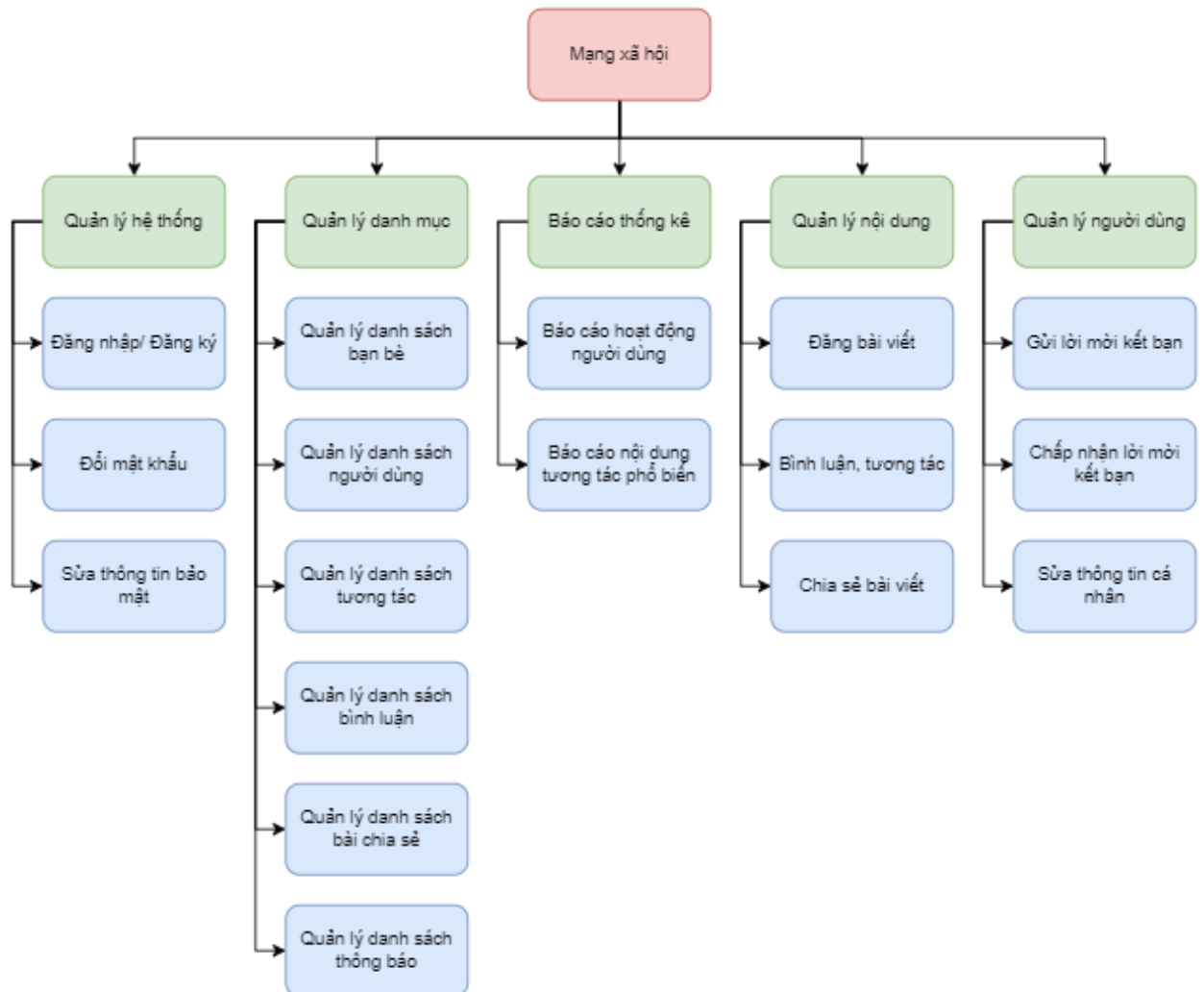


## CHƯƠNG 2. PHÂN TÍCH HỆ THỐNG

Link folder ảnh phần phân tích hệ thống: [Phân tích đồ án 1](#)

### 2.1 Sơ đồ phân rã chức năng



Hình 1. Sơ đồ phân rã chức năng

- Hệ thống mạng xã hội gồm 5 chức năng chính gồm: Quản lý hệ thống, Quản lý danh mục, Báo cáo thống kê hay các chức năng nghiệp vụ như Quản lý nội dung, Quản lý người dùng.
- **Quản lý hệ thống** bao gồm: Đăng ký/Đăng nhập, Đổi mật khẩu, Sửa thông tin bảo mật.
- **Quản lý danh mục**: Quản lý danh sách bạn bè, Quản lý danh sách người dùng, Quản lý danh sách tương tác, bình luận, chia sẻ, Quản lý danh sách thông báo. Mỗi chức năng trong quản lý danh mục bao gồm xem, thêm, sửa, xóa với sự phân quyền khác nhau.
- **Báo cáo thống kê**: Báo cáo hoạt động người dùng, Báo cáo nội dung tương tác phổ biến.
- Nghiệp vụ **quản lý nội dung**: Đăng bài viết, bình luận, tương tác, chia sẻ bài viết.

- Nghiệp vụ **quản lý người dùng**: Gửi lời mời kết bạn, chấp nhận lời mời kết bạn, sửa thông tin cá nhân.

## 2.2 Các actor và usecase chính theo các actor

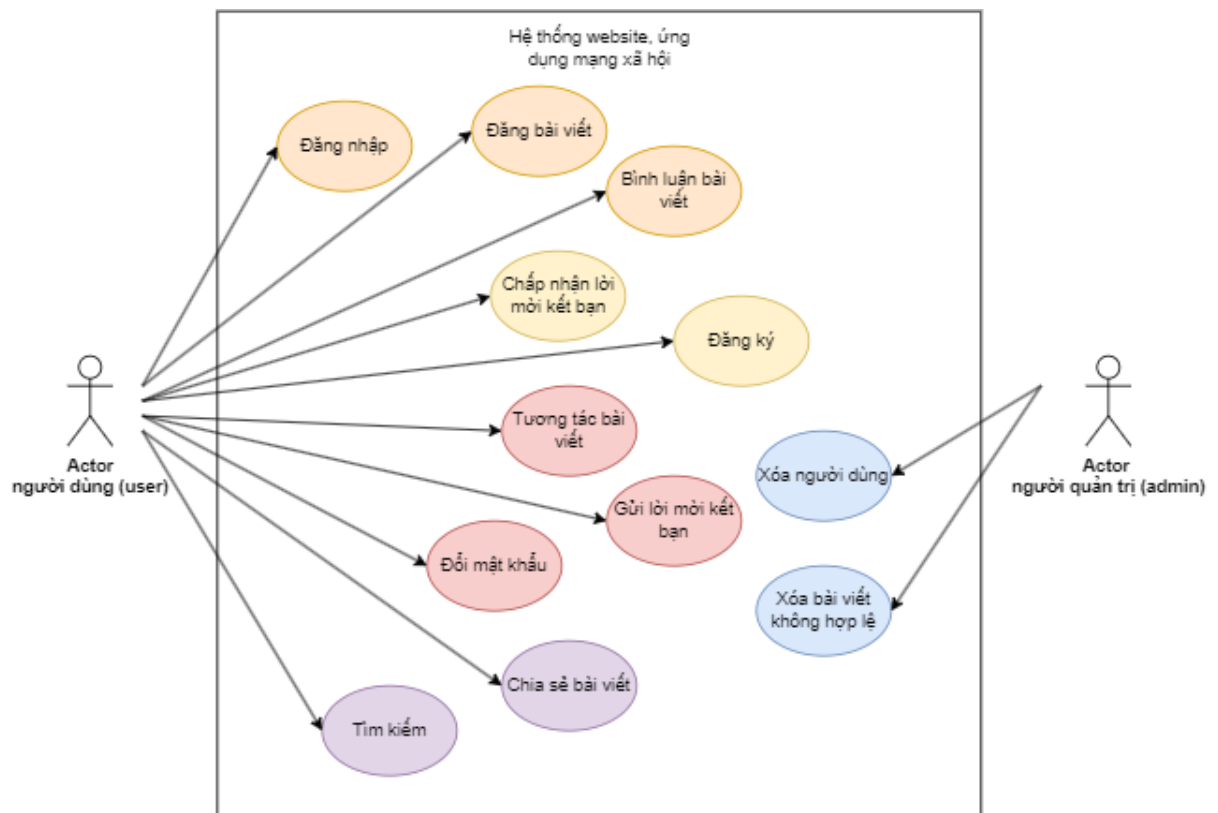
### - Actor: Người dùng (User)

- Đăng nhập
- Đăng kí
- Đổi mật khẩu
- Tìm kiếm
- Yêu cầu kết bạn
- Chấp nhận kết bạn
- Đăng bài viết
- Bình luận một bài viết, tương tác với một bài viết, Chia sẻ bài viết

### - Actor : Người quản trị (Admin)

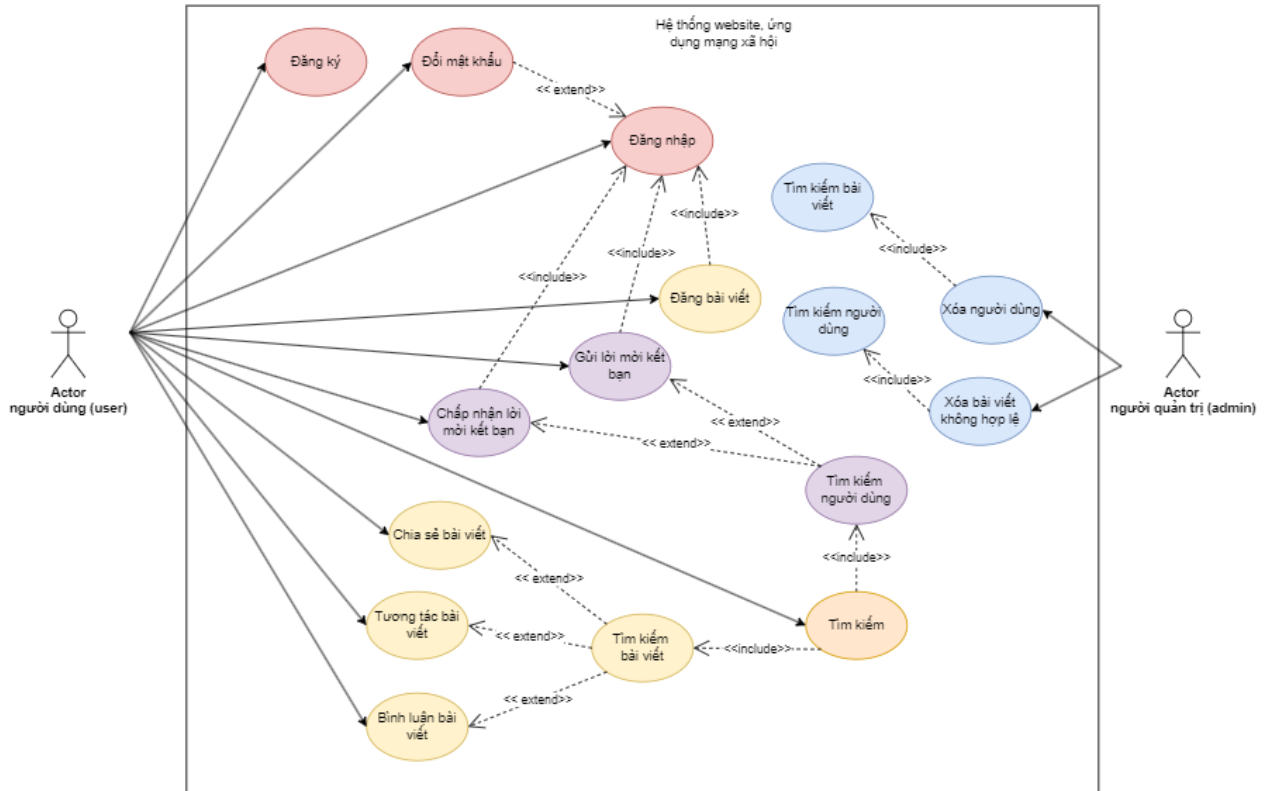
- Xóa một người dùng không hợp lệ
- Xóa bài viết không hợp lệ
- Đăng bài viết quảng cáo

### - Usecase Diagram tổng quát :



Hình 2. Usecase Diagram tổng quát

**- Usecase Diagram:**



*Hình 3. Usecase Diagram*

### 2.3 Đặc tả usecase và biểu diễn bằng sơ đồ activity

### 2.3.1 Usecase Đăng nhập:

- **Tên usecase:** Đăng nhập
- **Mô tả sơ lược:** Đăng nhập giúp người dùng quản lý thông tin cá nhân, đăng bài viết, chia sẻ bài viết, bình luận bài viết, thể hiện cảm xúc với bài viết, lấy danh sách bạn bè, danh sách người theo dõi, chấp nhận lời mời kết bạn, nhắn tin với bạn bè, nhận thông báo.
- **Actor chính:** người dùng (user)
- **Actor phụ:** Không có
- **Tiền điều kiện** (Pre-condition) : Người dùng đã có tài khoản, Thông tin đăng nhập chính xác
- **Hậu điều kiện** (Post-condition) : Đăng nhập thành công sẽ chuyển đến trang người dùng
- **Luồng sự kiện chính** (main flow) :
  1. Người dùng truy cập vào trang đăng nhập của website
  2. Người dùng nhập thông tin đăng nhập ( tài khoản , mật khẩu)
  3. Hệ thống kiểm tra thông tin đăng nhập và xác nhận tính hợp lệ của chúng
  4. Nếu thông tin không chính xác thì báo lỗi và yêu cầu nhập lại. Nếu thông tin chính xác , hệ thống sẽ cho phép người dùng truy cập và chuyển hướng họ đến trang chính của trang web

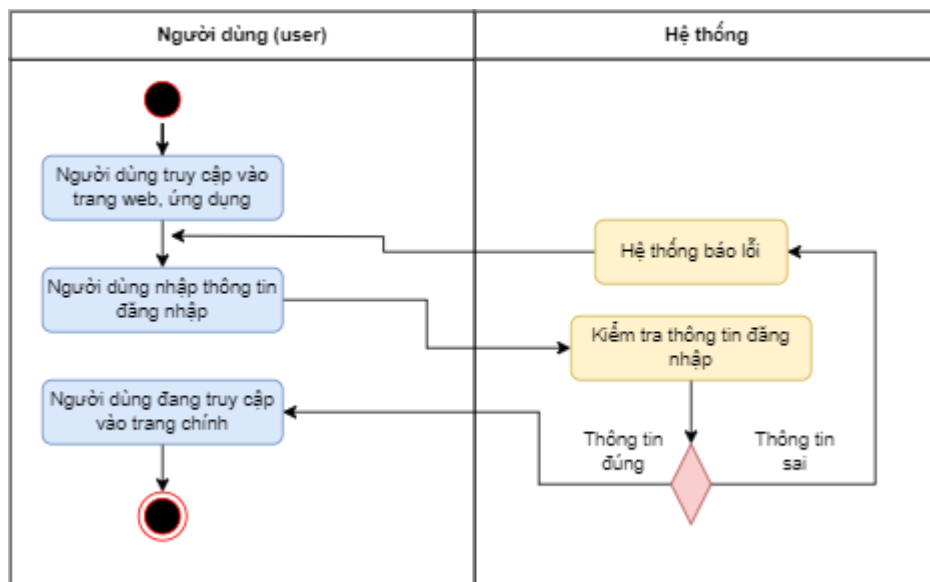
- **Luồng sự kiện thay thế** (alternate flow) : Nếu người dùng quên tài khoản hoặc mật khẩu(Bước 2)

1. Hệ thống hiển thị một liên kết “Quên mật khẩu” để người dùng có thể đặt lại mật khẩu
2. Người dùng nhấp vào liên kết và nhập tài khoản (email hoặc số điện thoại) của họ
3. Hệ thống gửi lại mã xác minh theo thông tin trên
4. Người dùng nhập mã và thiết lập lại mật khẩu

- **Luồng sự kiện ngoại lệ** (exception flow) : Nếu hệ thống không thể kết nối đến cơ sở dữ liệu, các lỗi xảy ra phía máy chủ.

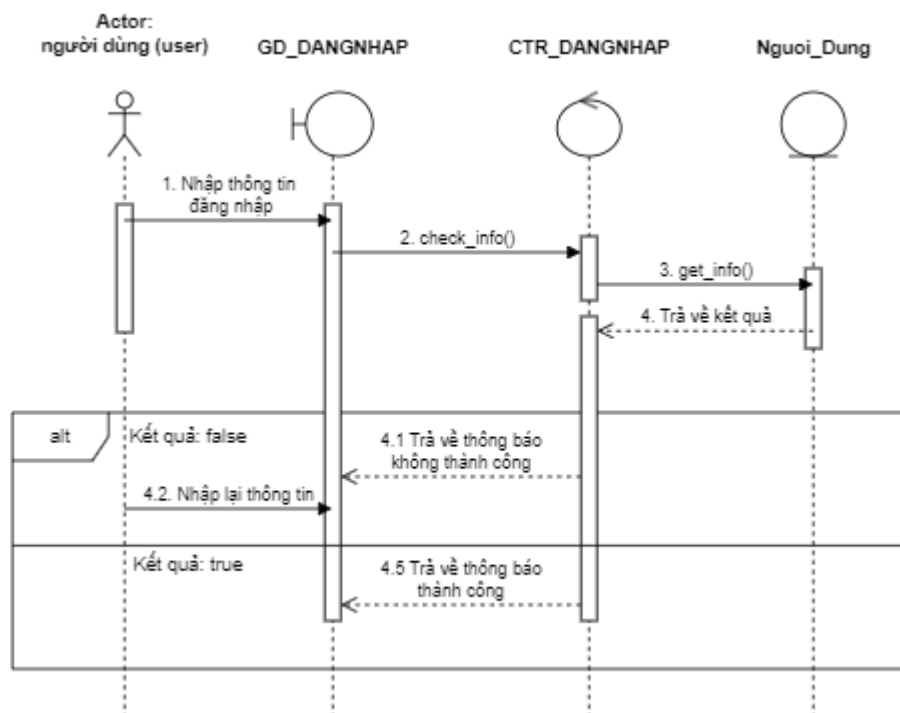
- Hệ thống hiển thị 1 thông báo đến cho người dùng : Thông báo không thể đăng nhập lúc này và yêu cầu thử lại sau

- **Activity diagram usecase Đăng nhập :**



Hình 4. Activity diagram usecase Đăng nhập

- **Sequence diagram usecase Đăng nhập :**



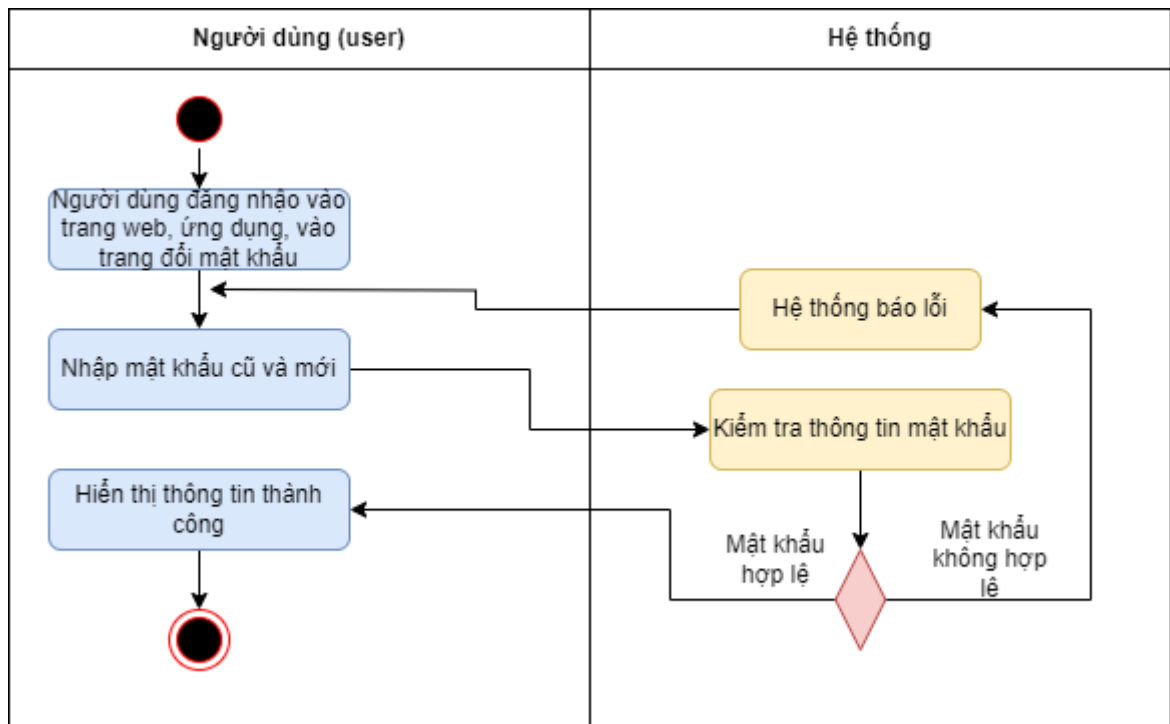
Hình 5. Sequence diagram usecase Đăng nhập

### 2.3.2 Usecase Đổi mật khẩu:

- **Tên usecase:** Đổi mật khẩu
- **Mô tả sơ lược:** Đổi mật khẩu phép người dùng thay đổi mật khẩu của tài khoản cá nhân trên trang web.
- **Actor chính:** người dùng (user)
- **Actor phụ:** Không có
- **Tiền điều kiện (Pre-condition):** Người dùng đã đăng nhập vào tài khoản của mình và có mật khẩu hiện tại.
- **Hậu điều kiện (Post-condition):** Mật khẩu của người dùng được thay đổi thành công.
- **Luồng sự kiện chính (main flow):**
  1. Người dùng đăng nhập vào website của hệ thống
  2. Người dùng truy cập vào trang thay đổi mật khẩu.
  3. Người dùng nhập mật khẩu hiện tại và mật khẩu mới của mình. Hệ thống kiểm tra tính hợp lệ của mật khẩu hiện tại và mật khẩu mới.
  4. Nếu mật khẩu hiện tại đúng, hệ thống thay đổi mật khẩu thành công và hiển thị thông báo xác nhận cho người dùng.
- **Luồng sự kiện thay thế (alternate flow):** 5. Nếu mật khẩu hiện tại không đúng , hay mật khẩu mới không khớp, hợp lệ, thông báo lỗi và yêu cầu thử lại sau.
- **Luồng sự kiện ngoại lệ (exception flow):** Nếu hệ thống gặp sự cố trong quá trình thay đổi mật khẩu (ví dụ: lỗi kết nối, lỗi máy chủ, lỗi cơ sở dữ liệu):

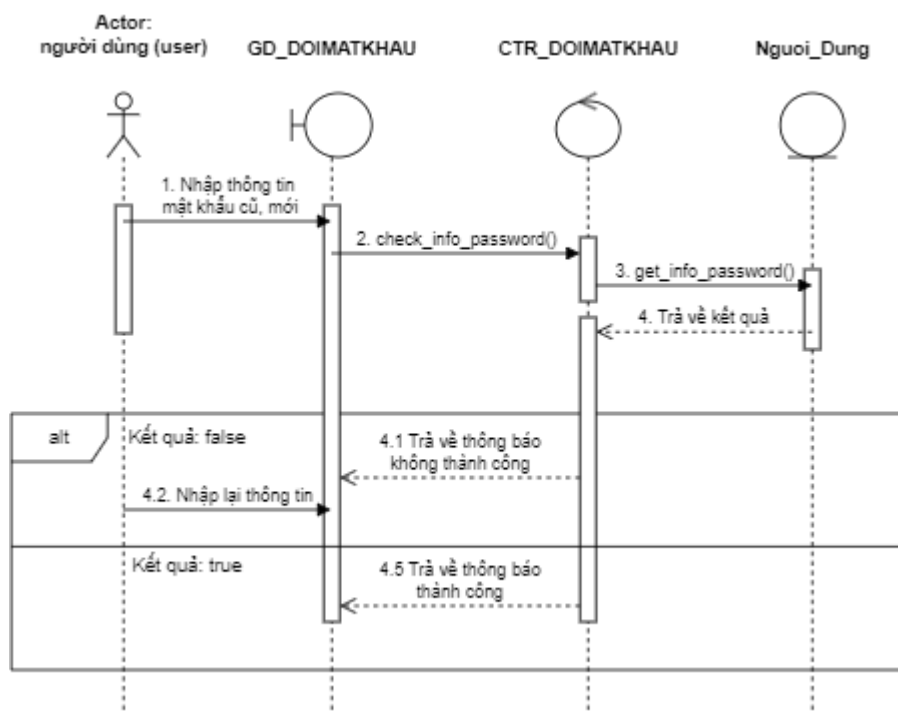
- Hệ thống hiển thị thông báo lỗi và yêu cầu người dùng thử lại sau.

