TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ I, NĂM HỌC 2025-2026

XÂY DỰNG WEB CHAT REAL TIME SỬ DỤNG ANGULAR VÀ SPRING BOOT

Giảng viên hướng dẫn

ThS. Ngô Thanh Huy

Sinh viên thực hiện
Họ tên:Võ Chí Hải
MSSV:110122068
Lớp:DA22TTD

Vĩnh Long, tháng 12, năm 2025

TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ I, NĂM HỌC 2025-2026

XÂY DỰNG WEB CHAT REAL TIME SỬ DỤNG ANGULAR VÀ SPRING BOOT

Giảng viên hướng dẫn

ThS. Ngô Thanh Huy

Sinh viên thực hiện
Họ tên:Võ Chí Hải
MSSV:110122068
Lớp:DA22TTD

Vĩnh Long, tháng 12, năm 2025

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Vĩnh Long, ngày ..... tháng …… năm ……

Giảng viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Vĩnh Long, ngày ..... tháng …… năm ……

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Để hoàn thành bài đồ án này, em xin gửi lời cảm ơn chân thành đến Quý Thầy, Cô thuộc Khoa Công nghệ thông tin, Trường Kỹ thuật và Công nghệ – Trường Đại học Trà Vinh đã tạo điều kiện thuận lợi để em được học tập, rèn luyện và tích lũy kiến thức, kỹ năng nhằm thực hiện đò án này.

Đặc biệt, em xin gửi lời cảm ơn đến thầy Ngô Thanh Huy đã tận tình chỉ dẫn, theo dõi và đưa ra những lời khuyên bổ ích giúp em giải quyết được các vấn đề gặp phải trong quá trình nghiên cứu và hoàn thành đề tài một cách tốt nhất.

Do kiến thức của bản thân còn hạn chế và thiếu kinh nghiệm thực tiễn nên nội dung bài báo cáo khó tránh những thiếu sót. Nhóm chúng em rất mong nhận được sự góp ý, chỉ dạy thêm từ Quý Thầy cô.

Cuối cùng, em xin chúc Quý Thầy Cô luôn thật nhiều sức khỏe và đạt được nhiều thành công trong công việc.

Mục lục

DANH MỤC HÌNH ẢNH VÀ BẢNG BIỂU

CHƯƠNG 1. TỔNG QUAN

1.1. Giới thiệu đề tài

Trong bối cảnh công nghệ 4.0 phát triển mạnh mẽ, nhu cầu trao đổi thông tin nhanh chóng, linh hoạt và thuận tiện trở thành một phần quan trọng trong đời sống số. Các nền tảng giao tiếp trực tuyến không chỉ được sử dụng trong trò chuyện cá nhân mà còn xuất hiện phổ biến trong môi trường doanh nghiệp, giáo dục và dịch vụ. Điều này đặt ra yêu cầu về những hệ thống chat có khả năng hoạt động ổn định, thời gian phản hồi thấp và mang lại trải nghiệm người dùng tốt. Xuất phát từ nhu cầu đó, với đề tài "Xây dựng web chat real time sử dụng Angular và Spring Boot" được thực hiện với mục tiêu xây dựng một ứng dụng web chat hoạt động theo thời gian thực, giao diện thân thiện, dễ sử dụng và có khả năng mở rộng trong tương lai. Bên cạnh đó, đề tài còn tạo cơ hội giúp người học củng cố kỹ năng lập trình web full-stack, làm quen với các công nghệ hiện đại và nắm vững quy trình xây dựng một hệ thống hoạt động theo mô hình client–server kết hợp xử lý dữ liệu thời gian thực.

1.2. Mục đích nghiên cứu

Mục tiêu chính của đề tài là xây dựng một ứng dụng web chat đáp ứng khả năng gửi và nhận tin nhắn ngay lập tức thông qua WebSocket. Đồng thời, đề tài tập trung vào việc ứng dụng các công nghệ tiêu biểu như Spring Boot cho backend, Angular cho frontend, SCSS cho giao diện, MySQL để lưu trữ dữ liệu và Redis dùng cho cache, session và hàng đợi tin nhắn. Ngoài ra, việc triển khai REST API kết hợp WebSocket giúp người học nắm rõ sự khác biệt giữa truyền dữ liệu đồng bộ và bất đồng bộ, đồng thời tăng khả năng thực hành lập trình full-stack trong một môi trường thực tế.

1.3. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài bao gồm các kỹ thuật, công nghệ và mô hình kiến trúc phục vụ xây dựng hệ thống chat thời gian thực. Cụ thể là kiến trúc client–server trong ứng dụng web, cơ chế hoạt động của WebSocket và khả năng truyền dữ liệu hai chiều giữa máy chủ và trình duyệt, phương pháp quản lý thông tin người dùng

và tin nhắn thông qua MySQL, cơ chế caching và message queue bằng Redis để tối ưu hiệu suất, và quá trình tích hợp frontend Angular với backend Spring Boot nhằm tạo nên một hệ thống vận hành liền mạch. Việc nghiên cứu các thành phần này giúp đảm bảo hệ thống đạt được tính ổn định, khả năng mở rộng và tốc độ phản hồi cần thiết.

1.4. Phạm vi nghiên cứu

Phạm vi nghiên cứu tập trung vào việc xây dựng một ứng dụng trò chuyện thời gian thực giữa các người dùng đã đăng ký. Ứng dụng không đi sâu vào các tính năng nâng cao như gửi file dung lượng lớn, cuộc gọi video, chat nhóm phức tạp, mã hóa đầu cuối hay tích hợp trí tuệ nhân tạo. Hệ thống được triển khai và kiểm thử trên môi trường local và server nội bộ nhằm đảm bảo tính ổn định và phù hợp với điều kiện thử nghiệm. Giao diện được thiết kế theo hướng trực quan, thân thiện với người dùng.

1.5. Phương pháp thực hiện

Để hoàn thành đề tài, nhiều phương pháp nghiên cứu được kết hợp nhằm đảm bảo tính khoa học và hiệu quả. Phương pháp nghiên cứu lý thuyết được áp dụng để tìm hiểu các tài liệu, tài nguyên liên quan đến WebSocket, Angular, Spring Boot, Redis và các mô hình kiến trúc hỗ trợ ứng dụng thời gian thực. Bên cạnh đó, phương pháp thực nghiệm được sử dụng thông qua việc xây dựng mô hình hệ thống, lập trình từng chức năng, triển khai và kiểm thử thực tế. Phương pháp so sánh và đánh giá được áp dụng nhằm đo lường hiệu năng hệ thống, đánh giá độ trễ phản hồi giữa trường hợp có sử dụng Redis và không sử dụng Redis. Cuối cùng là phương pháp tổng hợp nhằm phân tích kết quả, đánh giá ưu điểm – hạn chế của hệ thống và đề xuất hướng phát triển trong tương lai.

1.6. Kết quả đạt được

Kết quả của quá trình nghiên cứu và triển khai cho thấy hệ thống web chat đã được xây dựng thành công, đáp ứng khả năng truyền nhận tin nhắn thời gian thực với giao diện trực quan, dễ sử dụng. Việc tích hợp WebSocket giúp cải thiện tốc độ phản hồi, trong khi Redis hỗ trợ giảm tải cho cơ sở dữ liệu và đảm bảo dữ liệu tin nhắn được xử lý ổn định ngay cả khi có nhiều người dùng hoạt động đồng thời. Hệ thống hỗ trợ đầy đủ chức năng đăng ký, đăng nhập, xác thực người dùng, quản lý dữ liệu tin nhắn và hoạt động liên tục trên môi trường thử nghiệm. Kết quả cho thấy việc kết hợp

Spring Boot và Angular hoàn toàn phù hợp để phát triển các hệ thống web hiện đại có yêu cầu cao về tốc độ và tính ổn định, đồng thời mở ra nhiều hướng phát triển mở rộng trong tương lai.

Ngoài các công nghệ chính được sử dụng trong quá trình xây dựng hệ thống, một số công cụ hỗ trợ phát triển cũng đóng vai trò quan trọng trong việc đảm bảo tiến độ, chất lượng mã nguồn và khả năng trực quan hóa mô hình . **Visual Studio Code** được sử dụng làm môi trường lập trình cho phần frontend, hỗ trợ tốt cho Angular nhờ hệ thống tiện ích mở rộng phong phú, khả năng kiểm tra lỗi theo thời gian thực và giao diện trực quan. Đối với **backend**, **IntelliJ IDEA** là công cụ mạnh mẽ dành cho Java, hỗ trợ tự động gợi ý mã, quản lý cấu trúc dự án Spring Boot và tích hợp tốt với Maven cũng như các plugin hỗ trợ WebSocket. Trong quá trình phát triển và chạy thử cơ sở dữ liệu trên môi trường local, **XAMPP** được sử dụng để khởi tạo **MySQL** Server, giúp việc tạo bảng, kiểm tra kết nối và quản lý dữ liệu trở nên thuận tiện hơn. Bên cạnh đó, Draw.io được dùng để xây dựng các sơ đồ kiến trúc, mô hình dữ liệu và luồng xử lý, hỗ trợ trình bày báo cáo một cách khoa học và dễ hiểu. Cuối cùng **Postman** đóng vai trò là công cụ kiểm thử **API** quan trọng, cho phép gửi yêu cầu đến các **endpoint** của **Spring Boot**, kiểm tra phản hồi, đánh giá tính đúng đắn của dữ liệu và đảm bảo hệ thống hoạt động ổn định trước khi tích hợp vào **frontend**. Sự kết hợp của các công cụ này giúp quá trình phát triển diễn ra hiệu quả, rõ ràng và giảm thiểu sai sót trong toàn bộ vòng đời thực hiện đề tài.

CHƯƠNG 2. NGHIÊN CỨU LÝ THUYẾT

2.2. Cơ sở lý thuyết về ứng dụng web thời gian thực

2.2.1. Khái niệm ứng dụng thời gian thực

Ứng dụng thời gian thực (Real-Time Application – RTA) là những hệ thống trong đó dữ liệu được tạo ra, truyền đi, xử lý và hiển thị gần như ngay lập tức, đồng bộ với hành động của người dùng. Điểm đặc trưng của ứng dụng thời gian thực là độ trễ cực thấp, thường chỉ tính bằng vài mili–giây, đảm bảo dữ liệu luôn phản ánh đúng trạng thái hiện tại của hệ thống.

Trong thực tế, mô hình ứng dụng thời gian thực được ứng dụng rộng rãi trong nhiều lĩnh vực như trò chuyện trực tuyến (chat online), theo dõi định vị GPS, hệ thống giao dịch tài chính, giám sát cảm biến IoT, trò chơi trực tuyến nhiều người chơi (multiplayer online games)… Các hệ thống này đều yêu cầu khả năng cập nhật tức thời để đảm bảo trải nghiệm liên tục và chính xác.

Đối với ứng dụng chat – trường hợp được đề cập trong đề tài – khi một người dùng gửi tin nhắn, dữ liệu phải ngay lập tức được truyền tới máy chủ, xử lý và chuyển tiếp đến người nhận. Nếu sử dụng mô hình truyền thống dựa trên HTTP request–response, tốc độ phản hồi sẽ bị giới hạn và người dùng thường phải tải lại trang hoặc đợi một yêu cầu mới để nhận tin nhắn. Vì vậy, các công nghệ hỗ trợ truyền dữ liệu thời gian thực như WebSocket trở nên cần thiết để đảm bảo quá trình trao đổi thông tin luôn nhanh chóng và liền mạch.

2.2.2. Cơ chế hoạt động của WebSocket

WebSocket là một giao thức truyền thông hiện đại được thiết kế nhằm cho phép thiết lập kết nối hai chiều (full-duplex) giữa client và server thông qua một TCP socket duy nhất. Khác với HTTP truyền thống vốn hoạt động theo chu kỳ request–response, WebSocket tạo ra một kết nối liên tục và ổn định, giúp server có thể chủ động gửi dữ liệu về client bất cứ lúc nào mà không cần yêu cầu mới.

Quá trình hoạt động của WebSocket bắt đầu bằng một yêu cầu “bắt tay” (handshake) dựa trên HTTP. Sau khi handshake thành công, kết nối được nâng cấp từ HTTP lên

WebSocket và trở thành một kênh truyền thông toàn song công. Điều này mang lại hai lợi ích lớn:

Giảm độ trễ truyền dữ liệu vì không phải thiết lập kết nối nhiều lần.

Tiết kiệm tài nguyên so với việc liên tục gửi yêu cầu HTTP.

WebSocket đặc biệt phù hợp với các ứng dụng yêu cầu cập nhật liên tục như trò chuyện trực tuyến, thông báo thời gian thực, giao dịch tài chính theo thời gian thực, bảng xếp hạng game, ứng dụng streaming hoặc các dashboard theo dõi dữ liệu.

Trong đề tài, WebSocket được tích hợp vào hệ thống Spring Boot nhằm duy trì kênh truyền tin ổn định giữa người dùng và server. Khi có tin nhắn mới được gửi, server ngay lập tức đẩy dữ liệu đến tất cả người dùng liên quan, đảm bảo việc trao đổi tin nhắn diễn ra tức thời mà không chịu độ trễ đáng kể.

2.3. Kiến trúc tổng thể của hệ thống web chat

Hệ thống web chat được thiết kế dựa trên mô hình N-tier (đa tầng), chia làm ba lớp chính nhằm tối ưu khả năng mở rộng, dễ bảo trì và đảm bảo sự tách biệt giữa giao diện, logic nghiệp vụ và dữ liệu.

Frontend (client):

Lớp giao diện được xây dựng bằng Angular kết hợp với SCSS để tạo nên giao diện trực quan, hiện đại và thân thiện với người dùng. Angular giúp quản lý trạng thái ứng dụng hiệu quả, hỗ trợ đồng bộ dữ liệu thời gian thực và tối ưu trải nghiệm người dùng thông qua cơ chế component-based. Phần frontend chịu trách nhiệm hiển thị danh sách người dùng, danh sách tin nhắn, xử lý sự kiện gửi/nhận tin và kết nối đến WebSocket từ server.

Backend (server):

Backend được phát triển bằng Spring Boot, cung cấp các API REST phục vụ việc đăng nhập, đăng ký, quản lý người dùng và lưu trữ tin nhắn. Đồng thời, backend là nơi thiết lập endpoint WebSocket và xử lý logic chuyển tiếp tin nhắn giữa các client. Spring Boot mang lại khả năng mở rộng, dễ cấu hình và

tích hợp tốt với các công nghệ quản lý kết nối thời gian thực như STOMP hoặc SockJS.

Kiến trúc tổng thể hướng N-tier của hệ thống giúp tăng tính linh hoạt, dễ dàng mở rộng trong tương lai. Nếu cần, hệ thống có thể tích hợp thêm tầng cơ sở dữ liệu riêng biệt, tầng bảo mật, hoặc triển khai trên các dịch vụ cloud để đáp ứng số lượng người dùng lớn mà không ảnh hưởng đến hiệu năng.

Bên cạnh đó, việc sử dụng WebSocket trong các hệ thống web hiện đại còn mang lại nhiều lợi ích về mặt kiến trúc và hiệu năng. Nhờ khả năng truyền tải dữ liệu liên tục, WebSocket giúp giảm tải đáng kể cho máy chủ so với các kỹ thuật truyền thống như polling hoặc long-polling, vốn buộc server phải xử lý nhiều yêu cầu dư thừa. Ngoài ra, WebSocket hỗ trợ truyền dữ liệu dưới dạng khung (frame) với kích thước nhỏ và linh hoạt, giúp tối ưu băng thông và hạn chế độ trễ. Điều này đặc biệt quan trọng đối với các ứng dụng có lưu lượng giao tiếp cao hoặc cần phản hồi theo thời gian thực, ví dụ như hệ thống đặt xe, hỗ trợ khách hàng trực tuyến, hoặc nền tảng họp video. Nhờ sự ổn định của kết nối hai chiều, lập trình viên cũng dễ dàng quản lý trạng thái phiên làm việc của người dùng cũng như triển khai các tính năng nâng cao như thông báo trạng thái "đang nhập tin nhắn", "đã xem", hoặc đồng bộ hóa dữ liệu đa thiết bị. Tất cả những yếu tố này góp phần làm cho WebSocket trở thành công nghệ cốt lõi trong phát triển các ứng dụng web tương tác mạnh mẽ và yêu cầu tốc độ phản hồi tức thời.

2.4. Công nghệ sử dụng

2.4.1. Spring Boot

Spring Boot là một trong những framework mạnh mẽ và phổ biến nhất dành cho lập trình Java hiện đại, hỗ trợ xây dựng các ứng dụng web có tính ổn định, hiệu năng cao và dễ triển khai. Spring Boot giúp đơn giản hóa quá trình cấu hình bằng cách cung cấp cơ chế auto–configuration, cho phép lập trình viên tập trung vào logic nghiệp vụ thay vì phải xử lý cấu hình phức tạp. Framework này tích hợp sẵn các máy chủ như Tomcat hoặc Jetty, hỗ trợ đầy đủ Spring Data JPA/Hibernate, WebSocket và các module mở rộng khác. Ngoài ra, Spring Boot sử dụng Maven hoặc Gradle để quản lý dependencies giúp dự án có cấu trúc rõ ràng và dễ bảo trì. Spring Boot cũng dễ dàng tích hợp với Redis, MySQL, Docker và các nền tảng triển khai hiện đại. Trong đề tài,

Spring Boot đóng vai trò xương sống của hệ thống, đảm nhiệm xử lý logic nghiệp vụ, xác thực người dùng, quản lý dữ liệu và truyền – nhận tin nhắn thông qua WebSocket kết hợp MySQL.

2.4.2. Angular và SCSS

Angular là framework frontend do Google phát triển, hoạt động dựa trên kiến trúc Component–Based giúp tổ chức mã nguồn khoa học, tái sử dụng cao và dễ dàng mở rộng khi hệ thống phát triển. Angular hỗ trợ mạnh mẽ cơ chế xử lý hai chiều (two-way binding), routing, quản lý trạng thái và tối ưu hiệu suất hiển thị. SCSS (Sassy CSS) là phần mở rộng của CSS cho phép sử dụng biến, vòng lặp, mixin, kế thừa và nhiều kỹ thuật nâng cao khác giúp việc xây dựng giao diện trở nên dễ dàng, logic và thống nhất hơn. Trong đề tài, Angular đảm nhiệm toàn bộ phần hiển thị giao diện, xử lý các sự kiện gửi – nhận tin nhắn theo thời gian thực thông qua WebSocket và cập nhật dữ liệu tức thì trên màn hình mà không cần tải lại trang. SCSS hỗ trợ xây dựng giao diện thân thiện, trực quan, hiện đại và dễ tùy chỉnh, đảm bảo trải nghiệm người dùng mượt mà trong quá trình trò chuyện.

