TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI

KHOA

A blue and white logo

Description automatically generated

**ĐỀ TÀI**

|  |  |  |
| --- | --- | --- |
| Giảng viên bộ môn  Lớp môn học  Nhóm  Sinh viên thực hiện | :  :  :  : |  |

***Hà Nội, 2024***

**MỤC LỤC**

[MỞ ĐẦU 4](#_Toc199254737)

[CHƯƠNG 1:GIỚI THIỆU ĐỀ TÀI 5](#_Toc199254738)

[1.1.Lý do chọn đề tài 5](#_Toc199254739)

[1.2. Mục tiêu của đồ án 5](#_Toc199254740)

[1.3. Phạm vi và giới hạn 6](#_Toc199254741)

[1.4. Phương pháp thực hiện 7](#_Toc199254742)

[CHƯƠNG 2:PHÂN TÍCH HỆ THỐNG 9](#_Toc199254743)

[2.1. Mô tả bài toán 9](#_Toc199254744)

[2.2. Yêu cầu chức năng 9](#_Toc199254745)

[2.3. Yêu cầu phi chức năng 10](#_Toc199254746)

[2.4. Use Case Diagram tổng quát: 11](#_Toc199254747)

[2.4.1. Use Case tổng quát: 11](#_Toc199254748)

[2.4.2. Use Case đăng ký & đăng nhập & đăng xuất: 12](#_Toc199254749)

[2.4.3. Use Case xem tin nhắn: 13](#_Toc199254750)

[2.4.4. Use Case xem danh sách bạn bè: 14](#_Toc199254751)

[2.4.5. Use Case cập nhật Profile: 15](#_Toc199254752)

[2.4.6. Use Case xem trạng thái online: 16](#_Toc199254753)

[2.4.7. Use Case gửi tin nhắn: 17](#_Toc199254754)

[2.4.8. Use Case nhận tin nhắn: 18](#_Toc199254755)

[2.4.9.Use Case thu hồi tin nhắn: 19](#_Toc199254756)

[2.5. Mô tả các Use Case chính: 19](#_Toc199254757)

[2.5.1. Use Case đăng ký & đăng nhập & đăng xuất: 19](#_Toc199254758)

[2.5.2. Use Case xem tin nhắn: 20](#_Toc199254759)

[2.5.3. Use Case xem danh sách bạn bè: 20](#_Toc199254760)

[2.5.4. Use Case xem trạng thái online: 20](#_Toc199254761)

[2.5.5. Use Case cập nhật Profile: 21](#_Toc199254762)

[2.5.6. Use Case gửi tin nhắn: 21](#_Toc199254763)

[2.5.7. Use Case nhận tin nhắn Real-Time: 21](#_Toc199254764)

[2.5.8. Use Case thu hồi tin nhắn: 22](#_Toc199254765)

[CHƯƠNG 3. THIẾT KẾ HỆ THỐNG 23](#_Toc199254766)

[3.1. Lựa chọn mô hình kiến trúc (Monolithic, Microservices, MVC, Layered, Clean Architecture, ...) 23](#_Toc199254767)

[3.2. Sơ đồ kiến trúc tổng thể: 23](#_Toc199254768)

[3.2.1. Sơ đồ: 23](#_Toc199254769)

[3.2.2. Mô tả các thành phần: 24](#_Toc199254770)

[3.3. Biểu đồ tuần tự (các chức năng chính) 25](#_Toc199254771)

[3.3.1 Đăng ký: 25](#_Toc199254772)

[3.3.2 Đăng nhập: 26](#_Toc199254773)

[3.3.3 Chỉnh sửa profile: 27](#_Toc199254774)

[3.3.4 Gửi và nhận tin nhắn: 28](#_Toc199254775)

[3.3.5 Thu hồi tin nhắn: 29](#_Toc199254776)

[3.5. Thiết kế giao diện người dùng (Mockup / Wireframe): 30](#_Toc199254777)

[3.5.1. Thiết kế giao diện người dùng: Wireframe 30](#_Toc199254778)

[3.5.2 Thiết kế giao diện người dùng: Mockup 33](#_Toc199254779)

[CHƯƠNG 4. CÀI ĐẶT VÀ TRIỂN KHAI HỆ THỐNG 35](#_Toc199254780)

[4.1. Môi trường triển khai 35](#_Toc199254781)

[4.2. Cài đặt hệ thống: 35](#_Toc199254782)

[4.2.1 Yêu cầu hệ thống: 35](#_Toc199254783)

[4.2.2 Các bước cài đặt chương trình 35](#_Toc199254784)

[CHƯƠNG 5. KẾT QUẢ VÀ ĐÁNH GIÁ 37](#_Toc199254785)

[5.1. Kết quả thử nghiệm hệ thống 37](#_Toc199254786)

[5.2. Đánh giá hiệu quả hệ thống 37](#_Toc199254787)

[Tài liệu tham khảo 39](#_Toc199254788)

# MỞ ĐẦU

Trong thời đại công nghệ số, việc giao tiếp trực tuyến đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày. Các ứng dụng chat thời gian thực không chỉ giúp kết nối con người một cách nhanh chóng mà còn hỗ trợ hiệu quả trong công việc, học tập và giải trí. Với sự phát triển mạnh mẽ của các công nghệ web hiện đại, việc xây dựng một ứng dụng chat thời gian thực không chỉ đáp ứng nhu cầu giao tiếp mà còn là cơ hội để nghiên cứu và áp dụng các công nghệ tiên tiến.

Nhận thức được tầm quan trọng của việc giao tiếp trực tuyến và sự phổ biến của các ứng dụng chat, chúng em đã lựa chọn đề tài xây dựng Ứng dụng Chat Thời Gian Thực. Đề tài này được phát triển dựa trên MERN stack (MongoDB, Express.js, React.js, Node.js) kết hợp với Socket.IO để xử lý giao tiếp thời gian thực. Ngoài ra, ứng dụng còn tích hợp các tính năng như gửi tin nhắn văn bản, hình ảnh, tự động kết bạn, cập nhật thông tin cá nhân và thay đổi giao diện (theme).

Trong quá trình thực hiện đề tài, chúng em đã nhận được sự hướng dẫn tận tình từ thầy. Chúng em xin gửi lời cảm ơn chân thành đến thầy Phạm Hữu Tùng đã hỗ trợ và định hướng cho chúng em trong suốt quá trình nghiên cứu và phát triển ứng dụng này.

Chúng em hy vọng rằng sản phẩm này không chỉ là một bài tập thực hành mà còn là một ứng dụng có tính thực tiễn cao, đáp ứng nhu cầu giao tiếp trực tuyến của người dùng trong thời đại số.

# CHƯƠNG 1:GIỚI THIỆU ĐỀ TÀI

## 1.1.Lý do chọn đề tài

Trong thời đại công nghệ số phát triển mạnh mẽ, nhu cầu giao tiếp trực tuyến ngày càng trở nên thiết yếu trong cả học tập, công việc lẫn đời sống cá nhân. Các ứng dụng nhắn tin thời gian thực như Messenger, Zalo hay Telegram đã trở thành một phần không thể thiếu, đặc biệt trong bối cảnh làm việc từ xa và học trực tuyến ngày càng phổ biến.

Trước xu thế đó, việc xây dựng một ứng dụng chat thời gian thực (real-time chat) không chỉ có giá trị thực tiễn mà còn mang lại cơ hội học tập, nghiên cứu và áp dụng các công nghệ web hiện đại. Nhóm em quyết định lựa chọn đề tài "Xây dựng ứng dụng web Chat Real-Time sử dụng MERN Stack và Socket.IO" với mong muốn phát triển một hệ thống giao tiếp trực tuyến đơn giản, dễ sử dụng nhưng mang lại trải nghiệm tương tác hiệu quả cho người dùng.

Đề tài này giúp em:

* Áp dụng toàn diện kiến thức về phát triển ứng dụng web qua MERN stack: sử dụng ReactJS cho giao diện người dùng, NodeJs và ExpressJs để xây dựng API backend, và MongoDB để lưu trữ dữ liệu.
* Tìm hiểu và triển khai mô hình giao tiếp thời gian thực (Real-Time Communication) thông qua công nghệ SocketIo, giúp truyền nhận tin nhắn nhanh chóng và đồng bộ giữa người dùng.
* Học cách quản lý và xử lý dữ liệu đa phương tiện (hình ảnh) thông qua dịch vụ như Cloudinary, đồng thời cải thiện kỹ năng tích hợp API bên ngoài.
* Rèn luyện khả năng phân tích hệ thống, thiết kế giao diện người dùng (UI/UX), xử lý trạng thái, đồng bộ dữ liệu và tối ưu hiệu suất cho ứng dụng web.

Ngoài ra, việc thực hiện đề tài này còn là cơ hội để sinh viên làm quen với các thách thức thực tế trong phát triển ứng dụng, từ việc xử lý đồng bộ dữ liệu thời gian thực đến triển khai bảo mật cơ bản và mở rộng chức năng người dùng. Đây là bước đệm quan trọng giúp trang bị kỹ năng chuyên môn vững chắc trong lĩnh vực lập trình web và phát triển phần mềm hiện đại.

## 1.2. Mục tiêu của đồ án

Mục tiêu của đồ án là xây dựng một ứng dụng web chat thời gian thực (real-time) với giao diện hiện đại, thân thiện với người dùng, đồng thời tích hợp đầy đủ các chức năng thiết yếu phục vụ nhu cầu giao tiếp trực tuyến. Cụ thể, đồ án hướng đến các mục tiêu sau:

**Về chức năng người dùng:**

* Cho phép người dùng đăng ký và đăng nhập tài khoản.
* Tự động kết bạn giữa các tài khoản ngay sau lần đăng nhập đầu tiên.
* Hỗ trợ gửi và nhận tin nhắn văn bản giữa các người dùng trong thời gian thực.
* Hỗ trợ gửi và nhận hình ảnh, cho phép chia sẻ đa phương tiện nhanh chóng.
* Hiển thị trạng thái online/offline của người dùng để tăng tính tương tác.
* Cho phép người dùng cập nhật thông tin cá nhân như tên hiển thị, avatar, ...
* Hỗ trợ tùy chỉnh giao diện theo sở thích (ví dụ: chế độ sáng/tối - light/dark theme).