### - Activity diagram Đổi mật khẩu



Hình 6. Activity Diagram usecase Đổi mật khẩu

### - Sequence diagram usecase Đổi mật khẩu :

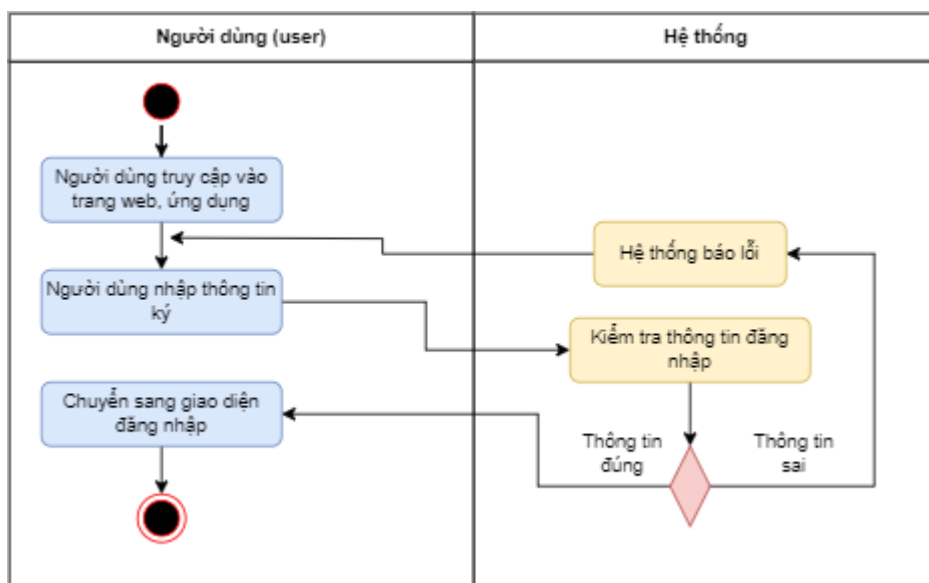


Hình 7. Sequence Diagram usecase Đổi mật khẩu

### 2.3.3 Usecase Đăng ký

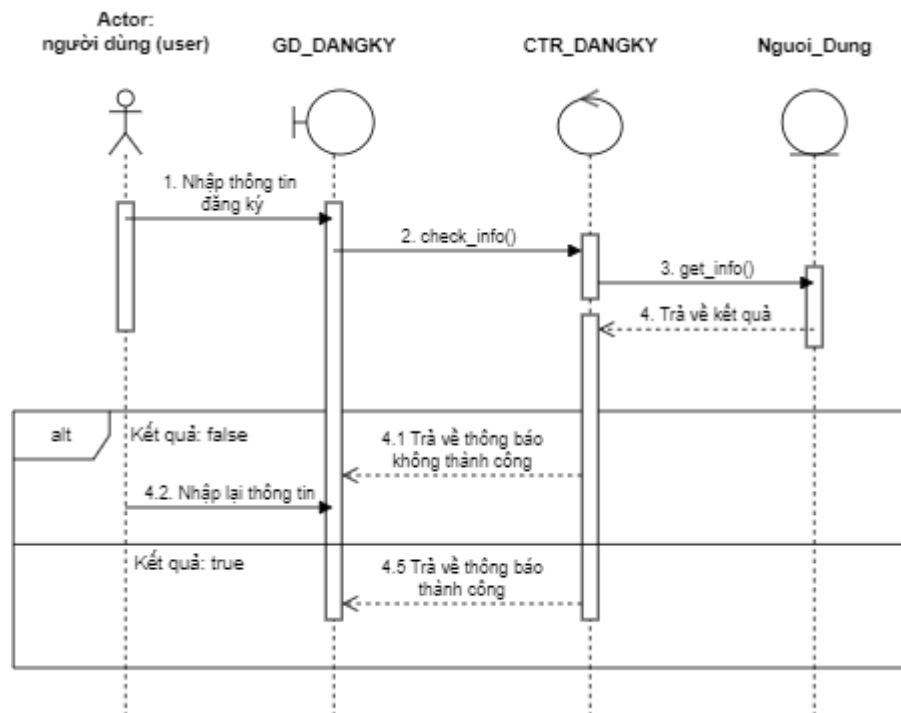
- **Tên usecase:** Đăng ký
- **Mô tả sơ lược:** Đăng ký cho phép người dùng tạo ra một tài khoản mới trên trang web để truy cập và sử dụng các tính năng của hệ thống.
- **Actor chính:** người dùng (user)
- **Actor phụ:** Không có
- **Tiền điều kiện (Pre-condition):** Truy cập thành công vào website đăng ký
- **Hậu điều kiện (Post-condition):** Tài khoản mới được tạo thành công và người dùng được đăng nhập tự động vào hệ thống.
- **Luồng sự kiện chính (main flow):**
  1. Người dùng truy cập vào trang đăng ký trên trang web.
  2. Người dùng nhập thông tin cá nhân cần thiết như tên, email và mật khẩu mong muốn.
  3. Hệ thống kiểm tra tính hợp lệ của thông tin đăng ký.
  4. Nếu thông tin không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại. Nếu hợp lệ, hệ thống tạo 1 tài khoản mới.
  5. Nếu thành công, hiển thị thông báo thành công, chuyển sang màn hình đăng nhập vào hệ thống.
- **Luồng sự kiện thay thế (alternate flow):** Nếu người dùng đã có tài khoản và cố gắng đăng ký một tài khoản mới với thông tin đã tồn tại:
  - Hệ thống hiển thị thông báo lỗi và yêu cầu người dùng sử dụng một địa chỉ email khác hoặc đăng nhập vào tài khoản hiện có.
- **Luồng sự kiện ngoại lệ (exception flow):** Không có.

#### -Activity diagram usecase Đăng ký:



Hình 8. Activity diagram usecase Đăng ký

- Sequence diagram usecase Đăng ký:



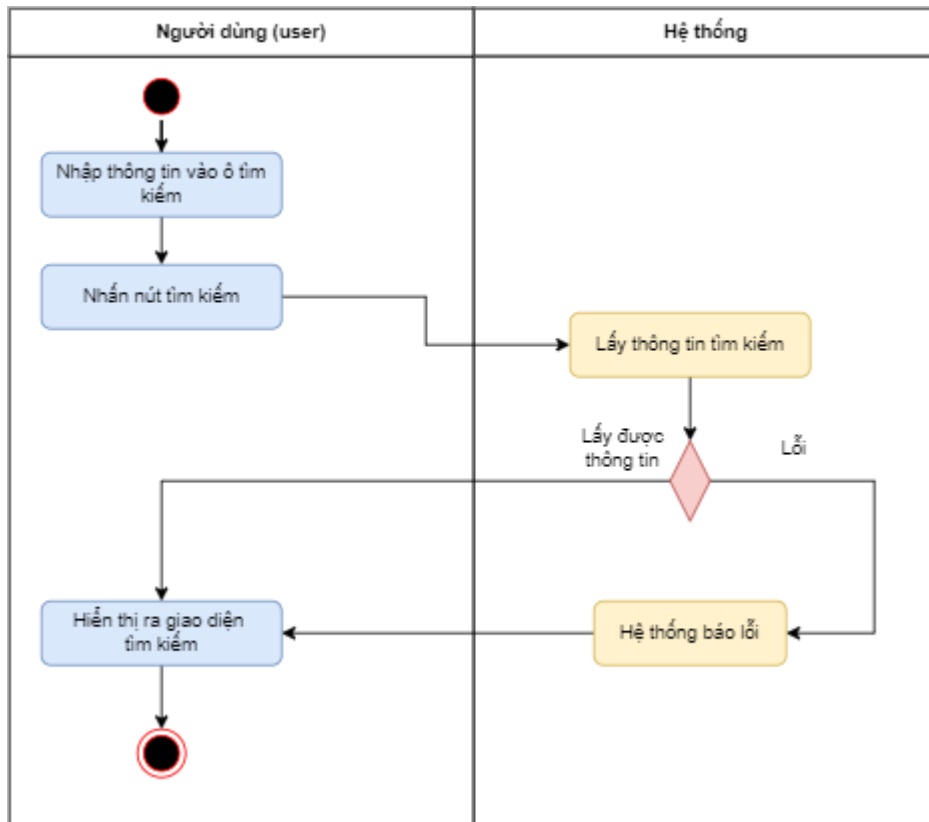
Hình 9. Sequence diagram usecase Đăng ký

### 2.3.4 Usecase Tìm kiếm

- **Tên usecase:** Tìm kiếm
- **Mô tả sơ lược:** Tìm kiếm cho phép người dùng tìm kiếm người dùng hoặc bài viết trên website, ứng dụng.
- **Actor chính:** người dùng (user)
- **Actor phụ:** không có
- **Tiền điều kiện (Pre-condition):** người dùng đã đăng nhập vào website, ứng dụng.
- **Hậu điều kiện (Post-condition):** thông tin về người dùng khác, bài đăng
- **Luồng sự kiện chính (main flow):**
  1. Người dùng truy cập vào màn hình chính website, ứng dụng.
  2. Người dùng nhập cụm từ cần tìm kiếm và ấn nút biểu tượng tìm kiếm.
  3. Hệ thống hiển thị một số người dùng và một số bài viết cùng với nút bấm “Xem thêm”
- **Luồng sự kiện thay thế (alternate flow):** người dùng có thể ấn nút “Xem thêm” để hiển thị thêm người dùng, bài viết.
- **Luồng sự kiện ngoại lệ (exception flow):** nếu có lỗi xảy ra trong quá trình tìm kiếm (lỗi máy chủ, lỗi truy xuất cơ sở dữ liệu)
  - Hệ thống hiển thị thông báo lỗi cụ thể cho người dùng và yêu cầu họ thử lại.

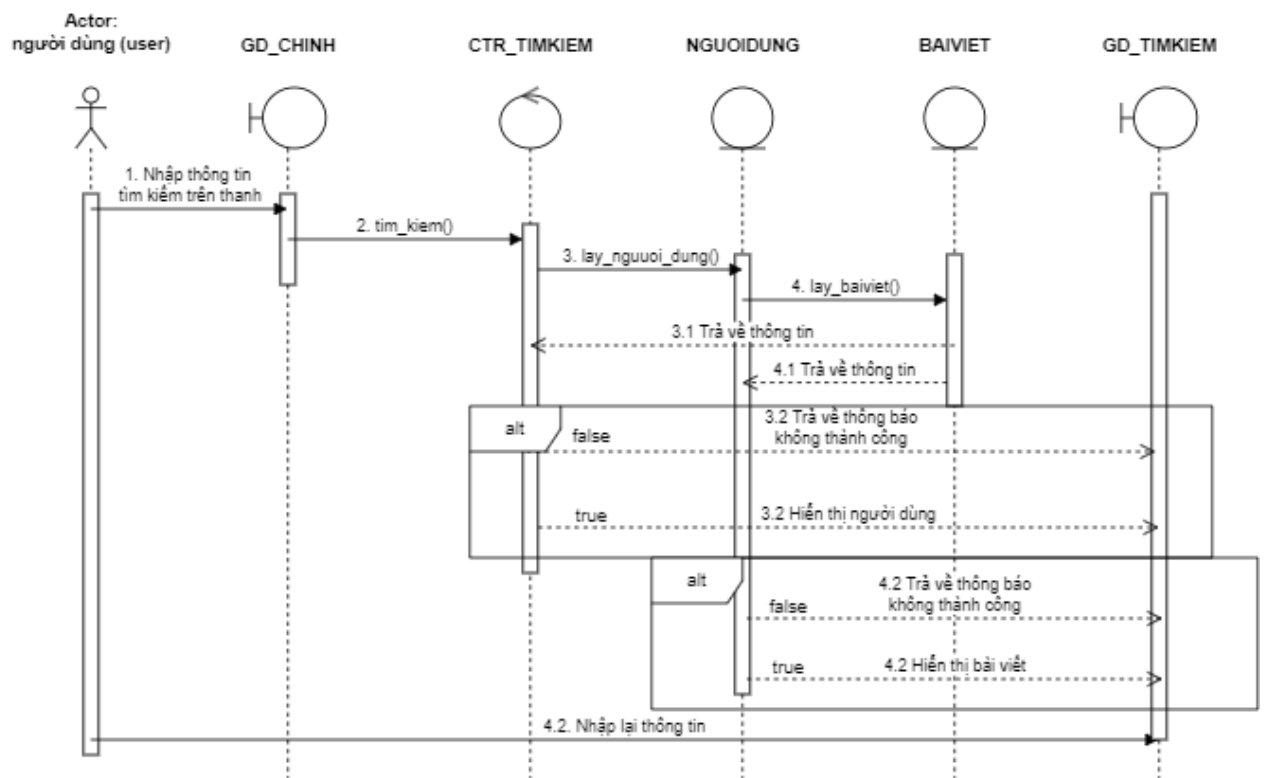


**- Activity diagram usecase Tìm kiếm:**



Hình 10. Activity diagram usecase Tìm kiếm

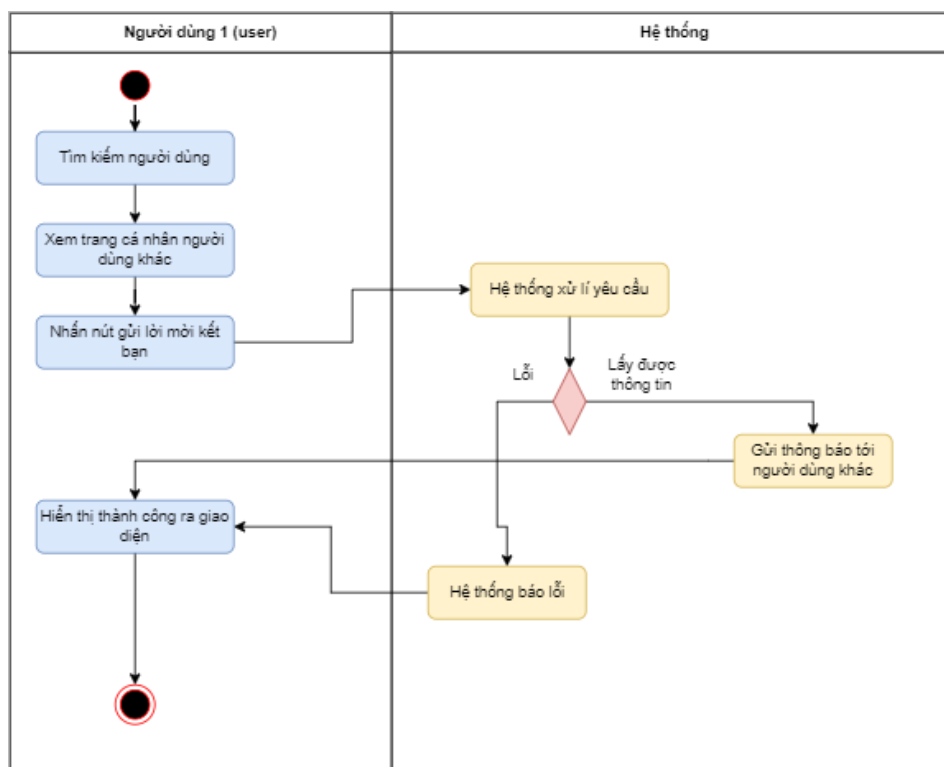
**- Sequence diagram Tìm kiếm**



Hình 11. Sequence diagram usecase Tìm kiếm

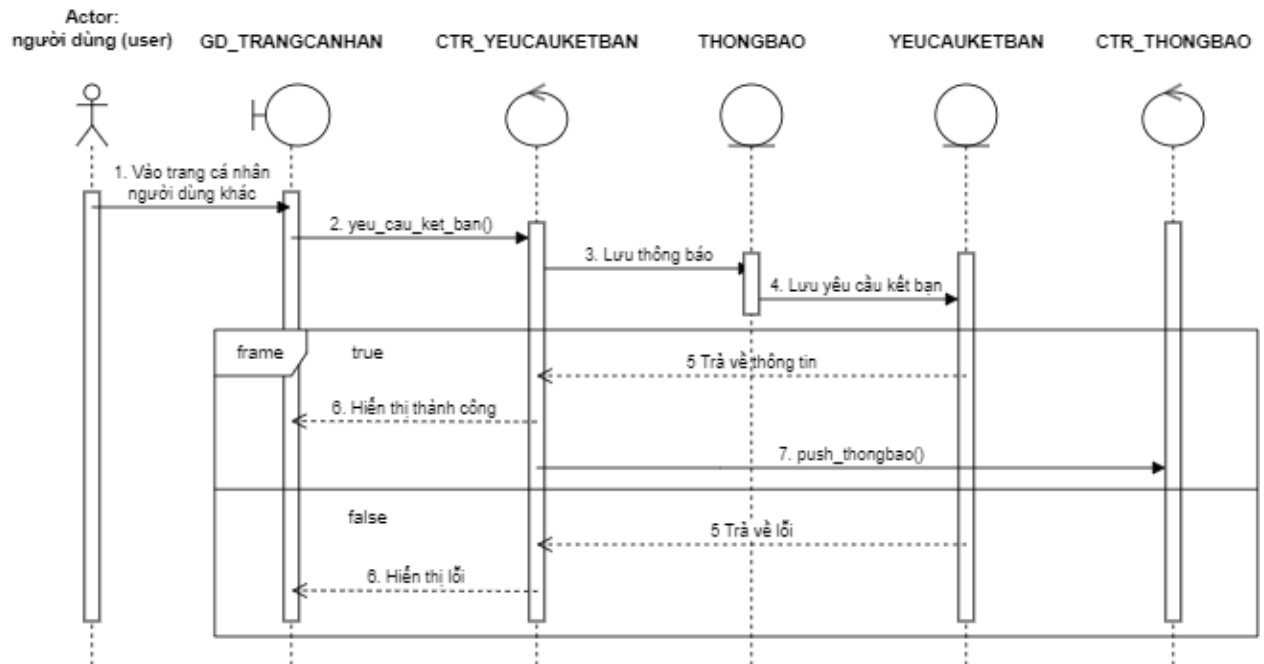
### 2.3.5 Usecase Yêu cầu kết bạn

- **Tên usecase:** Yêu cầu kết bạn
- **Mô tả sơ lược:** Yêu cầu kết bạn cho phép người dùng gửi một yêu cầu kết bạn cho người dùng khác.
- **Actor chính:** người dùng (user)
- **Actor phụ:** không có
- **Tiền điều kiện (Pre-condition):** người dùng đã đăng nhập vào website, ứng dụng.
- **Hậu điều kiện (Post-condition):** không có.
- **Luồng sự kiện chính (main flow):**
  1. Người dùng tìm kiếm người dùng khác trên thanh tìm kiếm hay qua các bài viết
  2. Vào trang cá nhân của người dùng đó
  3. Nhấn vào nút thêm bạn bè
  4. Hệ thống hiển thị thành công
  5. Người dùng khác nhận được thông báo
- **Luồng sự kiện thay thế (alternate flow):** không có
  - Hệ thống hiển thị thông báo cho người dùng và yêu cầu họ chọn sản phẩm khác hoặc quay lại sau.
- **Luồng sự kiện ngoại lệ (exception flow):** xảy ra lỗi truy xuất cơ sở dữ liệu.
  - Hệ thống hiển thị thông báo cho người dùng về lỗi, thử lại sau.
- **Activity diagram usecase Yêu cầu kết bạn**