2.4.3. MySQL

MySQL là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) phổ biến nhờ tính ổn định, tốc độ cao và khả năng xử lý tốt cho các ứng dụng có quy mô nhỏ đến trung bình. Dữ liệu được tổ chức dưới dạng bảng có mối quan hệ rõ ràng, hỗ trợ đầy đủ khóa chính, khóa ngoại, chỉ mục và các cơ chế giúp tối ưu truy vấn. MySQL dễ dàng tích hợp với Spring Data JPA/Hibernate nên rất phù hợp cho các hệ thống sử dụng Spring Boot. Trong ứng dụng chat, MySQL dùng để lưu trữ thông tin người dùng (username, mật khẩu đã mã hóa), lịch sử tin nhắn kèm thời gian gửi, người gửi – người nhận và dữ liệu về phòng chat cũng như các thành viên đang tham gia. Việc lưu trữ tập trung đảm bảo tính nhất quán và khả năng truy xuất dữ liệu khi cần thiết.

2.4.4. Redis

Redis là cơ sở dữ liệu dạng key–value hoạt động trực tiếp trên bộ nhớ RAM, nổi tiếng với tốc độ truy xuất cực nhanh và phù hợp cho các tác vụ thời gian thực. Redis thường được dùng như cache, hệ thống lưu trữ tạm hoặc message queue. Trong đề tài, Redis đảm nhiệm ba vai trò chính: Cache dữ liệu để giảm tải cho MySQL khi

truy xuất những thông tin được dùng thường xuyên, quản lý session người dùng giúp duy trì trạng thái đăng nhập ổn định trong quá trình sử dụng, hỗ trợ xử lý hàng đợi tin nhắn nhằm đảm bảo tin nhắn không bị mất khi nhiều người dùng hoạt động đồng thời.

2.4.5. Swagger

Swagger là bộ công cụ hỗ trợ xây dựng, mô tả và trực quan hóa API, giúp việc kiểm thử và phát triển backend trở nên đơn giản, rõ ràng và chuyên nghiệp hơn. Swagger cho phép tự động tạo tài liệu API dựa trên các annotation trong mã nguồn, giúp lập trình viên không cần viết tài liệu thủ công. Giao diện trực quan của Swagger UI giúp người dùng có thể xem toàn bộ các endpoint, thông số đầu vào, kiểu dữ liệu trả về và thử nghiệm trực tiếp các API ngay trên trình duyệt mà không cần dùng các công cụ ngoài như Postman. Trong đề tài, Swagger được sử dụng để mô tả các API liên quan đến đăng nhập, đăng ký, quản lý người dùng và xử lý tin nhắn. Việc tích hợp Swagger không chỉ giúp nhóm phát triển kiểm thử nhanh hơn mà còn hỗ trợ quá trình bảo trì, mở rộng hệ thống về sau. Đồng thời, nhờ tài liệu API được cập nhật tự động, việc phối hợp giữa các thành viên trong nhóm trở nên thuận tiện và hạn chế tối đa sai sót do hiểu sai yêu cầu hoặc cấu trúc dữ liệu.

2.4.6. Docker

Docker là công nghệ container hóa giúp đóng gói ứng dụng cùng toàn bộ môi trường, thư viện và cấu hình cần thiết vào các container độc lập, đảm bảo khả năng triển khai đồng nhất trên mọi nền tảng. Docker mang lại nhiều ưu điểm như môi trường chạy nhất quán, khả năng mở rộng và nhân bản hệ thống nhanh chóng, tối ưu tài nguyên và giảm chi phí triển khai. Trong đề tài, Docker được sử dụng để đóng gói backend (Spring Boot), frontend (Angular) và Redis thành các container riêng biệt, giúp dễ dàng triển khai trên Render và Vercel mà không lo vấn đề môi trường giữa các máy khác nhau.

2.4.7. GitHub Actions

GitHub Actions là công cụ tự động hóa quy trình CI/CD (Continuous Integration/Continuous Deployment), cho phép kiểm thử, build và triển khai ứng dụng mỗi khi có thay đổi trên repository. Trong giai đoạn CI, mã nguồn được kiểm tra, build và phân tích tự động để đảm bảo không phát sinh lỗi trước khi hợp nhất vào nhánh

chính. Trong giai đoạn CD, hệ thống sẽ tự động triển khai phiên bản mới lên môi trường thực tế nếu quá trình kiểm thử thành công. Trong đề tài, nhóm đã thiết lập pipeline CI/CD để tự động build Docker image sau mỗi commit, phân tích chất lượng mã bằng SonarQube Cloud và triển khai frontend lên Vercel, backend lên Render. Quy trình này giúp việc phát triển trở nên ổn định, nhanh chóng và giảm thiểu lỗi con người.

CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Mô tả bài toán:

Trong bối cảnh các hệ thống trao đổi thông tin ngày càng yêu cầu tốc độ và tính liên tục, bài toán đặt ra là xây dựng một ứng dụng **chat real-time** cho phép người dùng có thể gửi và nhận tin nhắn ngay lập tức thông qua môi trường web.

Ứng dụng cần đáp ứng các yêu cầu:

Gửi tin nhắn theo thời gian thực

Lưu trữ tin nhắn để người dùng có thể xem lại các tin nhắn cũ

Để giải quyết bài toán, đề tài sử dụng:

Frontend: Angular, sử dụng STOMP để truyền tải tin nhắn.

Backend: Spring Boot, cung cấp API REST và WebSocket endpoint.

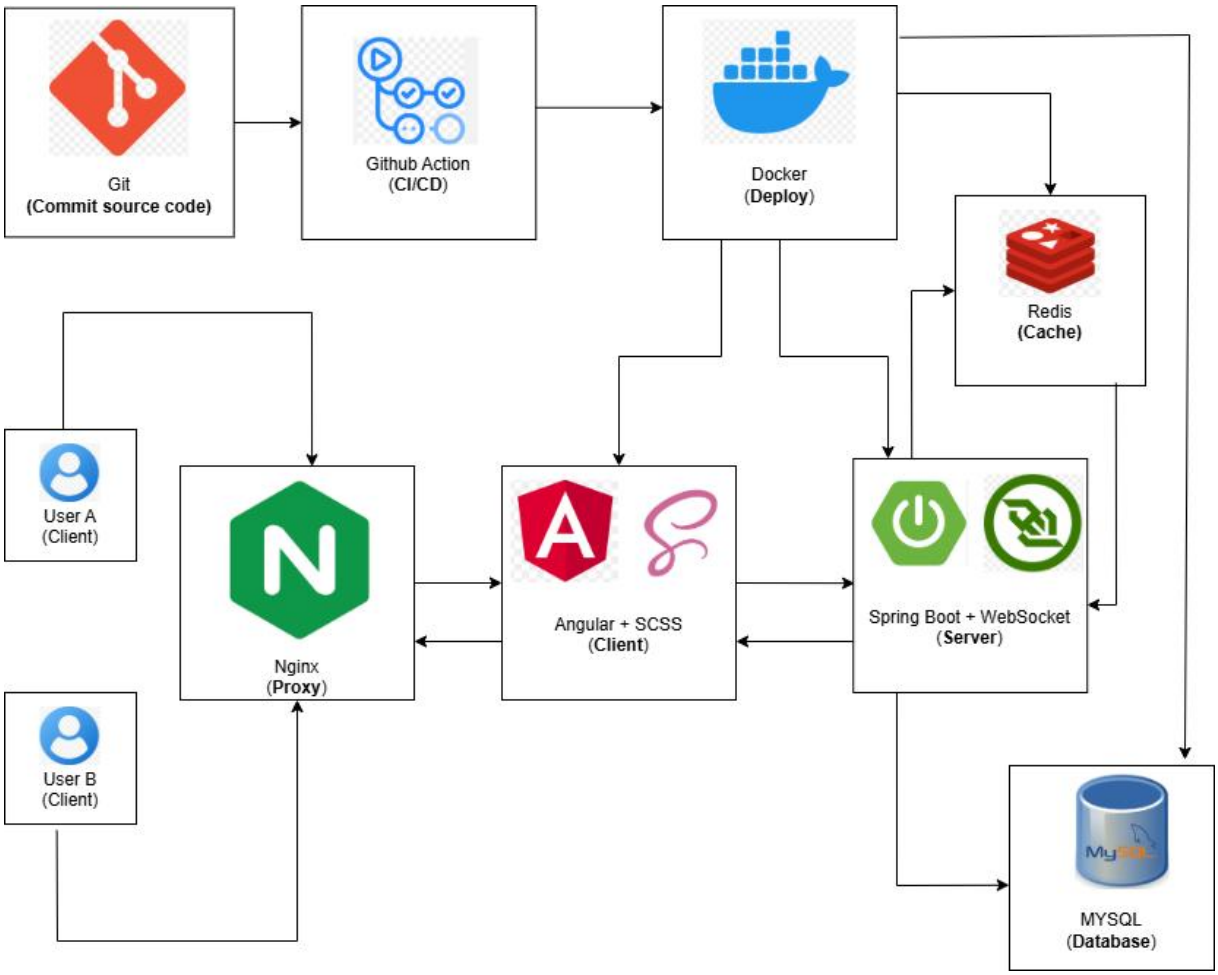
Database: MySQL để lưu trữ dữ liệu người dùng, phòng chat , tin nhắn.

3.2. Phân tích thiết kế hệ thống

Stt	Chức năng
1	Xác thực người dùng
2	Cập nhật avatar
3	Quản lý nhóm chat
4	Gửi tin nhắn
5	Chống spam tin nhắn
6	Khóa tài khoản người dùng
7	Chế độ sáng và tối
8	Đặt ảnh nền cho nhóm chat
9	Hiển thị trạng thái người dùng
10	Chọn màu chủ đề
11	Hiển thị danh sách các nhóm
12	Thông báo tin nhắn chưa xem

Bảng 1. Danh sách các chức năng

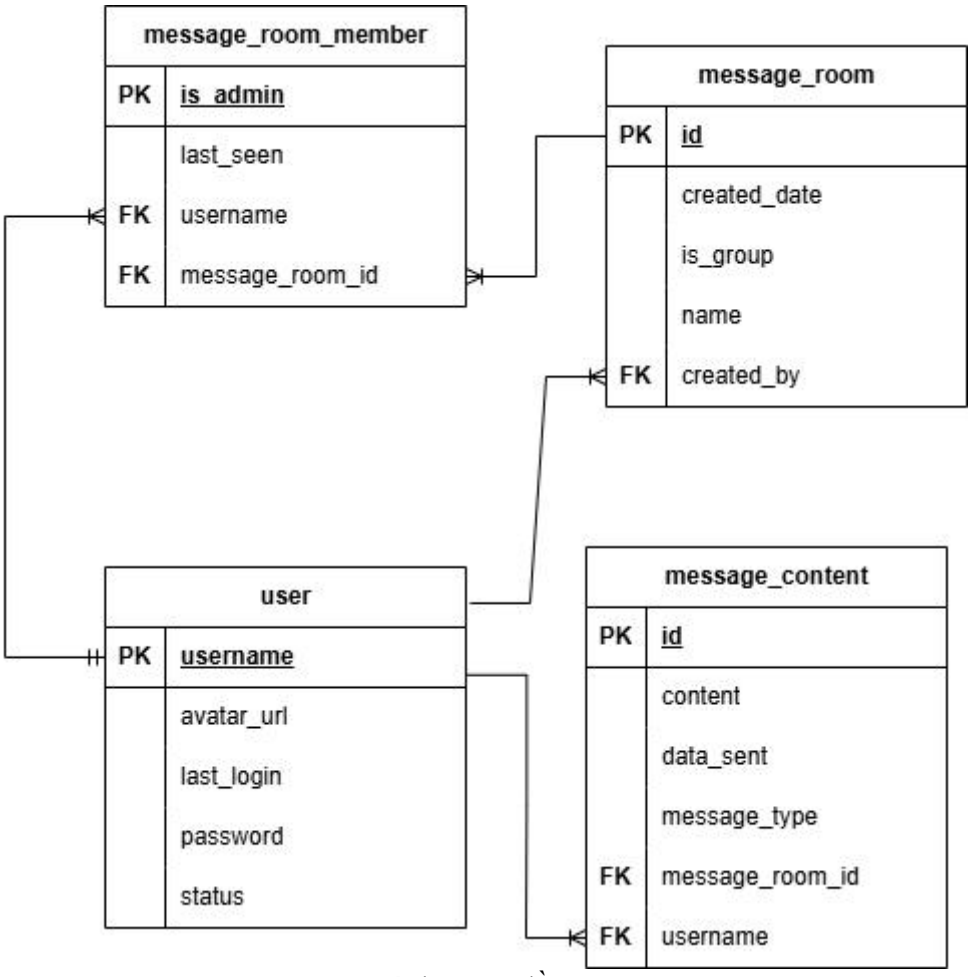
3.3. Kiến trúc hệ thống



Hình 1. Sơ đồ kiến trúc hệ thống

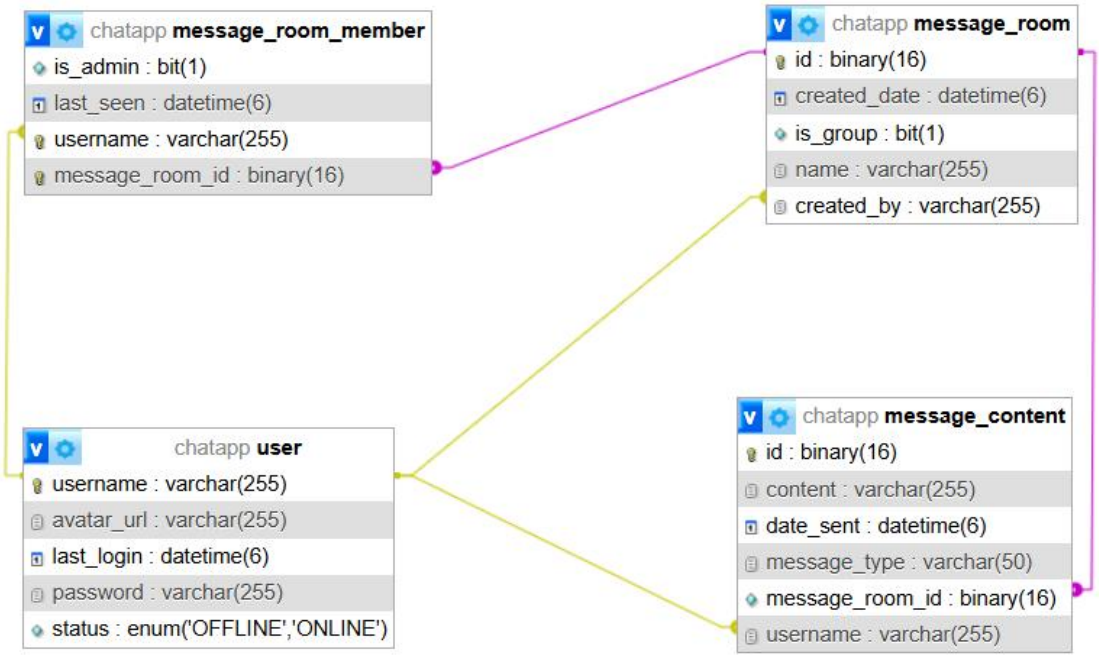
3.4. Thiết kế dữ liệu

3.4.1. Mô hình ERD



Hình 2. Sơ đồ ERD

3.4.2. Mô hình mức dữ liệu logic



Hình 3. Sơ đồ quan hệ

3.4.3. Danh sách chi tiết thực thể User

Stt	Tên thuộc tính	Kiểu dữ liệu	Diễn giải	Ràng buộc
1	username	varchar	Tên tài khoản người dùng	Khóa chính
2	avatar_url	varchar	Dường dẫn lưu file avatar	
3	last_login	datetime	Thời gian truy cập gần nhất	
4	password	varchar	mật khẩu của tài khoản	
5	status	enum	Trạng thái hoạt động	

Bảng 2. Thực thể User

3.4.4. Danh sách chi tiết thực thể MessageRoom

Stt	Tên thuộc tính	Kiểu dữ liệu	Diễn giải	Ràng buộc
1	id	binary	Id của nhóm chat	Khóa chính
2	created_date	datetime	Thời gian được tạo	
3	is_group	bit	Số thành viên trong nhóm	
4	name	varchar	Tên nhóm	
5	created_by	varchar	Người tạo nhóm	Khóa ngoại

Bảng 3. Thực thể MessageRoom

3.4.5. Danh sách chio tiết thực thể MessageRoomMember

Stt	Tên thuộc tính	Kiểu dữ liệu	Diễn giải	Ràng buộc
1	is_admin	bit	Quyền admin trong nhóm chat	
2	last_seen	datetime	Thời gian tin nhắn	
3	username	varchar	Tên người dùng	Khóa ngoại
4	message_room_id	binary	Id của nhóm chat	Khóa ngoại

Bảng 4. Thực thể MessageRoomMember

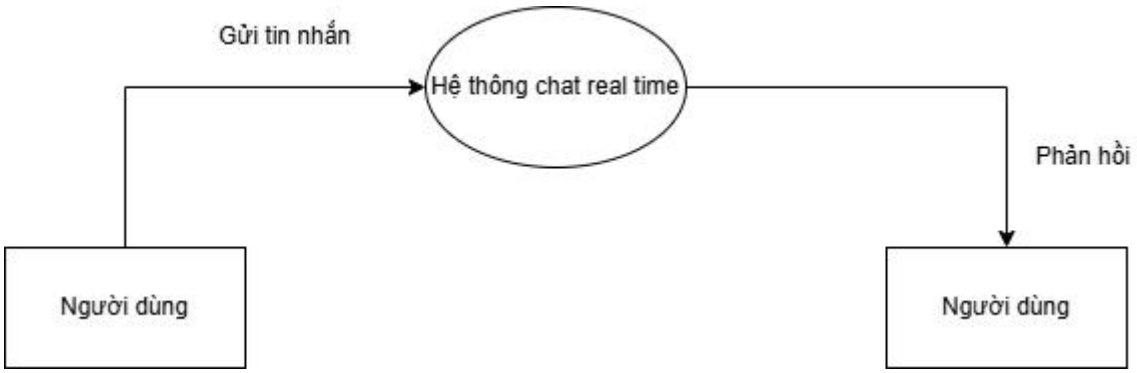
3.4.6. Danh sách chi tiết thực thể MessageContent

Stt	Tên thuộc tính	Kiểu dữ liệu	Diễn giải	Ràng buộc
1	id	binary	Id tin nhắn được gửi đi	Khóa chính
2	content	varchar	Nội dung tin nhắn	
3	date_sent	datetime	Ngày tin nhắn dược gửi	
4	message_type	varchar	Loại tin nhắn	
5	message_room_id	binary	Id của phòng chat	Khóa ngoại
6	username	varchar	Tên người dùng	Khóa ngoại