**Về công nghệ sử dụng:**

* Frontend được xây dựng bằng ReactJS, giúp tạo giao diện mượt mà và linh hoạt.
* Backend sử dụng NodeJs và ExpressJs để xử lý API và kết nối với cơ sở dữ liệu.
* MongoDB dùng làm hệ quản trị cơ sở dữ liệu NoSQL, lưu trữ thông tin người dùng và tin nhắn.
* Socket.IO đảm nhận việc giao tiếp thời gian thực, giúp tin nhắn được gửi/nhận tức thì qua WebSocket.
* Cloudinary tích hợp để lưu trữ và quản lý ảnh, hỗ trợ tối ưu hóa và truy xuất nhanh chóng.

**Về hiệu suất và bảo mật:**

* Tối ưu hóa hiệu suất để đảm bảo hệ thống hoạt động ổn định với nhiều người dùng đồng thời.
* Bảo mật thông tin người dùng thông qua việc:
  + Mã hóa mật khẩu trước khi lưu vào cơ sở dữ liệu.
  + Xác thực người dùng bằng JWT (JSON Web Token) trong các phiên làm việc.

**Về học tập và nghiên cứu:**

* Nắm vững mô hình client-server và phương pháp truyền dữ liệu thời gian thực bằng WebSocket.
* Hiểu và áp dụng kiến thức về quản lý trạng thái ứng dụng, RESTful API, xử lý bất đồng bộ, và tích hợp hệ thống đa công nghệ.

## 1.3. Phạm vi và giới hạn

**Phạm vi:**

Đồ án tập trung xây dựng một ứng dụng web chat thời gian thực (real-time chat) với các tính năng cơ bản, vận hành trên nền tảng trình duyệt web. Cụ thể, hệ thống bao gồm các chức năng chính như sau:

* Cho phép người dùng đăng ký và đăng nhập tài khoản.
* Tự động kết bạn giữa tất cả các tài khoản ngay khi đăng nhập lần đầu, giúp đơn giản hóa quá trình kết nối người dùng.Hỗ trợ giao tiếp 1-1 giữa các người dùng đã là bạn bè thông qua:

o Tin nhắn văn bản thời gian thực.

o Tin nhắn hình ảnh, có lưu trữ trên nền tảng đám mây.

* Giao diện người dùng đơn giản, dễ sử dụng, có hỗ trợ responsive để hoạt động tốt trên các thiết bị khác nhau.
* Cho phép cập nhật thông tin cá nhân và tùy chỉnh giao diện (theme với nhiều màu sắc đa dạng).

**Giới hạn:**

Do thời gian và phạm vi thực hiện đồ án có hạn, hệ thống hiện tại vẫn còn một số hạn chế:

* Chỉ hỗ trợ nhắn tin giữa hai người dùng (1-1), chưa hỗ trợ tạo và quản lý nhóm chat.
* Không có tính năng nâng cao như tìm kiếm tin nhắn, gọi thoại/video, gửi file lớn, hoặc thông báo đẩy (push notification).
* Chưa triển khai cơ chế mã hóa đầu cuối (end-to-end encryption), do đó nội dung tin nhắn vẫn có thể bị truy cập ở phía máy chủ.
* Ứng dụng mới chỉ hoạt động ổn định ở quy mô nhỏ, phù hợp cho mục đích demo hoặc thử nghiệm; chưa có khả năng mở rộng hoặc tối ưu hiệu suất để phục vụ lượng lớn người dùng đồng thời.
* Triển khai chủ yếu trong môi trường thử nghiệm, như localhost, chưa đưa vào môi trường sản xuất thực tế.

## 1.4. Phương pháp thực hiện

**Phân tích và thu thập yêu cầu:**

* Xác định các chức năng cần thiết của một ứng dụng chat cơ bản như: đăng ký, đăng nhập, gửi tin nhắn văn bản và hình ảnh, quản lý tài khoản cá nhân, hiển thị trạng thái người dùng và tùy chỉnh giao diện.
* Phân tích luồng tương tác giữa người dùng và hệ thống để đảm bảo logic hoạt động hợp lý.

**Thiết kế hệ thống:**

* Thiết kế kiến trúc tổng thể theo mô hình client-server sử dụng MERN Stack kết hợp Socket.IO.
* Thiết kế cơ sở dữ liệu MongoDB để lưu trữ thông tin người dùng, danh sách bạn bè, nội dung tin nhắn và dữ liệu liên quan.
* Thiết kế giao diện người dùng với giao diện đơn giản, trực quan và dễ sử dụng, hỗ trợ responsive.
* Vẽ các sơ đồ luồng dữ liệu và kiến trúc hệ thống để minh họa cách các thành phần hoạt động và tương tác với nhau.

**Cài đặt và phát triển:**

* **Frontend:**
  + Sử dụng ReactJs để xây dựng giao diện người dùng.
  + Kết nối với API của backend và tích hợp Socket.IO client để gửi/nhận tin nhắn thời gian thực.
  + Sử dụng Zustand để quản lý trạng thái toàn cục, và Tailwind CSS để thiết kế giao diện đẹp, hiện đại.
* **Backend:**
  + Dùng NodeJs và ExpressJs để xây dựng API và xử lý logic máy chủ.
  + Dùng Mongoose để tương tác với cơ sở dữ liệu MongoDB.
  + Tích hợp Socket.IO để xử lý kết nối thời gian thực giữa các client.
  + Triển khai xác thực bằng JWT (JSON Web Token) và mã hóa mật khẩu bằng bcrypt.
* **Lưu trữ hình ảnh:**
  + Sử dụng Cloudinary để lưu trữ và truy xuất ảnh gửi qua chat, giúp tối ưu dung lượng và hiệu suất.

**Kiểm thử hệ thống:**

* Kiểm tra các chức năng chính như: đăng nhập, gửi tin nhắn văn bản và hình ảnh, cập nhật thông tin cá nhân, đổi theme.
* Đặc biệt chú trọng kiểm thử thời gian thực với Socket.IO, đảm bảo tin nhắn hiển thị ngay lập tức giữa các người dùng.
* Kiểm thử giao diện trên nhiều kích thước màn hình để đảm bảo tính responsive.

**Triển khai và chạy thử:**

* Chạy thử nghiệm ứng dụng trên môi trường localhost để kiểm tra tính ổn định.

# CHƯƠNG 2:PHÂN TÍCH HỆ THỐNG

## 2.1. Mô tả bài toán

Bài toán đặt ra là xây dựng một ứng dụng web chat thời gian thực cho phép người dùng giao tiếp với nhau thông qua tin nhắn văn bản và hình ảnh, với giao diện hiện đại, dễ sử dụng.

Khi người dùng đăng nhập lần đầu, hệ thống sẽ tự động kết bạn với tất cả các tài khoản khác, giúp thuận tiện trong việc trao đổi thông tin mà không cần thao tác thủ công.

Ứng dụng cần đảm bảo tính năng thời gian thực (real-time), nghĩa là tin nhắn phải được gửi và hiển thị ngay lập tức giữa các người dùng mà không cần tải lại trang. Ngoài ra, người dùng có thể cập nhật thông tin cá nhân (tên hiển thị, avatar, v.v.) và tùy chỉnh giao diện (dark/light theme) theo sở thích cá nhân.

**Các yêu cầu chính của bài toán:**

* **Giao tiếp thời gian thực**: Tin nhắn được gửi/nhận ngay lập tức thông qua công nghệ WebSocket.
* **Kết bạn tự động**: Người dùng mới sẽ được kết nối với tất cả các tài khoản hiện có.
* **Gửi đa phương tiện**: Hỗ trợ gửi tin nhắn văn bản và hình ảnh.
* **Cập nhật thông tin cá nhân**: Cho phép người dùng chỉnh sửa tên, ảnh đại diện, ...
* **Tùy biến giao diện**: Cho phép chuyển đổi giữa giao diện sáng và tối.
* **Yêu cầu kỹ thuật**: Hệ thống phải có giao diện thân thiện, dễ sử dụng, bảo mật tốt, và hoạt động ổn định kể cả khi có nhiều người dùng đồng thời.

## 2.2. Yêu cầu chức năng

Ứng dụng cần đáp ứng các chức năng chính như sau:

* **Đăng ký tài khoản**: Cho phép người dùng nhập email, mật khẩu và thông tin cá nhân để tạo tài khoản mới.
* **Đăng nhập**: Người dùng có thể đăng nhập vào hệ thống bằng email và mật khẩu. Hệ thống sử dụng JWT (JSON Web Token) để xác thực người dùng một cách an toàn.
* **Tự động kết bạn**: Khi người dùng đăng nhập lần đầu tiên, hệ thống sẽ tự động kết nối tài khoản đó với tất cả các tài khoản khác hiện có trong hệ thống để tiện nhắn tin.
* **Chat thời gian thực**: Người dùng có thể gửi và nhận tin nhắn văn bản với người khác theo thời gian thực, sử dụng Socket.IO để đảm bảo việc truyền nhận dữ liệu diễn ra ngay lập tức, không cần tải lại trang.
* **Gửi hình ảnh**: Hệ thống hỗ trợ gửi ảnh đính kèm trong cuộc trò chuyện. Ảnh sẽ được tải lên và lưu trữ trên dịch vụ Cloudinary.
* **Hiển thị trạng thái online/offline**: Ứng dụng hiển thị trạng thái hoạt động (online hoặc offline) của từng người dùng dựa trên kết nối Socket.IO.
* **Cập nhật thông tin cá nhân**: Người dùng có thể thay đổi tên hiển thị, avatar, và các thông tin cá nhân khác sau khi đăng ký.
* **Tùy chỉnh giao diện**: Ứng dụng hỗ trợ chế độ sáng (light mode) và tối (dark mode), cho phép người dùng lựa chọn giao diện phù hợp với sở thích cá nhân.