Hình 12. Activity Diagram Yêu cầu kết bạn

## -Sequence diagram usecase Yêu cầu kết bạn:

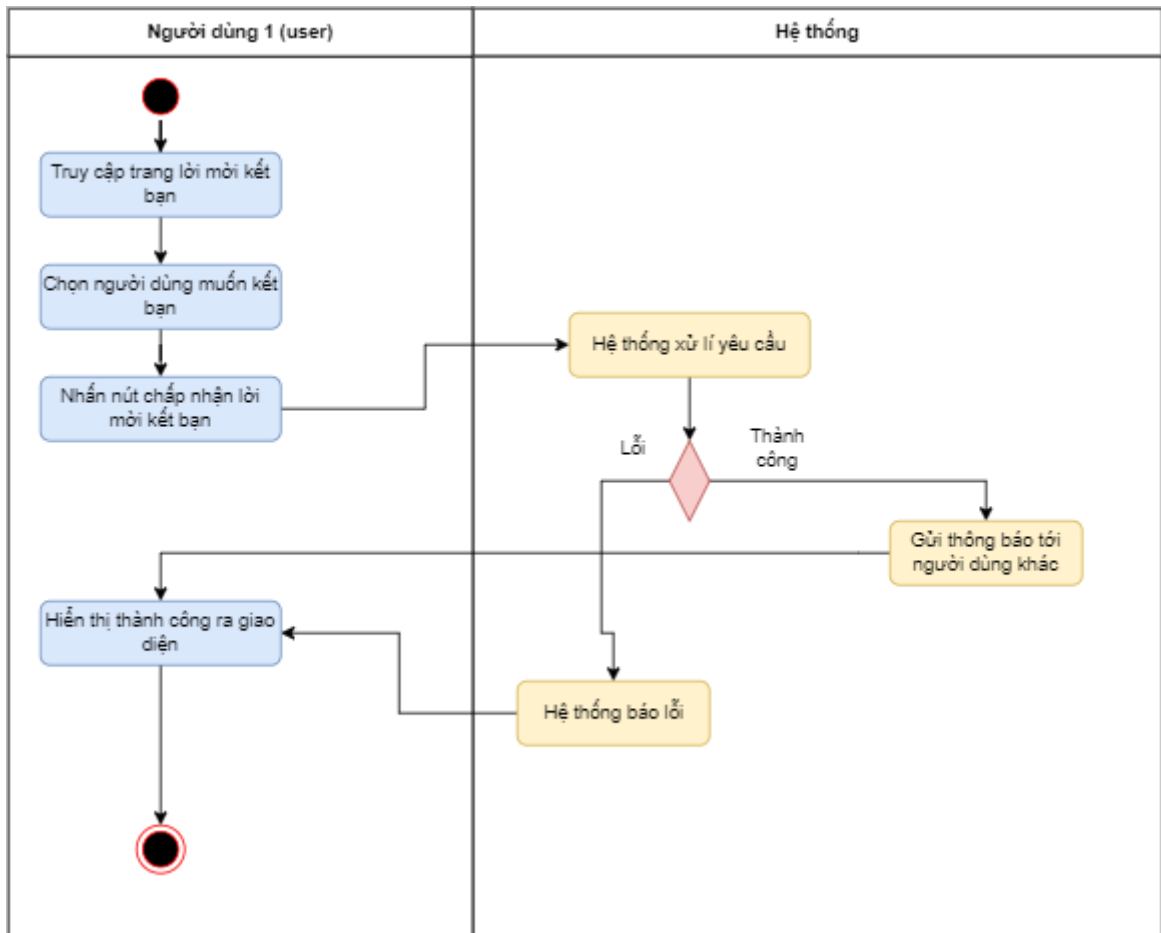


Hình 13. Sequence diagram Yêu cầu kết bạn

### 2.3.6 Usecase Chấp nhận kết bạn

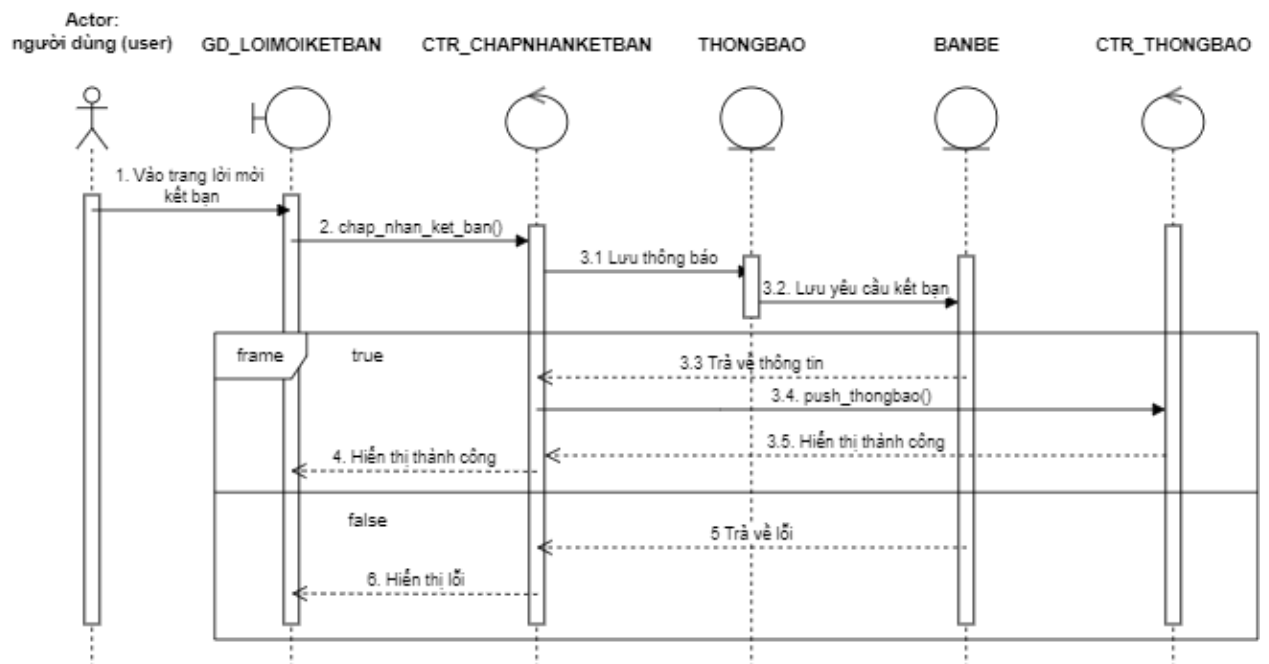
- **Tên usecase:** Chấp nhận kết bạn
- **Mô tả sơ lược:** Chấp nhận kết bạn cho phép người dùng chấp nhận một lời mời kết bạn từ người dùng khác
- **Actor chính:** người dùng (user)
- **Actor phụ:** không có
- **Tiền điều kiện (Pre-condition):** Người dùng phải đăng nhập
- **Hậu điều kiện (Post-condition):** Tồn tại lời mời kết bạn từ người dùng khác
- **Luồng sự kiện chính (main flow):**
  1. Người dùng truy cập vào trang có danh sách những lời mời kết bạn
  2. Người dùng chọn người dùng khác, nhấn nút “Chấp nhận”.
  3. Hệ thống xử lý gửi thông báo tới người dùng khác.
  4. Hệ thống phản hồi thành công về người dùng.
- **Luồng sự kiện thay thế (alternate flow):** Không có.
- **Luồng sự kiện ngoại lệ (exception flow):** Lỗi truy xuất dữ liệu trên cơ sở dữ liệu, lỗi máy chủ
  - Hệ thống hiển thị thông báo cho người dùng về lỗi, thử lại sau.

**-Activity diagram usecase Chấp nhận kết bạn:**



Hình 14. Activity diagram usecase Chấp nhận kết bạn

**- Sequence diagram usecase Chấp nhận kết bạn:**

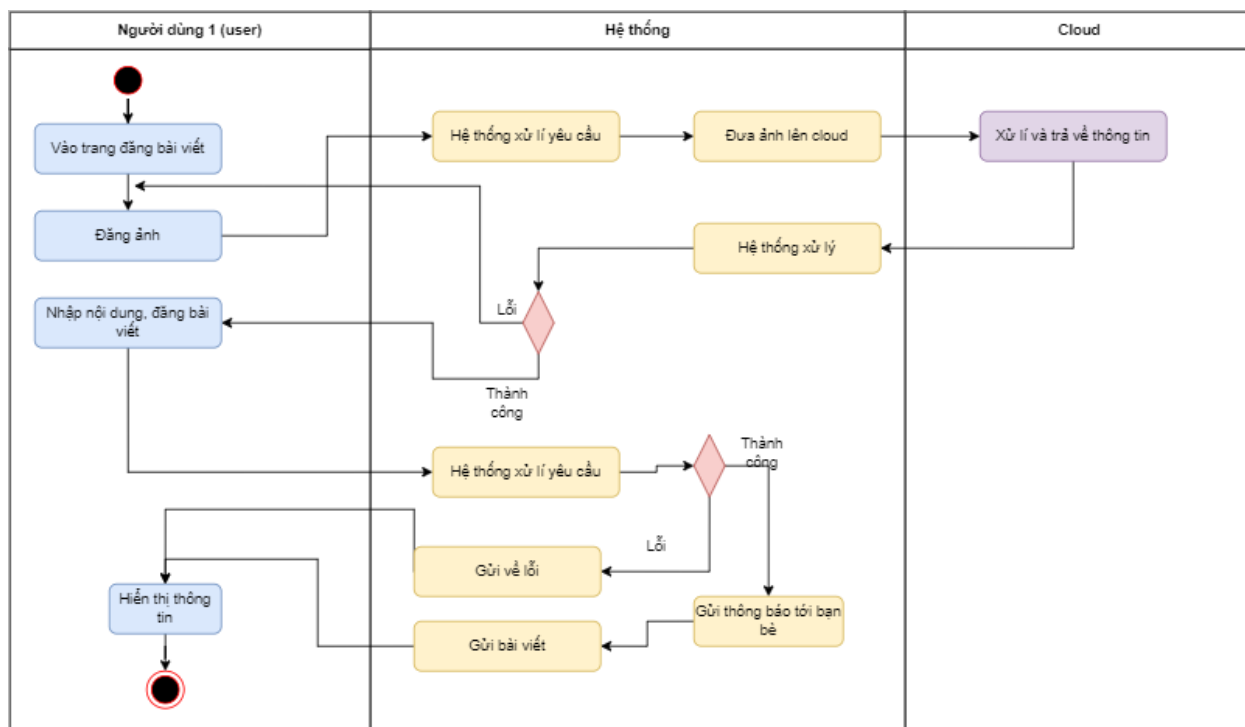


Hình 15. Sequence diagram usecase Chấp nhận kết bạn

### 2.3.7 Usecase Đăng bài viết

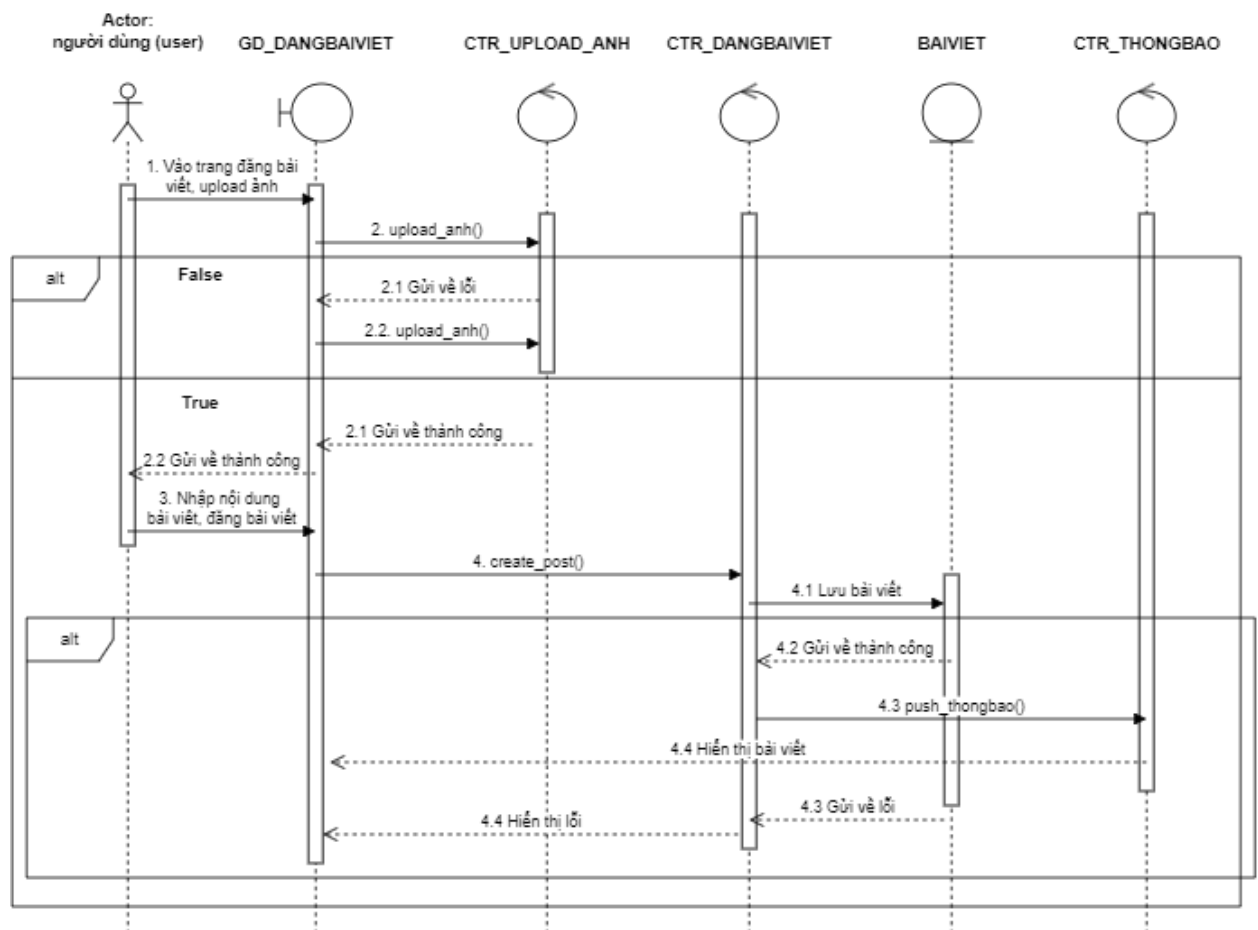
- **Tên usecase:** Đăng bài viết
- **Mô tả sơ lược:** Use case này mô tả quá trình tạo bài viết của người dùng.
- **Actor chính:** người dùng (user)
- **Actor phụ:** hệ thống lưu ảnh trên đám mây
- **Tiền điều kiện (Pre-condition):** Người dùng đã đăng nhập vào hệ thống.
- **Hậu điều kiện (Post-condition):** Bài viết hiển thị
- **Luồng sự kiện chính (main flow):**
  1. Người dùng vào trang đăng bài viết, hệ thống hiển thị giao diện đăng bài viết.
  2. Người dùng đăng ảnh, ảnh được tải lên hệ thống lưu ảnh trên đám mây.
  3. Người dùng nhập nội dung.
  4. Người dùng nhấn nút “Đăng”, hệ thống xử lý
  5. Hệ thống trả về thành công, hiển thị bài viết.
- **Luồng sự kiện thay thế (alternate flow):** Không có.
- **Luồng sự kiện ngoại lệ (exception flow):** Lỗi khi tải ảnh lên hệ thống lưu ảnh đám mây, truy xuất cơ sở dữ liệu, lỗi máy chủ
  - Hệ thống hiển thị thông báo cho người dùng về lỗi, thử lại sau.

#### -Activity diagram usecase Đăng bài viết:



Hình 16. Activity Diagram usecase Đăng bài viết

#### - Sequence diagram usecase Đăng bài viết :

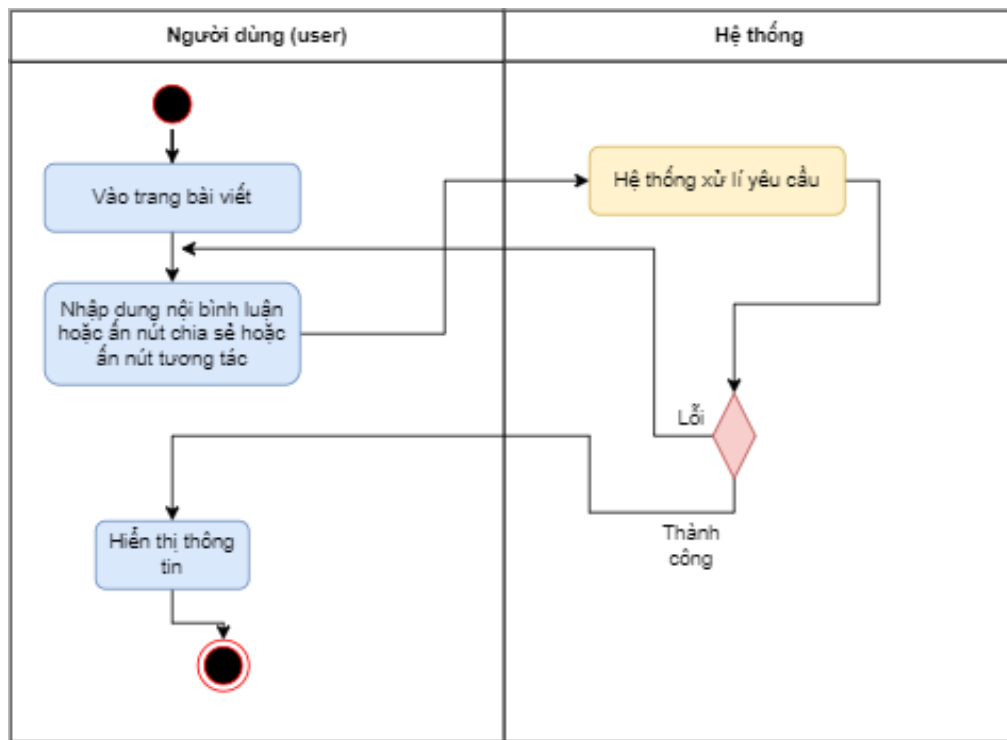


Hình 17. Sequence diagram usecase Đăng bài viết

### 2.3.8 Usecase Tương tác, bình luận, chia sẻ bài viết

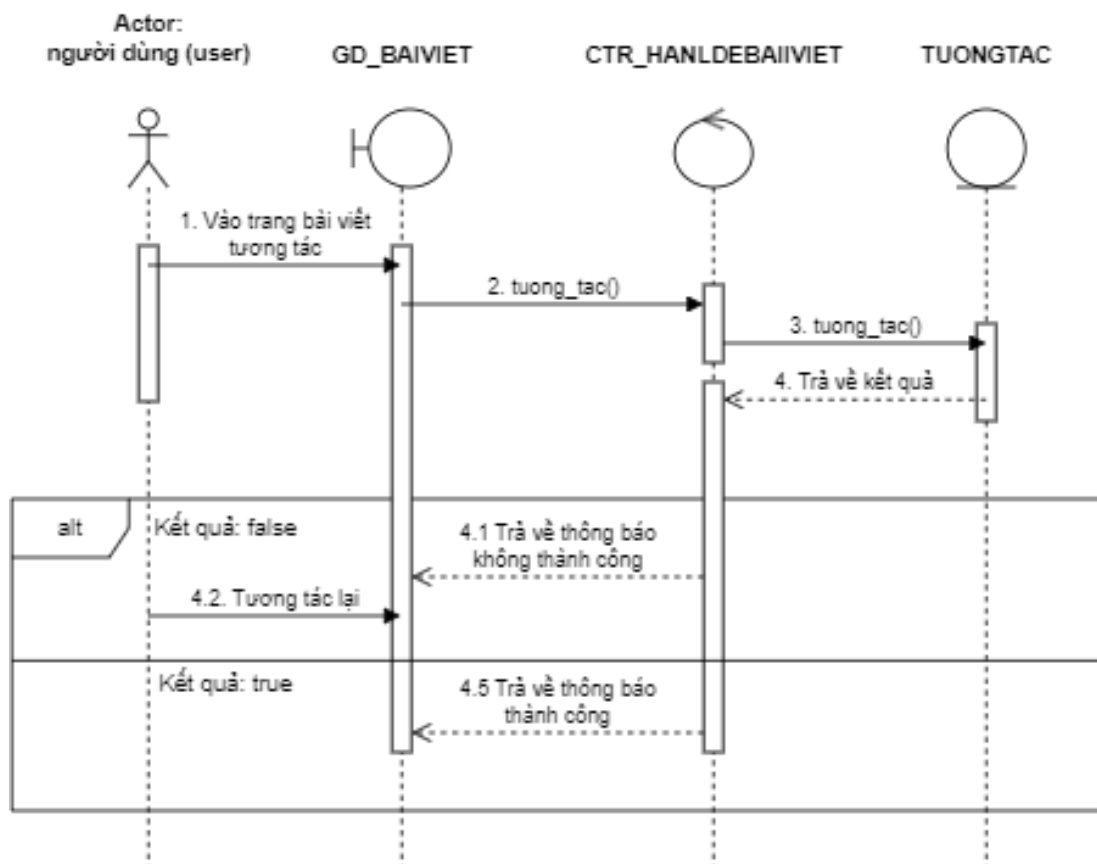
- **Tên usecase:** Tương tác, bình luận, chia sẻ bài viết
- **Mô tả sơ lược:** Tương tác, bình luận, chia sẻ bài viết
- **Actor:** Người quản lý(Admin)
- **Tiền điều kiện** (Pre-condition): Đăng nhập vào website, ứng dụng.
- **Hậu điều kiện** (Post-condition) : Tương tác, bình luận, chia sẻ bài viết.
- **Luồng sự kiện chính** (Main Flow):
  1. Người dùng chọn một bài viết, vào trang hiển thị bài viết.
  2. Người dùng tương tác, bình luận, chia sẻ.
  3. Hệ thống xử lý và trả về kết quả.
  4. Hiển thị kết quả,
- **Luồng sự kiện thay thế** (Alternate Flow): Không có
- **Luồng sự kiện ngoại lệ** (Exception Flow): Nếu có lỗi truy xuất cơ sở dữ liệu, lỗi máy chủ.
  - Hệ thống hiển thị thông báo cho người dùng về lỗi, thử lại sau.