Bảng 5. Thực thể MessageContent

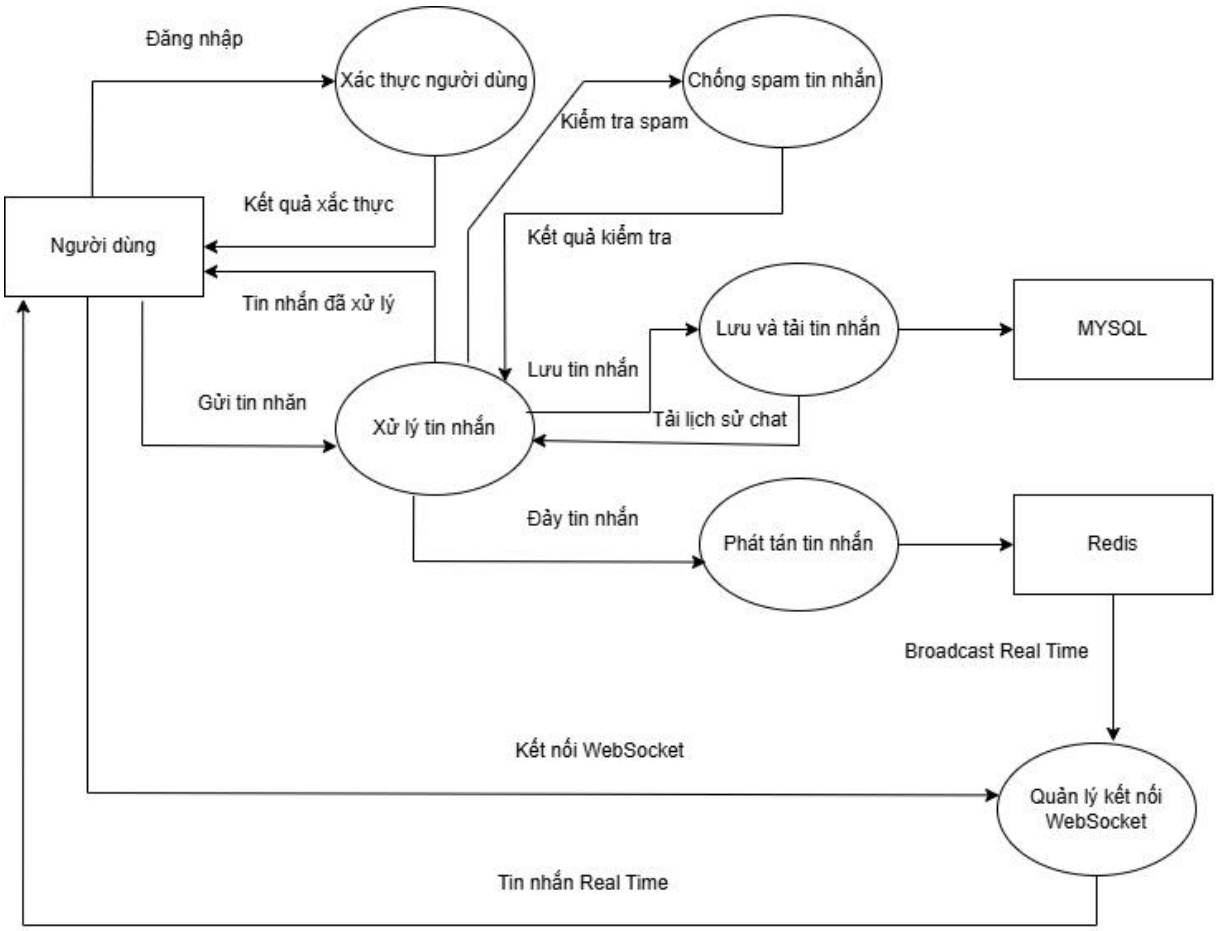
3.5. Thiết kế xử lý

3.5.1. Mô hình DFD mức ngữ cảnh



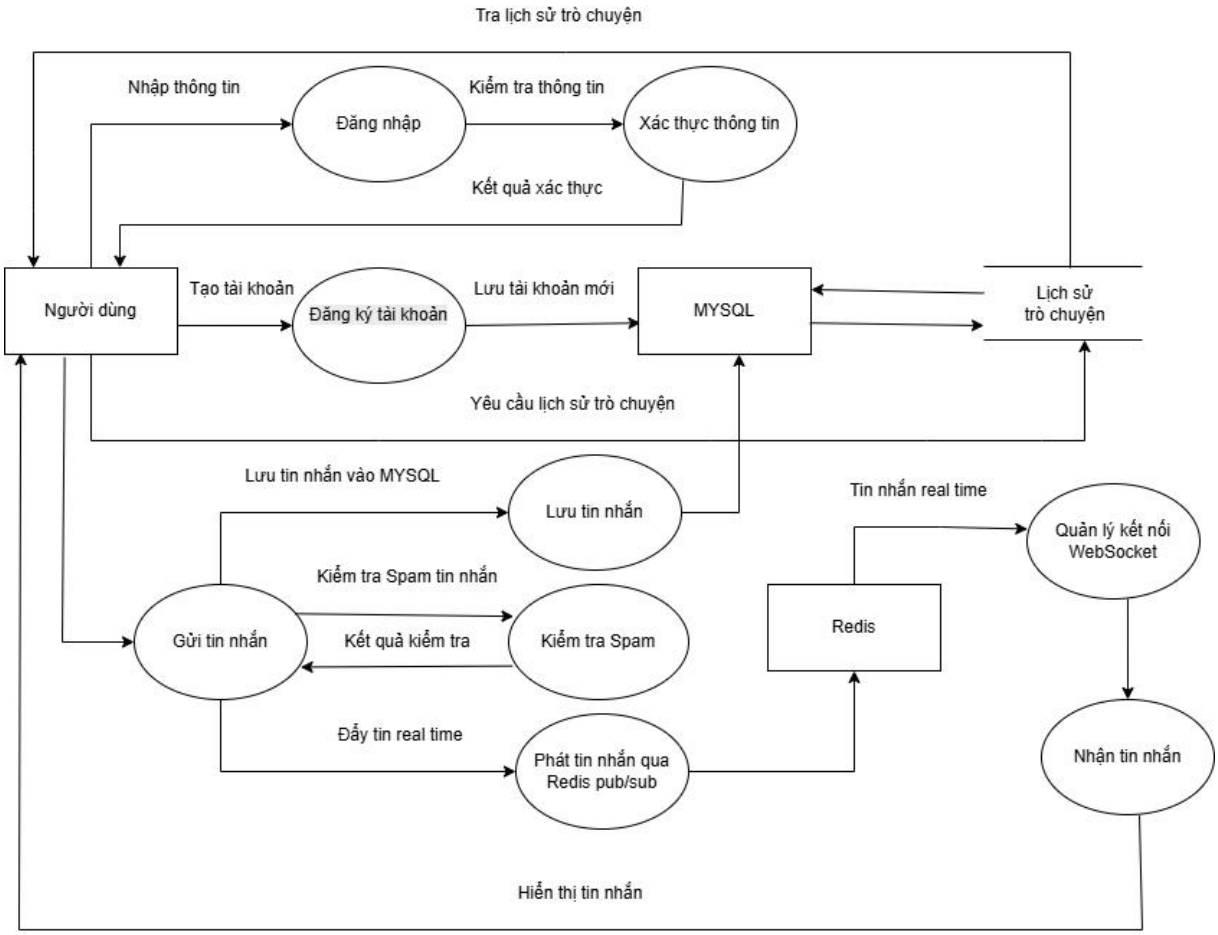
Hình 4. Sơ đồ kiến trúc hệ thống

3.5.2. Mô hình DFD mức 1



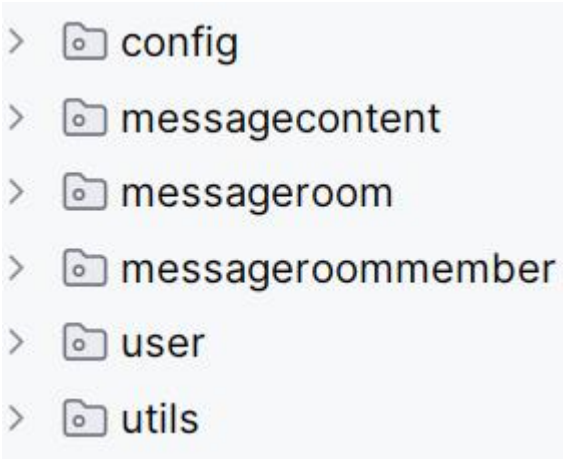
Hình 5. Sơ đồ DFD mức 1

3.5.3. Mô hình DFD mức 2



Hình 6. Sơ đồ DFD mức 2

3.6. Cấu trúc dự án



Hình 7. Cấu trúc thư mục Backend

Dự án được tổ chức thành các package điều này giúp tăng khả năng bảo trì và mở rộng trong tương lai dưới đây là mục đích và chức năng của từng package:

- config: Dùng để chứa các class cấu hình cho ứng dụng, như cấu hình bean cho Spring, cấu hình bảo mật, database hoặc bất kì cấu hình nào cần thiết cho ứng dụng.

- messagecontent: Chịu trách nhiệm xử lý và quản lý nội dung tin nhắn. Các class trong package này thường bao gồm entity, DTO, service hoặc repository phục vụ cho việc lưu trữ, truy xuất và xử lý nội dung tin nhắn mà người dùng gửi trong hệ thống.

- messageroom: Quản lý dữ liệu và logic liên quan đến phòng chat. Package này đảm nhiệm việc tạo phòng, cập nhật thông tin phòng, phân quyền truy cập cũng như các thao tác liên quan đến hoạt động của từng phòng chat.

- messageroommember: Đảm nhận việc quản lý thành viên trong từng phòng chat. Bao gồm chức năng thêm/xóa thành viên, theo dõi trạng thái tham gia phòng, phân quyền trong phòng chat và các nghiệp vụ liên quan đến người dùng trong phòng.

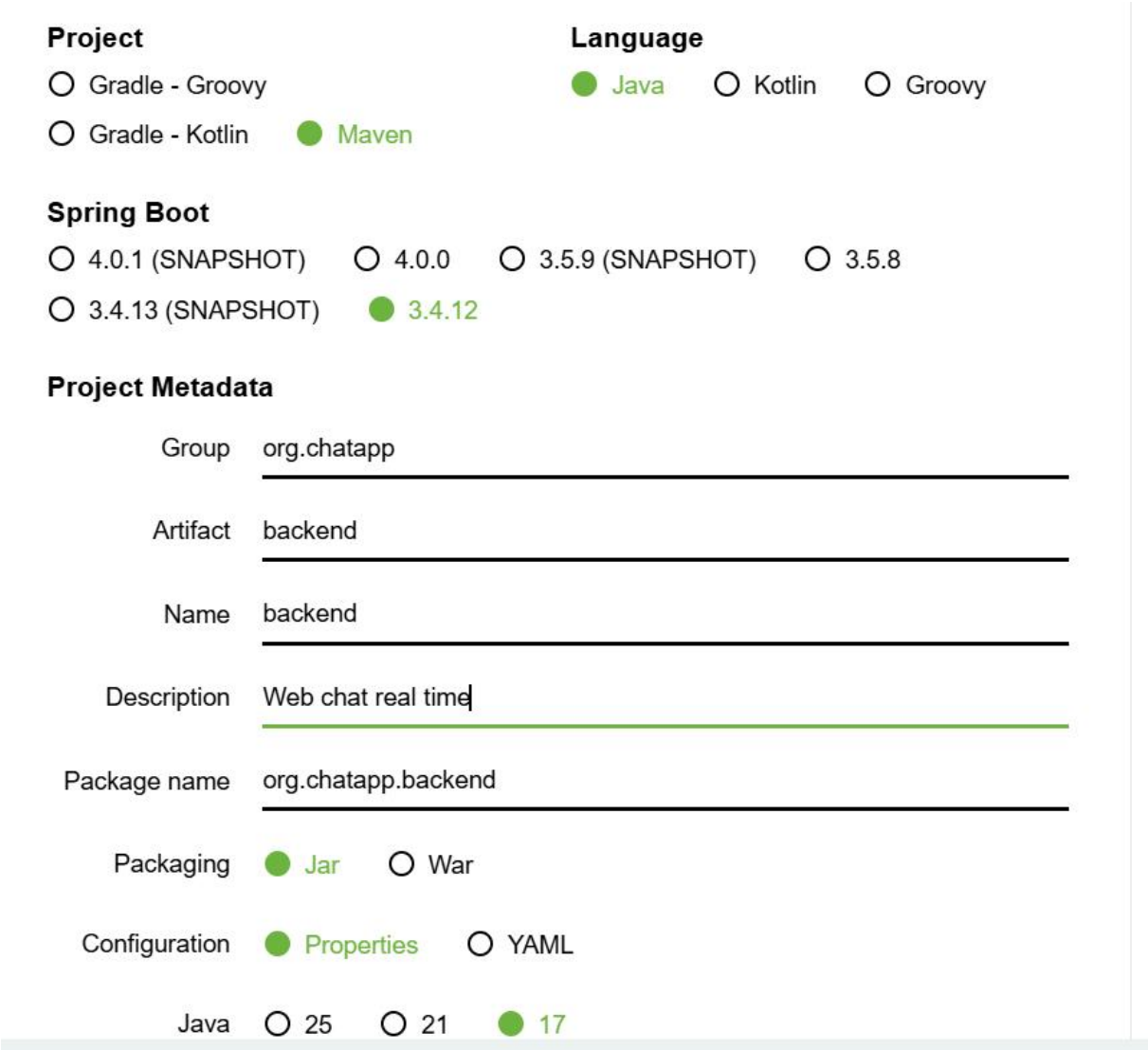
- user: Chứa các thành phần liên quan đến người dùng như thông tin người dùng, xử lý xác thực, phân quyền và quản lý tài khoản. Đây là package quan trọng trong việc kiểm soát quyền truy cập và đảm bảo an toàn hệ thống.

- utils: Chứa các đoạn mã tiện ích dùng chung trong toàn bộ dự án, chẳng hạn các hàm xử lý chuỗi, định dạng ngày giờ, mã hóa, hoặc các helper function khác. Các class tại đây giúp giảm lặp mã và tăng tính tái sử dụng.

CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU

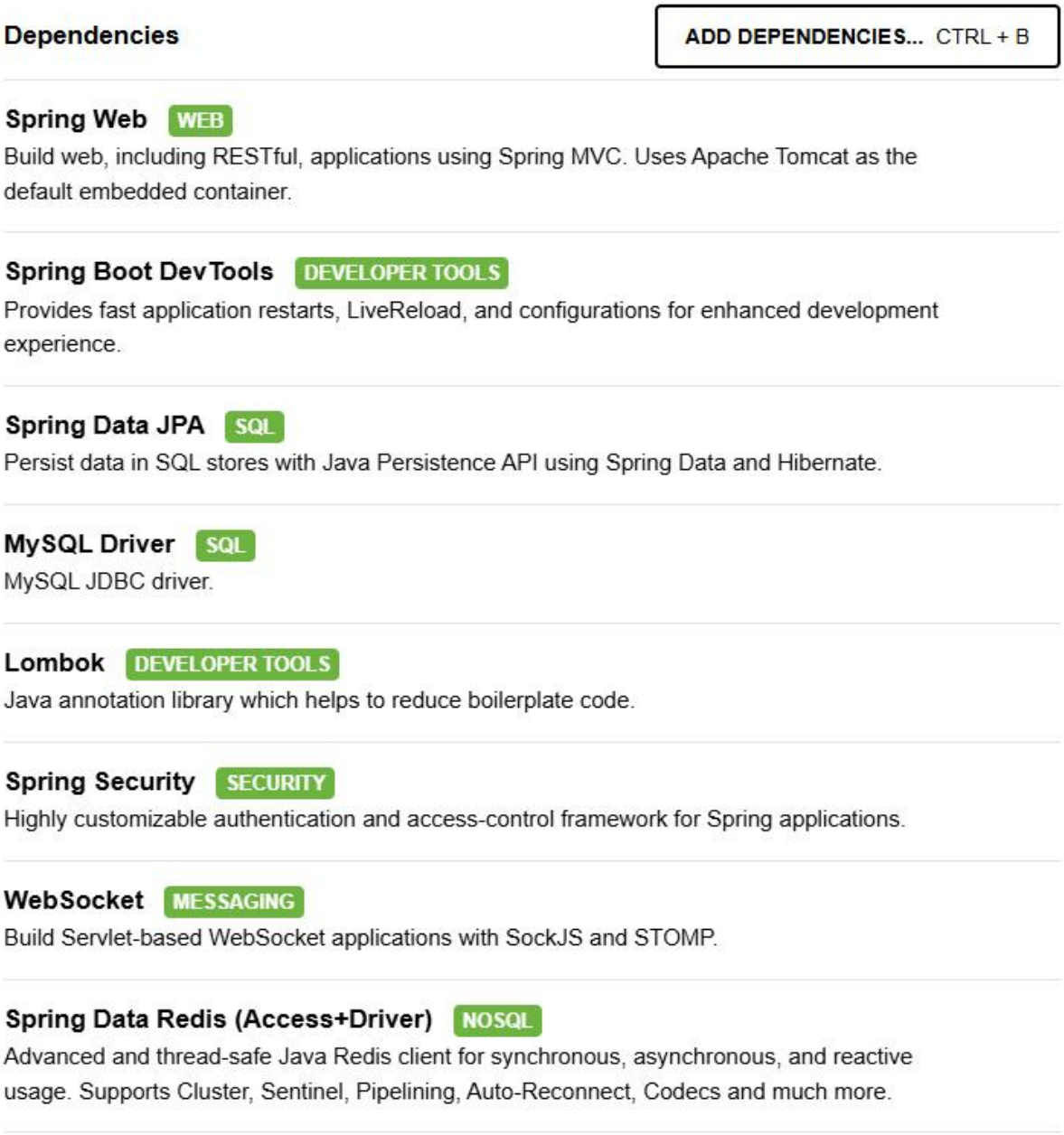
4.1. khởi tạo dự án

Truy cập vào trang https://start.spring.io/ để chọn phiên bản và ngôn ngữ phù hợp với dự án như sau:



Hình 8. Khởi tạo dự án Spring Boot

Và chọn các Dependencies phù hợp



Hình 9. Các Dependency của dự án

Các dependencies cho phép xây dựng một ứng dụng web có khả năng kết nối với sở dữ liệu MYSQL, sử dụng RESTful API trả về nhiều định dạng trong đó có định dạng Json nhẹ và phù hợp với ứng dụng hiện đại, sử dụng giao thức WebSocket để xử lý dữ liệu theo thời gian thực, giảm tải database với Redis và tạo tại liệu API tự động giúp cho việc test trở nên dễ dàng hơn và cuối cùng là bảo mật ứng dụng với spring security

Trang web start.spring.io được cung cấp bởi Spring giúp cho việc khởi tạo cấu trúc cơ bản của ứng dụng Spring Boot trở nên dễ dàng hơn.

4.2. Cấu hình dự án

Việc cấu hình trong dự án Spring Boot được thực hiện trên file application.properties

```
spring.application.name=backend

# ====================== DATABASE ======================
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/chatapp?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# ====================== REDIS ======================
spring.data.redis.host=localhost
spring.data.redis.port=6379
spring.data.redis.timeout=2000ms

# Lettuce connection pool
spring.data.redis.lettuce.pool.max-active=8
spring.data.redis.lettuce.pool.max-idle=8
spring.data.redis.lettuce.pool.min-idle=0

# ====================== CACHE ======================
spring.cache.type=redis
spring.cache.redis.time-to-live=600000
spring.cache.redis.cache-null-values=false
spring.cache.redis.use-key-prefix=true
spring.cache.redis.key-prefix=chatapp:

# ====================== API PREFIX ======================
api.prefix=/api/v1
backend.url=http://localhost:8080

# ====================== FILE UPLOAD ======================
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB

# ====================== LOGGING (debug Redis + SQL) ======================
logging.level.org.springframework.data.redis=DEBUG
logging.level.org.hibernate.SQL=DEBUG
```

Hình 10. File cấu hình ứng dụng

Trong đó:

driver-class-name: Khai báo driver dùng để kết nối MySQL.

spring.jpa.show-sql=true: Cho phép hiển thị các câu lệnh SQL trên console.

spring.jpa.hibernate.ddl-auto=update: Hibernate tự động cập nhật cấu trúc bảng.