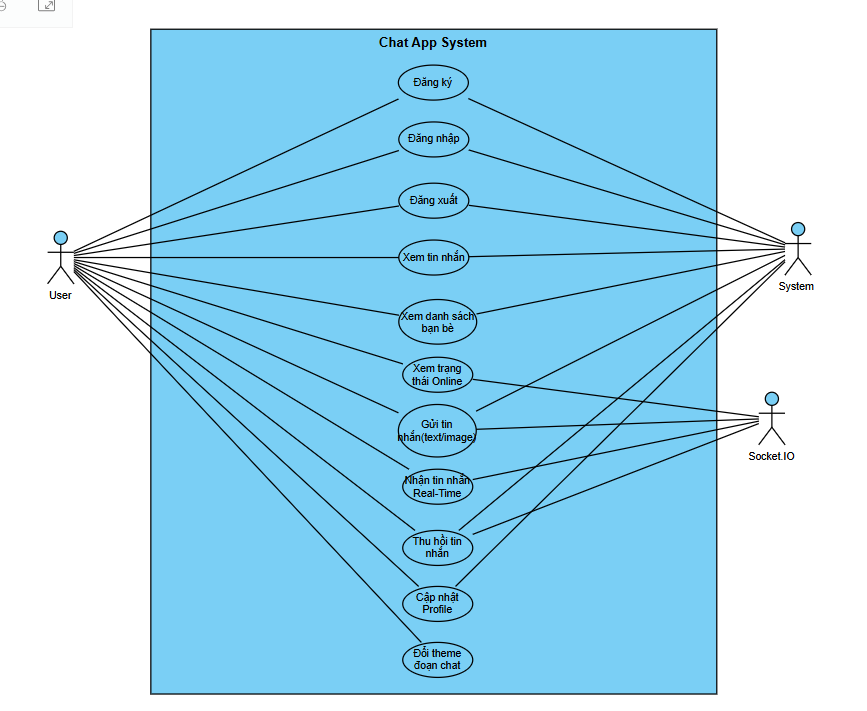
## 2.3. Yêu cầu phi chức năng

Ứng dụng cần đáp ứng các yêu cầu phi chức năng sau:

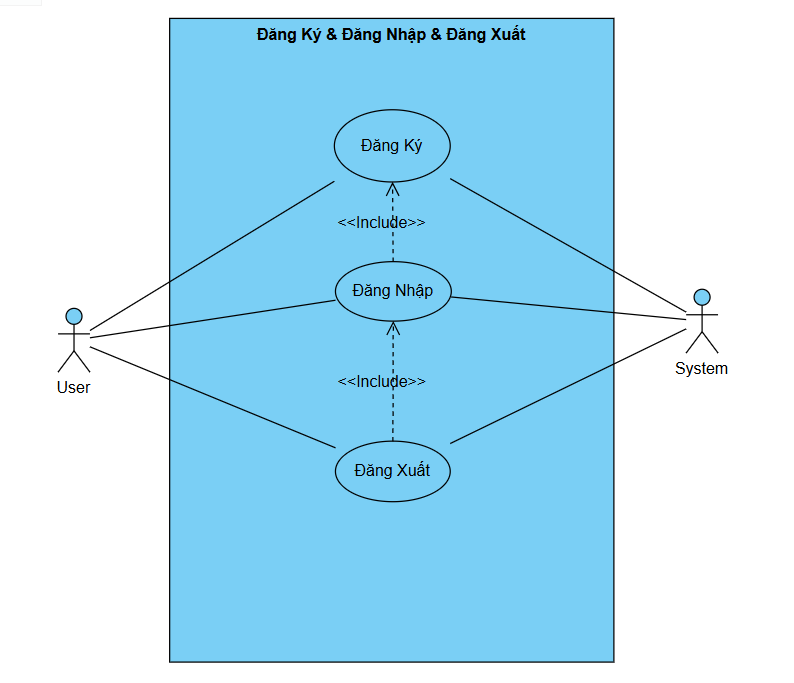
* **Hiệu suất:** Hệ thống phải đảm bảo phản hồi nhanh với độ trễ thấp, cụ thể là tin nhắn phải được gửi và nhận trong thời gian dưới 1 giây. Ứng dụng cần hoạt động ổn định với ít nhất 50 người dùng đồng thời mà không gặp lỗi hay gián đoạn dịch vụ.
* **Bảo mật:** Tất cả dữ liệu người dùng phải được lưu trữ một cách an toàn. Mật khẩu cần được mã hóa trước khi lưu trữ, ví dụ sử dụng thuật toán mã hóa bcrypt. Ngoài ra, việc xác thực và bảo vệ các API được thực hiện thông qua JWT (JSON Web Token) để đảm bảo tính toàn vẹn và riêng tư của thông tin người dùng.
* **Khả năng mở rộng:** Hệ thống được thiết kế với kiến trúc có thể mở rộng, cho phép dễ dàng tích hợp thêm các tính năng trong tương lai như nhắn tin nhóm, gọi thoại/video, hoặc mã hóa đầu cuối (end-to-end encryption) mà không cần thay đổi lớn trong kiến trúc hiện tại.
* **Tính tương thích:** Ứng dụng phải chạy ổn định trên các trình duyệt phổ biến như Google Chrome, Mozilla Firefox, và Microsoft Edge. Đồng thời, nó cũng cần tương thích với các thiết bị khác nhau bao gồm máy tính để bàn, máy tính bảng và điện thoại di động.
* **Tính dễ sử dụng:** Giao diện người dùng phải trực quan, hiện đại và thân thiện với người mới. Các thao tác cần được thiết kế đơn giản, dễ hiểu để đảm bảo trải nghiệm mượt mà và thuận tiện.
* **Khả năng bảo trì:** Mã nguồn của dự án cần được tổ chức hợp lý, rõ ràng và dễ bảo trì.

## 2.4. Use Case Diagram tổng quát:

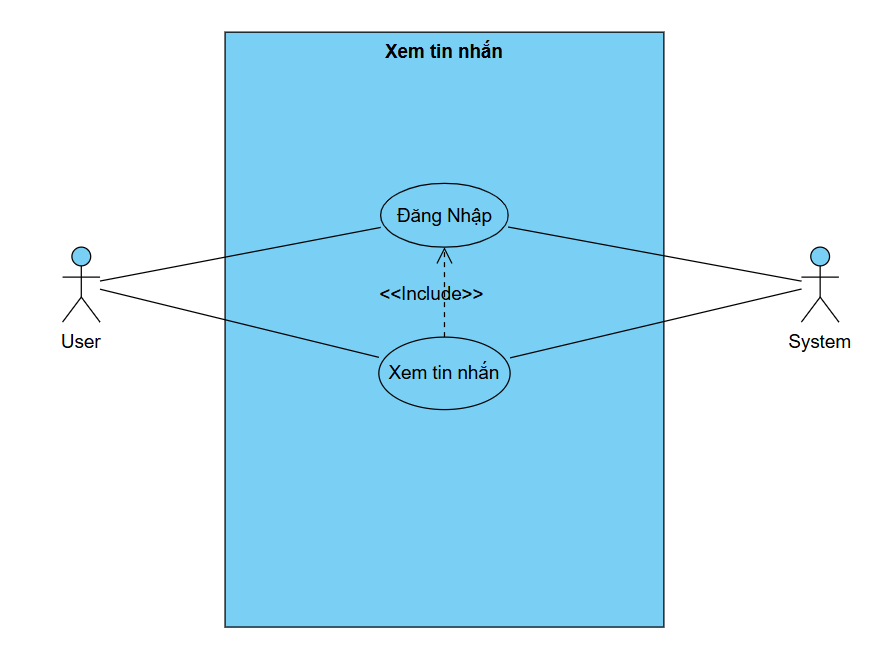
### 2.4.1. Use Case tổng quát:



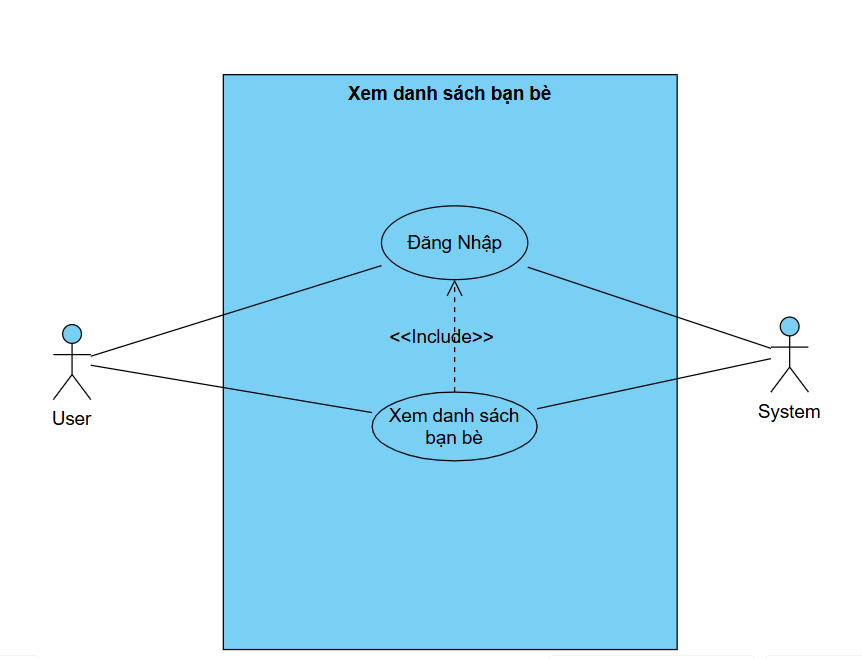
### 2.4.2. Use Case đăng ký & đăng nhập & đăng xuất:



