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 WEBSITE DIỄN ĐÀN SINH VIÊN

Giảng viên hướng dẫn:
ThS. Nguyễn Hoàng Duy Thiện

Sinh viên thực hiện:
Họ tên: Lê Khánh Đăng
MSSV: 110122047
Lớp: DA22TTA

Vĩnh Long, tháng 12 năm 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

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

Vĩnh Long, ngày 29 tháng 12 năm 2025

Giảng viên hướng dẫn

(Ký tên và ghi rõ họ tên)

Nguyễn Hoàng Duy Thiện

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

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc đến quý thầy cô trong khoa đã tận tình giảng dạy, truyền đạt kiến thức và tạo điều kiện thuận lợi cho em trong suốt quá trình học tập tại trường cũng như trong quá trình thực hiện đồ án này.

Em xin chân thành cảm ơn thầy hướng dẫn đã luôn dành thời gian, tâm huyết để hướng dẫn, góp ý và giải đáp những thắc mắc của em trong suốt quá trình nghiên cứu và thực hiện đề tài. Những ý kiến quý báu của thầy đã giúp em hiểu rõ hơn về nội dung đề tài, từ đó có thể hoàn thiện đồ án theo đúng mục tiêu và yêu cầu đề ra.

Em cũng xin gửi lời cảm ơn đến gia đình và bạn bè đã luôn động viên, khích lệ và hỗ trợ em trong quá trình học tập cũng như khi thực hiện đồ án. Sự quan tâm và giúp đỡ của mọi người là nguồn động lực to lớn giúp em vượt qua những khó khăn và hoàn thành đề tài đúng tiến độ.

Mặc dù đã rất cố gắng, nhưng do kiến thức và thời gian có hạn, đồ án không tránh khỏi những thiếu sót. Em rất mong nhận được những ý kiến đóng góp quý báu từ quý thầy cô và các bạn để đề tài có thể được hoàn thiện hơn trong tương lai.

Cuối cùng, em xin chân thành cảm ơn và kính chúc quý thầy cô cùng các bạn luôn dồi dào sức khỏe và thành công trong công tác và học tập.

Trân trọng,

Vĩnh Long, ngày ..... tháng …… năm ……

Sinh viên thực hiện

(Ký tên và ghi rõ họ tên)

Lê Khánh Đăng

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Đề tài "Xây dựng Website diễn đàn sinh viên" được thực hiện nhằm đáp ứng nhu cầu trao đổi học tập, chia sẻ thông tin và kết nối cộng đồng sinh viên trong môi trường trực tuyến. Trong bối cảnh chuyển đổi số giáo dục, nền tảng diễn đàn giúp sinh viên dễ dàng thảo luận, đặt câu hỏi, chia sẻ tài liệu và cập nhật thông tin là rất cần thiết.

Đồ án tập trung nghiên cứu các mô hình diễn đàn trực tuyến, phân tích yêu cầu người dùng và đề xuất giải pháp xây dựng một website diễn đàn sinh viên với các chức năng cơ bản như: đăng ký - đăng nhập tài khoản, đăng bài viết, bình luận, tìm kiếm bài viết, quản lý chủ đề và quản trị nội dung.

Hướng tiếp cận của đề tài là xây dựng hệ thống theo mô hình client - server, tách biệt giữa giao diện người dùng và xử lý phía máy chủ, đảm bảo khả năng mở rộng và bảo trì. Cơ sở dữ liệu được thiết kế để lưu trữ thông tin người dùng, bài viết, bình luận và các hoạt động liên quan.

Kết quả đạt được của đồ án là xây dựng được một website diễn đàn sinh viên hoạt động ổn định, giao diện thân thiện, đáp ứng được các nhu cầu trao đổi thông tin cơ bản của sinh viên, đồng thời là nền tảng có thể tiếp tục mở rộng thêm các chức năng trong tương lai.

MỞ ĐẦU

Lý do chọn đề tài

Trong bối cảnh chuyển đổi số mạnh mẽ hiện nay, việc ứng dụng công nghệ thông tin vào hoạt động học tập và trao đổi kiến thức của sinh viên ngày càng trở nên phổ biến và cần thiết. Sinh viên không chỉ học tập trên lớp mà còn có nhu cầu thảo luận bài tập, chia sẻ tài liệu, kinh nghiệm học tập, thông tin học vụ và các vấn đề trong đời sống sinh viên thông qua môi trường trực tuyến.

Hiện nay, các kênh trao đổi phổ biến như mạng xã hội hoặc các nhóm chat mang tính tiện lợi cao nhưng tồn tại nhiều hạn chế, đặc biệt là việc thông tin không được tổ chức khoa học, khó phân loại theo chủ đề, khó tìm kiếm lại nội dung cũ và không phù hợp cho việc lưu trữ lâu dài. Ngoài ra, các nền tảng này còn dễ gây nhiễu thông tin, thiếu tính học thuật và khó kiểm soát nội dung.

Xuất phát từ thực tế đó, việc xây dựng một website diễn đàn sinh viên chuyên biệt là cần thiết. Website diễn đàn không chỉ giúp sinh viên có một không gian trao đổi học tập tập trung, có tổ chức mà còn góp phần tạo ra môi trường giao lưu lành mạnh, thúc đẩy tinh thần tự học, hợp tác và chia sẻ kiến thức trong cộng đồng sinh viên. Đồng thời, đề tài cũng giúp sinh viên áp dụng kiến thức đã học vào một sản phẩm thực tế, có ý nghĩa ứng dụng cao.