-Activity diagram usecase Tương tác, bình luận, chia sẻ bài viết:



Hình 18. Sequence diagram usecase Tương tác, bình luận, chia sẻ bài viết

#### - Sequence diagram usecase Tương tác, bình luận, chia sẻ bài viết:



Hình 19. Sequence diagram usecase Tương tác, bình luận, chia sẻ bài viết

## CHƯƠNG 3. THIẾT KẾ CƠ SỞ DỮ LIỆU

### 3.1 Xác định các lớp khái niệm

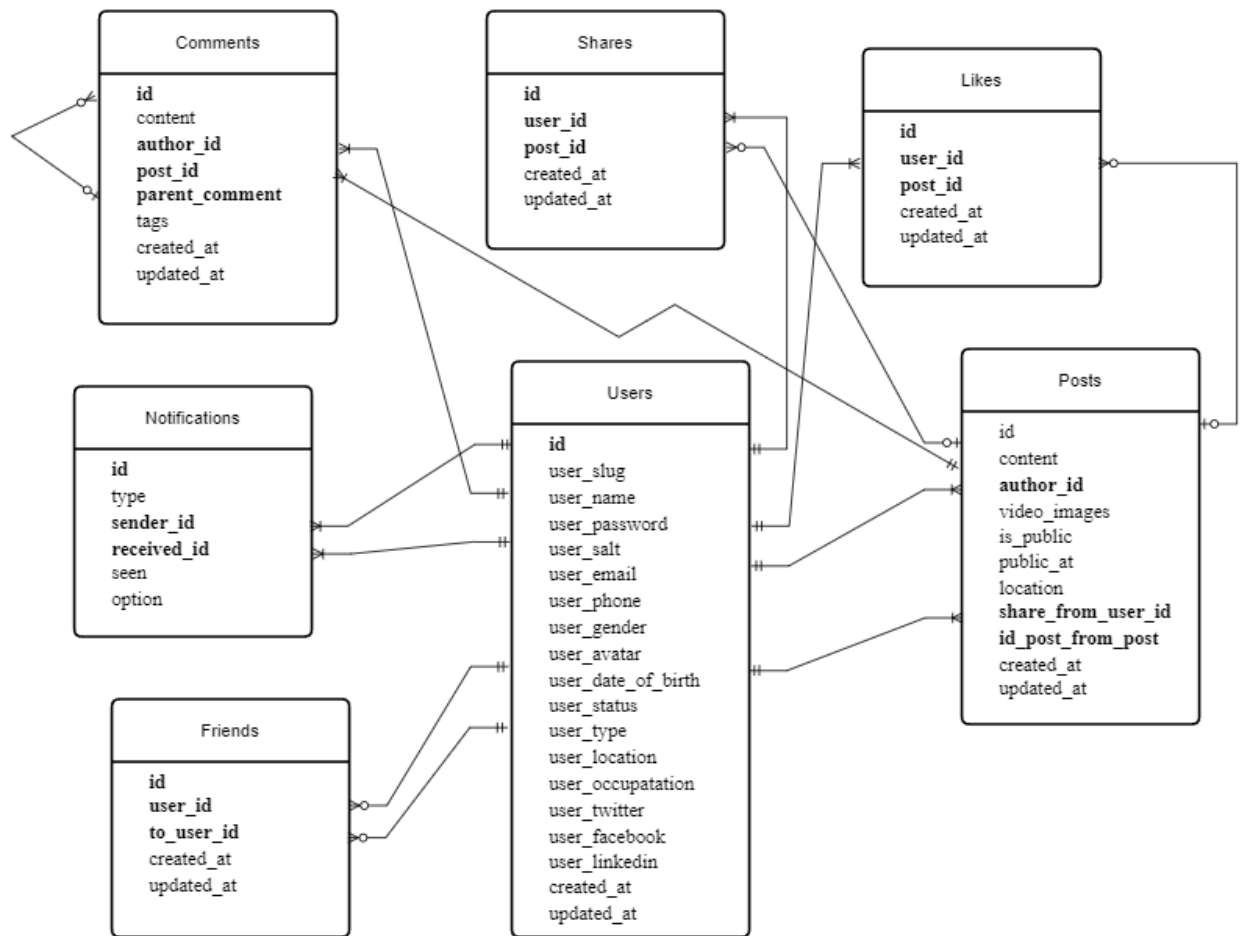
- **Lớp khái niệm** : người dùng (Users), bình luận (Comments), bài viết (Posts), theo dõi (Followers), lượt thích (Likes), chia sẻ (Shares), thông báo (Notifications), (bạn bè) Friends.

Lớp khái niệm	Thuộc tính
Users	<ul style="list-style-type: none"><li>• id (PK)</li><li>• user_slug</li><li>• user_name</li><li>• user_password</li><li>• user_salt</li><li>• user_email</li><li>• user_phone</li><li>• user_gender</li><li>• user_avatar</li><li>• user_date_of_birth</li><li>• user_status</li><li>• user_type</li><li>• user_location</li><li>• user_occupation</li><li>• user_twitter</li><li>• user_facebook</li><li>• user_linkedin</li><li>• created_at</li><li>• updated_at</li></ul>
Keys	<ul style="list-style-type: none"><li>• id (PK)</li><li>• user_id (FK)</li><li>• public_key</li><li>• private_key</li><li>• refresh_token</li><li>• created_at</li><li>• updated_at</li></ul>
RefreshTokens	<ul style="list-style-type: none"><li>• id (PK)</li><li>• key_id (FK)</li><li>• refresh_token</li><li>• created_at</li><li>• updated_at</li></ul>
Comments	<ul style="list-style-type: none"><li>• id (PK)</li><li>• content</li><li>• author_id (FK)</li><li>• post_id (FK)</li><li>• parent_comment (FK)</li><li>• tags</li></ul>



	<ul style="list-style-type: none"> <li>• created_at</li> <li>• updated_at</li> </ul>
<b>Followers</b>	<ul style="list-style-type: none"> <li>• id (PK)</li> <li>• follower_user_id (FK)</li> <li>• followed_user_id (FK)</li> <li>• created_at</li> <li>• updated_at</li> </ul>
<b>Friends</b>	<ul style="list-style-type: none"> <li>• id (PK)</li> <li>• user_id (FK)</li> <li>• to_user_id (FK)</li> <li>• created_at</li> <li>• updated_at</li> </ul>
<b>Likes</b>	<ul style="list-style-type: none"> <li>• id (PK)</li> <li>• user_id (FK)</li> <li>• post_id (FK)</li> <li>• created_at</li> <li>• updated_at</li> </ul>
<b>Shares</b>	<ul style="list-style-type: none"> <li>• id (PK)</li> <li>• user_id (FK)</li> <li>• post_id (FK)</li> <li>• created_at</li> <li>• updated_at</li> </ul>
<b>Posts</b>	<ul style="list-style-type: none"> <li>• id (PK)</li> <li>• content</li> <li>• author_id</li> <li>• videos_image</li> <li>• is_public</li> <li>• public_at</li> <li>• location</li> <li>• share_from_user_id</li> <li>• id_post_from_share</li> <li>• created_at</li> <li>• updated_at</li> </ul>
<b>Notifications</b>	<ul style="list-style-type: none"> <li>• id (PK)</li> <li>• type</li> <li>• sender_id (FK)</li> <li>• receivedId (FK)</li> <li>• seen</li> <li>• option</li> </ul>

### 3.2 Sơ đồ domain



Hình 20. Sơ đồ domain

## CHƯƠNG 4. TỔNG QUAN HỆ THỐNG

### 4.1 Một số khái niệm, công nghệ

#### 4.1.1 API và Restful API

+ **Khái niệm:** API (Application Programming Interface) là một tập hợp các định nghĩa và giao thức được sử dụng để xây dựng và tích hợp phần mềm ứng dụng. API cho phép các ứng dụng và dịch vụ khác nhau giao tiếp với nhau. Một trong những loại API phổ biến nhất là RESTful API.

Theo [Wikipedia](#):

"An application programming interface (API) is a set of subroutine definitions, communication protocols, and tools for building software."

+ **RESTful API:** REST (Representational State Transfer) là một phong cách kiến trúc thường được sử dụng để phát triển các dịch vụ web. **RESTful API** là một API được thiết kế dựa trên các nguyên tắc REST, cho phép các hệ thống giao tiếp với nhau qua giao thức HTTP.

Theo [REST Wikipedia](#):

"Representational state transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services."

#### + Các Phương Thức HTTP Trong RESTful API:

RESTful API sử dụng các phương thức HTTP để thực hiện các thao tác CRUD (Create, Read, Update, Delete):

- **GET:** Truy vấn dữ liệu từ máy chủ.
- **POST:** Tạo mới tài nguyên trên máy chủ.
- **PUT:** Cập nhật tài nguyên hiện có trên máy chủ.
- **DELETE:** Xóa tài nguyên khỏi máy chủ.
- **PATCH:** Cập nhật từng phần của tài nguyên.

=> RESTful API đã trở thành một chuẩn mực trong việc phát triển các dịch vụ web và ứng dụng phân tán. Với các nguyên tắc rõ ràng và sự linh hoạt cao, RESTful API giúp các nhà phát triển xây dựng các hệ thống mạnh mẽ, dễ bảo trì và dễ mở rộng. Hy vọng qua bài viết này, bạn đã có cái nhìn tổng quan về API và RESTful API, cùng với những lợi ích và thách thức khi sử dụng chúng.

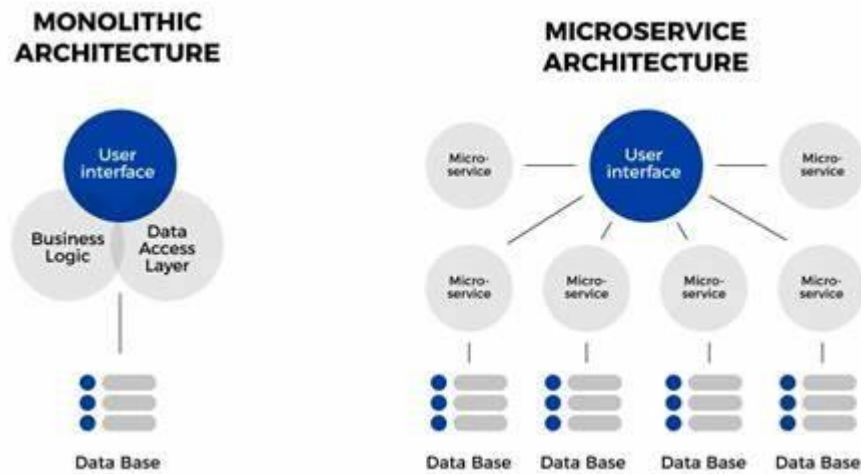
Việc hiểu rõ và áp dụng đúng các nguyên tắc của RESTful API sẽ giúp bạn xây dựng các dịch vụ web hiệu quả và đáp ứng được các yêu cầu kinh doanh ngày càng phức tạp.

#### 4.1.2 Mô hình microservice

- **Khái niệm:** Microservices - also known as the [microservice architecture](#) - is an architectural style that structures an application as a collection of services that are:

- [Independently deployable](#)
- [Loosely coupled](#)

Services are typically organized around business capabilities. Each service is often owned by a single, small team. ([What are microservices?](#))



Hình 21. Microservice architecture vs monolithic architecture

- Một số đặc điểm của Kiến trúc Microservice:

##### **Độc lập và Tự chủ:**

- Mỗi microservice là một đơn vị độc lập và có thể phát triển, triển khai và mở rộng riêng rẽ.
- Các dịch vụ không phụ thuộc vào việc triển khai cùng một thời điểm với các dịch vụ khác.

##### **Chia nhỏ và Tập trung:**

- Các dịch vụ được chia nhỏ dựa trên các chức năng kinh doanh cụ thể (domain-driven design).
- Mỗi dịch vụ đảm nhiệm một nhiệm vụ duy nhất và rõ ràng.

##### **Giao tiếp qua Giao thức Nhẹ:**

- Các microservice thường giao tiếp với nhau thông qua các giao thức nhẹ như HTTP/HTTPS, REST, hoặc các giao thức nhắn tin như AMQP (Advanced Message Queuing Protocol).

##### **Triển khai Độc lập:**

- Mỗi microservice có thể được triển khai, mở rộng và bảo trì một cách độc lập mà không ảnh hưởng đến các dịch vụ khác.

#### **Tính chịu lỗi và Độc lập:**

- Sự cố của một microservice không nên ảnh hưởng đến hoạt động của các dịch vụ khác.
- Kiến trúc microservice thường tích hợp các cơ chế chịu lỗi như circuit breakers và retries để đảm bảo tính liên tục của hệ thống.

#### **Khả năng mở rộng:**

- Mỗi microservice có thể được mở rộng độc lập dựa trên nhu cầu sử dụng của nó.

### **-Lợi ích của Kiến trúc Microservice**

#### **Phát triển và Triển khai Nhanh hơn:**

- Các nhóm phát triển có thể làm việc song song trên các dịch vụ khác nhau mà không phụ thuộc lẫn nhau.
- Việc triển khai và bảo trì từng dịch vụ có thể diễn ra một cách nhanh chóng và hiệu quả.

#### **Tính Linh hoạt và Dễ Dàng Mở Rộng:**

- Hệ thống dễ dàng mở rộng bằng cách triển khai thêm các instance của microservice cần mở rộng.
- Tính linh hoạt cao cho phép các dịch vụ được cập nhật, thay đổi hoặc thay thế mà không ảnh hưởng đến toàn bộ hệ thống.

#### **Khả Năng Chịu Lỗi Cao:**

- Các dịch vụ có thể tự phục hồi và tiếp tục hoạt động ngay cả khi có lỗi xảy ra trong một phần của hệ thống.

#### **Quản lý và Bảo trì Dễ Dàng:**

- Với từng dịch vụ nhỏ gọn, việc theo dõi, bảo trì và sửa lỗi trở nên đơn giản hơn.

#### **Độc lập về Công nghệ:**

- Các dịch vụ có thể được phát triển bằng các ngôn ngữ lập trình và công nghệ khác nhau, miễn là chúng tuân thủ các giao thức giao tiếp chung.

### **4.1.3 Ngôn ngữ lập trình javascripts**

#### **- Khái niệm:**

**JavaScript** là một ngôn ngữ lập trình bậc cao, đa năng, được sử dụng rộng rãi trong phát triển web. Ban đầu được thiết kế để chạy trên các trình duyệt web như một ngôn ngữ kịch bản phía khách hàng (client-side scripting), JavaScript đã

phát triển vượt ra ngoài phạm vi ban đầu của nó và hiện tại có thể được sử dụng ở cả phía khách hàng và phía máy chủ.

Theo [Mozilla Developer Network](#):

"JavaScript is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. JavaScript enables interactive web pages and is an essential part of web applications."

**- Các tính năng chính:**

- **Đa Năng:** Có thể được sử dụng cho cả phía khách hàng và phía máy chủ.
- **Không Đồng Bộ và Sự Kiện-Driven:** Hỗ trợ lập trình không đồng bộ qua callbacks, promises và async/await.
- **Đối Tượng và Hướng Đối Tượng:** Hỗ trợ cả lập trình đối tượng và hướng đối tượng.
- **Tích Hợp Dễ Dàng:** Dễ dàng tích hợp với HTML và CSS.
- **Cộng Đồng Lớn:** Có một cộng đồng lớn và nhiều thư viện, framework hỗ trợ.

#### 4.1.4 Môi trường Nodejs

**- Khái niệm:**

**Node.js** là một môi trường runtime mã nguồn mở, đa nền tảng cho JavaScript, được xây dựng trên engine V8 của Google Chrome. Node.js cho phép các nhà phát triển sử dụng JavaScript để viết mã chạy trên máy chủ, thay vì chỉ trên trình duyệt. Node.js nổi bật với khả năng xử lý I/O không đồng bộ và sự kiện-driven, làm cho nó lý tưởng cho các ứng dụng thời gian thực và xử lý dữ liệu lớn.

Theo Node.js Official Website:

"Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient."

**- Các tính năng chính:**

- **Event-Driven và Non-Blocking I/O:** Xử lý I/O không đồng bộ, giúp tối ưu hóa hiệu suất và khả năng mở rộng.
- **Single-Threaded:** Sử dụng một luồng duy nhất để xử lý các yêu cầu, nhưng có khả năng mở rộng thông qua các worker threads.
- **NPM (Node Package Manager):** Một hệ thống quản lý gói mạnh mẽ với hàng nghìn gói thư viện và công cụ.
- **Tích Hợp Dễ Dàng với JavaScript:** Sử dụng cùng một ngôn ngữ lập trình cho cả phía máy chủ và phía khách hàng.

#### 4.1.5 Framework ExpressJs

##### - Khái niệm:

**Express.js** là một framework web minimal và linh hoạt cho Node.js, cung cấp một bộ công cụ mạnh mẽ cho các ứng dụng web và API. Express.js giúp đơn giản hóa quá trình phát triển và triển khai các ứng dụng Node.js bằng cách cung cấp các tính năng cơ bản như routing, middleware, xử lý request và response.

Theo [Express.js Official Website](#):

"Express is a fast, unopinionated, minimalist web framework for Node.js."

##### - Các tính năng chính:

- **Routing Linh Hoạt:** Hỗ trợ các route động và tĩnh.
- **Middleware:** Hỗ trợ các middleware để xử lý các request, response và các thao tác trung gian.
- **Tích Hợp Dễ Dàng:** Dễ dàng tích hợp với các cơ sở dữ liệu, template engines và các thư viện khác.
- **Tài Liệu Phong Phú:** Có tài liệu chi tiết và cộng đồng hỗ trợ lớn.

Việc sử dụng ExpressJs sẽ là công cụ để xây dựng hệ thống theo chuẩn restful api.

#### 4.1.6 Cơ sở dữ liệu PostgreSQL

##### - Khái niệm:

**PostgreSQL** là một hệ quản trị cơ sở dữ liệu quan hệ đối tượng (ORDBMS) mã nguồn mở mạnh mẽ và tiên tiến. PostgreSQL được biết đến với sự ổn định, hiệu suất cao và tuân thủ các tiêu chuẩn SQL nghiêm ngặt. Nó được thiết kế để xử lý một loạt các khối lượng công việc từ các ứng dụng nhỏ đến các hệ thống dữ liệu lớn và phức tạp.

Theo [PostgreSQL Official Website](#):

"PostgreSQL is a powerful, open-source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance."

##### - Một số đặc điểm:

- **Mã Nguồn Mở:** PostgreSQL là phần mềm mã nguồn mở và miễn phí, được phát triển và duy trì bởi một cộng đồng lớn và tích cực.
- **Tuân Thủ SQL:** PostgreSQL hỗ trợ hầu hết các tiêu chuẩn SQL, bao gồm SQL:2011, giúp nó trở thành một trong những hệ quản trị cơ sở dữ liệu tuân thủ chuẩn nhất hiện nay.

- **Tích Hợp Đối Tượng:** Ngoài các tính năng cơ sở dữ liệu quan hệ truyền thống, PostgreSQL còn hỗ trợ các kiểu dữ liệu đối tượng, giúp quản lý dữ liệu phức tạp dễ dàng hơn.
- **Mở Rộng:** PostgreSQL có thể mở rộng với các mô-đun, thư viện và các ngôn ngữ lập trình khác nhau. Người dùng có thể thêm các chức năng mới bằng cách viết các hàm bằng ngôn ngữ như PL/pgSQL, PL/Tcl, PL/Perl, và PL/Python.
- **Bảo Mật:** Hỗ trợ các cơ chế xác thực và ủy quyền mạnh mẽ, bao gồm xác thực dựa trên vai trò và các phương thức mã hóa kết nối.

- **Các tính năng chính:**

- **Kiểu Dữ Liệu Phong Phú:** PostgreSQL hỗ trợ nhiều kiểu dữ liệu phong phú bao gồm số nguyên, số thực, chuỗi, ngày giờ, và cả các kiểu dữ liệu địa lý như PostGIS.
- **Khóa Ngoại, Trigger, và Views:** Hỗ trợ đầy đủ các tính năng của hệ quản trị cơ sở dữ liệu quan hệ như khóa ngoại, trigger, và views.
- **Giao Dịch (Transactions):** Hỗ trợ các giao dịch ACID (Atomicity, Consistency, Isolation, Durability) để đảm bảo tính nhất quán và độ tin cậy của dữ liệu.
- **Chỉ Mục (Indexes):** Hỗ trợ nhiều loại chỉ mục như B-tree, Hash, GIN, và GiST giúp tối ưu hóa truy vấn và tìm kiếm dữ liệu.
- **Sao Lưu và Phục Hồi:** Cung cấp các công cụ sao lưu và phục hồi mạnh mẽ, bao gồm cơ chế sao lưu liên tục (PITR - Point In Time Recovery).
- **Tối Ưu Hóa Truy Vấn:** PostgreSQL sử dụng các thuật toán tối ưu hóa truy vấn tiên tiến, giúp cải thiện hiệu suất truy vấn đáng kể.
- **Hỗ Trợ Đa Người Dùng:** Cho phép nhiều người dùng và nhiều phiên làm việc đồng thời mà không ảnh hưởng đến hiệu suất.

#### 4.1.7 Cơ sở dữ liệu MongoDB

- **Khái niệm:**

**MongoDB** là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở, sử dụng mô hình dữ liệu hướng tài liệu (document-oriented). Thay vì lưu trữ dữ liệu trong các bảng theo hàng và cột như các cơ sở dữ liệu quan hệ, MongoDB lưu trữ dữ liệu dưới dạng JSON-like documents, cho phép linh hoạt trong việc quản lý các dữ liệu phức tạp và không có cấu trúc rõ ràng.