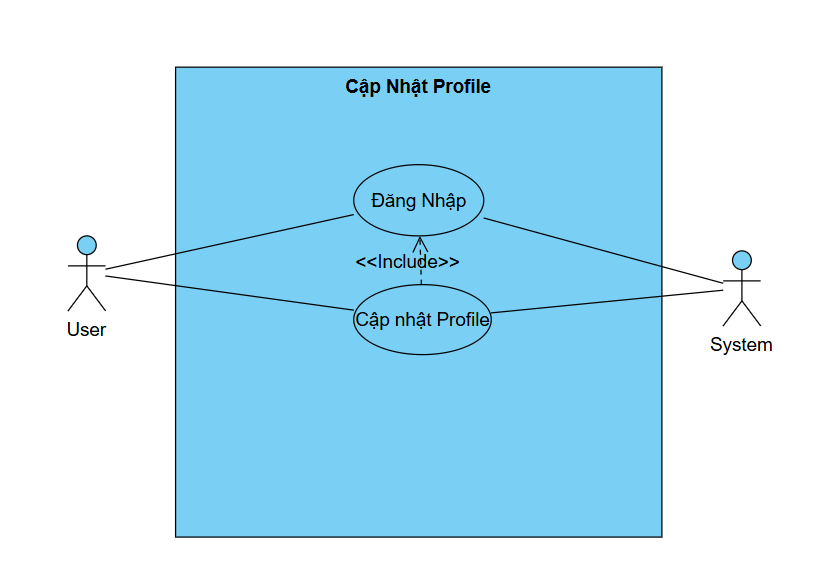
### 2.4.3. Use Case xem tin nhắn:



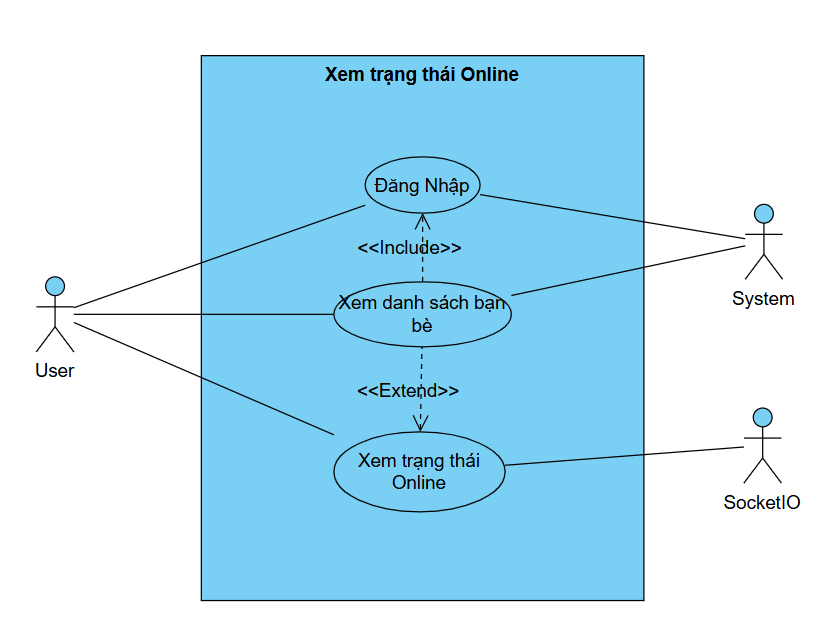
### 2.4.4. Use Case xem danh sách bạn bè:



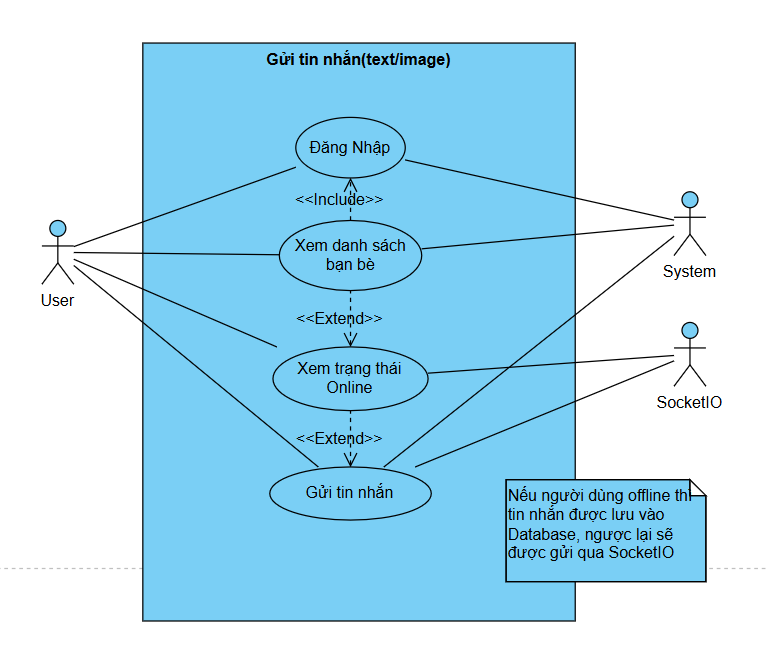
### 2.4.5. Use Case cập nhật Profile:



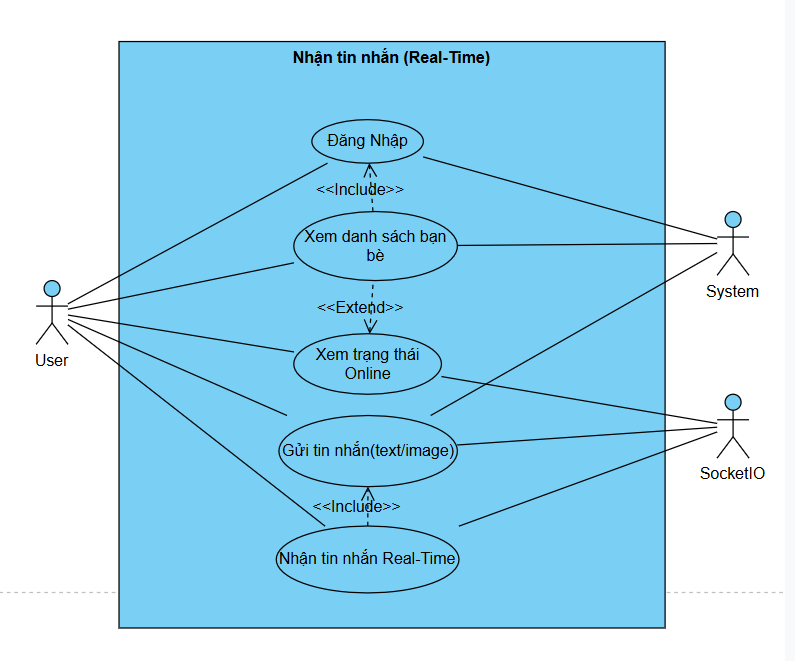
### 2.4.6. Use Case xem trạng thái online:



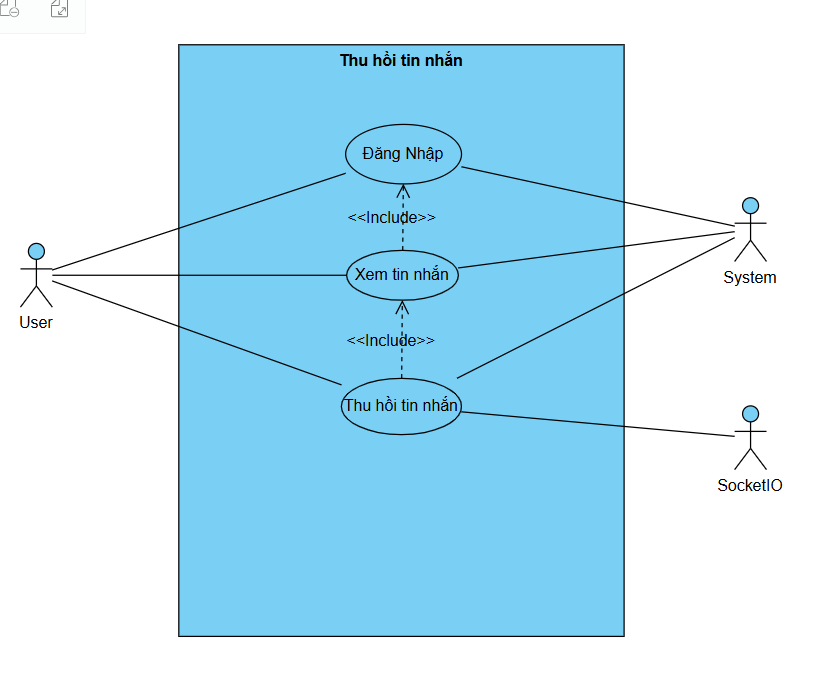
### 2.4.7. Use Case gửi tin nhắn:



### 2.4.8. Use Case nhận tin nhắn:



### 2.4.9.Use Case thu hồi tin nhắn:



## 2.5. Mô tả các Use Case chính:

### 2.5.1. Use Case đăng ký & đăng nhập & đăng xuất:

|  |  |
| --- | --- |
| **Thành Phần** | **Mô Tả** |
| Tên Use Case | Đăng Ký, Đăng Nhập, Đăng Xuất hệ thống |
| Mục tiêu | Người dùng đăng ký vào hệ thống, sau khi đăng ký thành công thì có thể đăng nhập, khi đăng nhập có thể đăng xuất để thoát ra |
| Tác nhân | Người dùng |
| Điều kiện tiên quyết | Người dùng đã có tài khoản hợp lệ |
| Luồng sự kiện chính | 1.Người dùng mở màn hình đăng ký.  2.Người dùng nhập đầy đủ thông tin và ấn nút đăng ký tài khoản.  3.Hệ thống xác minh thông tin đăng nhập.  4.Nếu hợp lệ, hệ thống cho phép truy cập vào trang chat.  5.Người dùng có thể đăng xuất tài khoản để trở về trang đăng nhập. |

### 2.5.2. Use Case xem tin nhắn:

|  |  |
| --- | --- |
| **Thành phần** | **Mô Tả** |
| Tên Use Case | Xem Tin Nhắn |
| Mục tiêu | Người dùng đăng nhập vào để có thể xem được tin nhắn. |
| Tác nhân | Người dùng |
| Điều kiện tiên quyết | Người dùng đã có tài khoản hợp lệ |
| Luồng sự kiện chính | 1.Sau khi người dùng có tài khoản và đã đăng nhập vào hệ thống  2.Người dùng chọn một cuộc trò chuyện bất kỳ của ai đó.  3.Người dùng có thể đọc được tin nhắn của mình và của người khác nhắn cho mình. |