Mục đích nghiên cứu

Mục đích của đề tài là nghiên cứu và xây dựng một hệ thống website diễn đàn sinh viên nhằm phục vụ nhu cầu trao đổi, thảo luận và chia sẻ thông tin trong môi trường học tập.

Cụ thể, đề tài hướng đến các mục tiêu sau:

- Nghiên cứu mô hình hoạt động, cấu trúc chức năng và cách tổ chức nội dung của các website diễn đàn trực tuyến hiện nay.

- Phân tích nhu cầu thực tế của sinh viên khi tham gia các hoạt động trao đổi học tập trên môi trường trực tuyến.

- Xây dựng một website diễn đàn sinh viên với các chức năng cơ bản như đăng ký - đăng nhập, đăng bài viết, bình luận, tìm kiếm và quản lý nội dung.

- Vận dụng và củng cố các kiến thức đã học về lập trình web, cơ sở dữ liệu, thiết kế hệ thống và mô hình client - server vào việc xây dựng một sản phẩm phần mềm hoàn chỉnh.

Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài bao gồm:

- Các hệ thống diễn đàn trực tuyến đang được sử dụng phổ biến hiện nay, từ đó phân tích ưu điểm, hạn chế và rút ra bài học áp dụng cho hệ thống đề tài.

- Sinh viên, là nhóm người dùng chính của website diễn đàn, bao gồm nhu cầu, hành vi và cách thức sử dụng diễn đàn trong học tập và trao đổi thông tin.

- Các công nghệ, công cụ và nền tảng phát triển website, bao gồm công nghệ lập trình web phía máy chủ, giao diện người dùng và hệ quản trị cơ sở dữ liệu.

Phạm vi nghiên cứu

Trong khuôn khổ của đồ án, đề tài tập trung nghiên cứu và xây dựng website diễn đàn sinh viên với các chức năng cơ bản, đáp ứng nhu cầu trao đổi thông tin và học tập của sinh viên, bao gồm:

- Quản lý người dùng và tài khoản (đăng ký, đăng nhập, phân quyền).

- Quản lý bài viết và bình luận theo chủ đề.

- Tìm kiếm và hiển thị thông tin trên diễn đàn.

Đề tài không đi sâu nghiên cứu và triển khai các chức năng nâng cao như trí tuệ nhân tạo, phân tích hành vi người dùng, hệ thống gợi ý thông minh hay tối ưu hóa ở quy mô lớn. Các nội dung này có thể được xem là hướng phát triển mở rộng trong các nghiên cứu và đồ án tiếp theo.

CHƯƠNG 1: TỔNG QUAN

1.1. Tổng quan về website diễn đàn

Website diễn đàn (Forum Website) là một hệ thống thông tin trực tuyến cho phép người dùng đăng tải nội dung, trao đổi ý kiến và thảo luận xoay quanh các chủ đề cụ thể. Nội dung trên diễn đàn thường được tổ chức theo cấu trúc phân cấp gồm các chuyên mục (category), chủ đề (topic) và bài viết (post). Cấu trúc này giúp người dùng dễ dàng tiếp cận thông tin, đồng thời hỗ trợ việc quản lý, lưu trữ và tìm kiếm dữ liệu một cách khoa học.

Khác với các nền tảng mạng xã hội, website diễn đàn không tập trung vào yếu tố giải trí hay tương tác tức thời mà hướng đến việc xây dựng kho tri thức chung, nơi các nội dung có giá trị được lưu trữ lâu dài và có thể tái sử dụng. Nhờ đó, diễn đàn trở thành công cụ hữu ích trong việc chia sẻ kiến thức, hỗ trợ học tập, nghiên cứu và trao đổi chuyên môn.

Trong những năm gần đây, cùng với sự phát triển của công nghệ web, các website diễn đàn ngày càng được cải tiến về giao diện, hiệu năng và khả năng tương tác, đáp ứng tốt hơn nhu cầu của người dùng trong nhiều lĩnh vực khác nhau, đặc biệt là lĩnh vực giáo dục.

1.2. Diễn đàn sinh viên trong bối cảnh chuyển đổi số giáo dục

Chuyển đổi số trong giáo dục đang diễn ra mạnh mẽ, kéo theo sự thay đổi trong phương thức dạy và học. Sinh viên không còn phụ thuộc hoàn toàn vào lớp học truyền thống mà ngày càng chủ động tiếp cận kiến thức thông qua các nền tảng trực tuyến. Trong bối cảnh đó, diễn đàn sinh viên đóng vai trò là một môi trường học tập mở, hỗ trợ hiệu quả cho quá trình tự học và trao đổi kiến thức.

Diễn đàn sinh viên cho phép sinh viên:

- Trao đổi và thảo luận bài tập, nội dung môn học ngoài giờ lên lớp.
- Đặt câu hỏi và nhận phản hồi từ các sinh viên khác hoặc người có kinh nghiệm.
- Chia sẻ tài liệu học tập, kinh nghiệm học tập và kỹ năng mềm.
- Cập nhật các thông tin học vụ, hoạt động đoàn - hội và đời sống sinh viên.

Thông qua các hoạt động này, diễn đàn góp phần nâng cao chất lượng học tập, thúc đẩy tinh thần học hỏi, hợp tác và chia sẻ trong cộng đồng sinh viên. Đồng thời, diễn đàn cũng là công cụ hỗ trợ giảng viên trong việc nắm bắt khó khăn của sinh viên và định hướng nội dung học tập phù hợp hơn.