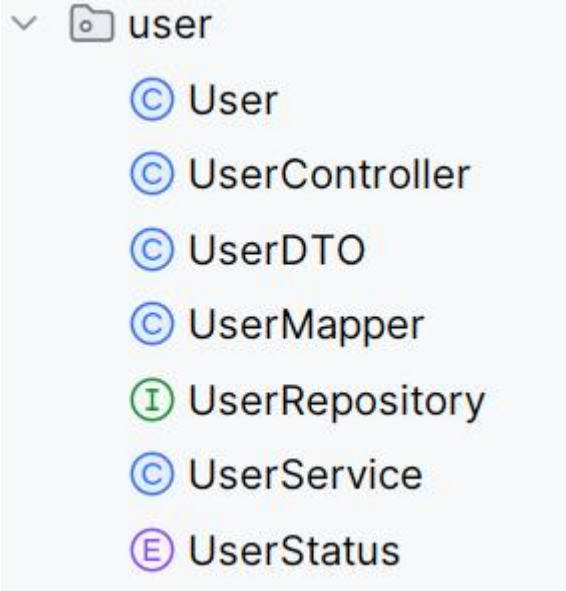
spring.datasource.url: Đường dẫn kết nối tới database chatapp

username / password: Thông tin đăng nhập MySQL.

api.prefix=/api/v1: Định nghĩa tiền tố chung cho tất cả endpoint để dễ quản lý.

4.3. Xây dựng API

4.3.1. Xây dựng API cho module User



Hình 11. Package User

User:

Là thực thể (entity) biểu diễn thông tin người dùng trong hệ thống, bao gồm các thuộc tính như id, username, password, email, role và status. Lớp này ánh xạ trực tiếp với bảng User trong cơ sở dữ liệu.

UserController:

Là lớp điều khiển (REST Controller) cung cấp các API xử lý các yêu cầu liên quan đến người dùng như đăng ký, đăng nhập, cập nhật thông tin hoặc thay đổi trạng thái. Controller nhận request từ client và trả response về frontend.

UserDTO:

Là lớp trung gian dùng để truyền dữ liệu giữa client và server, giúp tách biệt dữ liệu nhận/gửi với dữ liệu nội bộ của hệ thống. DTO thường không chứa logic mà chỉ chứa các trường cần thiết cho request/response.

UserMapper:

Là lớp chịu trách nhiệm chuyển đổi qua lại giữa User và UserDTO. Việc tách mapper giúp mã nguồn rõ ràng, giảm trùng lặp và đảm bảo dữ liệu được chuyển đổi đúng định dạng khi giao tiếp giữa các lớp.

UserRepository:

Là lớp giao tiếp trực tiếp với cơ sở dữ liệu, kế thừa từ JpaRepository để cung cấp các phương thức CRUD như lưu, tìm kiếm, cập nhật hoặc xóa người dùng. Repository giúp tách biệt logic truy vấn DB khỏi business logic.

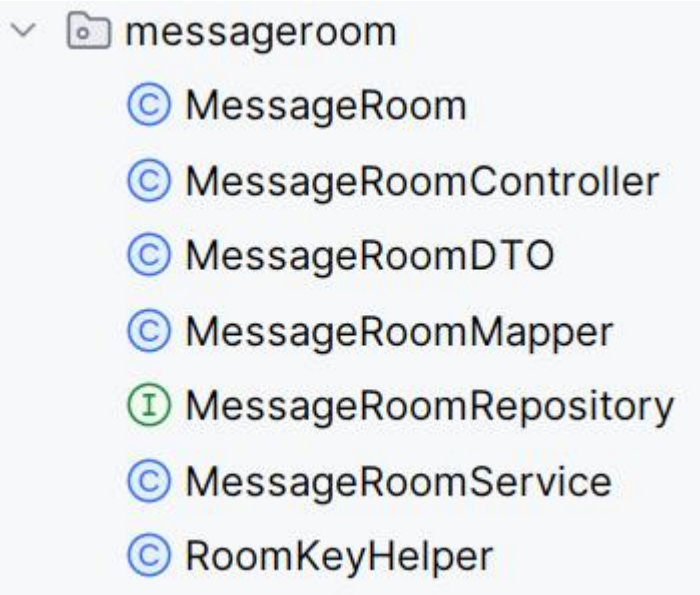
UserService:

Là lớp chứa toàn bộ business logic của module User, xử lý các nghiệp vụ như kiểm tra tồn tại người dùng, mã hóa mật khẩu, xác thực thông tin, cập nhật trạng thái và gọi repository để thao tác với cơ sở dữ liệu.

UserStatus:

Là enum biểu diễn trạng thái của người dùng trong hệ thống (ONLINE hoặc OFFLINE). Enum giúp định nghĩa trạng thái rõ ràng, tránh dùng giá trị "magic number" và dễ quản lý hơn.

4.3.2. Xây dựng API cho module MessageRoom



Hình 12. Package MessageRoom

MessageRoom

Đây có thể là lớp hoặc đối tượng chính đại diện cho một phòng nhắn tin. Nơi này sẽ chứa các thuộc tính và phương thức cơ bản cần thiết cho chức năng của phòng.

MessageRoomController

Chịu trách nhiệm xử lý các yêu cầu từ người dùng liên quan đến phòng nhắn tin. Nó hoạt động như một cầu nối giữa người dùng và logic nghiệp vụ.

MessageRoomDTO (Data Transfer Object)

Đây là lớp dùng để truyền dữ liệu giữa các lớp khác nhau. DTO giúp giảm thiểu số lượng các cuộc gọi cần thiết và làm cho dữ liệu dễ dàng xử lý hơn.

MessageRoomMapper

Chịu trách nhiệm chuyển đổi giữa các đối tượng từ lớp này sang lớp khác, đặc biệt là giữa các thực thể và DTO. Điều này giúp tách biệt logic xử lý dữ liệu và thực thể.

MessageRoomRepository

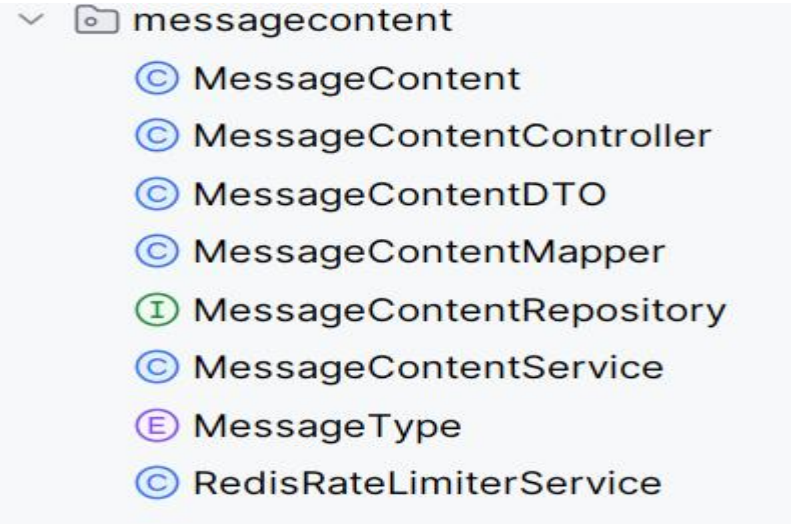
Đây là một phần quan trọng trong thiết kế DAO (Data Access Object). Nó chịu trách nhiệm truy xuất và quản lý dữ liệu từ cơ sở dữ liệu. Định nghĩa các phương thức cần có mà các lớp thực thi sẽ phải cụ thể hóa.**MessageRoomService**

Lớp này chứa logic nghiệp vụ liên quan đến phòng nhắn tin. Nó thực hiện các quy tắc của ứng dụng và gọi các phương thức trong Repository để lấy hoặc lưu dữ liệu.

RoomKeyHelper

Chứa các phương thức tiện ích liên quan đến khóa phòng (nếu có). Nó có thể bao gồm các logic để tạo hoặc xác thực khóa truy cập vào phòng, đảm bảo sự an toàn.

4.3.3. Xây dựng API cho module MessageContent



Hình 13. Package MessageContent

MessageContent

Lớp này đại diện cho nội dung tin nhắn, đóng vai trò là mô hình dữ liệu trong ứng dụng. Nó chứa các thuộc tính như nội dung tin nhắn, người gửi, thời gian gửi, và có thể bao gồm các thông tin bổ sung như trạng thái (đã đọc/chưa đọc). Điều này giúp hệ thống dễ dàng quản lý và lưu trữ thông tin tin nhắn một cách có tổ chức.

MessageContentController

Đóng vai trò là bộ điều khiển cho các yêu cầu HTTP liên quan đến nội dung tin nhắn. Tệp này thường chứa các phương thức để xử lý các hành động phổ biến như tạo, đọc, cập nhật và xóa nội dung tin nhắn (CRUD). Controller nhận các yêu cầu từ client, gọi đến dịch vụ thích hợp, và trả về phản hồi thích hợp.

MessageContentDTO

Lớp nội dung tin nhắn được sử dụng để truyền tải dữ liệu giữa các lớp trong ứng dụng. Tệp này định nghĩa các thuộc tính cần thiết để truyền thông tin một cách hiệu quả, giúp giảm thiểu dữ liệu không cần thiết. Nó cũng giúp đơn giản hóa quá trình serialization và deserialization dữ liệu, làm cho việc xử lý thông tin giữa client và server thuận tiện hơn.**MessageContentMapper**

Chịu trách nhiệm ánh xạ giữa các đối tượng mô hình như MessageContent và các DTO hoặc các đối tượng khác. Tệp này rất quan trọng trong việc chuyển đổi dữ liệu, giúp đồng bộ hóa các định dạng dữ liệu khác nhau giữa các lớp trong ứng dụng, từ đó đảm bảo rằng thông tin được truyền tải đầy đủ và chính xác.

MessageContentRepository

Đại diện cho lớp truy cập dữ liệu cho nội dung tin nhắn. Tệp này chứa các phương thức giúp tương tác với cơ sở dữ liệu, phục vụ cho việc lưu trữ, truy xuất hoặc tìm kiếm nội dung tin nhắn. Nó thực hiện các thao tác cơ bản với cơ sở dữ liệu, giúp tách biệt logic truy cập dữ liệu khỏi các phần khác của ứng dụng.

MessageContentService

Cung cấp các phương thức nghiệp vụ cho nội dung tin nhắn, nơi chứa logic xử lý

chính cho các chức năng liên quan đến tin nhắn. Tệp này sẽ gọi đến MessageContentRepository để thực hiện các thao tác với dữ liệu và có thể bao gồm các quy tắc nghiệp vụ như kiểm tra tính hợp lệ trước khi lưu tin nhắn.

MessageType

Định nghĩa các loại tin nhắn có thể có trong ứng dụng. Tệp này có thể chứa các hằng số hoặc enum để phân loại và xác định các loại tin nhắn khác nhau, giúp hệ thống dễ dàng quản lý và xử lý các loại tin nhắn đặc thù.

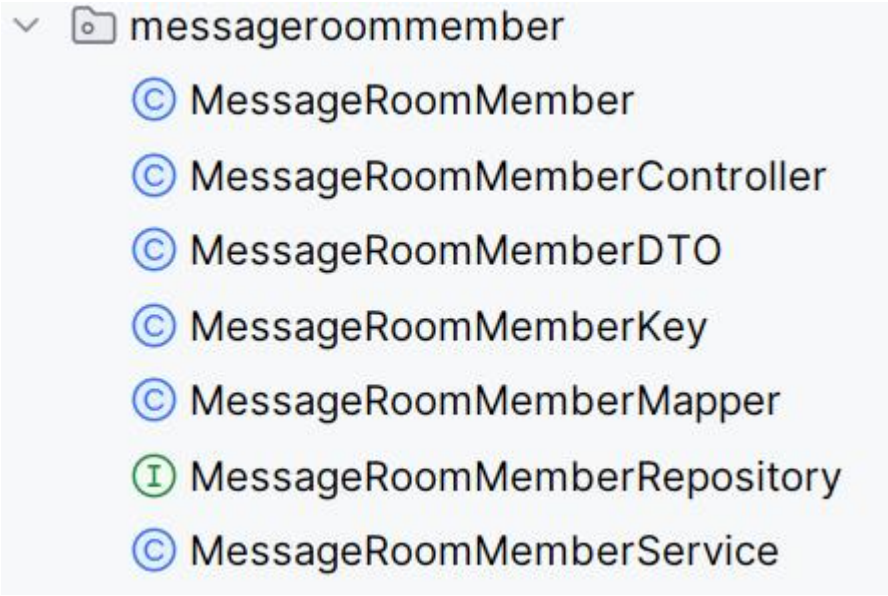
RedisRateLimiterService

chịu trách nhiệm quản lý việc giới hạn tần suất gửi yêu cầu từ người dùng. Điều này giúp giảm thiểu rủi ro lạm dụng dịch vụ và bảo vệ hệ thống khỏi các cuộc tấn công từ chối dịch vụ (DDoS)Khi một yêu cầu được gửi đến hệ thống.

Dịch vụ này sử dụng Redis, một kho dữ liệu in-memory có khả năng xử lý các thao tác nhanh chóng, giúp theo dõi và quản lý số lượng yêu cầu mà từng người dùng gửi trong một khoảng thời gian nhất định. Qua đó, **RedisRateLimiterService** thực hiện các chức năng như kiểm tra và ghi nhận số yêu cầu đã gửi, đồng thời giới hạn số yêu cầu mà một người dùng có thể gửi trong các khoảng thời gian cụ thể.
Khi một yêu cầu được gửi đến hệ thống

RedisRateLimiterService sẽ kiểm tra thông tin người dùng và số lượng yêu cầu đã được gửi trong khung thời gian đó. Nếu số lượng yêu cầu vượt quá giới hạn đã đặt ra, dịch vụ sẽ từ chối yêu cầu mới và có thể trả về một thông báo lỗi, thông báo cho người dùng rằng họ đã đạt đến giới hạn cho phép. Điều này không chỉ giúp bảo vệ hệ thống khỏi việc bị quá tải mà còn cải thiện trải nghiệm người dùng bằng cách đảm bảo rằng tất cả người dùng đều có thể truy cập dịch vụ mà không bị ảnh hưởng bởi sự lạm dụng của các người dùng khác.

4.3.4. Xây dựng API cho module MessageRoomMember



Hình 14. Package MessageRoomMember

MessageRoomMember

Đây có thể là lớp hoặc đối tượng chính đại diện cho thành viên trong một phòng nhắn tin. Nơi này sẽ chứa các thuộc tính và phương thức cần thiết để quản lý thông tin thành viên

MessageRoomMemberController

Chịu trách nhiệm xử lý các yêu cầu từ người dùng liên quan đến thành viên trong phòng nhắn tin. Nó hoạt động như cầu nối giữa người dùng và logic nghiệp vụ

MessageRoomMemberDTO (Data Transfer Object)

Đây là lớp dùng để truyền dữ liệu giữa các lớp khác nhau liên quan đến thành viên. DTO giúp giảm thiểu số lượng cuộc gọi cần thiết và làm cho dữ liệu dễ dàng xử lý hơn

MessageRoomMemberKey

Có thể là lớp hoặc đối tượng đại diện cho khóa chính hoặc định danh của thành viên trong phòng. Nó giúp xác định duy nhất mỗi thành viên trong hệ thống

MessageRoomMemberMapper

Chịu trách nhiệm chuyển đổi giữa các đối tượng từ lớp này sang lớp khác, đặc biệt là giữa các thực thể và DTO. Điều này giúp tách biệt logic xử lý dữ liệu và thực thể

MessageRoomMemberRepository

Đây là một phần quan trọng trong thiết kế DAO (Data Access Object). Nó chịu trách nhiệm truy xuất và quản lý dữ liệu từ cơ sở dữ liệu về các thành viên trong phòng. Mà các lớp thực thi sẽ phải cụ thể hóa

MessageRoomMemberService

Lớp này chứa logic nghiệp vụ liên quan đến các thành viên trong phòng nhắn tin. Nó thực hiện các quy tắc của ứng dụng và gọi các phương thức trong Repository để lấy hoặc lưu dữ liệu

4.4. Lớp cấu hình cho ứng dụng

Hình 15. Package Config

4.4.1. CorsConfig

```
package org.chatapp.backend.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration  no usages
public class CorsConfig implements WebMvcConfigurer {
    @Override  no usages
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping( pathPattern: "/**")
                .allowedOrigins("http://localhost:4200")
                .allowedMethods("*")
                .allowCredentials(true);
    }
}
```

Hình 16. Cấu hình CORS

Lớp này chịu trách nhiệm cấu hình chính sách CORS (Cross-Origin Resource Sharing). Nó cho phép hoặc từ chối các yêu cầu từ các nguồn khác nhau, đảm bảo an toàn cho ứng dụng.

4.4.2. ResourcesConfig

```java
package org.chatapp.backend.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration  no usages
public class ResourcesConfig implements WebMvcConfigurer {
    @Override  no usages
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler( ...pathPatterns: "/images/**")
                .addResourceLocations("file:uploads/");
    }
}
```

Hình 17. Cấu hình Resource

Chứa các cấu hình liên quan đến tài nguyên, như định nghĩa các đường dẫn hoặc cách mà ứng dụng sẽ xử lý và phục vụ các tài nguyên.

4.4.3. SwaggerConfig

```java
package org.chatapp.backend.config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Contact;
import io.swagger.v3.oas.models.info.Info;
import io.swagger.v3.oas.models.info.License;

@Configuration  no usages
public class SwaggerConfig {
    @Bean  no usages
    public OpenAPI customOpenAPI() {
        return new OpenAPI()
                .info(new Info()
                        .title("Ứng dụng web chat real time sử dụng Angular và Spring Boot")
                        .description("Ứng dụng cho phép nhắn tin theo thời gian thực từ chat với một người và với nhóm.")
                        .version("1.0")
                        .contact(new Contact()
                                .name("Web chat real time")
                                .email("110122068@st.tvu.edu.vn")
                                .url("https://github.com/haivoDA22TTD/cn-DA22TTD-vochihai-chat-app-real-time-Spring-Boot"))
                        .license(new License()
                                .name("MIT License")
                                .url("https://opensource.org/licenses/MIT"))
                );
    }
}
```

Hình 18. Cấu hình Swagger

Lớp này chịu trách nhiệm cấu hình Swagger, một công cụ giúp tạo tài liệu API. Nó cho phép các nhà phát triển dễ dàng tương tác và thử nghiệm các API.