### 2.5.3. Use Case xem danh sách bạn bè:

|  |  |
| --- | --- |
| **Thành Phần** | **Mô Tả** |
| Tên Use Case | Xem Danh Sách Bạn Bè |
| Mục tiêu | Người dùng đăng nhập vào để có thể xem được danh sách bạn bè của mình. |
| Tác nhân | Người dùng |
| Điều kiện tiên quyết | Người dùng đã có tài khoản hợp lệ |
| Luồng sự kiện chính | 1.Sau khi người dùng có tài khoản và đã đăng nhập vào hệ thống.  2.Khi đăng nhập thành công được đưa vào trang chủ đoạn chat.  3.Ở đây người dùng có thể xem được danh sách bạn bè của mình. |

### 2.5.4. Use Case xem trạng thái online:

|  |  |
| --- | --- |
| **Thành Phần** | **Mô Tả** |
| Tên Use Case | Xem Trạng Thái Online |
| Mục tiêu | Người dùng có thể xem được trạng thái online hay offline của bạn bè. |
| Tác nhân | Người dùng |
| Điều kiện tiên quyết | Người dùng đã có tài khoản hợp lệ |
| Luồng sự kiện chính | 1.Sau khi đã có tài khoản và đã đăng nhập vào hệ thống.  2.Ở giao diện trang chủ chat người dùng có thể xem được trạng thái online hay offline của bạn bè. |

### 2.5.5. Use Case cập nhật Profile:

|  |  |
| --- | --- |
| **Thành Phần** | **Mô Tả** |
| Tên Use Case | Cập nhật thông tin người dùng |
| Mục tiêu | Người dùng có thể thay đổi được ảnh đại diện của mình. |
| Tác nhân | Người dùng. |
| Điều kiện tiên quyết | Người dùng đã có tài khoản hợp lệ. |
| Luồng sự kiện chính | 1.Sau khi đã có tài khoản và đã đăng nhập vào hệ thống.  2.Người dùng chọn vào nút profile trên thanh header.  3.Lúc này sẽ hiện ra một mock up để người dùng có thể thay đổi được ảnh của mình. |

### 2.5.6. Use Case gửi tin nhắn:

|  |  |
| --- | --- |
| **Thành Phần** | **Mô tả** |
| Tên Use Case | Gửi Tin Nhắn |
| Mục tiêu | Cho phép người dùng gửi tin nhắn tới người khác và nhận phản hồi ngay lập tức |
| Tác nhân | Người dùng |
| Điều kiện tiên quyết | Người dùng đã đăng nhập và có kết nối mạng |
| Luồng sự kiện chính | 1.Sau khi đã đăng nhập vào hệ thống thành công.  2.Người dùng chọn một cuộc trò chuyện bất kì bên thanh sidebar.  3.Người dùng nhập tin nhắn trên thanh message input và nhấn gửi.  4.Tin nhắn được gửi đến máy chủ.  5.Máy chủ gửi tin nhắn cho người nhận qua WebSocket / socket.io.  6.Người nhận thấy tin nhắn ngay lập tức trong giao diện.  7.Nếu người dùng offline thì tin nhắn được lưu trên database. |

### 2.5.7. Use Case nhận tin nhắn Real-Time:

|  |  |
| --- | --- |
| **Thành Phần** | **Mô Tả** |
| Tên Use Case | Nhận tin nhắn Real Time. |
| Mục tiêu | Người dùng nhận tin nhắn từ người khác ngay lập tức. |
| Tác nhân | Người dùng. |
| Điều kiện tiên quyết | Người dùng đã đăng nhập và có kết nối mạng. |
| Luồng sự kiện chính | 1.Hệ thống (qua WebSocket) nhận tin nhắn mới.  2.Hiển thị tin nhắn trong giao diện chat. |

### 2.5.8. Use Case thu hồi tin nhắn:

|  |  |
| --- | --- |
| **Thành Phần** | **Mô Tả** |
| Tên Use Case | Thu hồi tin nhắn |
| Mục tiêu | Cho phép người dùng xóa/ tự thu hồi tin nhắn đã gửi để không hiển thị với người nhận. |
| Tác nhân | Người dùng (người gửi tin nhắn) |
| Điều kiện tiên quyết | Người dùng đã đăng nhập và đã gửi ít nhất một tin nhắn trong cuộc trò chuyện. |
| Luồng sự kiện chính | 1.Người dùng hover vào tin nhắn đã gửi.  2.Sau khi hover sẽ hiện biểu tượng thùng rác.  3.Người dùng chọn vào biểu tượng thùng rác.  4.Hiển thị mock up hỏi “Bạn có chắc muốn thu hồi tin nhắn hay không”?  5.Người dùng đồng ý thu hồi.  6.Ứng dụng gửi yêu cầu thu hồi tin nhắn đến máy chủ.  7.Máy chủ cập nhật trạng thái tin nhắn và gửi thông báo đến cả người gửi và người nhận.  8.Tin nhắn được cập nhật thành “Tin nhắn đã được thu hồi” trên cả hai phía. |

# CHƯƠNG 3. THIẾT KẾ HỆ THỐNG

## 3.1. Lựa chọn mô hình kiến trúc (Monolithic, Microservices, MVC, Layered, Clean Architecture, ...)

**Mô hình MVC:**

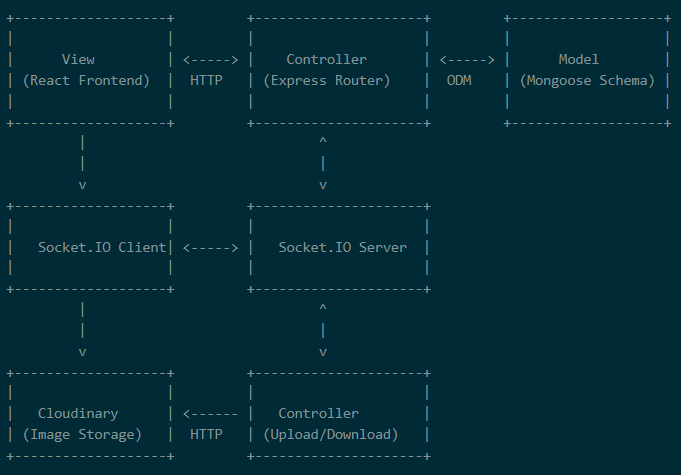
**\***Model – Mô hình dữ liệu: Đại diện cho dữ liệu và logic xử lý dữ liệu của ứng dụng. Model tương tác trực tiếp với cơ sở dữ liệu hoặc API.

\*View – Giao diện người dùng: Là phần hiển thị dữ liệu cho người dùng (UI). View nhận dữ liệu từ Model và hiển thị ra màn hình.

\*Controller – Bộ điều khiển: Là nơi xử lý các yêu cầu (request) từ người dùng. Controller nhận input từ View, xử lý, rồi cập nhật View

## 3.2. Sơ đồ kiến trúc tổng thể:

### 3.2.1. Sơ đồ:



### 

### 3.2.2. Mô tả các thành phần:

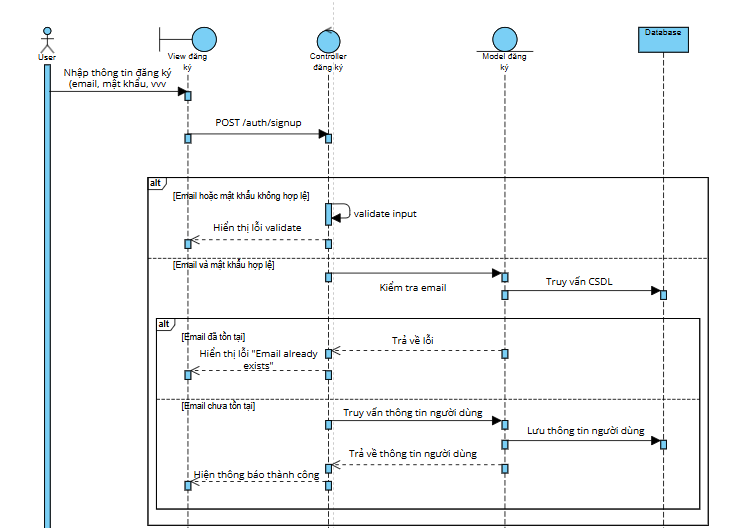
* **View (React Frontend):**
  + Giao diện người dùng, hiển thị dữ liệu, nhận thao tác từ user.
  + Gửi request API tới Controller (backend).
  + Kết nối Socket.IO client để nhận/gửi dữ liệu real-time.
* **Controller (Express Router):**
  + Xử lý request từ View, gọi Model để thao tác dữ liệu.
  + Xử lý logic nghiệp vụ (đăng nhập, gửi tin nhắn, upload ảnh...).
  + Giao tiếp với Cloudinary để upload ảnh.
  + Phát sự kiện real-time qua Socket.IO server.
* **Model (Mongoose Schema):**
  + Định nghĩa cấu trúc dữ liệu (User, Message...).
  + Thao tác với database (MongoDB).
* **Socket.IO:**
  + Kênh truyền thông real-time giữa client và server.
* **Cloudinary:**
  + Lưu trữ ảnh đại diện, ảnh gửi trong chat.

3.3.3. Luồng Dữ Liệu:

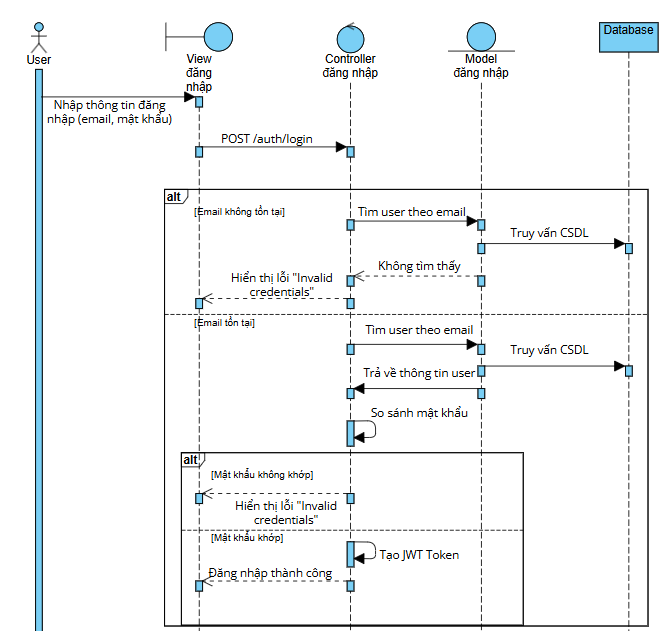
1. User thao tác trên View (React) → gửi request tới Controller (Express).
2. Controller xử lý logic, thao tác Model (Mongoose) để lấy/lưu dữ liệu.
3. Controller trả kết quả về View.
4. Với các thao tác real-time, Controller phát sự kiện qua Socket.IO server đến các client.
5. Khi upload ảnh, Controller gửi ảnh lên Cloudinary, nhận URL và lưu vào Model.