1.3. Thực trạng trao đổi thông tin của sinh viên hiện nay

Hiện nay, phần lớn sinh viên sử dụng các mạng xã hội hoặc ứng dụng nhắn tin để trao đổi thông tin học tập. Các nền tảng này có ưu điểm là nhanh chóng, tiện lợi và dễ tiếp cận. Tuy nhiên, khi áp dụng vào môi trường học tập lâu dài, các công cụ này bộc lộ nhiều hạn chế.

Thứ nhất, thông tin trao đổi trên mạng xã hội thường không được phân loại rõ ràng theo môn học hoặc chủ đề, dẫn đến tình trạng nội dung bị trôi, khó tìm kiếm lại sau một thời gian.

Thứ hai, các nhóm chat thường chứa nhiều thông tin không liên quan, gây nhiễu và làm giảm hiệu quả học tập.

Thứ ba, việc lưu trữ tài liệu và kiến thức trên các nền tảng này không mang tính hệ thống, thiếu tính học thuật và khó kiểm soát chất lượng nội dung.

Những hạn chế trên gây khó khăn cho sinh viên trong việc tổng hợp kiến thức, tra cứu thông tin và học tập. Do đó, cần có một nền tảng trao đổi học tập chuyên biệt, được thiết kế phù hợp với mục đích giáo dục và nhu cầu thực tế của sinh viên.

1.4. Vấn đề cần giải quyết

Từ thực trạng nêu trên, có thể nhận thấy vấn đề cốt lõi cần giải quyết là thiếu một môi trường trao đổi học tập trực tuyến có tổ chức, chuyên biệt và phù hợp với sinh viên. Một website diễn đàn sinh viên cần đáp ứng các yêu cầu sau:

- Nội dung được phân loại rõ ràng theo chuyên mục và chủ đề.

- Hỗ trợ trao đổi học tập hiệu quả thông qua bài viết và bình luận.

- Dễ dàng tìm kiếm, tra cứu và lưu trữ thông tin lâu dài.

- Có cơ chế quản lý nội dung và phân quyền người dùng phù hợp.

Việc giải quyết những vấn đề này sẽ góp phần nâng cao hiệu quả học tập, đồng thời tạo ra một cộng đồng sinh viên tích cực, có định hướng và mang tính học thuật.

1.5. Định hướng giải pháp của đề tài

Xuất phát từ các vấn đề đã phân tích, đề tài "Xây dựng Website diễn đàn sinh viên" đề xuất giải pháp xây dựng một hệ thống diễn đàn trực tuyến với kiến trúc rõ ràng, chức năng phù hợp và dễ sử dụng.

Hệ thống được xây dựng theo mô hình client - server, trong đó:

- Phía client đảm nhiệm việc hiển thị giao diện và tương tác với người dùng.

- Phía server xử lý nghiệp vụ, xác thực người dùng và quản lý dữ liệu.

- Cơ sở dữ liệu dùng để lưu trữ thông tin người dùng, bài viết, bình luận và các hoạt động liên quan.

Giải pháp này nhằm đảm bảo hệ thống hoạt động ổn định, dễ bảo trì và có khả năng mở rộng trong tương lai khi nhu cầu sử dụng tăng lên.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1. Giới thiệu về ReactJS

2.1.1. ReactJS là gì ?

ReactJS là một thư viện JavaScript mã nguồn mở do Facebook (nay là Meta) phát triển, được sử dụng để xây dựng giao diện người dùng (User Interface - UI) cho các ứng dụng web hiện đại. ReactJS cho phép xây dựng giao diện theo hướng component - based, trong đó giao diện được chia thành các thành phần nhỏ, độc lập và có thể tái sử dụng.

ReactJS thường được dùng để phát triển các ứng dụng web đơn trang (Single Page Application - SPA), nơi toàn bộ ứng dụng chỉ tải một lần ban đầu và sau đó cập nhật nội dung động mà không cần tải lại trang. Điều này giúp cải thiện tốc độ phản hồi và nâng cao trải nghiệm người dùng [1], [2].

Trong đề tài "Xây dựng Website diễn đàn sinh viên", ReactJS được sử dụng để xây dựng giao diện cho người dùng, hỗ trợ các chức năng như đăng ký - đăng nhập, hiển thị danh sách bài viết, xem chi tiết bài viết, đăng bài và bình luận.

2.1.2. Virtual DOM

Document Object Model (DOM) là một cấu trúc trừu tượng biểu diễn nội dung của tài liệu HTML dưới dạng cây, trong đó mỗi phần tử HTML được xem như một node trong cây DOM. Cây DOM mô phỏng toàn bộ cấu trúc của một trang web và cho phép trình duyệt thao tác, cập nhật giao diện dựa trên các thay đổi của dữ liệu.

Trong các ứng dụng web truyền thống, khi dữ liệu thay đổi, trình duyệt sẽ thao tác trực tiếp lên DOM thật. Mặc dù DOM được xử lý tương đối nhanh, nhưng đối với các ứng dụng web đơn trang (Single Page Application - SPA), việc cập nhật DOM liên tục, đặc biệt là khi ứng dụng có quy mô lớn và nhiều tương tác người dùng, sẽ gây tiêu tốn nhiều tài nguyên và làm giảm hiệu suất hệ thống.