4.4.5. WebSocketConfig

```java
package org.chatapp.backend.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.messaging.simp.config.MessageBrokerRegistry;
import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;
import org.springframework.web.socket.config.annotation.StompEndpointRegistry;
import org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer;

@Configuration  no usages
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {
    @Override  no usages
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint( ...paths: "/api/ws")
                .setAllowedOriginPatterns("*")
                .withSockJS();
    }

    @Override  no usages
    public void configureMessageBroker(MessageBrokerRegistry registry) {
        registry.enableSimpleBroker( ...destinationPrefixes: "/topic", "/user");
        registry.setApplicationDestinationPrefixes("/app");
        registry.setUserDestinationPrefix("/user");
    }
}
```

Hình 19. Cấu hình WebSocket

Chịu trách nhiệm cấu hình cho WebSocket, cho phép giao tiếp hai chiều trong thời gian thực giữa client và server. Cấu hình này tạo ra các endpoint WebSocket cho ứng dụng.

4.4.6. RedisConfig

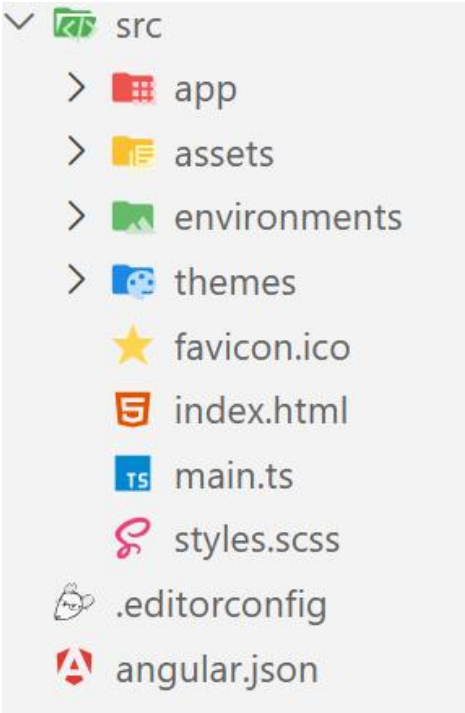
```java
package org.chatapp.backend.config;                                    ⚠ 2

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.core.StringRedisTemplate;

@Configuration  no usages
public class RedisConfig {
    @Bean  no usages
    public StringRedisTemplate stringRedisTemplate(RedisConnectionFactory factory) {
        return new StringRedisTemplate(factory);
    }
}
```

Hình 20. Cấu hình Redis

Định nghĩa cấu hình cho kết nối với Redis. Tệp này thiết lập các tham số cần thiết để tương tác với cơ sở dữ liệu Redis, bao gồm các thiết lập về URL, cổng và các tùy chọn khác.

4.5. Thiết kế giao diện

4.5.1. Cấu trúc dự án Angular



Hình 21. Cấu trúc dự án Angular

src chứa mã nguồn của ứng dụng

app gồm các thành phần, dịch vụ, mô hình và module chính của ứng dụng

assets lưu trữ các tài nguyên tĩnh như hình ảnh và các file css chứa màu themes

environments chứa các file môi trường cho cấu hình ứng dụng

themes bao gồm các file CSS cho các chủ đề khác nhau của ứng dụng

favicon.ico là file biểu tượng nhỏ hiển thị trên tab trình duyệt

index.html là file HTML chính của ứng dụng, khởi động ứng dụng Angular

main.ts là file TypeScript khởi động ứng dụng, thực hiện bootstrap module gốc

styles.scss là file SCSS chính cho toàn bộ kiểu dáng của ứng dụng

.editorconfig là file cấu hình cho trình soạn thảo mã

angular.json là file cấu hình cho dự án Angular.

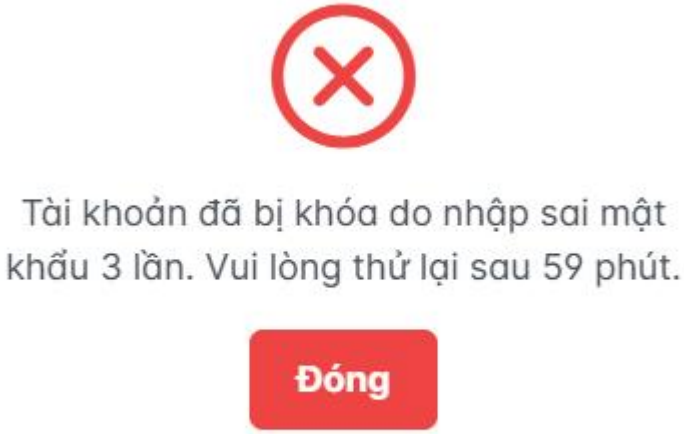
4.5.2. Giao diện trang đăng nhập

Hình 22. Trang đăng nhập

Người dùng cần đăng nhập vào ứng dụng để có thể nhắn tin trực tuyến tại trang đăng nhập điền vào lần lượt là tên đăng nhập và mật khẩu nếu người dùng điền không đầy đủ thông tin thì nút đăng nhập sẽ bị mờ người dùng không thể nhấn nút đăng nhập. Người dùng có thể ấn vào icon ở ô điền mật khẩu để hiển thị mật khẩu mình vừa nhập, nếu tài khoản mà người dùng nhập vào chưa có trong hệ thống thì khi họ nhấn vào nút đăng nhập thì tài khoản đó sẽ được tạo và điều hướng người dùng đến trang chủ của ứng dụng lúc này người dùng đã có thể tạo nhóm chat hoặc nhắn 1-1 với tài khoản vừa mới tạo còn nếu tài khoản đã có trong hệ thống mà người dùng nhập sai mật khẩu thì ứng dụng sẽ thông báo người dùng nhập sai mật khẩu và tối đa chỉ nhập sai mật khẩu hai lần nếu tới lần thứ ba mà người dùng vẫn nhập sai thì sẽ bị khóa tài khoản.

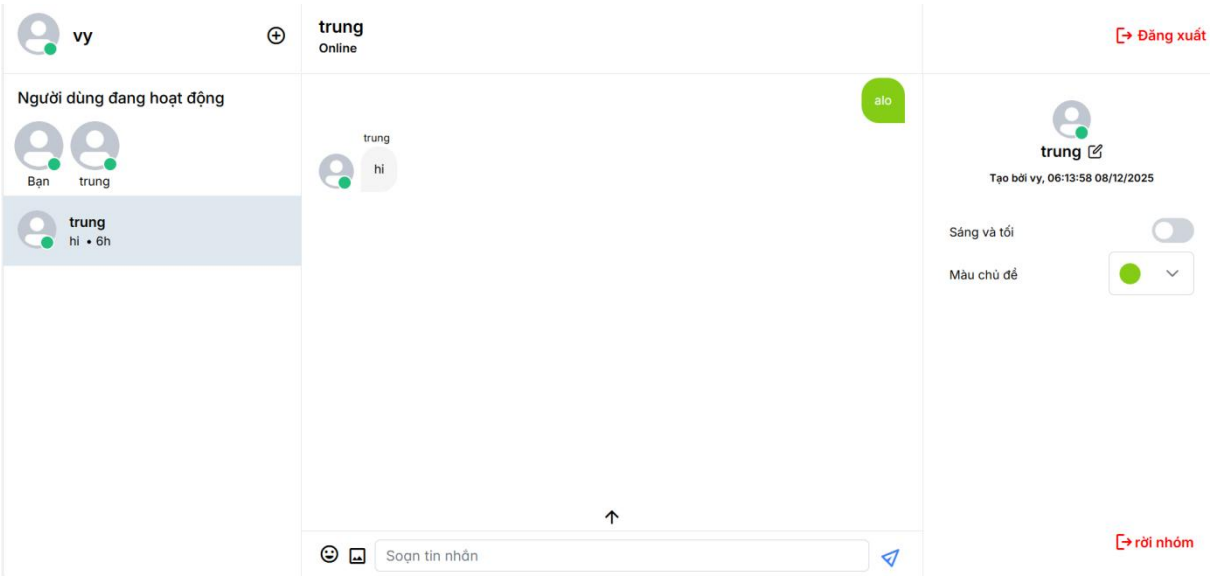
Hình 23. Thông báo nhập sai mật khẩu

Nếu như người dùng nhập sai tài khoản tới lần thứ ba thì tài khoản của họ sẽ bị khóa trong vòng một giờ.



Hình 24. Thông báo khóa tài khoản

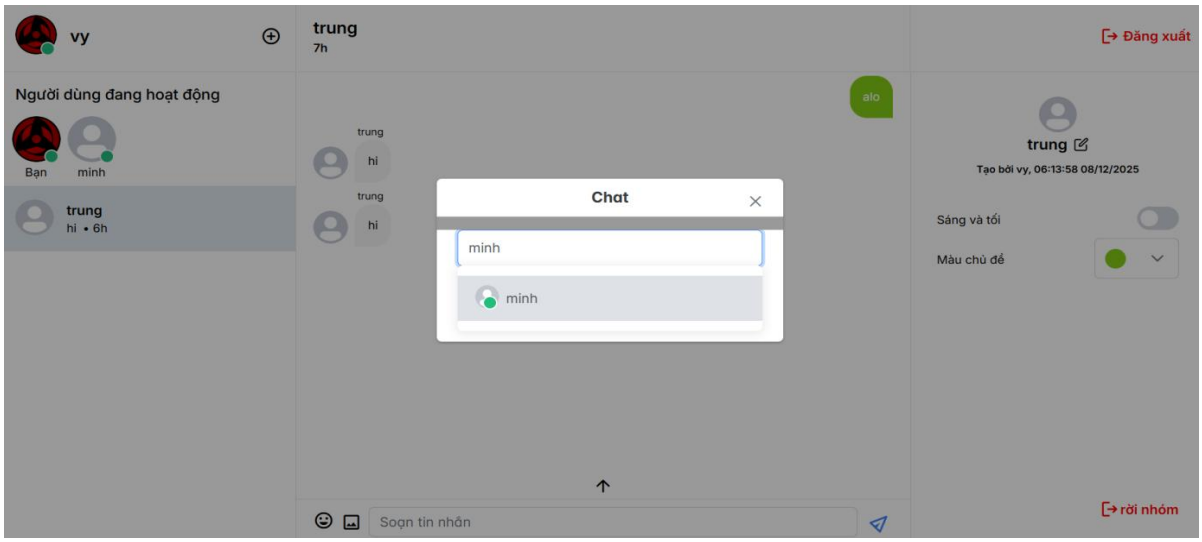
4.5.3 Giao diện trang chủ



Hình 25. Giao diện trang chủ ứng dụng

Đây là nơi sẽ hiển thị các cuộc trò chuyện của người dùng thanh bên trái là thanh sidebar gồm có ảnh đại diện của người dùng kế bên là nút tạo tạo cuộc trò chuyện, phía dưới là danh sách những người dùng đang hoạt động. Ở giữa sẽ hiển thị các tin nhắn của người trong cuộc trò chuyện dưới chân trang là ô soạn thảo tin nhắn, nút gửi tin nhắn. Bên phải là nơi chọn chế độ sáng và tối và chọn màu chủ đề của cuộc trò chuyện và cuối cùng là nút rời nhóm và đăng xuất.

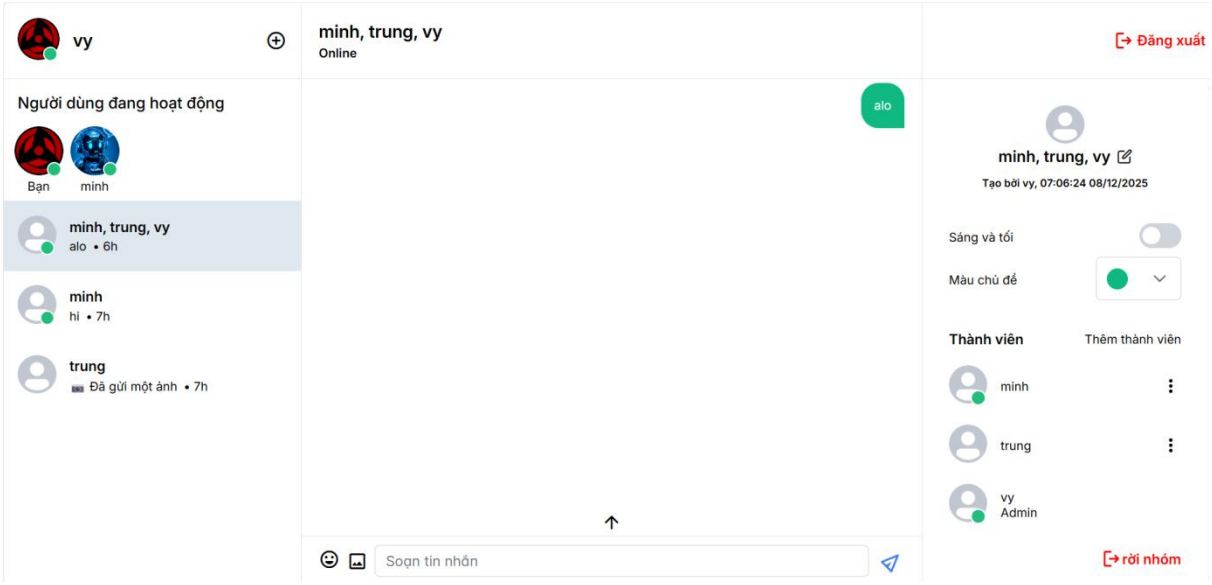
4.5.4. Tạo nhóm



Hình 26. Tạo nhóm

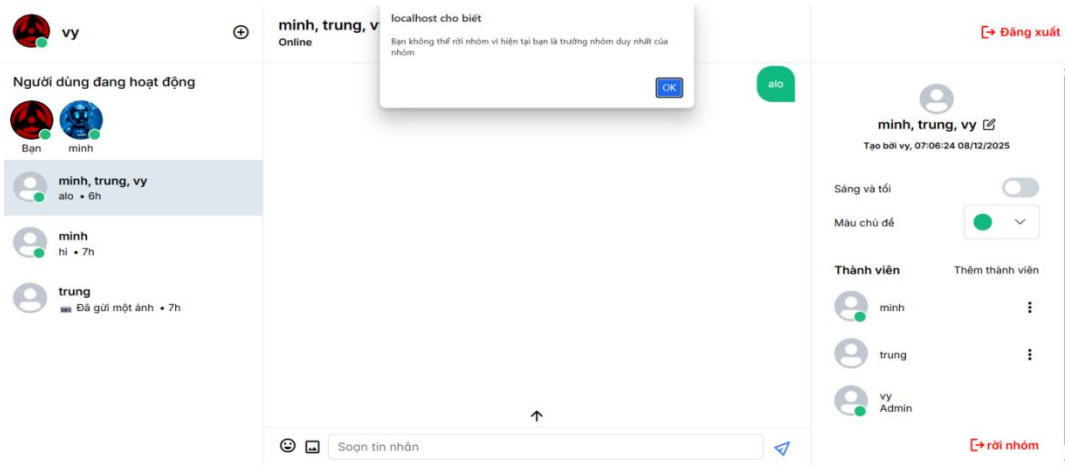
Người dùng sẽ tạo nhóm bằng cách nhấn vào biểu tượng dấu cộng ở thanh sidebar sau đó sẽ mở ra một hộp thoại người dùng sẽ điền tên của người mà họ muốn thêm vào nhóm. Nếu tài khoản đó có tồn tại trong hệ thống thì sẽ hiển thị người đó và nếu như ngời đó đang online thì ảnh đại diện của người dùng đó sẽ xuất hiện biểu tượng hình tròn màu xanh lá.

4.5.5. Quản lý nhóm



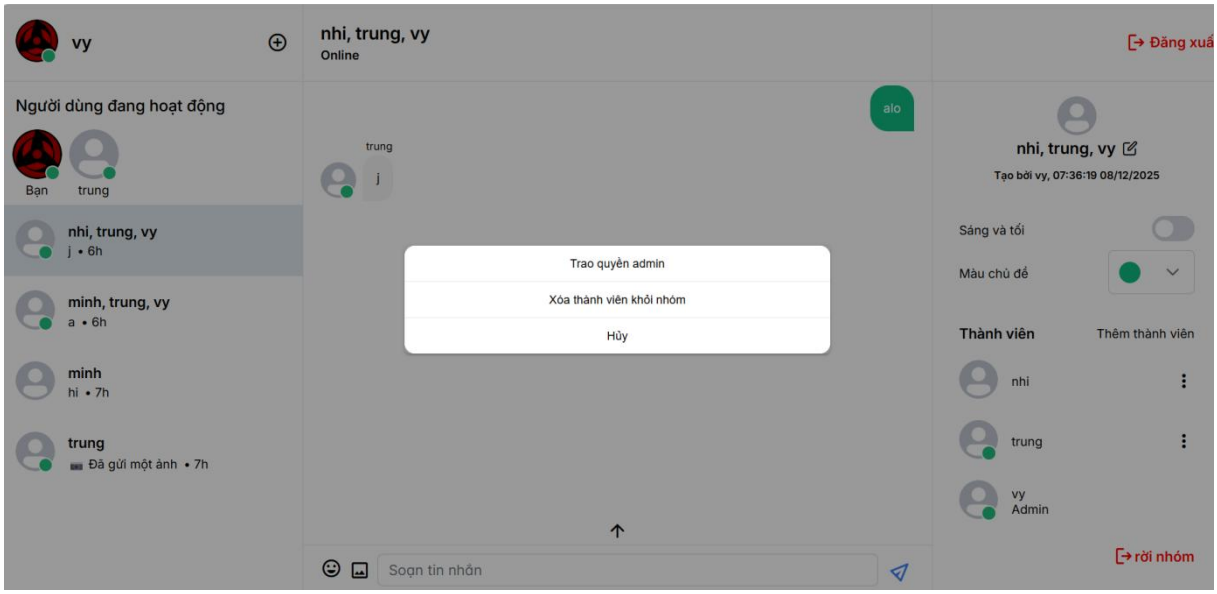
Hình 27. Quản lý nhóm

Sau khi tạo nhóm thành công ở thanh sidebar bên phải sẽ hiển thị các thành viên trong nhóm và người tạo nhóm sẽ là trưởng nhóm nếu trong nhóm chỉ có một trưởng nhóm duy nhất thì người trưởng nhóm này khi nhấn nút rời nhóm thì sẽ hiển thị thông báo và không cho người dùng này rời nhóm.