## 3.3. Biểu đồ tuần tự (các chức năng chính)

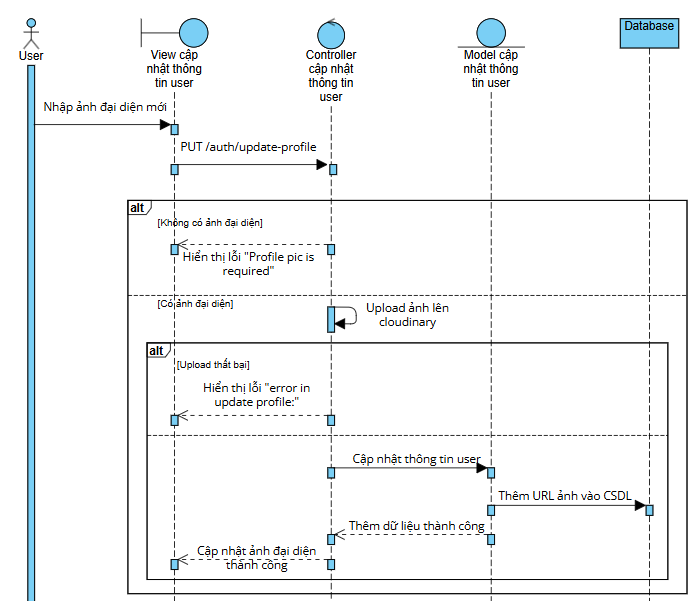
### 3.3.1 Đăng ký:



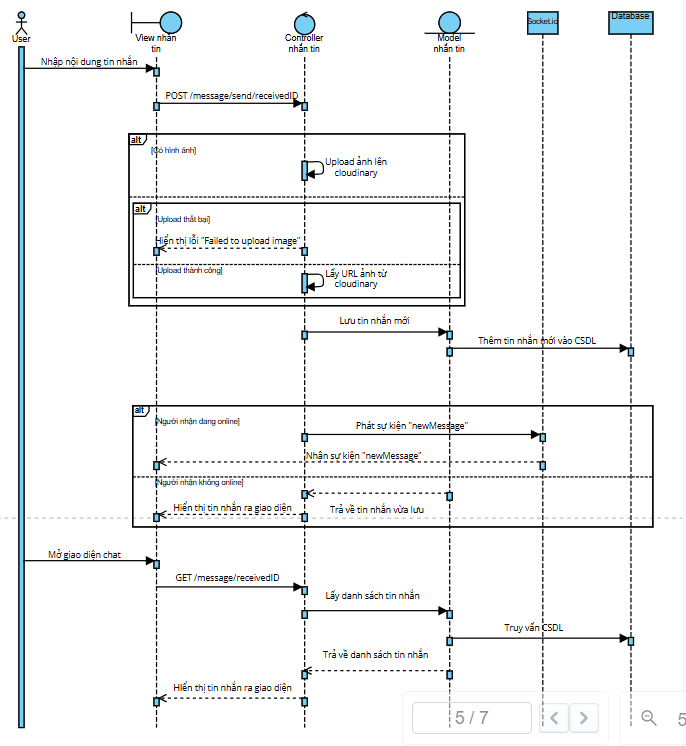
### 3.3.2 Đăng nhập:



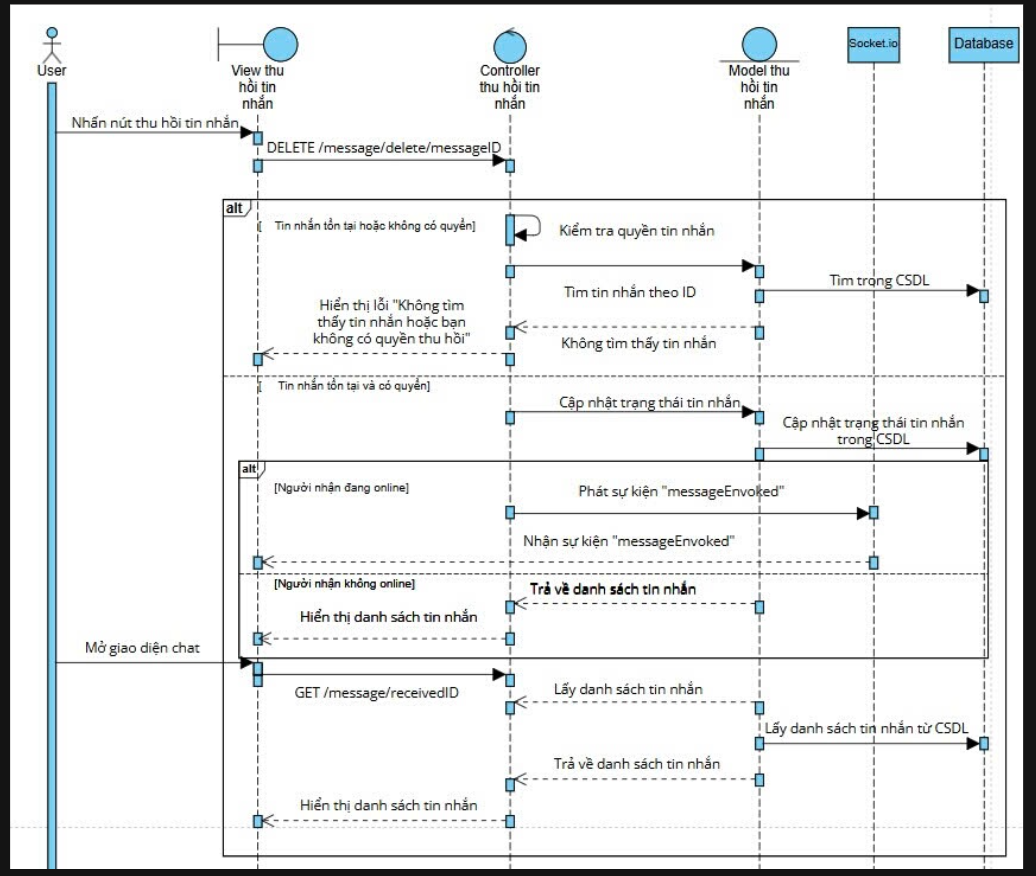
### 3.3.3 Chỉnh sửa profile:



### 3.3.4 Gửi và nhận tin nhắn:



### 3.3.5 Thu hồi tin nhắn:



**3.4.** **Thiết kế lớp (Class Diagram):**

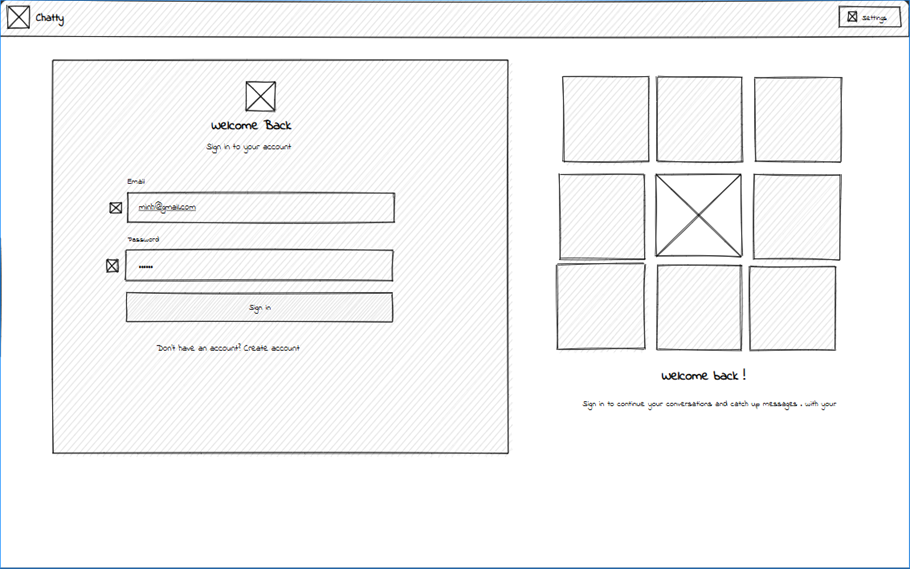
A screenshot of a computer

AI-generated content may be incorrect.