Để giải quyết vấn đề này, ReactJS sử dụng Virtual DOM (DOM ảo), là một bản sao nhẹ của DOM thật được lưu trữ trong bộ nhớ. Virtual DOM không trực tiếp hiển thị trên trình duyệt mà đóng vai trò trung gian trong quá trình cập nhật giao diện.

Khi trạng thái (state) hoặc dữ liệu của ứng dụng ReactJS thay đổi, ReactJS sẽ tạo ra một Virtual DOM mới. Sau đó, ReactJS sử dụng thuật toán so sánh (diffing

algorithm) để so sánh Virtual DOM mới với Virtual DOM trước đó, nhằm xác định chính xác những phần tử nào đã thay đổi. Dựa trên kết quả so sánh này, ReactJS chỉ cập nhật các phần tử cần thiết lên DOM thật thông qua quá trình gọi là reconciliation.

Cơ chế này giúp giảm đáng kể số lần thao tác trực tiếp lên DOM thật, từ đó cải thiện hiệu năng, tăng tốc độ xử lý và nâng cao trải nghiệm người dùng [1]. Đặc biệt, đối với các website lớn, phức tạp và có nhiều tương tác như website diễn đàn sinh viên, việc sử dụng Virtual DOM giúp hệ thống hoạt động ổn định và hiệu quả hơn.

Ngược lại, khi không sử dụng ReactJS và Virtual DOM, mỗi thay đổi trong giao diện có thể yêu cầu cập nhật lại toàn bộ hoặc phần lớn cây DOM, dẫn đến tiêu tốn nhiều tài nguyên tính toán và làm chậm quá trình hiển thị. Với ReactJS, hệ thống chỉ cập nhật đúng những vị trí cần thay đổi, giúp tiết kiệm tài nguyên và rút ngắn thời gian phản hồi của ứng dụng.

2.1.3. JSX

JSX (JavaScript XML) là một cú pháp mở rộng của JavaScript, cho phép lập trình viên viết mã có cấu trúc tương tự HTML ngay trong mã JavaScript [1]. JSX không phải là HTML thuần mà là một cú pháp đặc biệt được ReactJS sử dụng để mô tả cấu trúc giao diện người dùng một cách trực quan và dễ hiểu.

Về bản chất, JSX được biên dịch thành các lệnh JavaScript thông thường thông qua các công cụ như Babel trước khi được trình duyệt thực thi [1]. Do đó, trình duyệt không trực tiếp hiểu JSX mà chỉ xử lý mã JavaScript đã được biên dịch.

Việc sử dụng JSX mang lại nhiều lợi ích trong quá trình phát triển ứng dụng ReactJS. Thứ nhất, JSX giúp kết hợp chặt chẽ giữa logic xử lý và giao diện trong cùng một component, từ đó giảm sự tách rời giữa HTML, CSS và JavaScript như trong các mô hình phát triển web truyền thống. Thứ hai, cú pháp gần giống HTML giúp mã nguồn dễ đọc, dễ bảo trì và dễ phát hiện lỗi hơn.

Trong JSX, lập trình viên có thể nhúng các biểu thức JavaScript bằng cách sử dụng dấu ngoặc nhọn { }. Điều này cho phép hiển thị dữ liệu động, xử lý điều kiện và lặp danh sách ngay trong phần giao diện. Tuy nhiên, JSX cũng có một số quy tắc riêng, chẳng hạn như:

- Mỗi component chỉ được trả về một phần tử gốc.

- Các thuộc tính HTML phải tuân theo cú pháp camelCase (ví dụ: className thay cho class).

- Các thẻ phải được đóng đầy đủ.

JSX là một thành phần quan trọng trong ReactJS, giúp đơn giản hóa quá trình xây dựng giao diện người dùng, đồng thời nâng cao tính rõ ràng và khả năng bảo trì của mã nguồn trong các ứng dụng web hiện đại [1].

2.1.4. Components

Component là đơn vị cơ bản trong ReactJS, đại diện cho một phần giao diện người dùng cụ thể [1], [2]. Mỗi component có thể bao gồm cấu trúc giao diện, logic xử lý và dữ liệu liên quan. Thay vì xây dựng giao diện dưới dạng một khối lớn, ReactJS cho phép chia giao diện thành nhiều component nhỏ, độc lập và có thể tái sử dụng.

Trong ReactJS, mỗi component được xem như một hàm hoặc một lớp JavaScript, có nhiệm vụ nhận dữ liệu đầu vào và trả về giao diện tương ứng. Việc chia ứng dụng thành các component giúp mã nguồn trở nên rõ ràng, dễ quản lý và dễ mở rộng khi hệ thống phát triển.

Component trong ReactJS có thể được phân loại thành hai loại chính:

- Functional Component: là các hàm JavaScript đơn giản, thường được sử dụng trong các phiên bản ReactJS hiện đại. Functional Component dễ viết, dễ hiểu và kết hợp tốt với các cơ chế quản lý trạng thái.

- Class Component: là các lớp JavaScript kế thừa từ lớp React.Component. Loại component này được sử dụng phổ biến trong các phiên bản ReactJS cũ, hiện nay ít được sử dụng hơn trong các dự án mới.

Một trong những ưu điểm lớn của component là khả năng tái sử dụng. Một component có thể được sử dụng nhiều lần trong các vị trí khác nhau của ứng dụng, giúp giảm sự trùng lặp mã nguồn và tăng hiệu quả phát triển. Ngoài ra, việc chia giao diện thành các component nhỏ giúp dễ dàng kiểm thử và bảo trì hệ thống.