Theo [MongoDB Official Website](#):

"MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need."

- **Một số đặc điểm:**



- **Mô Hình Dữ Liệu Linh Hoạt:** Sử dụng JSON-like documents giúp lưu trữ dữ liệu có cấu trúc động, phù hợp với nhiều loại ứng dụng khác nhau.
- **Mở Rộng Theo Chiều Ngang:** Hỗ trợ sharding, cho phép mở rộng cơ sở dữ liệu bằng cách phân phối dữ liệu trên nhiều máy chủ.
- **Hiệu Suất Cao:** Tối ưu hóa cho các hoạt động đọc và ghi dữ liệu với khả năng quản lý lượng dữ liệu lớn một cách hiệu quả.
- **Tính Sẵn Sàng Cao:** Hỗ trợ replication, cho phép tạo ra các bản sao của cơ sở dữ liệu để đảm bảo tính sẵn sàng và khôi phục sau sự cố.
- **Hỗ Trợ Đa Nền Tảng:** Có thể chạy trên nhiều hệ điều hành khác nhau, bao gồm Windows, macOS và Linux.

**- Các tính năng chính:**

- **Document-oriented Storage:** Dữ liệu được lưu trữ dưới dạng BSON (Binary JSON), cho phép lưu trữ các cấu trúc dữ liệu phức tạp và nhúng.
- **Ad Hoc Queries:** Hỗ trợ các truy vấn linh hoạt, bao gồm tìm kiếm theo trường, truy vấn phạm vi và regex.
- **Indexing:** Cung cấp các cơ chế chỉ mục đa dạng để tăng hiệu suất truy vấn.
- **Replication:** Hỗ trợ tạo các bản sao cơ sở dữ liệu để đảm bảo tính sẵn sàng và khả năng chịu lỗi.
- **Sharding:** Phân mảnh dữ liệu trên nhiều máy chủ để mở rộng khả năng lưu trữ và xử lý dữ liệu.
- **Aggregation Framework:** Cung cấp các công cụ mạnh mẽ để xử lý và tổng hợp dữ liệu, bao gồm các phép toán như \$sum, \$avg, \$max, \$min, và \$push.

#### 4.1.8 Hàng đợi tin nhắn Rabbit MQ

**- Khái niệm:**

**RabbitMQ** là một phần mềm môi giới tin nhắn mã nguồn mở, được thiết kế để cho phép các hệ thống và ứng dụng trao đổi thông tin với nhau một cách hiệu quả và đáng tin cậy. Nó dựa trên giao thức AMQP (Advanced Message Queuing Protocol) và hỗ trợ nhiều giao thức khác như STOMP, MQTT. RabbitMQ thường được sử dụng để quản lý hàng đợi tin nhắn, điều phối công việc, và tích hợp các dịch vụ trong kiến trúc microservices.

Theo [RabbitMQ Official Website](#):

"RabbitMQ is the most widely deployed open-source message broker."

**- Một số đặc điểm:**

- **Mã Nguồn Mở:** RabbitMQ là phần mềm mã nguồn mở, miễn phí và có một cộng đồng phát triển lớn mạnh.

- **Giao Thức Linh Hoạt:** Hỗ trợ nhiều giao thức nhắn tin như AMQP, MQTT, STOMP, giúp tích hợp với nhiều hệ thống khác nhau.
- **Khả Năng Mở Rộng:** Có thể mở rộng bằng cách thêm các node vào cluster, giúp tăng khả năng chịu tải và tính sẵn sàng.
- **Hỗ Trợ Nhiều Ngôn Ngữ Lập Trình:** RabbitMQ có thư viện hỗ trợ nhiều ngôn ngữ lập trình như Java, Python, Ruby, C#, và nhiều ngôn ngữ khác.
- **Quản Lý Dễ Dàng:** Cung cấp giao diện quản lý web, giúp theo dõi và quản lý các hàng đợi và tin nhắn một cách trực quan.

- **Các tính năng chính:**

- **Hàng Đợi Tin Nhắn (Queues):** Lưu trữ các tin nhắn chờ được xử lý, đảm bảo việc gửi và nhận tin nhắn một cách đáng tin cậy.
- **Trao Đổi Tin Nhắn (Exchanges):** Xác định cách thức định tuyến tin nhắn đến các hàng đợi dựa trên các quy tắc định tuyến.
- **Binding:** Xác định mối quan hệ giữa các exchange và queue, giúp điều phối cách thức tin nhắn được chuyển đến hàng đợi.
- **Xác Nhận Tin Nhắn (Acknowledgement):** Đảm bảo tin nhắn chỉ được xóa khỏi hàng đợi khi đã được xử lý thành công.
- **Cơ Chế Lưu Trữ:** RabbitMQ hỗ trợ lưu trữ tin nhắn trên đĩa để đảm bảo tính bền vững và khôi phục sau sự cố.
- **Cluster và High Availability:** RabbitMQ hỗ trợ cluster để phân phối tải và tăng tính sẵn sàng của hệ thống.

#### 4.1.9 Api web socket

- **Khái niệm:**

**WebSocket** là một giao thức truyền thông máy tính, cung cấp kênh giao tiếp hai chiều qua một kết nối TCP duy nhất. WebSocket được thiết kế để hoạt động trên các cổng web tiêu chuẩn (80 và 443), và được sử dụng rộng rãi trong các ứng dụng yêu cầu giao tiếp thời gian thực, như ứng dụng chat, game online, cập nhật dữ liệu theo thời gian thực, và nhiều ứng dụng khác.

Theo [Mozilla Developer Network](#):

"The WebSocket API is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser and a server."

- **Một số đặc điểm:**

- **Kết Nối Hai Chiều:** Cho phép máy chủ và máy khách gửi và nhận dữ liệu trong cùng một kết nối.

- **Thời Gian Thực:** Cung cấp khả năng giao tiếp với độ trễ thấp, lý tưởng cho các ứng dụng yêu cầu cập nhật thời gian thực.
- **Tiết Kiệm Băng Thông:** Giảm tải so với HTTP vì chỉ cần thiết lập một lần kết nối duy nhất và không cần gửi đi các header HTTP mỗi lần trao đổi dữ liệu.
- **Khả Năng Mở Rộng:** Có thể mở rộng để xử lý hàng ngàn kết nối đồng thời nhờ vào tính chất bất đồng bộ của nó.

#### - Cấu trúc:

WebSocket sử dụng một kết nối TCP duy nhất và hoạt động qua hai giai đoạn chính:

**Handshake:** Bắt đầu bằng một yêu cầu HTTP từ máy khách tới máy chủ, sau đó nâng cấp kết nối lên WebSocket.

**Data Transfer:** Sau khi kết nối được thiết lập, dữ liệu có thể được truyền tải giữa máy khách và máy chủ dưới dạng các khung tin (frames).

### 4.1.10 ReactJs

#### - Khái niệm:

**ReactJS** là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, được sử dụng để xây dựng giao diện người dùng (UI) tương tác và linh hoạt. ReactJS tập trung vào việc phát triển các thành phần UI có thể tái sử dụng, giúp việc phát triển và quản lý ứng dụng trở nên dễ dàng và hiệu quả hơn.

Theo [React Official Documentation](#):

"React is a JavaScript library for building user interfaces."

#### - Một số đặc điểm:

- **Component-Based:** React cho phép xây dựng các ứng dụng bằng cách sử dụng các thành phần nhỏ và có thể tái sử dụng. Mỗi thành phần có thể quản lý trạng thái và giao diện của riêng nó.
- **Virtual DOM:** React sử dụng một cơ chế gọi là Virtual DOM để cập nhật giao diện một cách hiệu quả. Khi trạng thái của ứng dụng thay đổi, React sẽ cập nhật Virtual DOM trước, sau đó so sánh với DOM thật và chỉ cập nhật những phần cần thiết.
- **One-Way Data Binding:** React sử dụng cơ chế dữ liệu truyền một chiều, giúp dễ dàng kiểm soát luồng dữ liệu trong ứng dụng và dễ dàng gỡ lỗi.
- **JSX:** JSX là một cú pháp mở rộng của JavaScript, cho phép viết HTML trong JavaScript. Điều này giúp code trở nên dễ đọc và dễ hiểu hơn.

#### 4.1.11 Json web token (JWT)

##### - Khái niệm:

**JSON Web Token (JWT)** là một tiêu chuẩn mở (RFC 7519) định nghĩa một cách nhỏ gọn và tự chứa để truyền tải thông tin một cách an toàn giữa các bên dưới dạng JSON. Thông tin trong JWT có thể được xác minh và tin cậy bởi vì nó được ký số.

Theo JWT.io:

"JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties."

##### - Cấu trúc:

**Header:** Header thường bao gồm hai phần: loại token (JWT) và thuật toán ký (như HMAC SHA256 hoặc RSA).

**Payload:** Payload chứa các tuyên bố (claims). Các tuyên bố này là thông tin về một thực thể (thường là người dùng) và các dữ liệu khác. Có ba loại tuyên bố:

- **Registered claims:** Được định nghĩa trước bởi đặc tả JWT (ví dụ: iss, exp, sub, aud).
- **Public claims:** Có thể được định nghĩa tùy ý để chia sẻ thông tin với các bên khác.
- **Private claims:** Được tạo bởi hai bên để chia sẻ thông tin sử dụng nội bộ.

**Signature:** Để tạo chữ ký, bạn cần lấy header và payload, mã hóa chúng bằng base64, sau đó nối chúng lại với nhau bằng dấu chấm và ký chúng bằng thuật toán được chỉ định trong header. Ví dụ với HMAC SHA256:

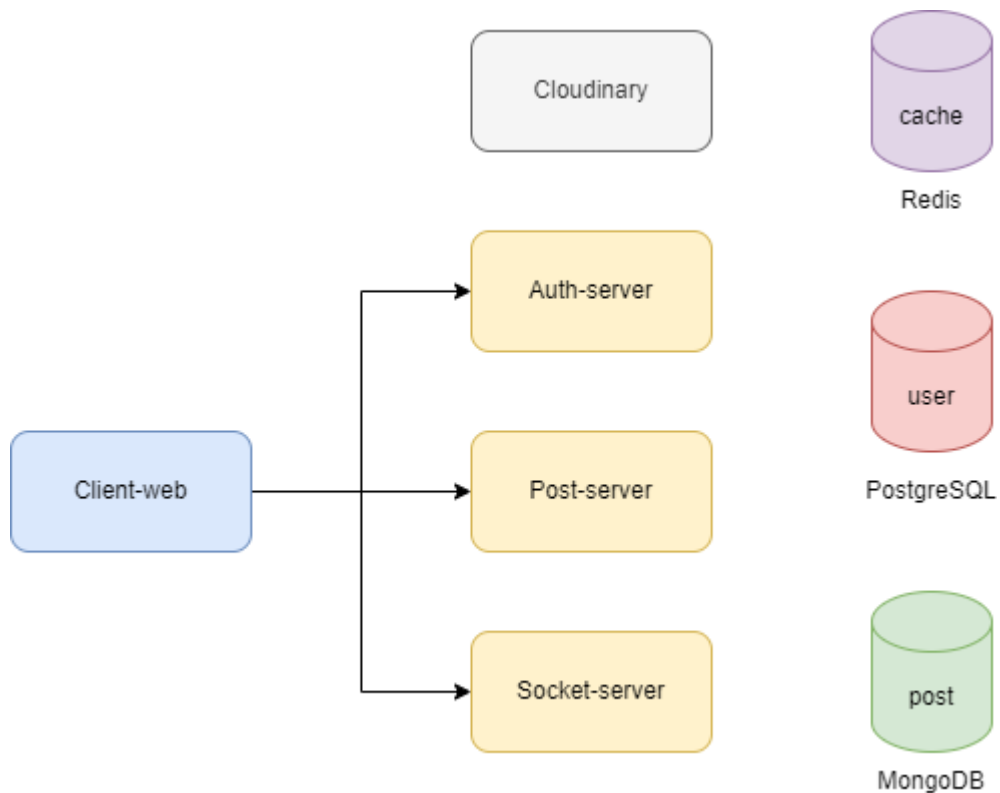
##### -Sử dụng:

JWT thường được sử dụng trong các hệ thống xác thực và ủy quyền:

- **Xác Thực (Authentication):** Khi người dùng đăng nhập, máy chủ sẽ tạo ra một JWT và trả về cho người dùng. Sau đó, người dùng sẽ sử dụng JWT này để truy cập các tài nguyên bảo mật.
- **Ủy Quyền (Authorization):** Sau khi người dùng đăng nhập, mỗi lần người dùng gửi yêu cầu, JWT sẽ được gửi kèm theo để xác minh và cấp quyền truy cập tài nguyên.

## 4.2 Kiến trúc

### 4.2.1 Kiến trúc tổng quan



## 22. Kiến trúc hệ thống

- Hệ thống được mở rộng thành microservice, với các service auth, post, socket, web client.

### 4.2.2 Client-web

- Sử dụng ReactJS xây dựng giao diện.

### 4.2.3 Auth-server:

- Khi đăng nhập, hệ thống sẽ lấy token (JWT) từ auth-server để có thể truy cập post-server cho các lượt truy cập tiếp theo.
- Các api của hệ thống:  
**post /v1/api/auth/register:** Tạo tài khoản cho người dùng  
**post /v1/api/auth/login:** Đăng nhập cho người dùng, nhận một mã token.  
**post /v1/api/auth/logout:** Xóa mã token sử dụng cho các lượt gửi yêu cầu tới post-server.  
**post /v1/api/auth/refresh\_token:** Sau khi mã token hết hạn thì tạo mã token mới

### 4.2.4 Post-server:

Các api của hệ thống:

- **Api post:**  
**get /v1/api/post:** Lấy bài viết từ cơ sở dữ liệu.  
**get /v1/api/post/:post\_id:** Lấy bài viết cụ thể

- post /v1/api/post :** Tạo một bài viết
- put /v1/api/post :** Sửa một bài viết
- delete /v1/api/post :** Xóa một bài viết
- get /v1/api/post/user/:id:** Lấy bài viết bằng id người dùng
  
- **Api user:**
  - get /v1/api/user:** Lấy thông tin người dùng
  - get /v1/api/user/:id:** Lấy thông tin người dùng bằng id người dùng
  
- **Api comment:**
  - post /v1/api/comment:** Tạo bình luận với một bài viết
  - put /v1/api/comment/:comment\_id:** Sửa bình luận
  - delete /v1/api/comment/:post\_id:** Xóa bình luận
  - get /v1/api/comment/:post\_id:** Lấy bình luận
  
- **Api follow**
  - put /v1/api/follow/:followed\_user\_id:** Tạo danh sách theo dõi của người dùng
  - get /v1/api/follow/:followed\_user\_id:** Lấy danh sách theo dõi người dùng
  - get /v1/api/follow/:follower\_user\_id:** Lấy danh sách người dùng theo dõi
  
- **Api friend**
  - get /v1/api/friend/:user\_id:** Lấy bạn bè của người dùng bằng mã người dùng
  
- **Api like**
  - patch /v1/api/like/:post\_id:** Cập nhật like bài viết
  - get /v1/api/like/:post\_id:** Lấy danh sách người dùng like bài viết
  
- **Api notification**
  - get /v1/api/notification:** Lấy danh sách thông báo
  
- **Api share**
  - get /v1/api/share/:post\_id:** Lấy danh sách người chia sẻ bài viết
  - post /v1/api/share:** Chia sẻ bài viết
  
- **Api upload**
  - post v1/api/upload:** tải ảnh lên cloudinary
  
  - delete v1/api/upload:** xóa ảnh trên cloudinary

#### **4.2.5 Socket-server**

- Nhận thông tin từ post-server qua rabbit mq
- Kết nối web socket với client web để gửi thông báo trực tiếp về cho người dùng.

#### **4.2.6 Cơ sở dữ liệu MongoDB:**

- Lưu trữ bài viết, bình luận, lượt thích, danh sách bạn bè,.....
- ⇒ Việc sử dụng mongoDB giúp các bài viết đa dạng trường dữ liệu, dễ dàng kiểm soát.

#### **4.2.7 Cơ sở dữ liệu PostgreSQL:**

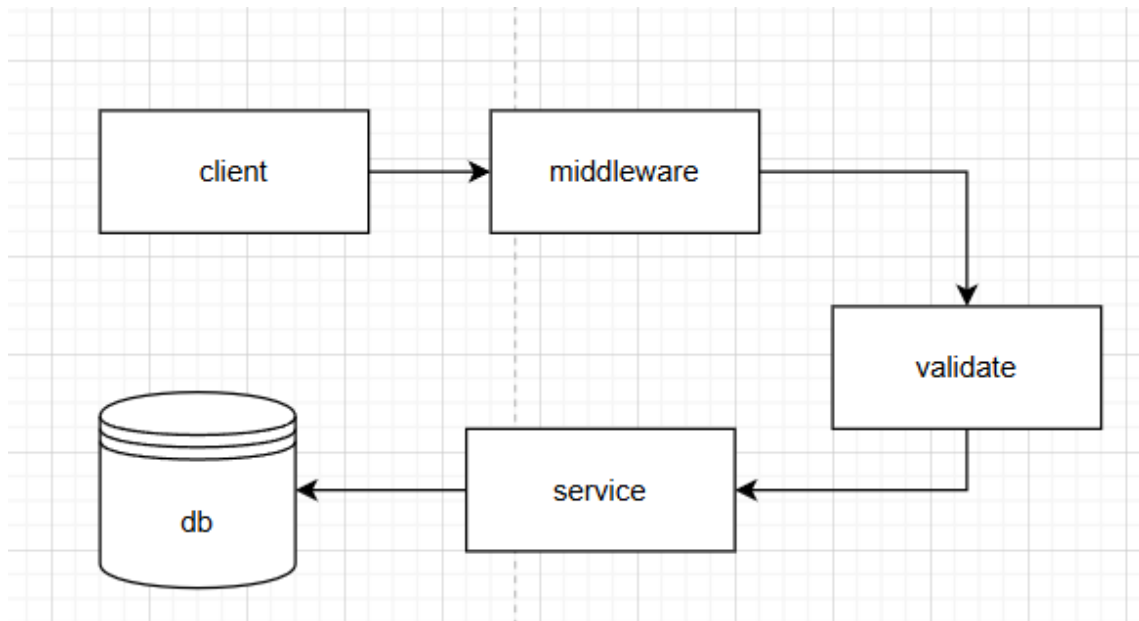
- Lưu trữ thông tin đăng nhập, bảo mật của người dùng
  - Lưu trữ token, phiên đăng nhập
  - Lưu trữ lịch sử token.
- ⇒ Cơ sở dữ liệu có tính bảo mật cao, thông tin người dùng cấu trúc không thay đổi cấu trúc theo thời gian.

#### **4.2.8 Redis:**

- Bộ nhớ đệm
- Kiểm tra số lượng kết nối
- Kiểm tra danh sách người dùng có hành vi lạ

## CHƯƠNG 5. LẬP TRÌNH API

### 5.1 Luồng hoạt động của một api



23. Luồng hoạt động của một api

- Middleware: kiểm tra yêu cầu từ website có hợp lệ không, đồng thời phân quyền, kiểm tra hành vi lạ.
- Validate: kiểm tra các trường dữ liệu trong yêu cầu có hợp lệ hay không, việc kiểm tra này cũng được thực hiện ở website.
- Service: Các chức năng nghiệp vụ thực hiện chi tiết, dữ liệu, lỗi được quyết định trả về ở đây.
- Database: Các thao tác đọc/ ghi với cơ sở dữ liệu, procedure/function/transaction.
- Cấu hình lỗi trả về:

```
{
  "status": "error",
  "code": 403,
  "stack": "Error: Error: Authentication error\n    at AuthService.login (D:\\Code\\Social Network\\auth-server\\src\\services\\auth.service.js:117:13)\n    at async login (D:\\Code\\Social Network\\auth-server\\src\\controllers\\auth.controller.js:16:23)",
  "message": "Error: Authentication error"
}
```

- Cấu hình dữ liệu trả về:

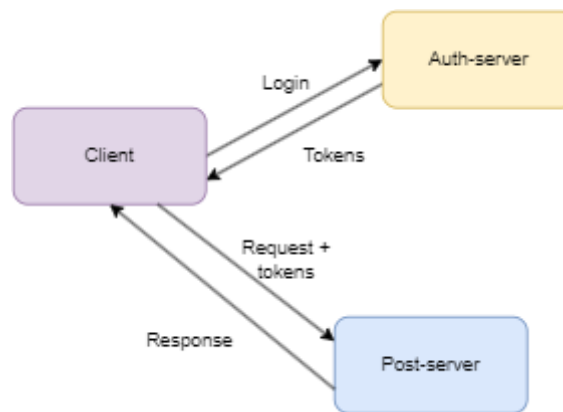
```
{
  "message": "Login successfully!",
  "status": 200,
  "metadata": { ... }
}
```

### 5.2 Middleware

- **Khái niệm:** Một hàm được thực hiện trả lại kết quả đính kèm trước khi yêu cầu được thực thi trên hệ thống
- Có rất nhiều middleware được sử dụng trong hệ thống, dưới đây là một middleware xác thực người dùng có đăng nhập trong hệ thống.