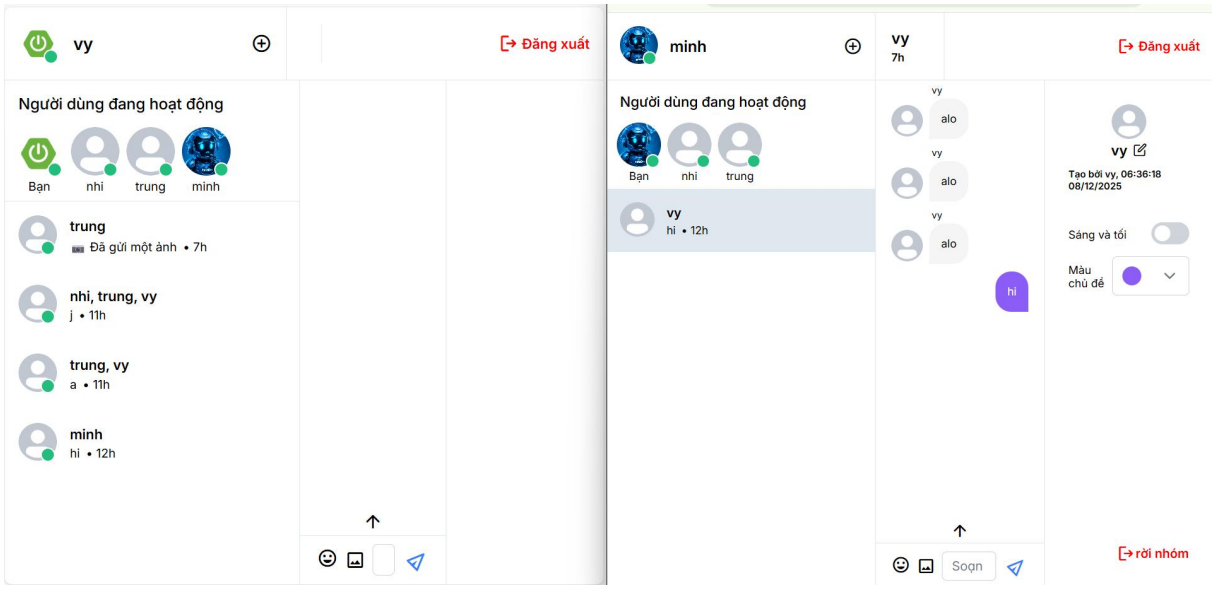
Hình 28. Cảnh báo không thể rời nhóm

Để rời nhóm thì người trưởng nhóm hiện tại phải chuyển quyền trưởng nhóm cho một thành viên trong nhóm. Để thực hiện chuyển quyền trưởng nhóm cho một thành viên trong nhóm trưởng nhóm sẽ ấn vào dấu ba chấm kế bên tên thành viên trong nhóm sau đó một hộp thoại sẽ hiện ra trưởng nhóm sẽ hiện tại sẽ nhấn vào mục chuyển quyền trưởng nhóm



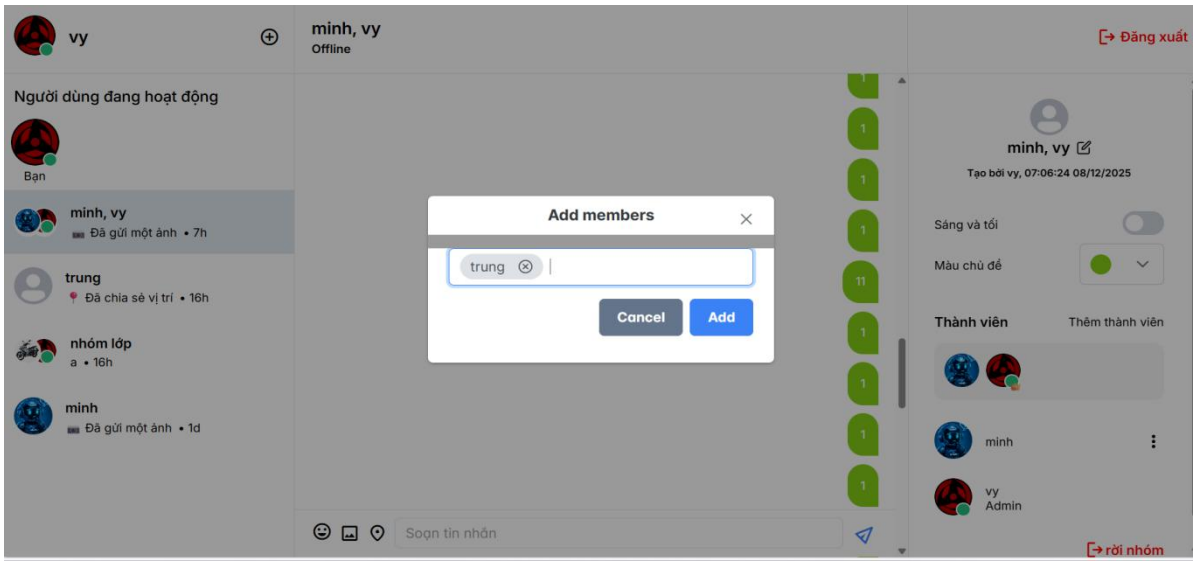
Hình 29. Tùy chọn

Trưởng nhóm có thể xóa một thành viên ra khỏi nhóm bằng cách ấn vào dấu ba chấm kế bên tên thành viên trong nhóm lúc này sẽ hiển thị hộp thoại chọn mục xóa thành viên khỏi nhóm. Lúc này danh sách thành viên nhóm sẽ được cập nhật lại và thành viên bị xóa không còn tồn tại trong nhóm nữa và trưởng nhóm có thể thêm một thành viên khác vào nhóm.



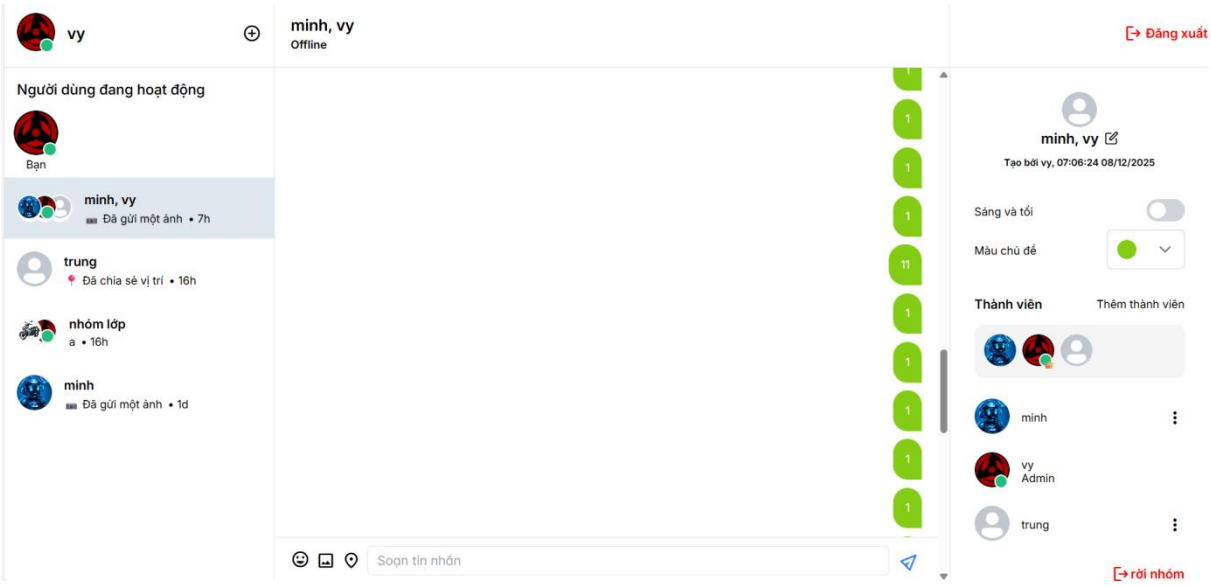
Hình 30. Danh sách người dùng online

Ngoài ra trưởng nhóm có thể thêm thành viên vào nhóm bằng nhập tên của người dùng và nhấn vào nút thêm.



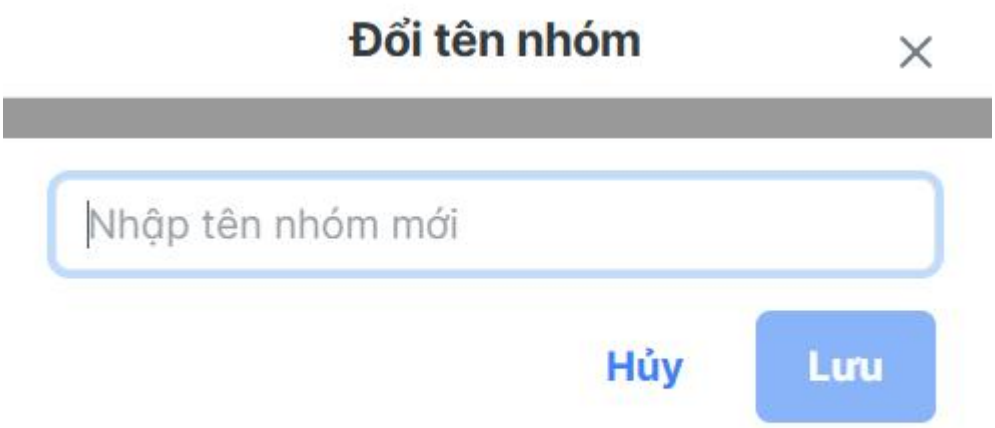
Hình 31. Thêm thành viên vào nhóm chat

Người dùng sau khi được thêm vào nhóm sẽ hiển thị trong danh sách thành viên trong nhóm



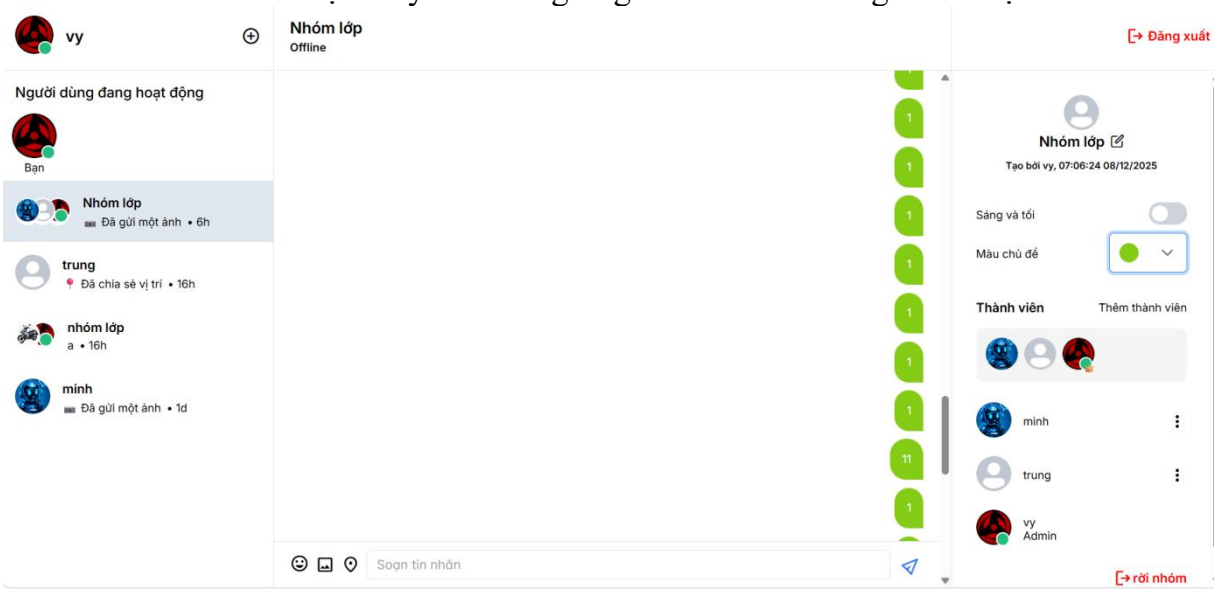
Hình 32. Thành viên nhóm sau được nhật

Trưởng nhóm còn có thể đổi tên nhóm bằng cách nhấn vào biểu tượng cây bút kế bên tên nhóm



Hình 33. Đổi tên nhóm

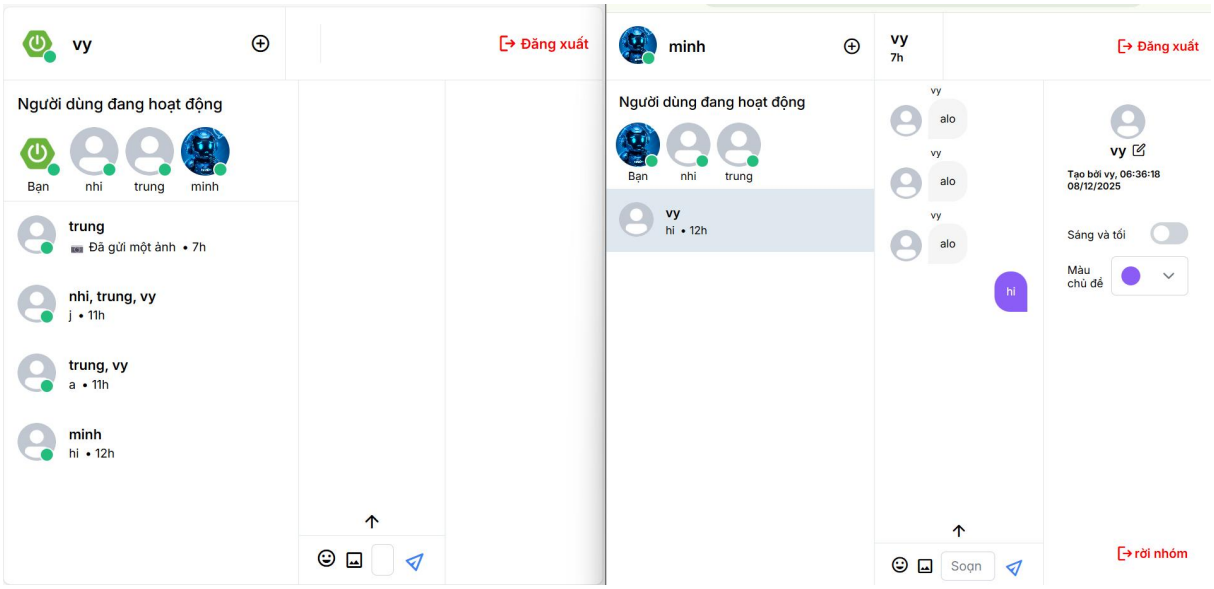
Tên nhóm sẽ được thay đổi tương ứng với tên mà trưởng nhóm đặt



Hình 34. Tên nhóm sau khi được cập nhật

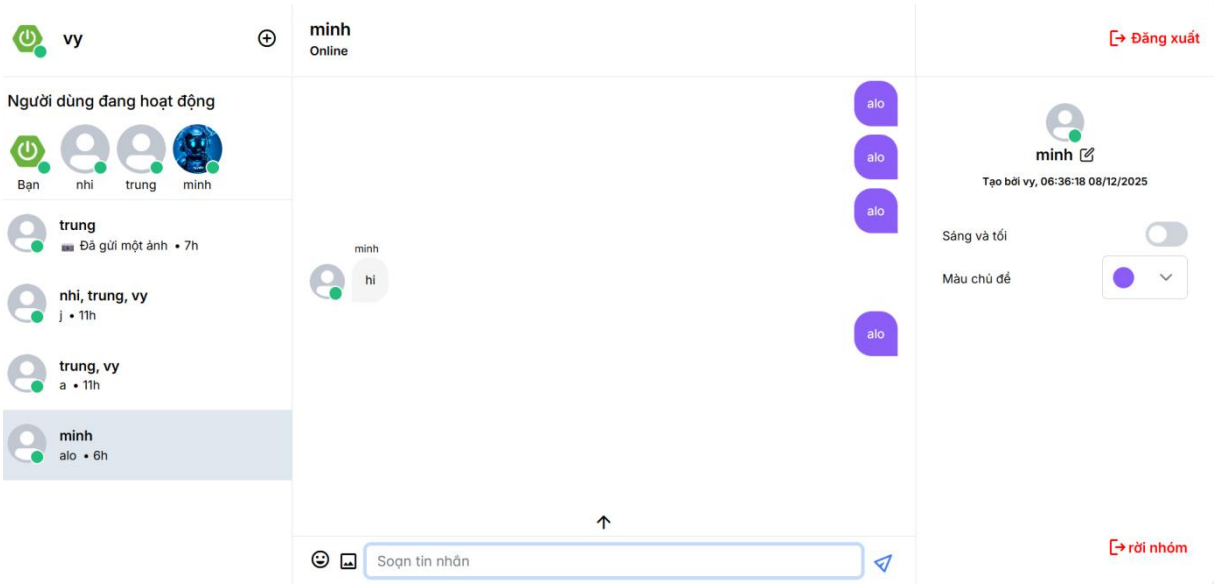
4.5.6. Cập nhật ảnh đại diện

Người dùng có thể cập nhật ảnh đại diện từ ảnh ngay trong thiết bị của mình sau khi cập nhật ảnh đại diện thì những người trong cuộc trò chuyện và những và những người dùng đang online sẽ có thể thấy được ảnh dại diện vừa cập mới được cập nhật của họ.



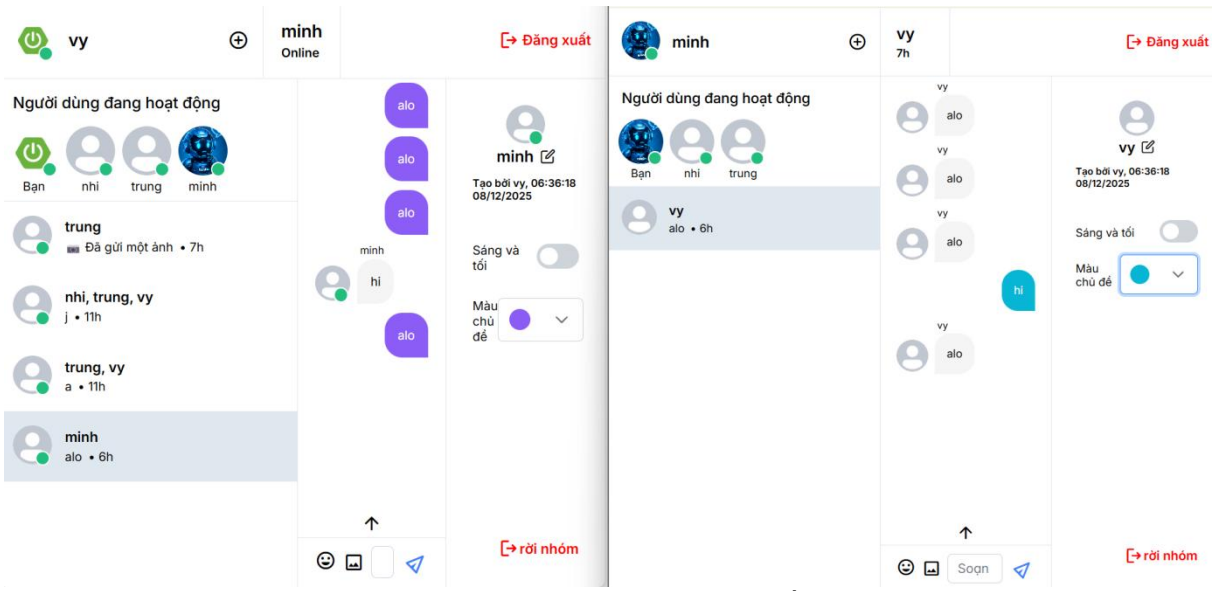
Hình 35. Cập nhật ảnh đại diện

4.5.7. Gửi tin nhắn



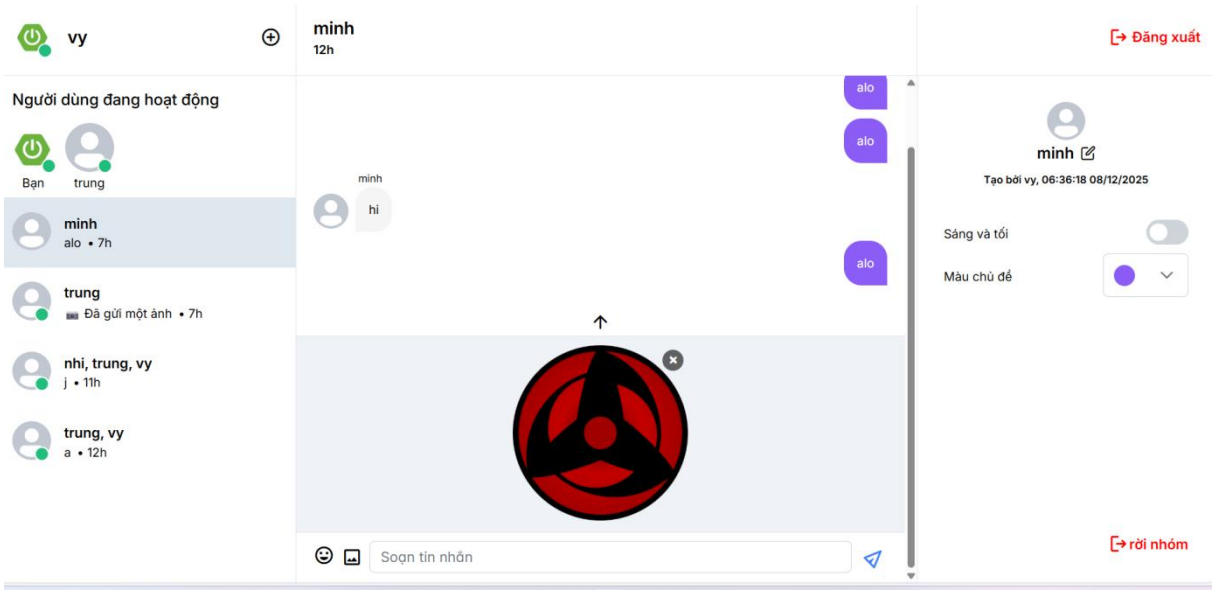
Hình 36. Tin nhắn văn bản

Người dùng nhập tin nhắn vào ô soạn thảo tin nhắn để gửi tin nhắn đi có thể nhấn nút enter hoặc nhấn vào nút gửi tin nhắn tin nhắn ngay lập tức được gửi đi và nhờ có giao thức WebSocket dữ liệu được gửi đi hai chiều mọi thao tác bên thứ nhất thực hiện thì bên thứ hai sẽ nhận được ngay lập tức.