2.1.5. Props & State

Trong ReactJS, Props (Properties) và State là hai khái niệm cốt lõi được sử dụng để quản lý và truyền dữ liệu giữa các component [1]. Việc hiểu rõ sự khác nhau giữa props và state giúp xây dựng ứng dụng có cấu trúc rõ ràng, dễ bảo trì và mở rộng.

Props là các tham số được truyền từ component cha xuống component con nhằm cung cấp dữ liệu hoặc cấu hình cho component. Props có tính chất chỉ đọc, nghĩa là component con không được phép thay đổi giá trị của props. Nhờ cơ chế này, ReactJS đảm bảo luồng dữ liệu một chiều, giúp việc kiểm soát và dự đoán trạng thái của ứng dụng trở nên đơn giản hơn.

State là dữ liệu nội bộ của component, được sử dụng để lưu trữ các thông tin có thể thay đổi trong quá trình tương tác với người dùng. Khi state thay đổi, ReactJS sẽ tự động cập nhật lại giao diện tương ứng thông qua cơ chế Virtual DOM. State thường được dùng để quản lý các trạng thái động như dữ liệu nhập từ biểu mẫu, trạng thái đăng nhập hoặc danh sách bài viết được cập nhật.

Sự khác biệt cơ bản giữa props và state có thể được tóm tắt như sau:

- Props được truyền từ component cha và không thể thay đổi bên trong component con.

- State do chính component quản lý và có thể thay đổi trong quá trình thực thi.

- Props giúp các component giao tiếp với nhau, trong khi state giúp quản lý dữ liệu nội bộ của component.

2.1.6. React Router

React Router là một thư viện phổ biến được sử dụng trong ReactJS để quản lý điều hướng (routing) giữa các trang trong ứng dụng web đơn trang (Single Page Application - SPA) [1]. Thay vì tải lại toàn bộ trang khi người dùng chuyển sang một trang khác, React Router cho phép thay đổi nội dung hiển thị dựa trên đường dẫn URL mà không làm gián đoạn trải nghiệm người dùng.

Trong các ứng dụng ReactJS, React Router đóng vai trò ánh xạ giữa đường dẫn URL và component tương ứng. Khi người dùng truy cập một URL cụ thể, React Router sẽ xác định component nào cần được hiển thị. Cơ chế này giúp ứng dụng có cấu trúc rõ ràng, dễ quản lý và mở rộng.

React Router cung cấp nhiều thành phần hỗ trợ điều hướng, trong đó phổ biến nhất là:

- BrowserRouter: quản lý lịch sử điều hướng của ứng dụng dựa trên URL.

- Route: định nghĩa mối quan hệ giữa một đường dẫn và component tương ứng.

- Link / NavLink: tạo liên kết điều hướng giữa các trang mà không cần tải lại.

- useParams, useNavigate: hỗ trợ lấy tham số từ URL và điều hướng.

Việc sử dụng React Router giúp tổ chức ứng dụng theo dạng nhiều trang logic trong cùng một ứng dụng SPA, đồng thời đảm bảo tính thân thiện với người dùng và dễ phát triển. Ngoài ra, React Router còn hỗ trợ truyền tham số trên URL, giúp hiển thị các nội dung động như trang chi tiết bài viết hoặc trang hồ sơ người dùng.

Ngoài ra, ReactJS có hệ sinh thái phong phú với nhiều thư viện hỗ trợ như React Router, Redux, Axios,… cùng cộng đồng lớn, giúp lập trình viên dễ dàng tìm kiếm tài liệu và giải pháp trong quá trình phát triển.

2.1.7. Ưu điểm

ReactJS mang lại nhiều ưu điểm nổi bật trong quá trình phát triển các ứng dụng web hiện đại, đặc biệt là các hệ thống có mức độ tương tác cao như website diễn đàn sinh viên.

Thứ nhất, ReactJS áp dụng mô hình component - based, cho phép chia giao diện người dùng thành các thành phần nhỏ, độc lập và có thể tái sử dụng. Điều này giúp mã nguồn trở nên rõ ràng, dễ quản lý, giảm sự trùng lặp và tăng hiệu quả phát triển.

Thứ hai, cơ chế Virtual DOM giúp ReactJS tối ưu hiệu năng bằng cách chỉ cập nhật những phần giao diện thực sự thay đổi thay vì cập nhật toàn bộ DOM thật. Nhờ đó, ứng dụng hoạt động mượt mà hơn, đặc biệt khi có nhiều tương tác người dùng và dữ liệu thay đổi liên tục.

Thứ ba, ReactJS hỗ trợ tốt việc xây dựng Single Page Application (SPA), giúp giảm số lần tải lại trang, tăng tốc độ phản hồi và nâng cao trải nghiệm người dùng. Đây là ưu điểm quan trọng đối với các hệ thống diễn đàn, nơi người dùng thường xuyên xem bài viết, bình luận và chuyển đổi giữa các trang nội dung [1], [2].

2.1.8. Nhược điểm

Bên cạnh các ưu điểm, ReactJS cũng tồn tại một số nhược điểm nhất định.

Thứ nhất, ReactJS chỉ là một thư viện giao diện, không phải là một framework hoàn chỉnh. Do đó, khi xây dựng một ứng dụng hoàn chỉnh, lập trình viên cần kết hợp thêm nhiều thư viện khác để xử lý các chức năng như routing, quản lý trạng thái hay gọi API, dẫn đến việc tăng độ phức tạp trong quá trình phát triển ban đầu.