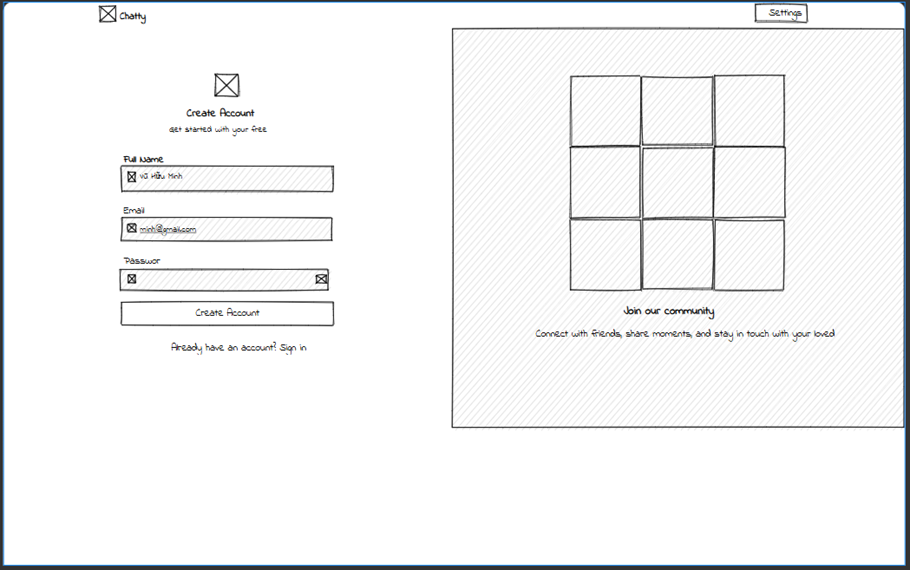
## 3.5. Thiết kế giao diện người dùng (Mockup / Wireframe):