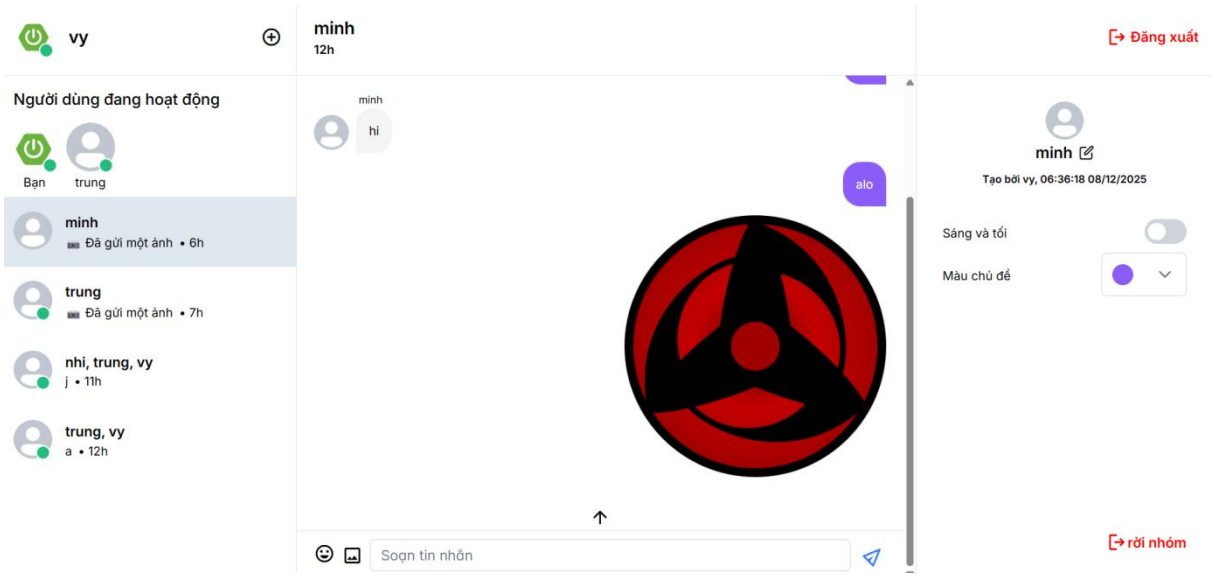
Hình 37. Người nhận xem tin nhắn

Bên cạnh việc gửi tin nhắn văn bản người dùng còn có thể gửi hình ảnh từ thiết bị của mình vào nhóm sau khi chọn hình ảnh từ thiết bị hình ảnh mà người dùng vừa chọn sẽ được hiển thị người dùng có thể bỏ ảnh đã chọn bằng cách nhấn vào nút dấu x phía trên bên phải của ảnh.



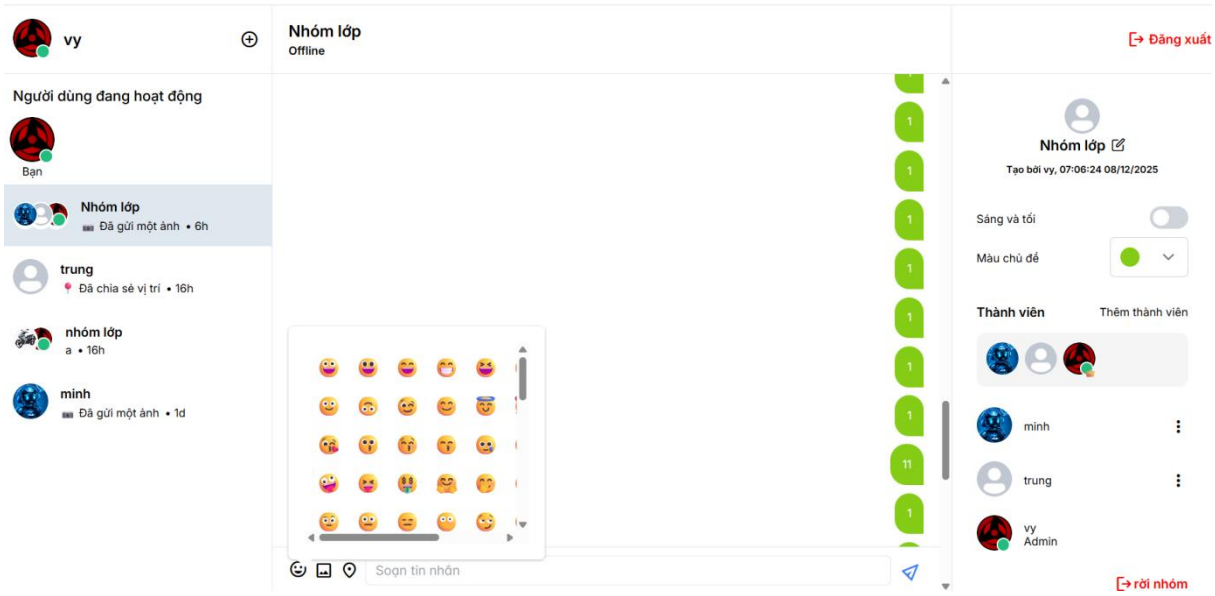
Hình 38. Chọn ảnh từ thiết bị

Và nếu như người dùng nhấn nút gửi hoặc nhấn enter thì hình ảnh sẽ được gửi đi và hiển thị trong cuộc trò chuyện hoặc nhóm.

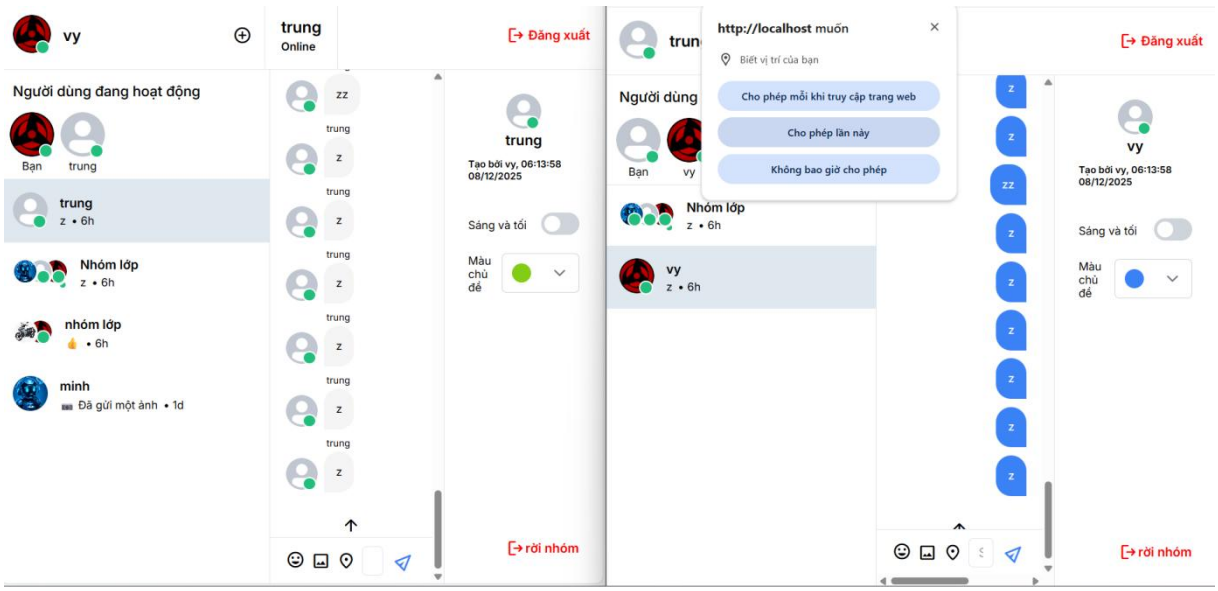


Hình 39. Gửi hình ảnh

Ngoài việc gửi tin nhắn văn hản và hình ảnh người dùng còn có thể gửi icon bằng cách nhấn vào biểu tượng icon và chọn icon mà mình muốn gửi sau đó nhấn enter hoặc nhấn nút gửi.

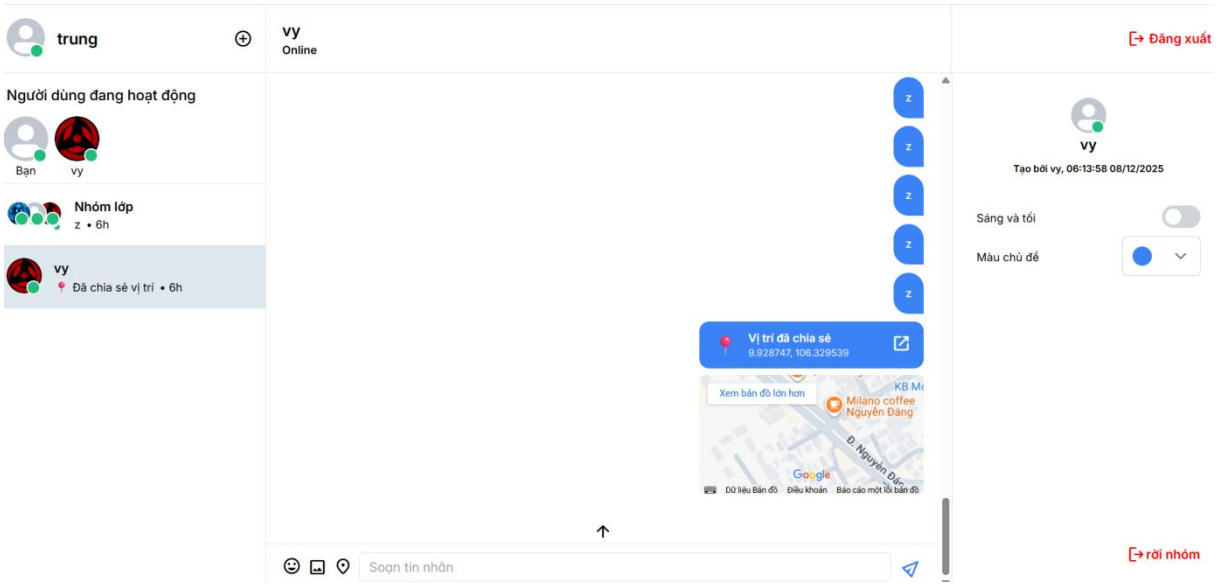


Hình 40. Gửi icon



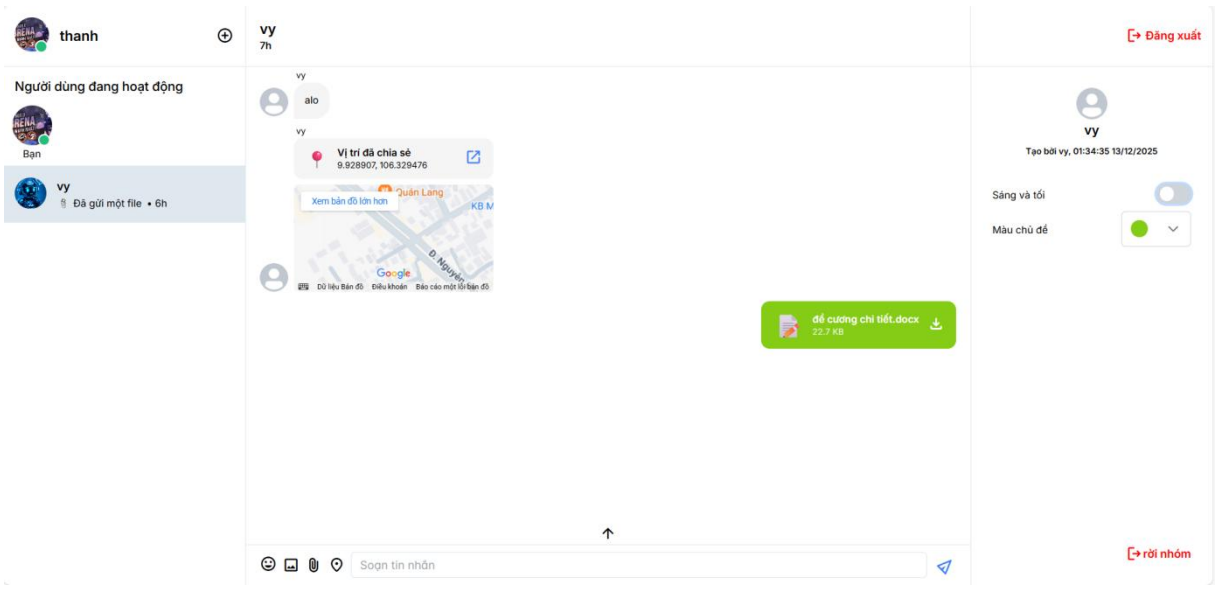
Hình 41. Thông báo cho phép truy cập

Người dùng có thể gửi định vị GPS để cho những người trong nhóm chat có thể biết được vị trí của hiện tại của mình trình duyệt lúc này sẽ hiển thị thông báo cho phép truy cập vào vị trí sau khi người nhấn cho phép thì vị trí GPS sẽ được gửi đi.



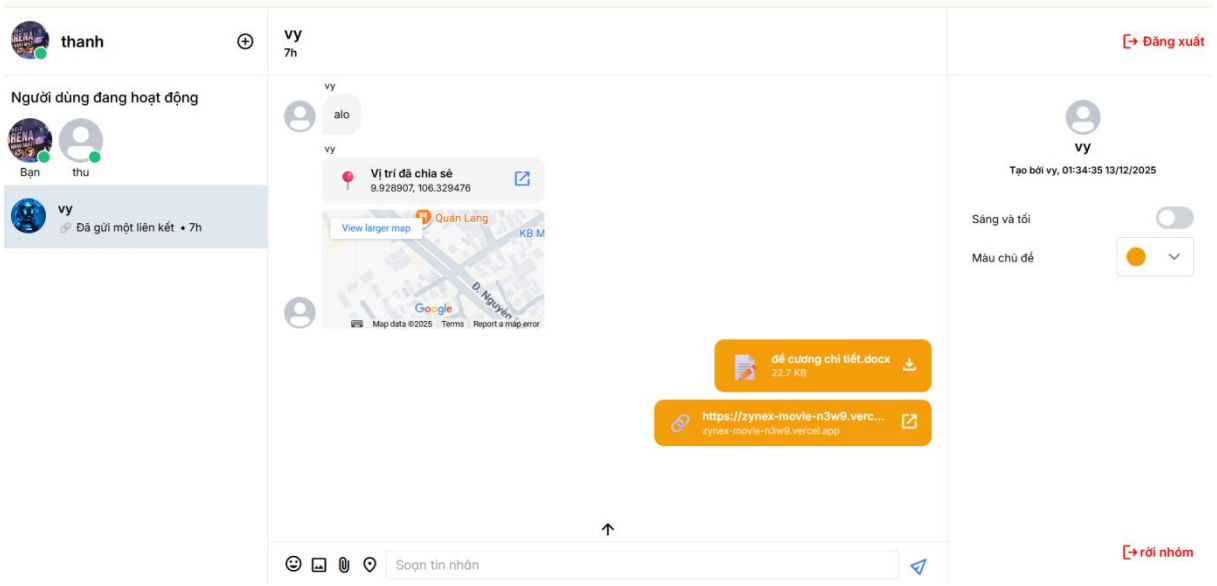
Hình 42. Vị trí GPS được gửi đi

Người dùng còn có thể gửi file từ thiết bị của mình



Hình 43. Gửi file

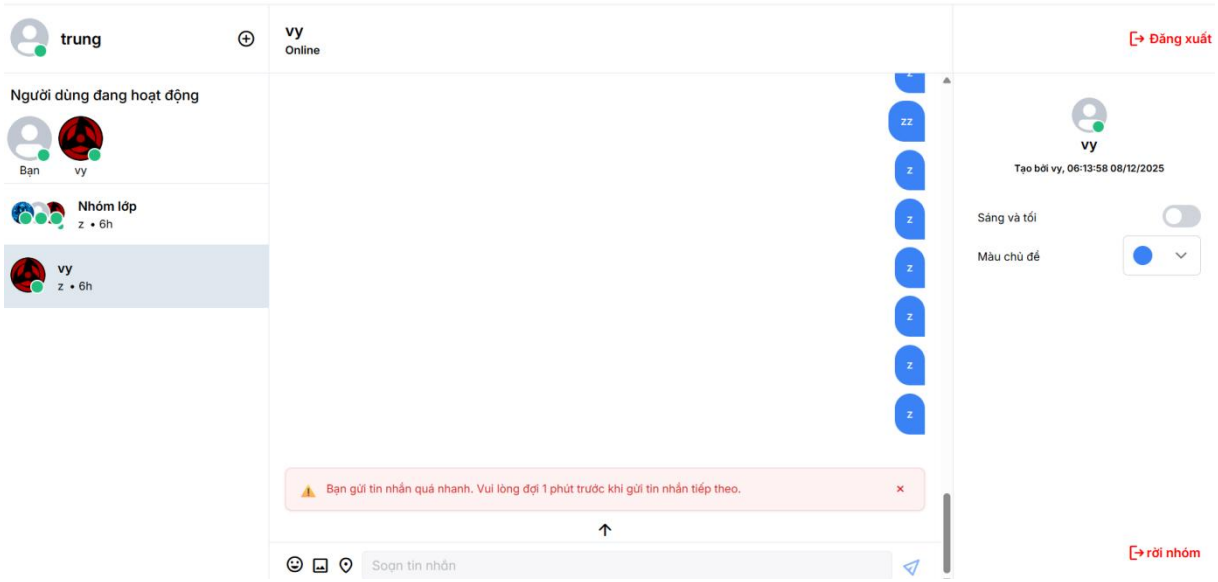
Những người trong cuộc trò chuyện có thể tải file này về máy của họ và cuối cùng người dùng có thể gửi link.