Thứ hai, cú pháp JSX có thể gây khó khăn cho người mới bắt đầu, đặc biệt là những người chưa quen với JavaScript hiện đại. Việc kết hợp giữa HTML và JavaScript trong cùng một file đòi hỏi lập trình viên cần thời gian để làm quen.

Thứ ba, ReactJS yêu cầu kiến thức tương đối tốt về JavaScript ES6+, bao gồm các khái niệm như arrow function, destructuring, hook,… Điều này có thể là rào cản đối với người mới tiếp cận lập trình web.

Tuy nhiên, các nhược điểm trên có thể được khắc phục thông qua việc học tập và thực hành, và không làm giảm đi tính hiệu quả của ReactJS trong việc xây dựng các ứng dụng web hiện đại.

2.2. Giới thiệu về Node.js

2.2.1. NodeJS là gì?

NodeJS là một nền tảng (runtime environment) cho phép thực thi mã JavaScript ở phía máy chủ, được xây dựng dựa trên V8 JavaScript Engine của Google. Trước khi NodeJS ra đời, JavaScript chủ yếu chỉ được sử dụng ở phía trình duyệt để xử lý giao diện người dùng. Sự xuất hiện của NodeJS đã mở rộng phạm vi sử dụng của JavaScript, cho phép lập trình viên xây dựng các ứng dụng phía máy chủ, các dịch vụ web và API một cách hiệu quả.

Một đặc điểm quan trọng của NodeJS là việc sử dụng mô hình bất đồng bộ (asynchronous) và hướng sự kiện (event - driven). Theo mô hình này, các tác vụ như truy vấn cơ sở dữ liệu hoặc xử lý yêu cầu từ người dùng không làm chặn luồng xử lý chính, mà được thực hiện thông qua các cơ chế như callback, promise hoặc async/await. Nhờ đó, NodeJS có khả năng xử lý nhiều yêu cầu đồng thời với hiệu năng cao.

Mặc dù NodeJS hoạt động theo mô hình single-threaded, nhưng nó sử dụng cơ chế event loop để quản lý và điều phối các tác vụ bất đồng bộ. Cơ chế này cho phép NodeJS xử lý hàng nghìn kết nối cùng lúc mà không cần tạo nhiều luồng xử lý riêng biệt, từ đó giúp tiết kiệm tài nguyên hệ thống và nâng cao hiệu suất hoạt động.

Trong kiến trúc ứng dụng web, NodeJS thường được sử dụng để xây dựng phần backend của hệ thống. NodeJS tiếp nhận các yêu cầu từ phía client, xử lý logic nghiệp vụ, tương tác với cơ sở dữ liệu và trả kết quả về cho client thông qua các API. Với hiệu năng cao, khả năng mở rộng tốt và cộng đồng hỗ trợ lớn, NodeJS phù hợp cho các ứng dụng web có lượng truy cập lớn và yêu cầu thời gian phản hồi nhanh [3].

Trong đề tài Xây dựng Website diễn đàn sinh viên, NodeJS đóng vai trò là nền tảng xây dựng hệ thống backend, đảm nhiệm các chức năng như xác thực người dùng, quản lý bài viết, bình luận và phân quyền người dùng. Việc sử dụng NodeJS góp phần đảm bảo hệ thống hoạt động ổn định và có khả năng mở rộng trong tương lai.

2.2.2. Giới thiệu về Express.js

Express.js là một framework mã nguồn mở, nhẹ và phổ biến được xây dựng trên nền tảng NodeJS, nhằm hỗ trợ việc phát triển các ứng dụng web và dịch vụ API một cách nhanh chóng và hiệu quả. Express.js cung cấp các công cụ và cấu trúc cần thiết để xử lý các yêu cầu HTTP, định tuyến (routing) và quản lý middleware, giúp đơn giản hóa quá trình xây dựng hệ thống backend [4].

Một trong những đặc điểm nổi bật của Express.js là kiến trúc middleware, cho phép xử lý các yêu cầu theo từng bước thông qua chuỗi middleware. Mỗi middleware đảm nhiệm một chức năng cụ thể như xác thực người dùng, kiểm tra dữ liệu đầu vào, xử lý lỗi hoặc ghi log hệ thống. Cách tiếp cận này giúp mã nguồn rõ ràng, dễ quản lý và dễ mở rộng khi bổ sung các chức năng mới.

Express.js hỗ trợ mạnh mẽ việc xây dựng RESTful API, cho phép định nghĩa các endpoint tương ứng với các phương thức HTTP như GET, POST, PUT và DELETE. Điều này rất phù hợp với kiến trúc client - server, trong đó frontend (ReactJS) giao tiếp với backend thông qua các API. Nhờ Express.js, việc tổ chức và quản lý các API trở nên linh hoạt và hiệu quả hơn.

2.2.3. Socket. IO

Socket.IO là một thư viện hỗ trợ giao tiếp thời gian thực (real-time) giữa client và server, cho phép trao đổi dữ liệu hai chiều (full-duplex) thông qua một kết nối liên tục [5]. Socket.IO được xây dựng dựa trên giao thức WebSocket và các cơ chế dự phòng khác như HTTP long-polling, giúp đảm bảo khả năng kết nối ổn định trên nhiều môi trường và trình duyệt khác nhau.