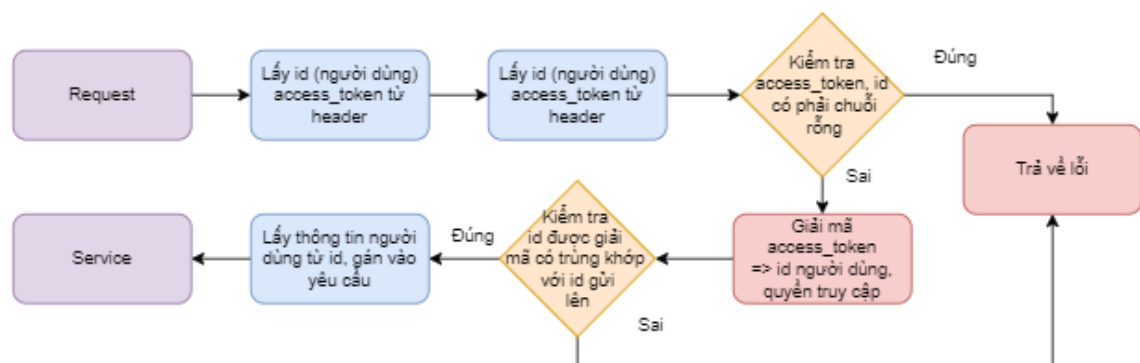


- **Middleware authentication:**



Hình 22. Luồng hoạt động phân quyền của hệ thống

- + Để dùng được hệ thống (website), người dùng phải đăng nhập hệ thống
- + Khi đăng nhập hệ thống, hệ thống gửi về một cặp token (JWT) gồm access\_token (mã truy cập), refresh\_token (mã cấp lại mã truy cập).
- + Các thao tác tiếp theo với website sẽ dùng mã truy cập để lấy dữ liệu, xử lý dữ liệu trên một server khác.



Hình 23. Logic middleware authentication

- + Yêu cầu gửi lên server, trước khi xử lý các thao tác nghiệp vụ, xác thực xem access\_token có hợp lệ, có trùng khớp.

### 5.3 Api (method: Post)

#### 5.3.1 Tổng Quan về POST Method

POST là một trong bốn phương thức HTTP cơ bản (cùng với GET, PUT và DELETE) được sử dụng trong RESTful API. Phương thức này thường được sử dụng để gửi dữ liệu tới máy chủ để tạo mới một tài nguyên. Khi một yêu cầu POST được gửi đi, dữ liệu được bao gồm trong phần thân (body) của yêu cầu.

### 5.3.2 Đặc Điểm của POST

- **Tạo Tài Nguyên Mới:** POST thường được sử dụng để tạo mới một tài nguyên trên máy chủ.
- **Không Idempotent:** POST không idempotent, nghĩa là nếu bạn gửi nhiều yêu cầu POST giống nhau, bạn sẽ tạo ra nhiều tài nguyên khác nhau.
- **Thường Trả Về 201 (Created):** Nếu tài nguyên được tạo thành công, máy chủ sẽ trả về mã trạng thái HTTP 201.
- **Bao Gồm Dữ Liệu Trong Body:** Dữ liệu được gửi qua POST thường nằm trong phần thân của yêu cầu và có thể ở định dạng JSON, XML hoặc các định dạng khác.

### 5.3.3 Cấu Trúc của Yêu Cầu POST

Một yêu cầu POST điển hình bao gồm:

- **URL:** Đường dẫn đến tài nguyên API.
- **Headers:** Bao gồm thông tin như Content-Type (thường là application/json nếu dữ liệu là JSON).
- **Body:** Chứa dữ liệu cần gửi.

Ví dụ:

```
bash Copy code

POST /api/posts HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "content": "This is the content of the post.",
  "author_id": 1,
  "video_images": ["image1.jpg", "image2.jpg"],
  "location": "Hanoi"
}
```

### 5.3.4 Ví dụ api (method: Post) create post

Giả sử chúng ta có một RESTful API để quản lý các bài viết trên blog. Để tạo một bài viết mới, chúng ta sẽ sử dụng phương thức POST đến endpoint `v1/api/post`.

Yêu Cầu Tạo Bài Viết

**Request:**

```
bash Copy code

POST /api/posts HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "content": "This is the content of the post.",
  "author_id": 1,
  "video_images": ["image1.jpg", "image2.jpg"],
  "location": "Hanoi"
}
```

### Headers:

- Content-Type: application/json chỉ ra rằng dữ liệu được gửi ở định dạng JSON.

### Body:

- Chứa dữ liệu bài viết với các trường như content, author\_id, video\_images, và location.

### Phản Hồi từ Máy Chủ

Nếu yêu cầu POST thành công, máy chủ sẽ trả về một phản hồi với mã trạng thái HTTP 201 (Created) và dữ liệu chi tiết về bài viết vừa được tạo.

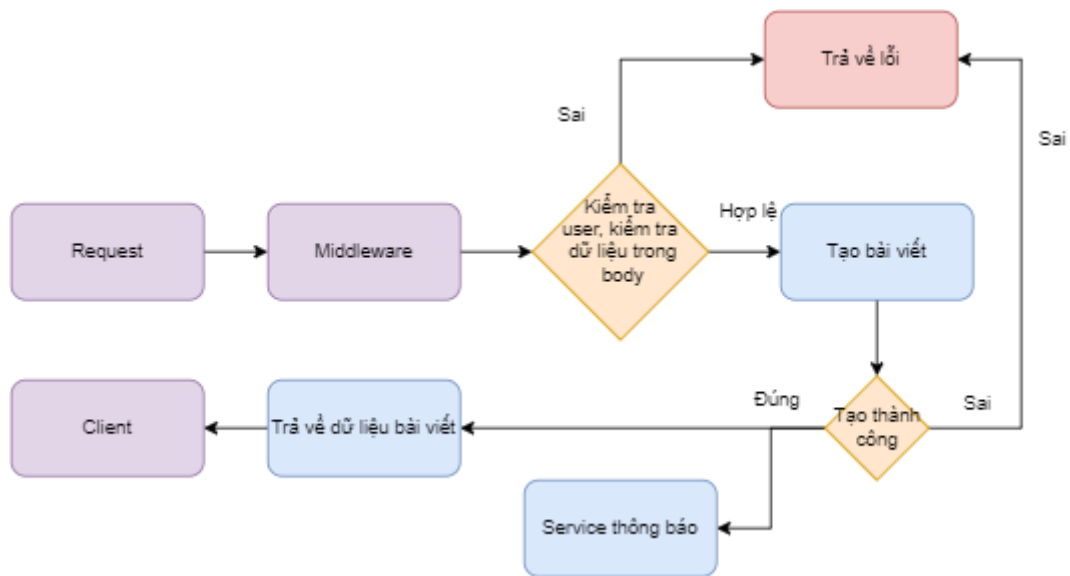
### Response:

```
json Copy code

HTTP/1.1 201 Created
Content-Type: application/json

{
  "id": "60b8d295f4d5c82650d2f1c1",
  "content": "Understanding RESTful APIs",
  "author_id": 1,
  "video_images": ["image1.jpg", "image2.jpg"],
  "likes": 0,
  "shares": 0,
  "comments": 0,
  "isPublished": false,
  "publishedAt": null,
  "location": "Hanoi",
  "share_from_user_id": null,
  "id_post_from_share": null,
  "createdAt": "2024-06-11T12:34:56Z",
  "updatedAt": "2024-06-11T12:34:56Z"
}
```

## Luồng hoạt động trong của create post:



Hình 24. Logic service create post

- Bài viết được tạo từ thông tin người từ middleware authentication và dữ liệu body gửi lên từ yêu cầu.

## 5.4 Api (method: GET)

### 5.4.1 Tổng Quan về Phương Thức GET

Phương thức GET trong RESTful API được sử dụng để truy xuất dữ liệu từ máy chủ. Khi gửi yêu cầu GET đến một endpoint, máy chủ sẽ trả về dữ liệu mà không thay đổi trạng thái của tài nguyên trên máy chủ.

### 5.4.2 Cấu Trúc của Yêu Cầu GET

Một yêu cầu GET điển hình bao gồm:

- **URL:** Đường dẫn đến tài nguyên API.
- **Headers:** Thông thường chỉ cần Authorization nếu yêu cầu xác thực, không cần body.

Ví dụ:

```
vbnet Copy code

GET /api/posts HTTP/1.1
Host: example.com
Authorization: Bearer YOUR_ACCESS_TOKEN
```

### 5.4.3 Ví Dụ về API Get List Posts

Giả sử chúng ta có một RESTful API để quản lý các bài viết trên blog. Để lấy danh sách các bài viết, chúng ta sẽ sử dụng phương thức GET đến endpoint `/api/posts`.

Yêu Cầu Lấy Danh Sách Bài Viết

#### Request:

```
makefile                                                                    Copy code

GET /api/posts HTTP/1.1
Host: blogapi.com
Content-Type: application/json
Authorization: Bearer YOUR_ACCESS_TOKEN
```

#### Headers:

- Authorization: Bearer YOUR\_ACCESS\_TOKEN nếu API yêu cầu xác thực.

Phản Hồi từ Máy Chủ

Nếu yêu cầu GET thành công, máy chủ sẽ trả về một phản hồi với mã trạng thái HTTP 200 (OK) và danh sách các bài viết.

#### Response:

HTTP/1.1 200 OK

Content-Type: application/json

```
[
  {
    "id": "60b8d295f4d5c82650d2f1c1",
    "content": "Understanding RESTful APIs",
    "author_id": 1,
    "video_images": ["image1.jpg", "image2.jpg"],
    "likes": 0,
    "shares": 0,
    "comments": 0,
    "isPublished": false,
    "publishedAt": null,
    "location": "Hanoi",
    "share_from_user_id": null,
    "id_post_from_share": null,
    "createdAt": "2024-06-11T12:34:56Z",
    "updatedAt": "2024-06-11T12:34:56Z"
  },
  {
    "id": "60b8d295f4d5c82650d2f1c2",
    "content": "Another Post",
    "author_id": 2,
    "video_images": [],
    "likes": 10,
    "shares": 2,
    "comments": 5,
    "isPublished": true,
    "publishedAt": "2024-06-10T12:00:00Z",
    "location": "HCM",
    "share_from_user_id": null,
    "id_post_from_share": null,
    "createdAt": "2024-06-10T12:34:56Z",
    "updatedAt": "2024-06-10T12:34:56Z"
  }
]
```

## CHƯƠNG 6. GIAO DIỆN, DEMO CODE

Link source code:

[Auth-server](#)

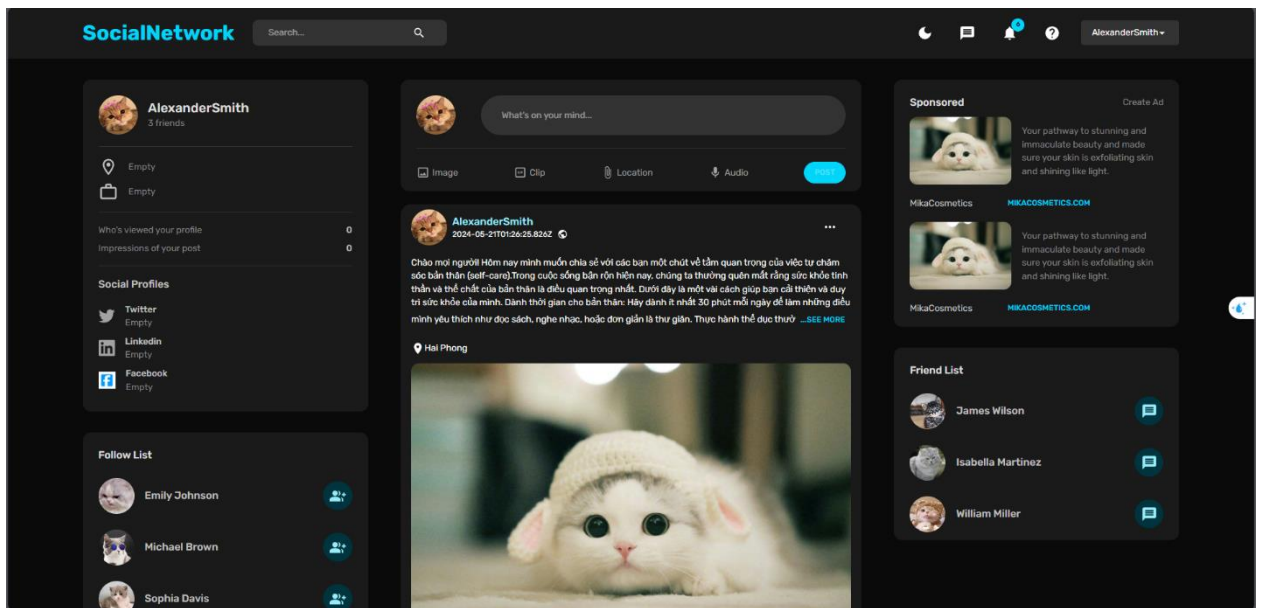
[Post-server](#)

[Socket-server](#)

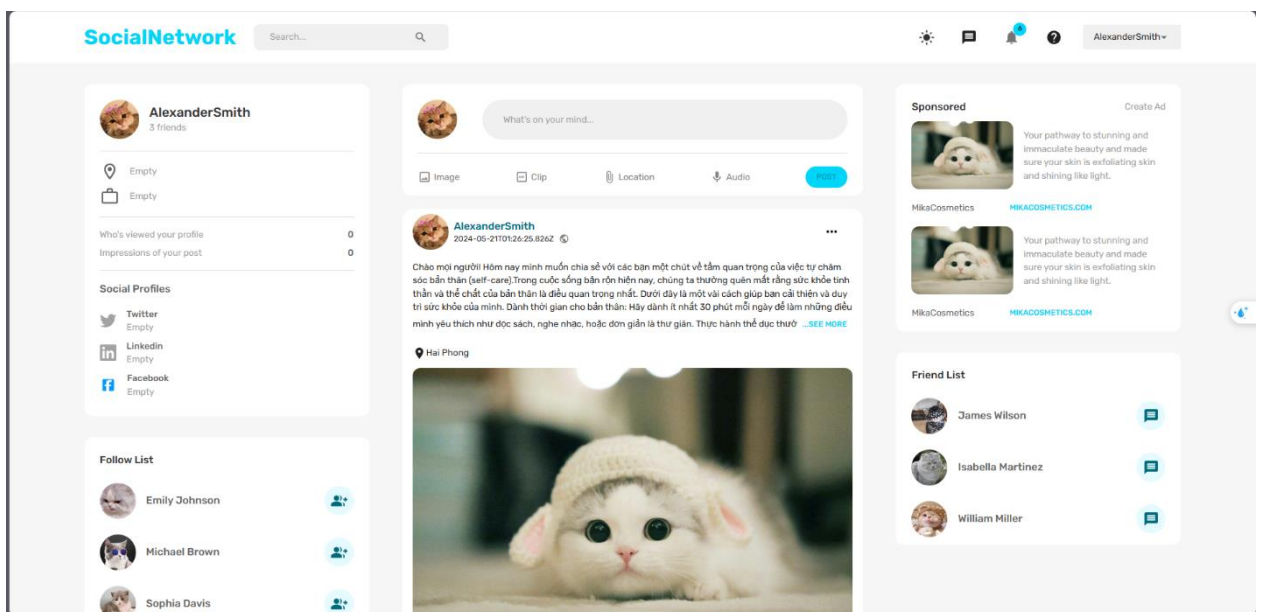
[Client\\_web](#)

### 6.1 Giao diện

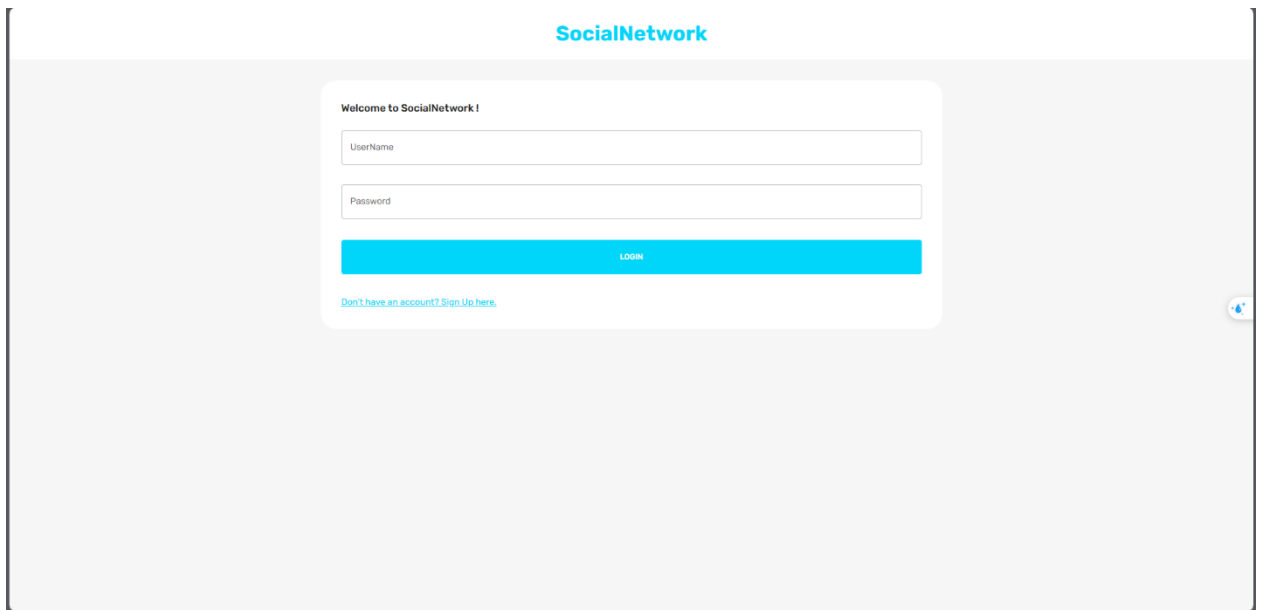
- Giao diện chính của trang web, có thể đổi màu sáng tối



Hình 25. Trang chủ (Tối)

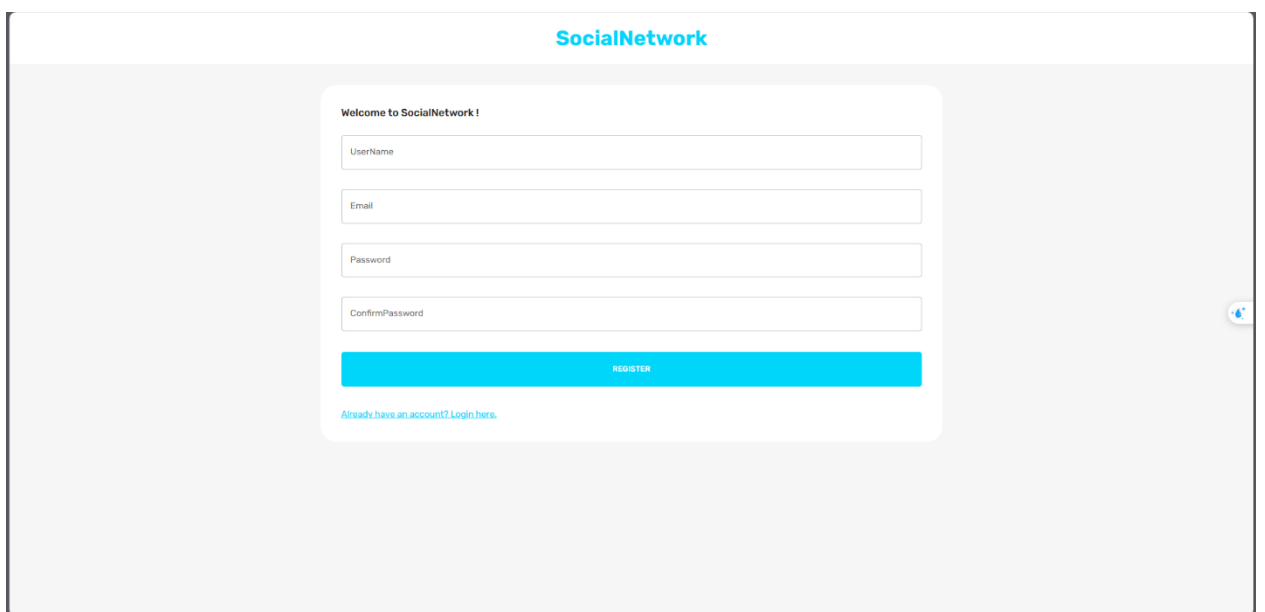


Hình 26. Trang chủ (sáng)



The image shows a login page for a website titled "SocialNetwork". The page has a light gray background. At the top center, the title "SocialNetwork" is displayed in a blue font. Below the title, there is a white rectangular box containing the login form. The form is titled "Welcome to SocialNetwork !" and includes two input fields: "UserName" and "Password". Below these fields is a blue button labeled "LOGIN". At the bottom of the form, there is a link that says "Don't have an account? Sign Up here.".