Hình 44. Gửi link

4.5.8. Chống spam tin nhắn

Ứng dụng chỉ cho phép người dùng gửi tối đa 20 tin nhắn trong một phút nếu như người dùng nhắn quá 20 tin nhắn thì sẽ bị cảnh báo và không thể gửi tin nhắn nào kể cả gửi tin nhắn văn bản, hình ảnh, icon, file, link hay là định vị GPS đều không gửi được.



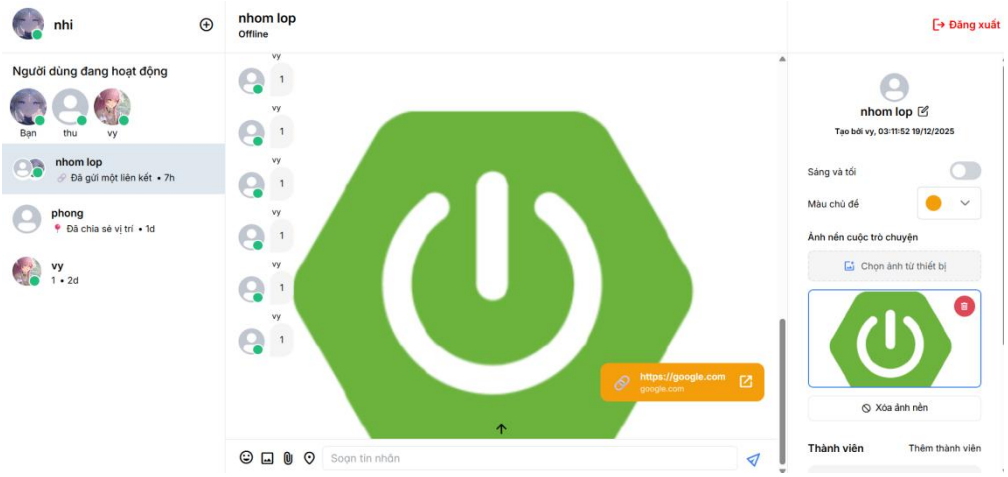
Hình 45. Thông báo đạt giới hạn

Dữ liệu được cache và giới hạn request được lưu trong bộ nhớ của redis

STRING	chatapp:onlineUsers::SimpleKey []	9 min	144 B
STRING	chatapp:lastMessage::2d549568-166c-47ae-a8bb-d032a4a0...	9 min	744 B
STRING	rate_limit:vy:3ffc42d5-c81e-4b87-bed0-2c769a5a329e	10 s	96 B
STRING	chatapp:lastMessage::ffca08ed-0db8-4828-8497-9df1850c3...	9 min	744 B
STRING	chatapp:lastMessage::7e495202-8157-4df0-a9cf-b59c1d72e...	9 min	744 B
STRING	chatapp:users::vy	9 min	448 B

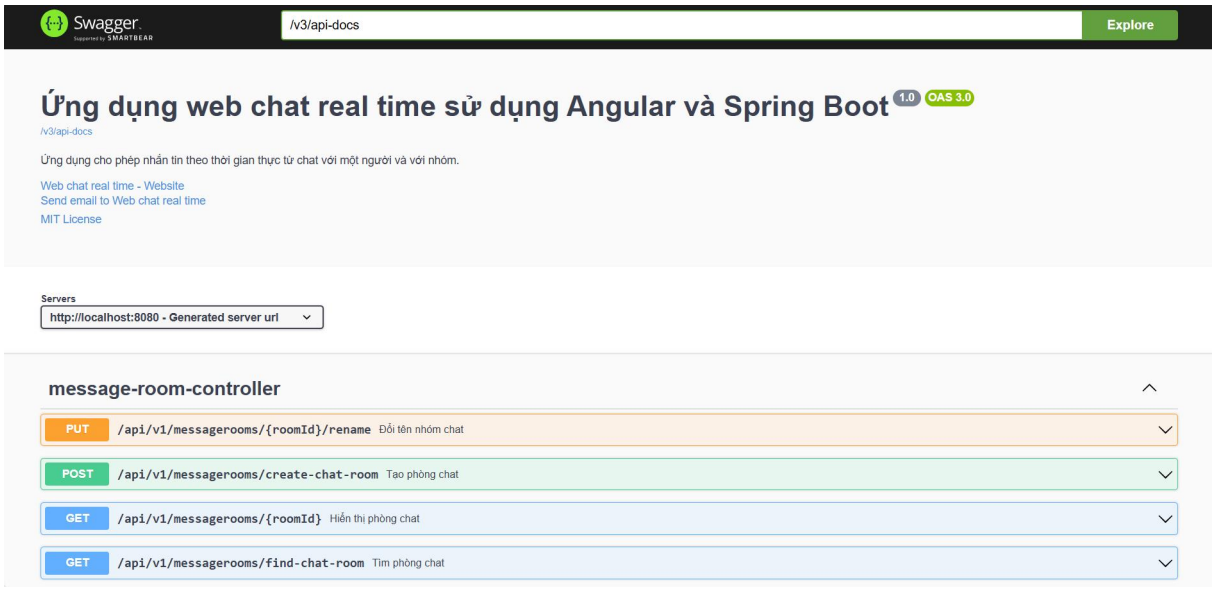
Hình 46. Dữ liệu được Cache trong Redis

4.5.9. Đặt ảnh nền cho cuộc trò chuyện



Hình 47. Ảnh nền trong phòng chat

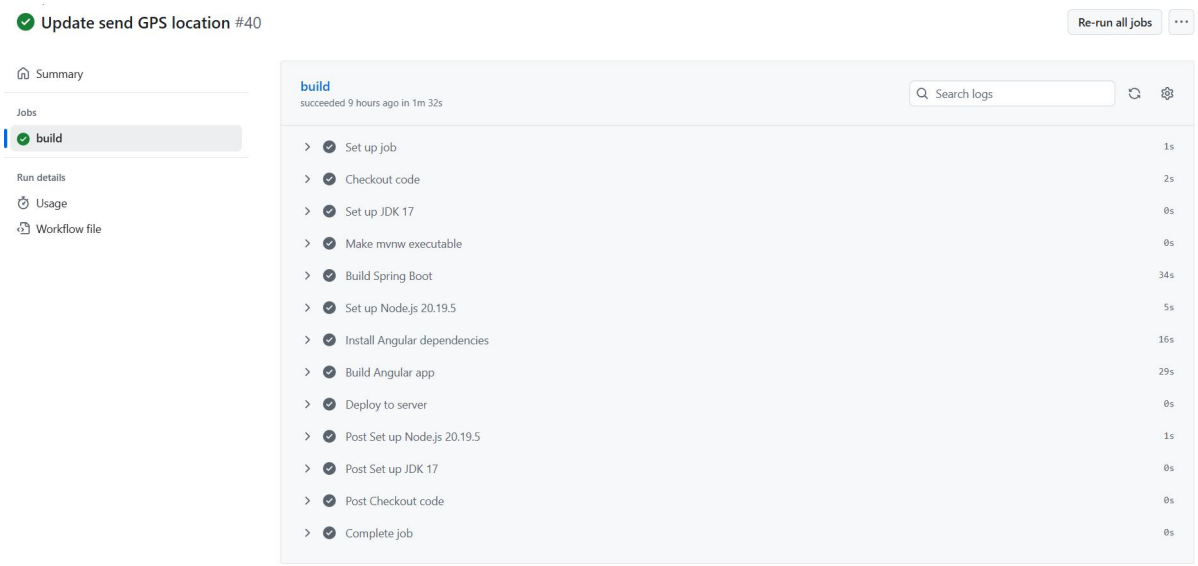
4.6. Tạo tài liệu API với Swagger



Hình 48. Tài liệu API với Swagger

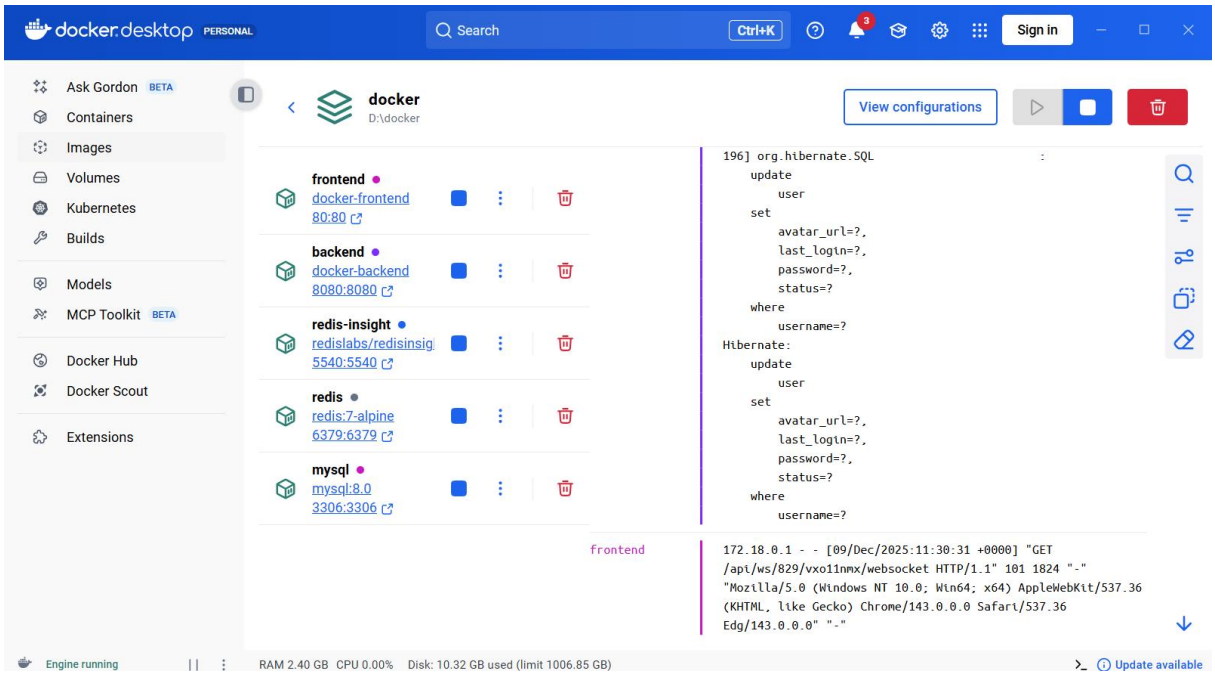
4.7. Quy trình kiểm thử tự động với Github Action

Gồm 3 giai đoạn build, test và deploy



Hình 49. CI/CD với Github Actions

4.8. Đóng gói và triển khai ứng dụng dưới dạng các container với Docker



Hình 50. Đóng gói ứng dụng với Docker

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được

Đề tài đã khảo sát các khái niệm cơ bản của WebSocket sử dụng Angular và Spring Boot để xây dựng một ứng dụng web chat theo thời gian thực bao gồm những chức năng như xác thực người dùng với JWT, mã hóa mật khẩu với Bcrypt, khóa tài khoản tạm thời trong vòng một giờ đối với trường hợp người dùng nhập sai tài khoản ba lần. Người dùng có thể tạo nhóm và quản lý nhóm, gửi tin nhắn với nhiều loại tin nhắn khác nhau như: văn bản, hình ảnh, icon, file, link, định vị GPS, chống spam tin nhắn người dùng chỉ có thể gửi tối đa hai mươi tin nhắn trong một phút nếu như người dùng gửi quá mức tin nhắn giới hạn cho phép thì sẽ không thể gửi tin nhắn và sẽ nhận được cảnh báo, tạo tài liệu API với Swagger, áp dụng quy trình kiểm thử tự động với GitHub Actions để phát hiện lỗi kịp thời trước khi triển khai lên môi trường production. Và cuối cùng ứng dụng được đóng gói dưới dạng các container để triển khai lên Docker giúp cho việc demo ứng dụng tiện lợi và tiết kiệm thời gian hơn so với việc chạy local trên môi trường phát triển.

5.2. Hạn chế

Hiện tại, ứng dụng vẫn còn một số hạn chế như chức năng emoji chỉ hỗ trợ bộ icon sẵn có mà chưa có cơ chế gợi ý emoji tự động dựa trên từ khóa nhập trong ô soạn thảo khi gửi link chỉ hiển thị URL thuần túy mà chưa có tính năng preview (xem trước tiêu đề, mô tả, hình ảnh thumbnail của trang web), một số tính năng nâng cao như thông báo đẩy khi offline hoặc xử lý tải cao chưa được tối ưu hoàn toàn, chưa hỗ trợ voice/video call hoặc các tính năng AI thông minh. Những hạn chế này chủ yếu xuất phát từ phạm vi đồ án tập trung vào phần core realtime chat.

5.3. Hướng phát triển

Trong tương lai, ứng dụng có thể được mở rộng với nhiều tính năng nâng cao hơn nữa để nâng cao trải nghiệm người dùng, tăng khả năng mở rộng hệ thống và tích hợp các công nghệ hiện đại, phù hợp với xu hướng phát triển ứng dụng chat thời gian thực năm 2025 và các năm tiếp theo. Các hướng phát triển cụ thể bao gồm: tích hợp Keycloak nhằm hỗ trợ đăng nhập bên thứ ba (như Google, Facebook, GitHub...) và

Single Sign-On (SSO) giữa nhiều ứng dụng, giúp người dùng đăng nhập một lần duy nhất mà vẫn truy cập được nhiều dịch vụ liên kết, đồng thời tăng cường bảo mật và tiện lợi; sử dụng Kafka để xử lý bất đồng bộ cho việc gửi thông báo email (quên mật khẩu, đặt lại mật khẩu, thông báo tin nhắn mới, xác nhận đăng ký) giúp giảm tải cho server chính, đảm bảo độ tin cậy cao hơn và dễ dàng mở rộng khi lượng người dùng tăng; tích hợp RabbitMQ làm hàng đợi để xử lý lượng truy cập cao, giảm tải hệ thống, tăng khả năng chịu lỗi và hỗ trợ horizontal scaling mượt mà; thay thế Redis bằng DragonflyDB – một alternative hiện đại được viết bằng C++ với kiến trúc multi-threaded, mang lại throughput cao hơn đáng kể (có benchmark cho thấy lên đến 25x nhanh hơn Redis trong một số trường hợp trên hardware multi-core), tiết kiệm tài nguyên, chi phí vận hành thấp hơn và tương thích hoàn toàn với protocol Redis hiện tại; tích hợp AI (sử dụng mô hình , OpenAI GPT) để tự động trả lời thay khi người nhận offline, gợi ý emoji thông minh dựa trên nội dung tin nhắn, phân tích sentiment để phát hiện cảm xúc người dùng và đưa ra phản hồi phù hợp, hỗ trợ dịch real-time đa ngôn ngữ, hỗ trợ WebRTC để cho phép voice call, video call và chia sẻ màn hình trực tiếp trong phòng chat (P2P với signaling qua WebSocket hiện tại, kết hợp STUN/TURN server để vượt NAT/firewall), kèm theo các tính năng nâng cao như real-time transcription (chuyển giọng nói thành văn bản bằng Whisper hoặc tương tự), noise cancellation, adaptive bitrate để tối ưu chất lượng kết nối trên mạng yếu và recording cuộc gọi nếu cần; cải thiện các UX nhỏ như thêm link preview (sử dụng OpenGraph meta tags hoặc service bên thứ ba như Microlink để hiển thị tiêu đề, mô tả, thumbnail một cách đẹp mắt), emoji picker với gợi ý tự động dựa trên từ khóa, hỗ trợ sticker/GIF tùy chỉnh từ Giphy API, voice note (ghi âm gửi nhanh) và multimodal input (gửi hình ảnh để AI mô tả hoặc xử lý)(Elasticsearch, Logstash, Kibana) cho log management, và Sentry cho error tracking deploy lên Kubernetes cluster (sử dụng Helm chart), hỗ trợ blue-green deployment hoặc canary release để giảm rủi ro khi update production. Những hướng phát triển này dễ dàng scale lên môi trường production thực tế với hàng triệu người dùng hoạt động đồng thời, đồng thời tận dụng tối đa các xu hướng công nghệ đang bùng nổ như AI multimodal, realtime communication và cloud-native architecture trong năm 2025 trở đi.

DANH MỤC TÀI LIỆU THAM KHẢO

1. **Spring Boot**

Spring Boot – Framework giúp xây dựng các ứng dụng Java độc lập, sẵn sàng cho môi trường production.

Truy cập: https://spring.io/projects/spring-boot

2. **Angular v16 Documentation**

Angular là một nền tảng và framework dùng để xây dựng các ứng dụng web dạng single-page, sử dụng HTML, CSS và TypeScript.

Truy cập: https://v16.angular.io/docs

3. **PrimeNG – Angular UI Component Library**

PrimeNG là thư viện các thành phần giao diện người dùng phong phú, hỗ trợ phát triển ứng dụng Angular một cách nhanh chóng và hiệu quả.

Truy cập: https://primeng.org/

4. **PrimeFlex – Utility-First CSS Library**

PrimeFlex là thư viện CSS theo hướng utility-first, cung cấp các lớp hỗ trợ bố cục, hiển thị và thiết kế giao diện responsive.

Truy cập: https://primeflex.org/

5. **PrimeIcons – Icon Library for PrimeNG**

PrimeIcons là thư viện biểu tượng mặc định của PrimeNG, cung cấp nhiều biểu tượng mã nguồn mở phục vụ cho các ứng dụng web.

Truy cập: https://primeng.org/icons

6. **Redis**

Redis là hệ quản trị lưu trữ dữ liệu trong bộ nhớ (in-memory), thường được sử dụng làm cơ sở dữ liệu, bộ nhớ đệm và hệ thống truyền thông điệp.

Truy cập: https://redis.io/

7. **Docker**

Docker là nền tảng hỗ trợ đóng gói, triển khai và vận hành ứng dụng thông qua công nghệ container hóa.

Truy cập: https://www.docker.com/

8. **Docker Hub**

Docker Hub là dịch vụ lưu trữ trực tuyến cho các Docker image, hỗ trợ chia sẻ và quản lý container giữa các nhà phát triển..

Truy cập: https://hub.docker.com/

9. **MySQL**

MySQL là hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, sử dụng ngôn ngữ SQL để quản lý và truy vấn dữ liệu.

Truy cập: https://www.mysql.com/