Khác với mô hình giao tiếp HTTP truyền thống, trong đó client phải gửi request để nhận response từ server, Socket.IO cho phép server chủ động gửi dữ liệu đến client ngay khi có sự kiện xảy ra. Sau khi quá trình kết nối ban đầu (handshake) được thiết lập, client và server duy trì một kết nối liên tục, giúp giảm độ trễ và tăng hiệu quả truyền dữ liệu.

Socket.IO hoạt động hiệu quả trong môi trường NodeJS nhờ tận dụng mô hình bất đồng bộ (asynchronous) và hướng sự kiện (event - driven). Mỗi kết nối Socket.IO được quản lý như một sự kiện, cho phép hệ thống xử lý đồng thời nhiều kết nối từ người dùng mà không làm ảnh hưởng đến hiệu năng tổng thể.

Một số đặc điểm nổi bật của Socket.IO bao gồm:

- Tự động kết nối lại (auto - reconnect): Khi kết nối bị gián đoạn, Socket.IO tự động thiết lập lại kết nối mà không cần người dùng thao tác.

- Hỗ trợ nhiều cơ chế truyền tải: Socket.IO ưu tiên sử dụng WebSocket và tự động chuyển sang các cơ chế dự phòng khi cần thiết.

- Giao tiếp theo sự kiện: Dữ liệu được truyền đi dưới dạng các sự kiện (event), giúp lập trình viên dễ dàng quản lý luồng giao tiếp.

- Hỗ trợ broadcast và room: Cho phép gửi dữ liệu đến tất cả client hoặc một nhóm client cụ thể.

- Khả năng mở rộng: Phù hợp với các ứng dụng web có số lượng người dùng lớn.

Trong kiến trúc ứng dụng web, Socket.IO thường được tích hợp cùng NodeJS và Express.js để xây dựng các chức năng giao tiếp thời gian thực. Socket.IO không thay thế hoàn toàn HTTP mà bổ sung cho HTTP trong các trường hợp cần cập nhật dữ liệu tức thời, chẳng hạn như thông báo, trò chuyện hoặc cập nhật nội dung động.

2.2.4. Ưu điểm

NodeJS mang lại nhiều ưu điểm nổi bật trong việc xây dựng các hệ thống backend hiện đại. Nhờ sử dụng mô hình bất đồng bộ (asynchronous) và hướng sự kiện (event - driven), NodeJS có khả năng xử lý đồng thời nhiều yêu cầu với hiệu năng cao, phù hợp cho các ứng dụng web có lượng truy cập lớn. Việc sử dụng cùng một ngôn ngữ JavaScript cho cả frontend và backend giúp giảm độ phức tạp trong quá trình phát triển và tăng tính nhất quán của hệ thống. Ngoài ra, NodeJS sở hữu cộng đồng lớn và hệ sinh thái phong phú, hỗ trợ nhiều thư viện và công cụ phục vụ phát triển ứng dụng.

Express.js là một framework nhẹ, linh hoạt, giúp đơn giản hóa việc xây dựng backend trên nền NodeJS. Express.js hỗ trợ mạnh mẽ việc phát triển RESTful API, cung cấp cơ chế routing rõ ràng và hệ thống middleware linh hoạt. Nhờ đó, mã nguồn backend trở nên dễ đọc, dễ bảo trì và dễ mở rộng khi bổ sung thêm các chức năng mới như xác thực, ghi log hay xử lý lỗi.

Socket.IO mang lại lợi thế lớn trong việc xây dựng các chức năng giao tiếp thời gian thực. Thư viện này cho phép trao đổi dữ liệu hai chiều giữa client và server với độ trễ thấp, đồng thời hỗ trợ các cơ chế như broadcast, room và tự động kết nối lại. Socket.IO hoạt động hiệu quả khi kết hợp với NodeJS, giúp hệ thống dễ dàng triển khai các chức năng như chat, thông báo tức thời và cập nhật dữ liệu động.

2.2.5. Nhược điểm

Bên cạnh các ưu điểm, NodeJS tồn tại một số hạn chế. Do hoạt động theo mô hình single - threaded, NodeJS không phù hợp cho các tác vụ tính toán nặng (CPU - intensive), vì các tác vụ này có thể làm chậm event loop và ảnh hưởng đến hệ thống. Ngoài ra, việc lập trình bất đồng bộ đòi hỏi lập trình viên phải nắm vững các khái niệm như callback, promise và async/await để tránh lỗi và khó khăn trong việc debug.

Express.js mặc dù nhẹ và linh hoạt, nhưng không cung cấp một cấu trúc chặt chẽ như các framework lớn. Điều này khiến lập trình viên cần tự tổ chức cấu trúc dự án và lựa chọn các thư viện bổ trợ phù hợp, có thể gây khó khăn cho người mới bắt đầu.

Đối với Socket.IO, việc duy trì kết nối liên tục giữa client và server có thể làm tăng mức tiêu thụ tài nguyên hệ thống nếu số lượng người dùng lớn. Bên cạnh đó, việc

triển khai và quản lý các kết nối thời gian thực cũng đòi hỏi lập trình viên phải chú ý đến vấn đề bảo mật và khả năng mở rộng của hệ thống.

2.3. Giới thiệu về MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở, được thiết kế để lưu trữ và quản lý dữ liệu dưới dạng document theo định dạng JSON (Binary JSON - BSON). Khác với các hệ quản trị cơ sở dữ liệu quan hệ truyền thống sử dụng bảng, hàng và cột, MongoDB tổ chức dữ liệu theo các collection và document, giúp việc lưu trữ và truy xuất dữ liệu trở nên linh hoạt hơn.