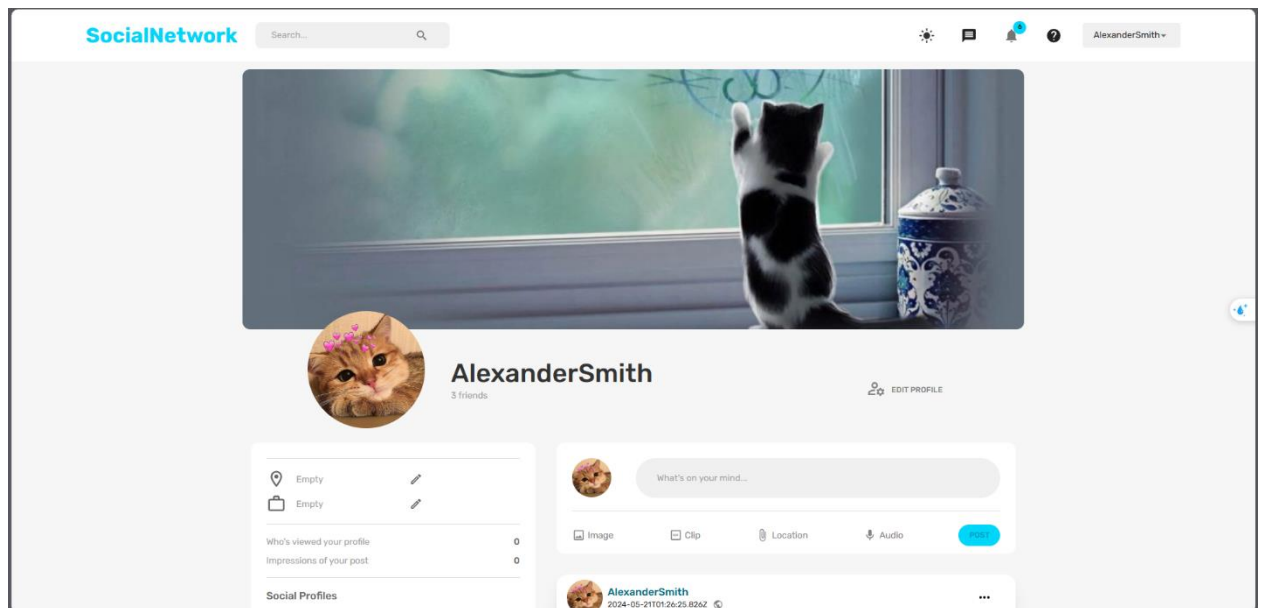
Hình 27. Trang đăng nhập



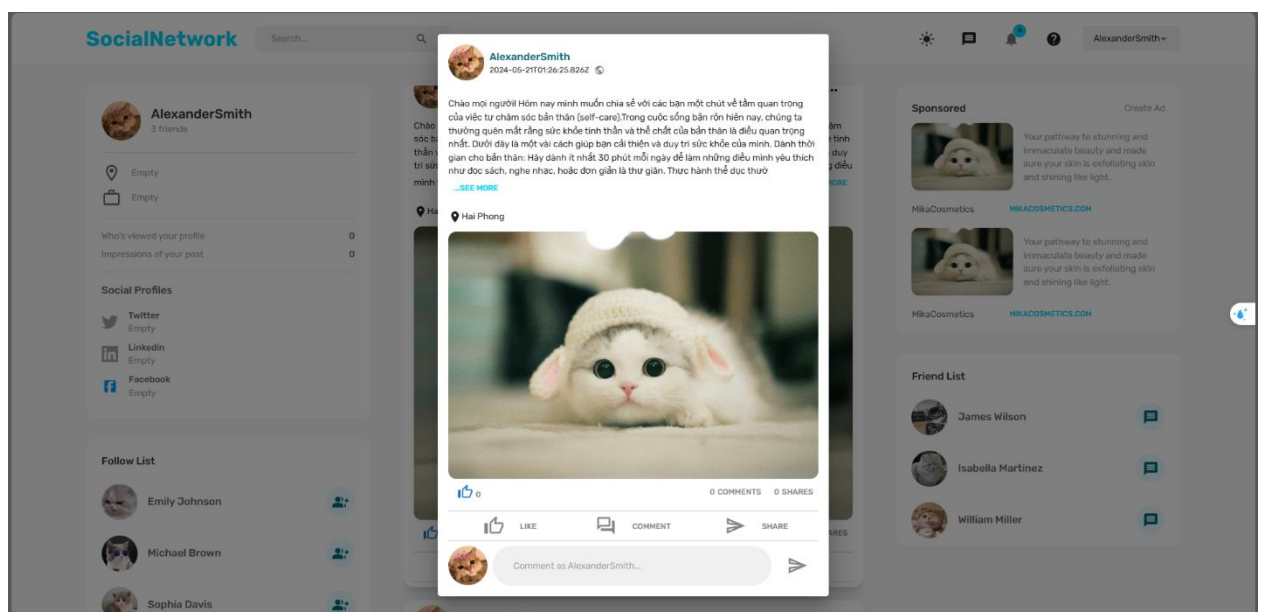
The image shows a registration page for a website titled "SocialNetwork". The page has a light gray background. At the top center, the title "SocialNetwork" is displayed in a blue font. Below the title, there is a white rectangular box containing the registration form. The form is titled "Welcome to SocialNetwork !" and includes four input fields: "UserName", "Email", "Password", and "ConfirmPassword". Below these fields is a blue button labeled "REGISTER". At the bottom of the form, there is a link that says "Already have an account? Login here.".

Hình 28. Trang đăng ký

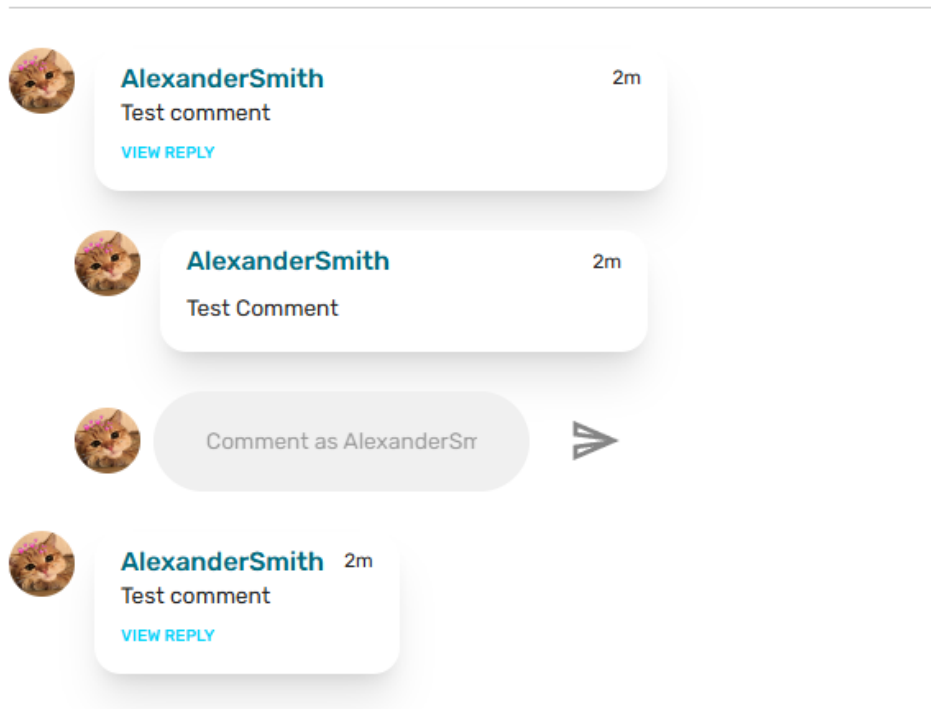




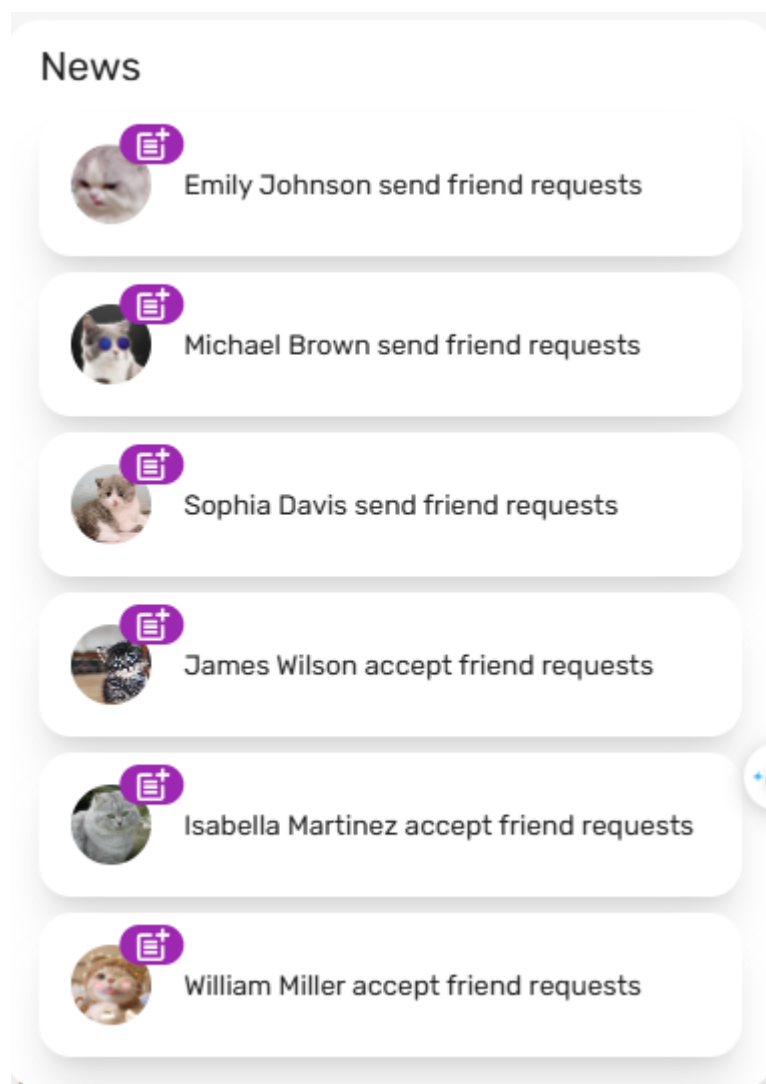
Hình 29. Trang cá nhân



Hình 30. Trang bài viết



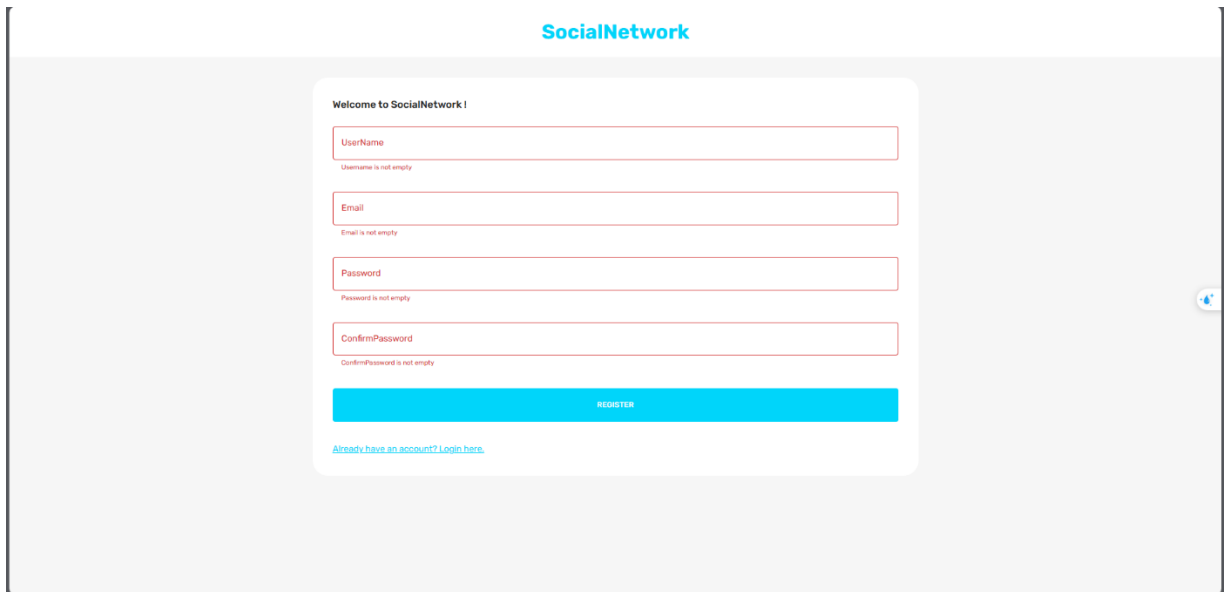
Hình 31. Giao diện phần bình luận



Hình 32. Giao diện phần thông báo

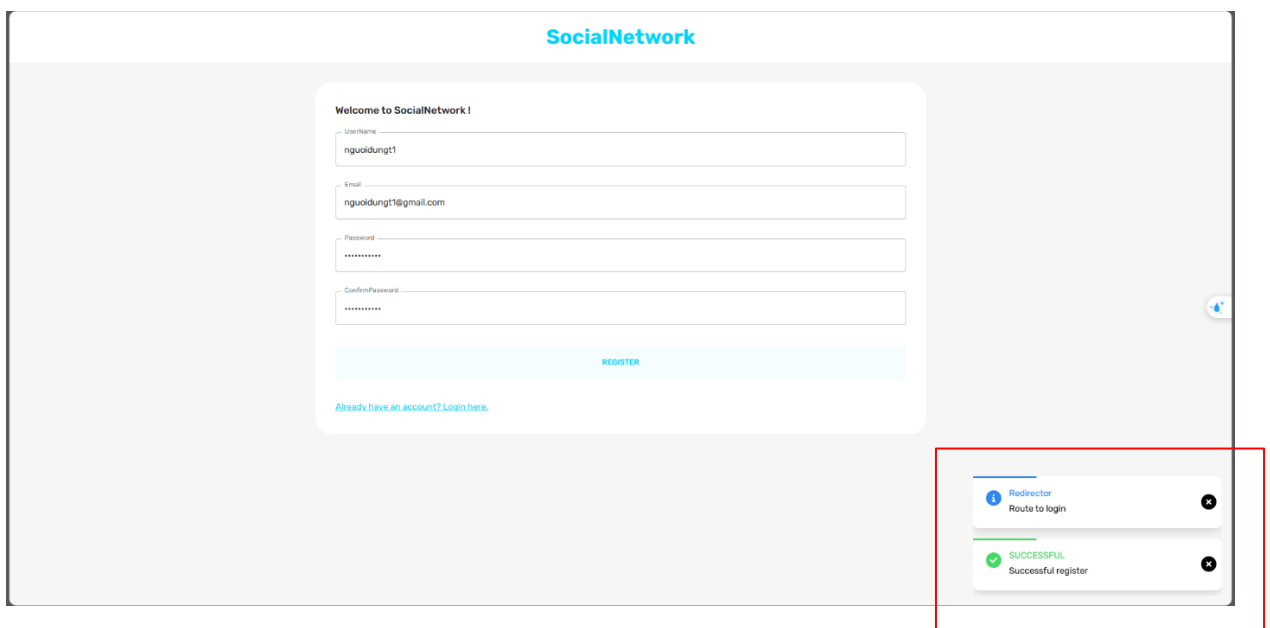
## 6.2 Kịch bản

- Người dùng 1 muốn tham gia trang mạng xã hội trên, bắt đầu với thao tác đăng ký, trang đăng ký, hay đăng nhập luôn kiểm tra các trường dữ liệu không hợp lệ, phù hợp với luật của trang. Việc kiểm tra này diễn ra ở web và server.



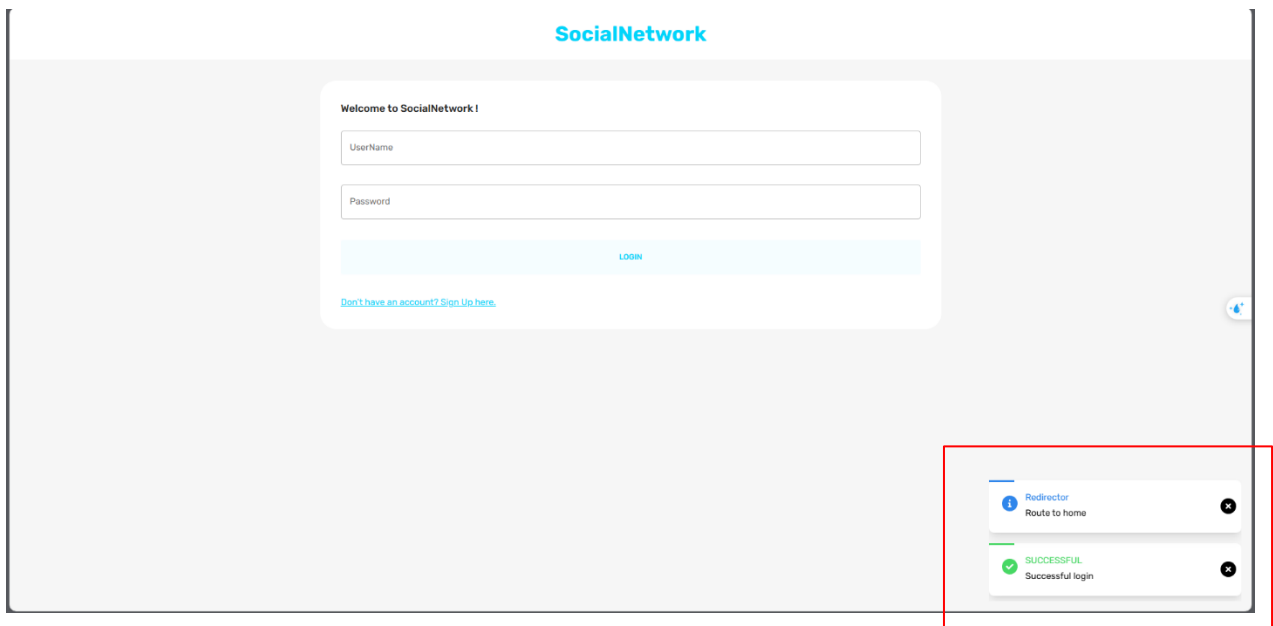
The screenshot shows a web page titled "SocialNetwork" with a registration form. The form is titled "Welcome to SocialNetwork !" and contains four input fields: "UserName", "Email", "Password", and "ConfirmPassword". Each field has a red border and a small error message below it: "Username is not empty", "Email is not empty", "Password is not empty", and "ConfirmPassword is not empty". A blue "REGISTER" button is at the bottom of the form. Below the button is a link: "Already have an account? Login here."

- Sau khi đăng ký thành công, góc dưới bên phải màn hình hiện một thông báo, đó có thể là lỗi hoặc thông báo thành công.

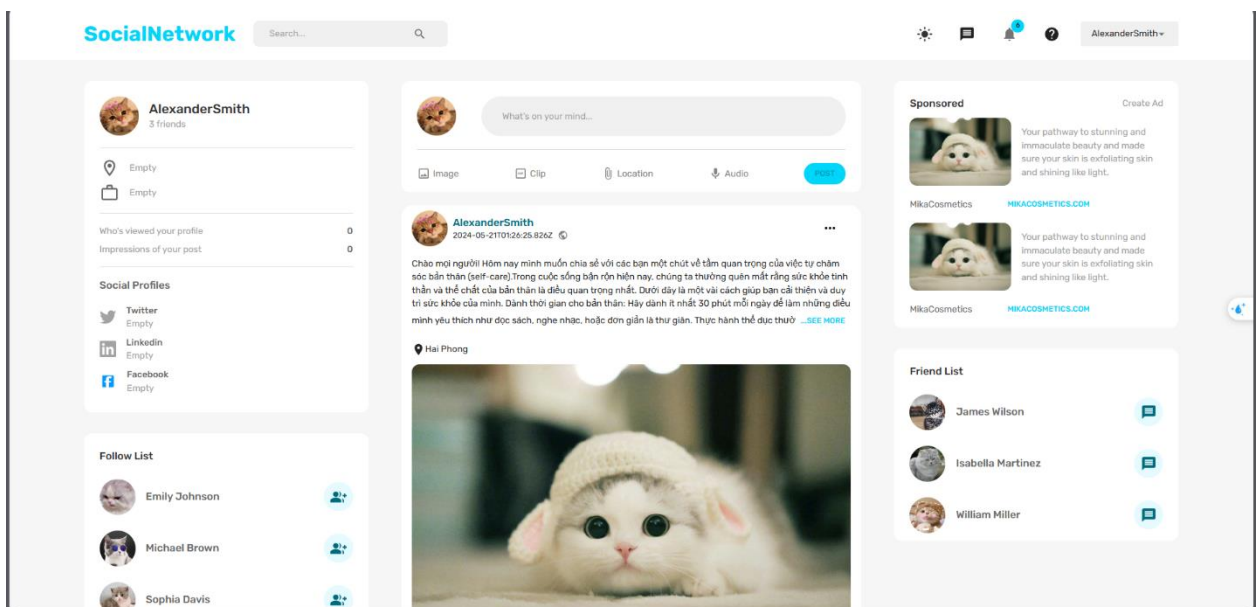


The screenshot shows the same registration form as before, but now it is filled with data: "UserName" is "nguoidung1", "Email" is "nguoidung1@gmail.com", "Password" is "\*\*\*\*\*", and "ConfirmPassword" is "\*\*\*\*\*". The "REGISTER" button is now light blue. Below the button is the same link: "Already have an account? Login here." In the bottom right corner, there is a notification box with two messages: "Redirector Route to login" (with a blue 'i' icon) and "SUCCESSFUL Successful register" (with a green checkmark icon). The notification box is highlighted with a red border.

- Sau khi đăng ký thành công, màn hình chuyển sang trang đăng nhập, sau khi đăng nhập thành công, hiện thông báo tương tự chuyển sang trang chủ.