### 3.5.1. Thiết kế giao diện người dùng: Wireframe

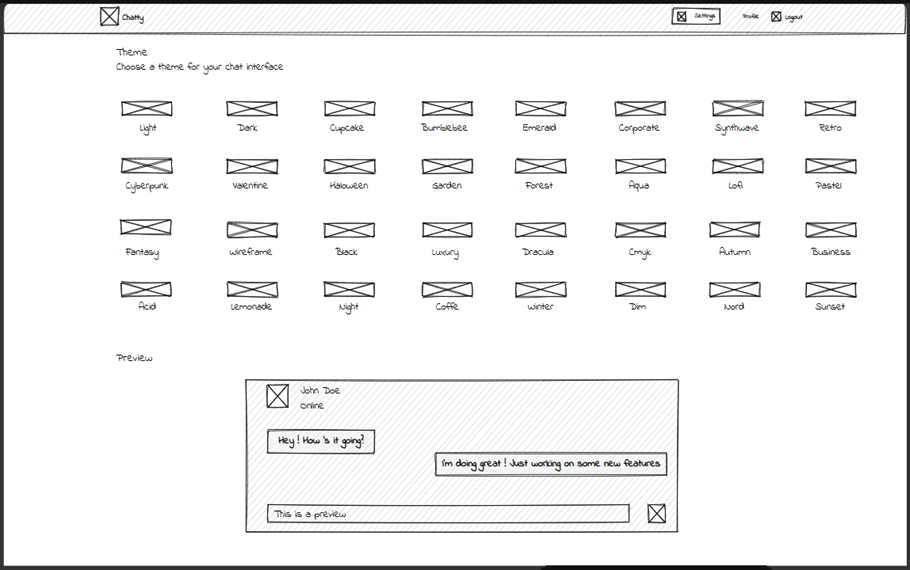
1. Giao diện đăng nhập



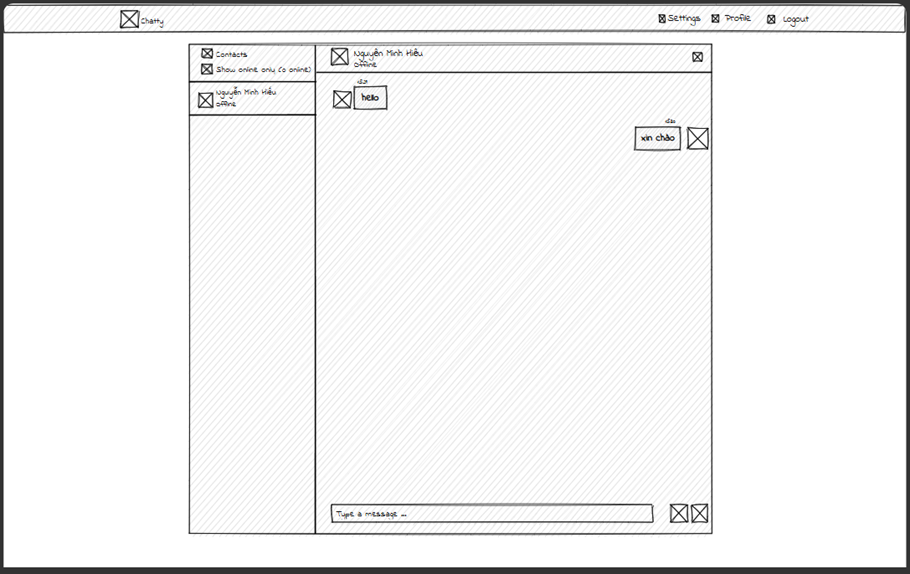
1. Giao diện đăng ký



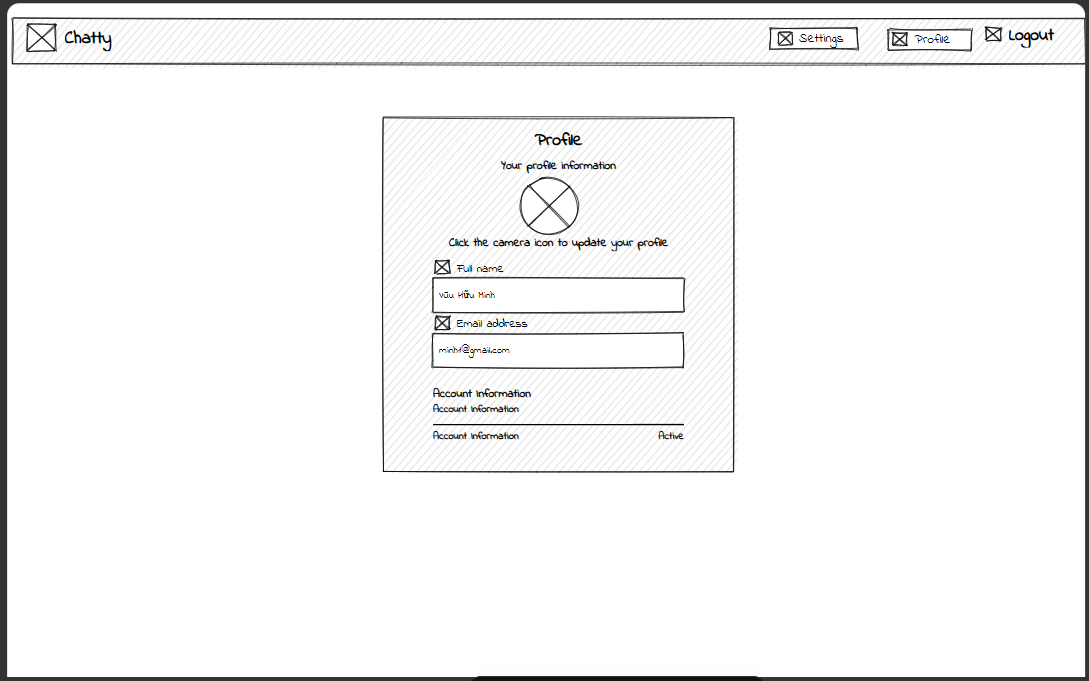
1. Giao diện setting đổi theme



1. Giao diện nhắn tin

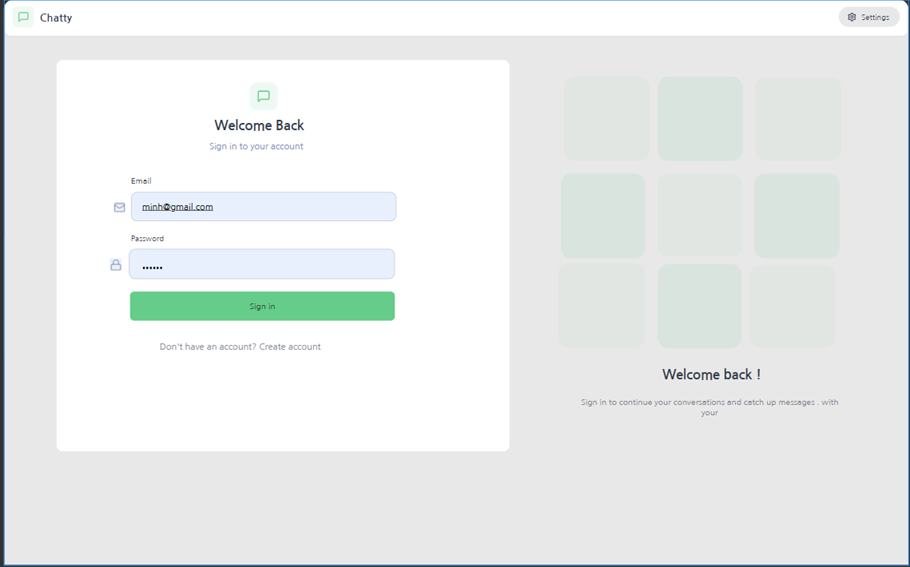


1. Giao diện cập nhật thông tin cá nhân người dùng

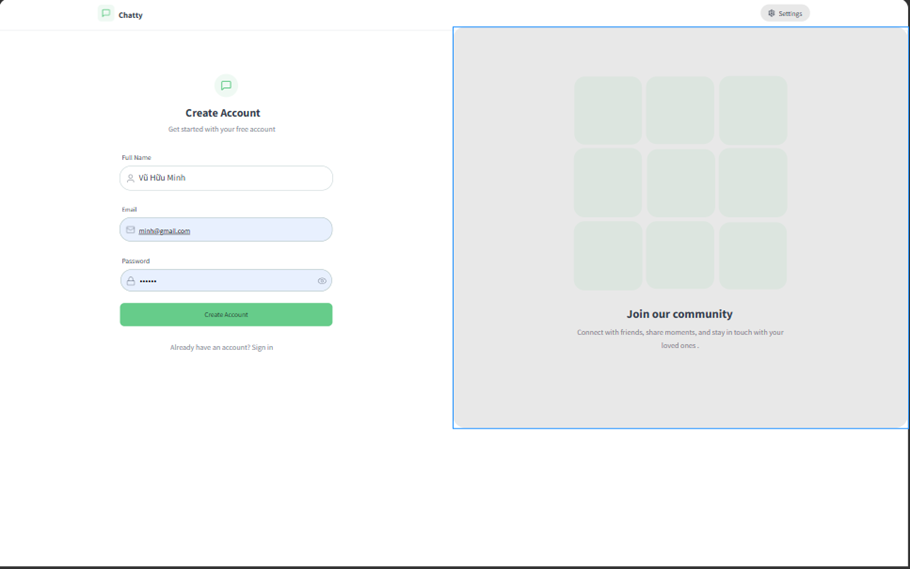


### 3.5.2 Thiết kế giao diện người dùng: Mockup

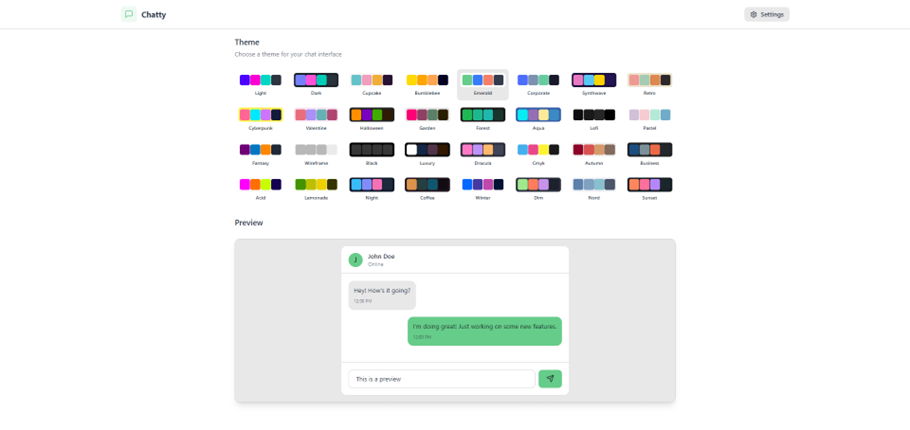
1. Giao diện đăng nhập



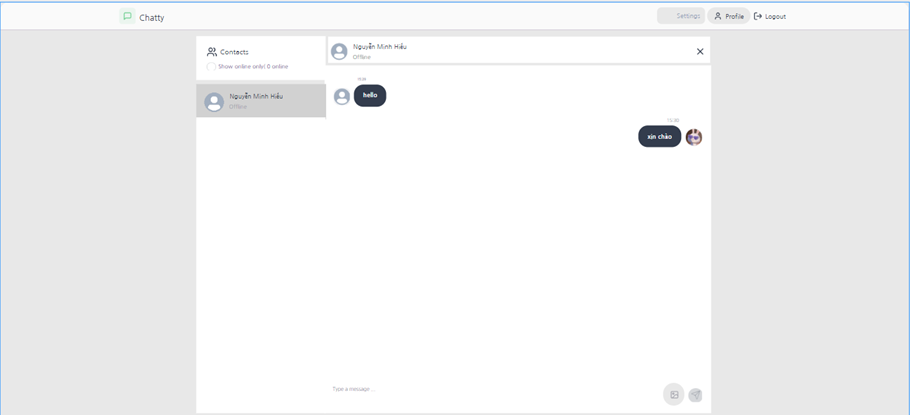
1. Giao diện đăng ký



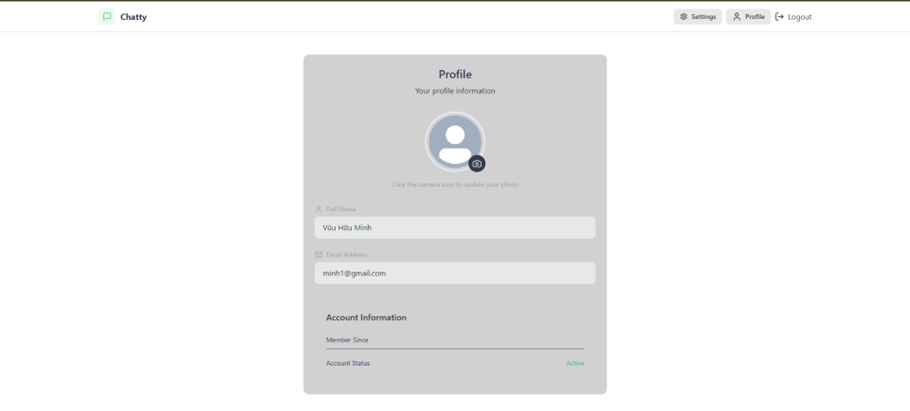
1. Giao diện setting đổi theme



1. Giao diện nhắn tin



1. Giao diện cập nhật thông tin người dùng



# CHƯƠNG 4. CÀI ĐẶT VÀ TRIỂN KHAI HỆ THỐNG

## 4.1. Môi trường triển khai

Hệ thống được phát triển và triển khai trong môi trường như sau:

* **Hệ điều hành:** Windows 10
* **Trình duyệt:** Google Chrome, Cốc cốc
* **Trình quản lý mã nguồn:** Git + GitHub
* **Trình soạn thảo mã:** Visual Studio Code
* **Cơ sở dữ liệu:** MongoDB (MongoDB Atlas)
* **Server Backend:** Node.js (v18+), Express.js
* **Frontend:** ReactJs với Vite, Tailwind CSS
* **Real-Time Communication:** Socket.IO
* **Quản lý trạng thái:** Zustand Store
* **Lưu trữ ảnh:** Cloudinary (sử dụng API miễn phí)
* **Triển khai thử nghiệm:** Localhost

## 4.2. Cài đặt hệ thống:

### 4.2.1 Yêu cầu hệ thống:

**Phần mềm:**

* Node.js (v18 trở lên)
* MongoDB (MongoDB Atlas)
* Trình duyệt hiện đại (Chrome/Cốc Cốc)
* Công cụ quản lý gói: npm hoặc yarn
* IDE: Visual Studio Code

**Phần cứng:**

* CPU tối thiểu: Intel Core i5 hoặc tương đương
* RAM: từ 4GB trở lên
* Dung lượng ổ đĩa còn trống: tối thiểu 1GB

### 4.2.2 Các bước cài đặt chương trình

Bước 1: Cài đặt môi trường:

* Cài đặt Node.js từ nodejs.org.
* Cài đặt MongoDB từ mongodb.com.
* Cài đặt Git từ git-scm.com.

Bước 2: Clone mã nguồn:

* Mở terminal và chạy lệnh:

git clone https://github.com/BuiTrung1710/Real-Time-ChatApp.git

cd real-time-chat-app

Bước 3: Cài đặt backend

Di chuyển vào thư mục backend:

* cd backend

Cài đặt các package:

* npm install

Tạo file .env và cấu hình:

MONGO\_URI = …….

CLOUDINARY\_NAME=your\_cloudinary\_name

CLOUDINARY\_API\_KEY=your\_api\_key

CLOUDINARY\_API\_SECRET=your\_api\_secret

JWT\_SECRET=your\_jwt\_secret

Chạy server backend:

* npm run dev

Bước 4: Cài đặt frontend:

Di chuyển vào thư mục frontend:

* cd frontend

Cài đặt các package:

* npm install

Chạy ứng dụng frontend:

* npm run dev

Bước 5: Kiểm tra hệ thống:

Mở trình duyệt và truy cập:

* Frontend: [http://localhos](http://localhost:3000)t:5174.
* Backend API: <http://localhost:5001/api>.

# CHƯƠNG 5. KẾT QUẢ VÀ ĐÁNH GIÁ

## 5.1. Kết quả thử nghiệm hệ thống

* Đăng ký và đăng nhập:
  + Người dùng có thể đăng ký tài khoản mới và đăng nhập thành công.
  + Hệ thống kiểm tra tính hợp lệ của email và mật khẩu trước khi cho phép truy cập.
* Tự động kết bạn:
  + Khi người dùng đăng nhập lần đầu, hệ thống tự động kết bạn với tất cả người dùng khác để thuận tiện nhắn tin.
* Nhắn tin thời gian thực:
  + Tin nhắn văn bản được gửi và nhận ngay lập tức nhờ sử dụng [Socket.IO](http://socket.io).
  + Tin nhắn được lưu trữ đầy đủ trong cơ sở dữ liệu và hiển thị chính xác ở cả hai phía.
* Gửi và nhận hình ảnh:
  + Người dùng có thể gửi hình ảnh qua giao diện chat.
  + Hình ảnh được lưu trữ và quản lý thông qua Cloudinary, đảm bảo truy xuất nhanh và ổn định.
* Thu hồi tin nhắn:
  + Người gửi có thể thu hồi tin nhắn đã gửi.
  + Trạng thái tin nhắn được cập nhật và phản ánh rõ trên cả hai giao diện người gửi và người nhận.
* Cập nhật thông tin cá nhân:
  + Cho phép thay đổi avatar của người dùng.
* Giao diện người dùng:
  + Giao diện gọn gàng dễ nhìn, thuận tiện cho người dùng
  + Hỗ trợ chuyển đổi theme đoạn chat, ghi nhớ theme người dùng đã chọn.
* Hiển thị trạng thái hoạt động:
  + Trạng thái online/offline của người dùng được hiển thị chính xác dựa trên kết nối socket.
* Hiệu suất và độ ổn định:
  + Tin nhắn được xử lý nhanh với độ trễ thấp.
  + Hệ thống hoạt động ổn định khi thử nghiệm với 10 người dùng đồng thời, không xuất hiện lỗi hay gián đoạn.

## 5.2. Đánh giá hiệu quả hệ thống

Hệ thống đáp ứng được các mục tiêu đặt ra ban đầu đối với một ứng dụng nhắn tin cơ bản. Giao diện dễ sử dụng, tốc độ phản hồi tin nhắn nhanh, và đảm bảo trải nghiệm người dùng tốt trên nhiều trình duyệt. Tuy nhiên, do giới hạn tài nguyên và mục đích thử nghiệm, hệ thống vẫn còn một số điểm cần cải thiện:

* Chưa hỗ trợ nhóm chat hoặc gọi video.
* Không có thông báo đẩy hoặc tính năng tìm kiếm tin nhắn.
* Khả năng mở rộng và bảo mật chưa được tối ưu cho môi trường sản phẩm thực tế.

Nhìn chung, hệ thống phù hợp để làm nền tảng cho việc mở rộng thành một ứng dụng chat hoàn chỉnh trong tương lai.

# Tài liệu tham khảo

<https://chatgpt.com/>

<https://www.cursor.com/>