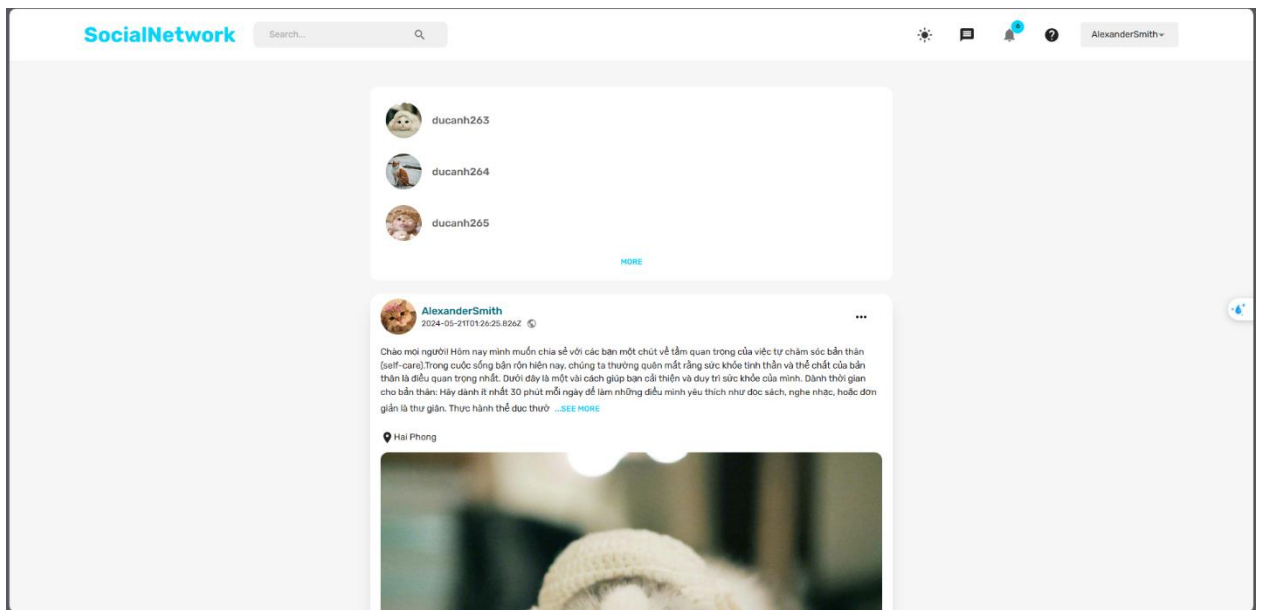
- Trang chủ sẽ hiện ra với giao diện



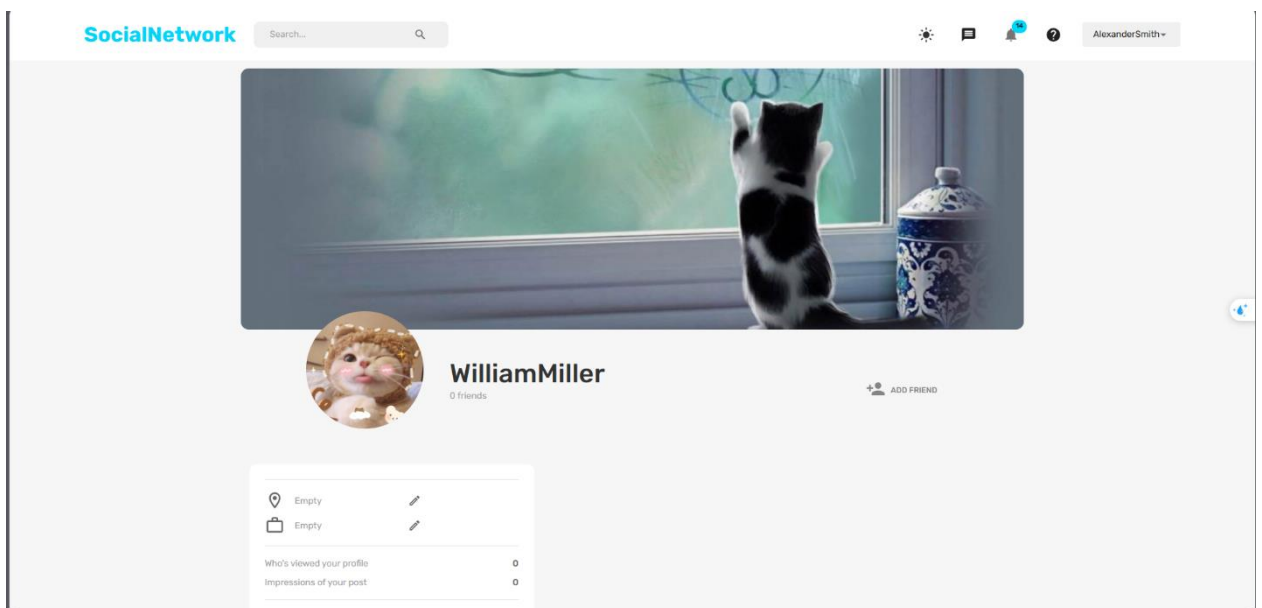
- Thực hiện thao tác tìm kiếm trên thanh header của trang:



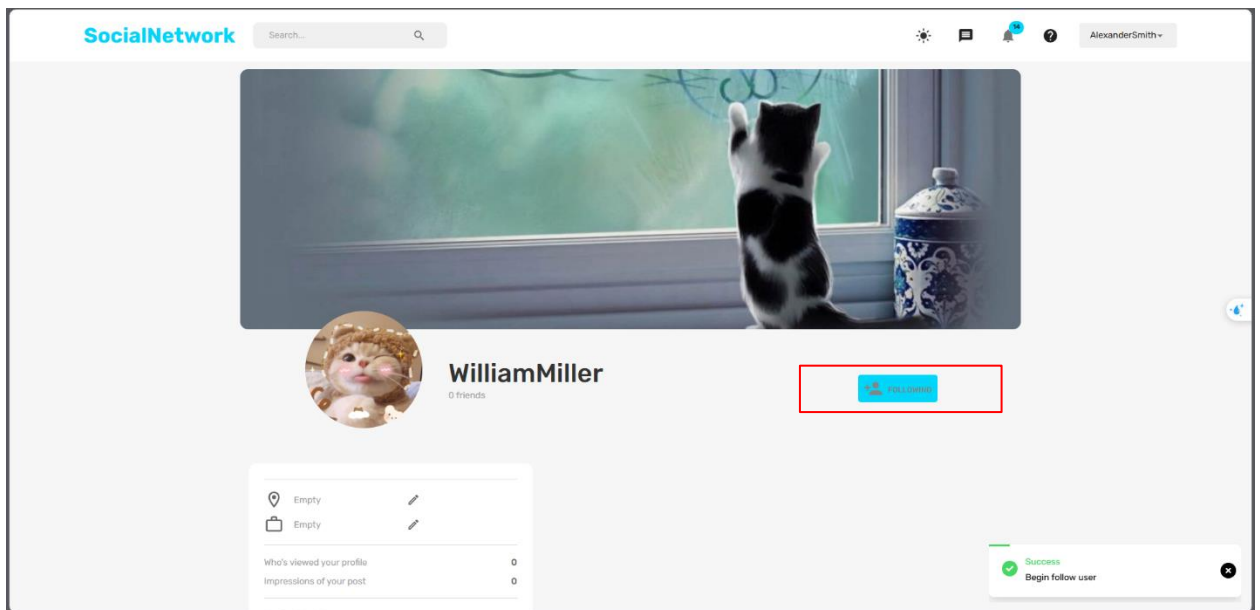
- Giao diện tìm kiếm hiện ra:



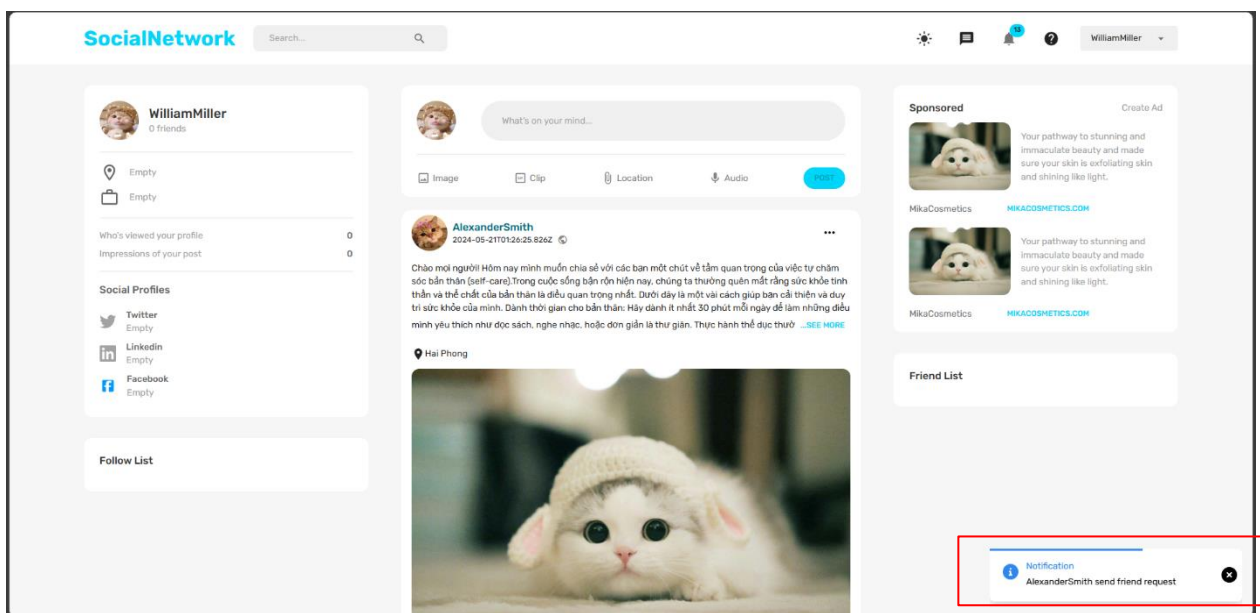
- Chọn một người dùng cụ thể và vào trang chủ của người dùng đó, ở đây là William Miller



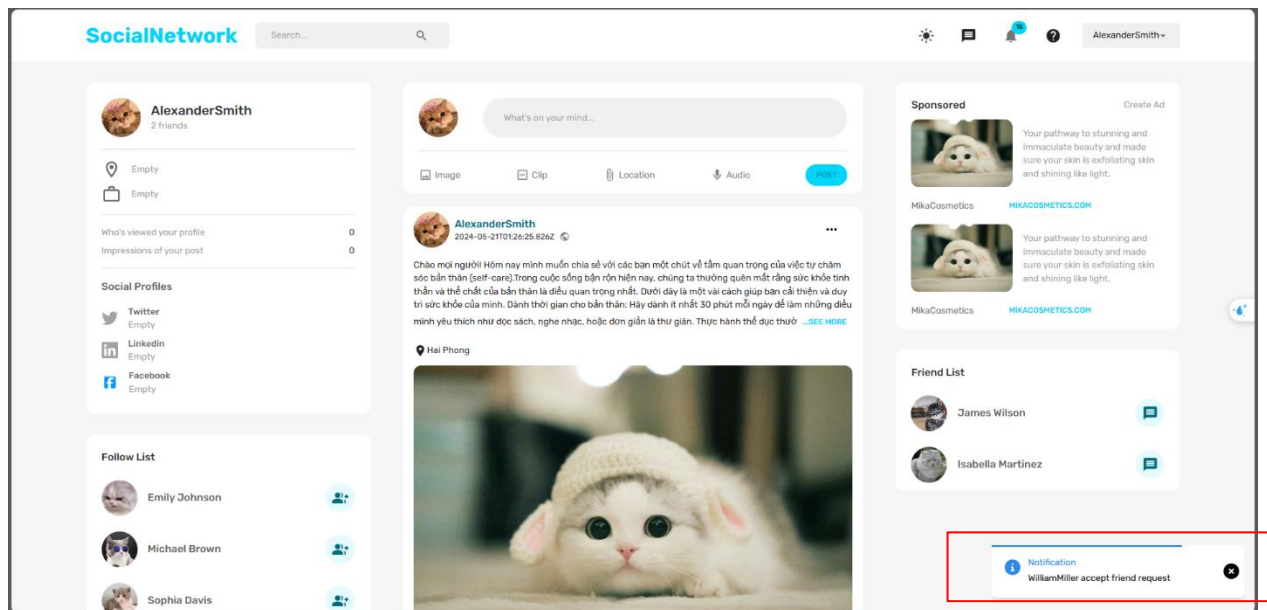
- Chọn nút “Add Friend” để tiến hành gửi lời mời kết bạn với Wiliam Miller.



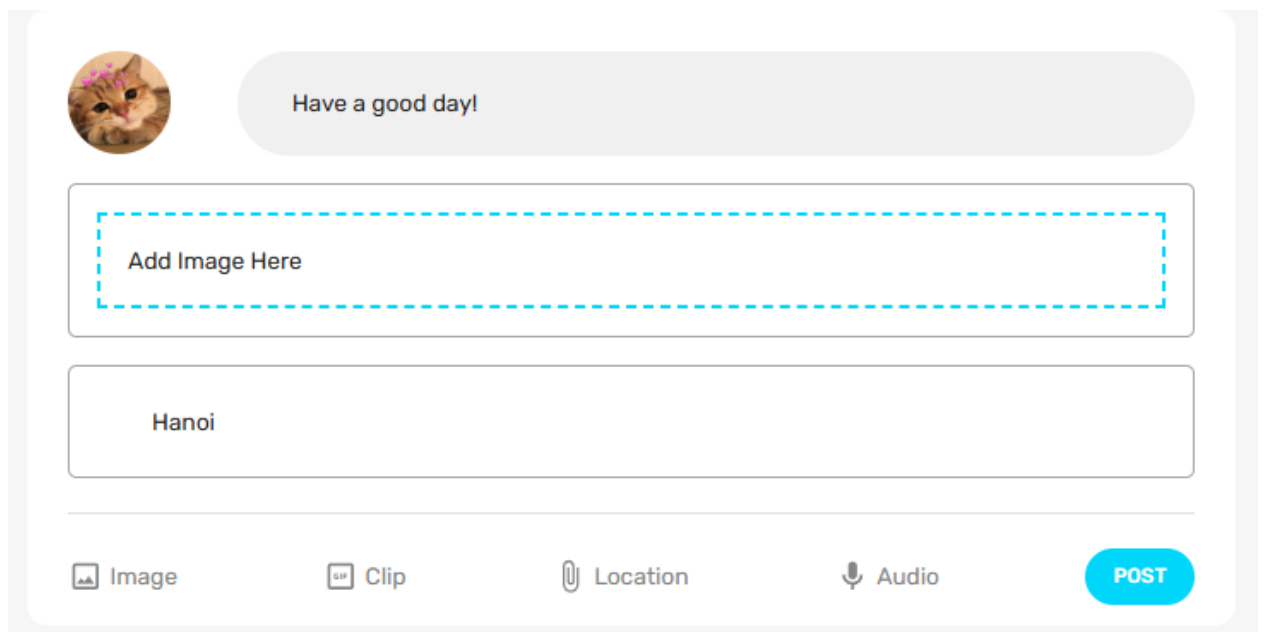
- Sau khi lời mời kết bạn được gửi, người dùng được gửi lời mời kết bạn sẽ nhận được thông báo trong thời gian thực qua kết nối web socket, vậy là Alexander Smith đã gửi lời mời kết bạn cho WilliamMiller



- Tương tự, WilliamMiller khi chấp nhận lời mời kết bạn, thông báo sẽ được gửi về AlexanderSmith

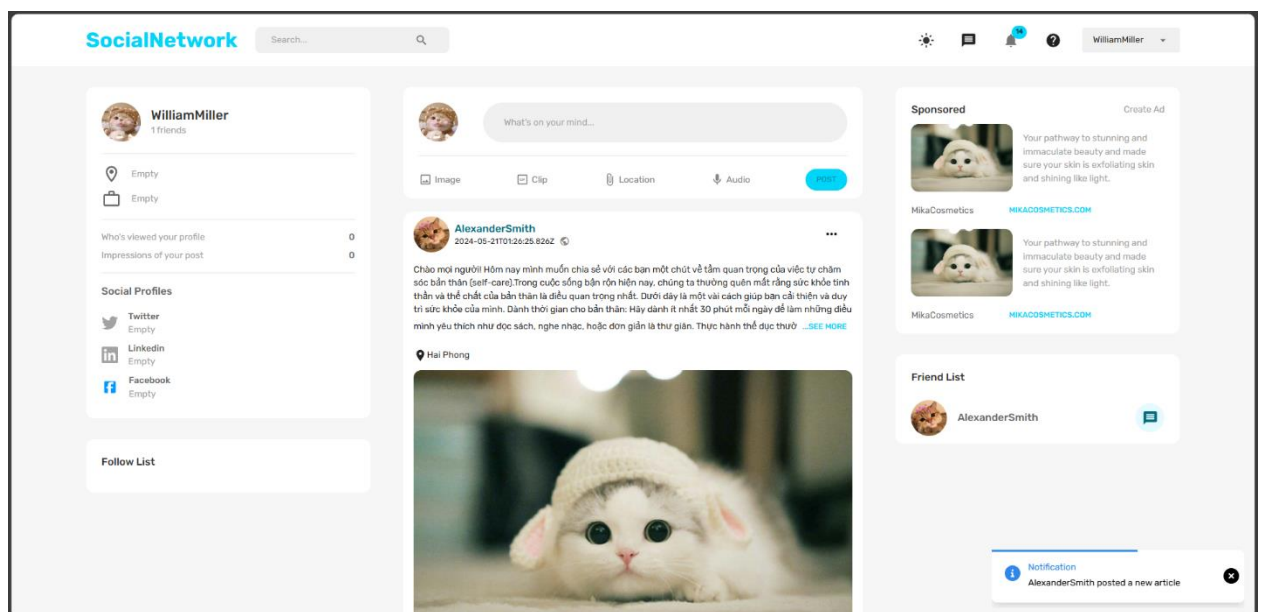


- Lúc này, AlexsanderSmith và WilliamMiller đã là bạn bè. Khi AlexsanderSmith đăng một bài viết. Ở phần thêm ảnh, khi ảnh được thêm vào, ngay lập tức sẽ được tải lên cloudinary để lưu trữ.



- Sau khi tải lên thành công, ảnh sẽ hiển thị tên.

- Khi AlexanderSmith đăng ảnh, bạn bè của anh ấy sẽ nhận được thông báo trong thời gian thực, khi đó WilliamMiller sẽ nhận được thông báo.



- Khi đó, William có thể thích bài viết, bình luận bài viết, chia sẻ bài viết. Và tương tự Alexander cũng nhận được thông báo.





AlexanderSmith

2024-06-11T02:09:08.321Z

Have a good day!

Hanoi



0 COMMENTS 0 SHARES



LIKE



COMMENT



SHARE



What's wrong with you?



## CHƯƠNG 7. KẾT LUẬN

### 7.1 Kết luận

Đồ án 1 đã hoàn thành việc khảo sát, phân tích, thiết kế và triển khai một hệ thống mạng xã hội đơn giản. Trong quá trình thực hiện đồ án, các yêu cầu của đề bài đã được đáp ứng đầy đủ, bao gồm việc xây dựng các chức năng cơ bản như đăng ký, đăng nhập, đổi mật khẩu, tìm kiếm, yêu cầu kết bạn, chấp nhận kết bạn, đăng bài viết, và tương tác với bài viết.

Tính chính xác của các chức năng đã được kiểm tra và xác nhận thông qua quá trình thử nghiệm, tuy nhiên có thể có một số lỗi trong quá trình thử nghiệm không thể phát hiện ra. Hệ thống được thiết kế với kiến trúc microservice, sử dụng các công nghệ hiện đại như Node.js, Express.js, React.js, MongoDB, PostgreSQL, và RabbitMQ, đã cho thấy tính thực tế và khả năng mở rộng của đồ án.

Đồ án có tính áp dụng vào thực tế cuộc sống cao, qua các chức năng được trình bày. Đồ án cũng đã minh chứng cho khả năng áp dụng các kiến thức lý thuyết vào thực tiễn, từ việc khảo sát nhu cầu, phân tích hệ thống, thiết kế cơ sở dữ liệu đến lập trình và triển khai hệ thống. Các kết quả đạt được phản ánh sự nỗ lực và tinh thần học hỏi của sinh viên trong quá trình thực hiện đồ án.

Qua quá trình thực hiện đồ án tốt nghiệp, em đã thu nhận được nhiều kiến thức và kỹ năng quý báu. Cụ thể:

#### **Kiến thức chuyên môn:**

- Hiểu rõ về các khái niệm và công nghệ liên quan đến mạng xã hội và API.
- Nắm vững cách xây dựng và triển khai hệ thống sử dụng kiến trúc microservice.
- Sử dụng thành thạo các công nghệ và ngôn ngữ lập trình như Node.js, Express.js, React.js, MongoDB, PostgreSQL.

#### **Kỹ năng thực hành:**

- Kỹ năng lập trình: Tự tin trong việc viết mã nguồn, tạo API, xử lý dữ liệu và triển khai hệ thống.
- Kỹ năng tìm kiếm và tổng hợp thông tin: Khả năng tự học hỏi, tra cứu tài liệu và áp dụng vào thực tiễn.
- Kỹ năng chế bản: Biết cách thiết kế cơ sở dữ liệu, xây dựng sơ đồ hệ thống, và phát triển giao diện người dùng.
- Kỹ năng trình bày: Biết cách trình bày ý tưởng, báo cáo kết quả và viết tài liệu kỹ thuật.

## 7.2 Hướng phát triển của đồ án trong tương lai

Trong tương lai, hệ thống mạng xã hội này có thể được phát triển thêm với nhiều tính năng nâng cao như:

- **Tính năng bảo mật:** Nâng cao cơ chế bảo mật, áp dụng các biện pháp mã hóa dữ liệu, bảo vệ thông tin cá nhân người dùng.
- **Tính năng AI:** Tích hợp trí tuệ nhân tạo để gợi ý bạn bè, phân tích hành vi người dùng, và đề xuất nội dung.
- **Tính năng tương tác đa phương tiện:** Hệ thống chat với thời gian thực, hỗ trợ livestream, video call, và các hình thức tương tác đa dạng khác.
- **Tính năng mở rộng:** Mở rộng khả năng kết nối với các mạng xã hội khác, hỗ trợ nhiều ngôn ngữ và khu vực khác nhau.

Đồ án này đã đặt nền móng cho những phát triển tiếp theo, và với sự hoàn thiện không ngừng, hệ thống mạng xã hội này có thể trở thành một sản phẩm hoàn chỉnh và có giá trị thực tiễn cao.

## **TÀI LIỆU THAM KHẢO**

- [1] Ferguson, P. (2020). Implementing JWT Authentication in RESTful APIs. Packt Publishing.
- [2] Mullins, C. (2018). PostgreSQL Database Performance Tuning. Apress.
- [3] Chodorow, K. (2019). MongoDB: The Definitive Guide (3rd ed.). O'Reilly Media.
- [4] Node.js Documentation - Tài liệu chính thức của Node.js với hướng dẫn và API.
- [5] Express.js Documentation - Tài liệu chính thức của Express.js, bao gồm hướng dẫn và API.
- [6] React Documentation - Tài liệu chính thức của React, cung cấp hướng dẫn và API.
- [7] MongoDB Documentation - Tài liệu chính thức của MongoDB với hướng dẫn cài đặt và sử dụng.
- [8] PostgreSQL Documentation - Tài liệu chính thức của PostgreSQL với hướng dẫn và API.
- [9] JWT.io - Trang web cung cấp thông tin chi tiết và công cụ liên quan đến JSON Web Tokens (JWT).
- [10] RabbitMQ Documentation - Tài liệu chính thức của RabbitMQ, bao gồm hướng dẫn cài đặt và sử dụng.