2.3.1. Đặc điểm của MongoDB

Một số đặc điểm nổi bật của MongoDB bao gồm:

- Lưu trữ dữ liệu dạng document: Dữ liệu được lưu trữ dưới dạng document có cấu trúc gần giống JSON, dễ đọc và dễ thao tác trong các ứng dụng web.

- Không yêu cầu schema cố định: MongoDB cho phép các document trong cùng một collection có cấu trúc khác nhau, phù hợp với các hệ thống có dữ liệu thay đổi thường xuyên.

- Khả năng mở rộng cao: MongoDB hỗ trợ mở rộng theo chiều ngang (horizontal scaling), đáp ứng tốt khi dữ liệu và số lượng người dùng tăng.

- Hiệu năng tốt: MongoDB được tối ưu cho các thao tác đọc/ghi nhanh, phù hợp với các ứng dụng web có lượng truy cập lớn.

Các đặc điểm trên cho thấy MongoDB phù hợp với các hệ thống web hiện đại có dữ liệu lớn, linh hoạt và yêu cầu khả năng mở rộng cao [6].

2.3.2. MongoDB trong kiến trúc ứng dụng web

Trong kiến trúc ứng dụng web hiện đại, MongoDB thường được sử dụng làm cơ sở dữ liệu phía máy chủ, kết hợp với NodeJS để xây dựng hệ thống backend. MongoDB cho phép lưu trữ dữ liệu dưới dạng JSON, rất tương thích với JavaScript - ngôn ngữ chính được sử dụng trong NodeJS và ReactJS.

Việc kết hợp MongoDB với NodeJS giúp quá trình trao đổi dữ liệu giữa backend và database trở nên đơn giản và hiệu quả, giảm bớt sự phức tạp trong việc chuyển đổi dữ liệu.

2.3.3. Vai trò của MongoDB trong website diễn đàn sinh viên

Trong đề tài Xây dựng Website diễn đàn sinh viên, MongoDB được sử dụng để lưu trữ và quản lý các dữ liệu chính của hệ thống, bao gồm:

- Thông tin tài khoản người dùng.

- Dữ liệu bài viết và bình luận.

- Thông tin phân quyền và các hoạt động liên quan của người dùng.

Do đặc thù dữ liệu của diễn đàn thường xuyên thay đổi và có cấu trúc linh hoạt, MongoDB là lựa chọn phù hợp giúp hệ thống dễ dàng mở rộng, bảo trì và nâng cấp trong tương lai.

2.3.4. Ưu và nhược điểm của MongoDB

Ưu điểm của MongoDB:

- Linh hoạt trong thiết kế cấu trúc dữ liệu.

- Dễ dàng mở rộng khi quy mô hệ thống tăng.

- Tương thích tốt với các công nghệ web hiện đại.

- Hiệu năng cao đối với các ứng dụng có dữ liệu lớn.

Nhược điểm của MongoDB:

- Không phù hợp với các hệ thống yêu cầu tính toàn vẹn quan hệ phức tạp.

- Việc quản lý dữ liệu không có schema cố định đòi hỏi thiết kế cẩn thận để tránh dữ liệu không nhất quán.

- Cần cấu hình và tối ưu hợp lý để đạt hiệu suất tốt nhất.

2.4. Phương pháp nghiên cứu

2.4.1. Phương pháp nghiên cứu tài liệu

Phương pháp nghiên cứu tài liệu được sử dụng để thu thập, phân tích và tổng hợp các tài liệu liên quan đến lĩnh vực nghiên cứu. Các nguồn tài liệu bao gồm giáo trình, sách chuyên ngành, bài báo khoa học và tài liệu kỹ thuật về các công nghệ như ReactJS, NodeJS, Express.js, Socket.IO và MongoDB.

Việc nghiên cứu tài liệu giúp đề tài xây dựng cơ sở lý thuyết vững chắc, làm nền tảng cho quá trình phân tích yêu cầu và lựa chọn giải pháp công nghệ phù hợp.

2.4.2. Phương pháp phân tích yêu cầu

Dựa trên nhu cầu thực tế của sinh viên và mục tiêu của đề tài, hệ thống website diễn đàn sinh viên được phân tích theo hai nhóm yêu cầu chính:

Yêu cầu chức năng: quản lý tài khoản người dùng, đăng bài viết, bình luận, tìm kiếm nội dung và phân quyền người dùng.

Yêu cầu phi chức năng: tính bảo mật, hiệu năng, khả năng mở rộng và tính thân thiện với người dùng.

Phương pháp phân tích yêu cầu giúp xác định rõ phạm vi và chức năng của hệ thống, tránh phát sinh ngoài mục tiêu nghiên cứu.

2.4.3. Phương pháp thiết kế hệ thống

Trên cơ sở các yêu cầu đã phân tích, đề tài tiến hành thiết kế hệ thống theo mô hình client - server. Các nội dung thiết kế bao gồm:

- Thiết kế kiến trúc tổng thể của hệ thống.

- Thiết kế cơ sở dữ liệu MongoDB.

- Thiết kế giao diện người dùng bằng ReactJS.

- Thiết kế các API phía backend bằng NodeJS và Express.js.

Phương pháp thiết kế giúp hệ thống có cấu trúc rõ ràng, dễ triển khai, bảo trì và mở rộng trong tương lai.

2.4.4. Phương pháp thực nghiệm

Phương pháp thực nghiệm được thực hiện thông qua việc xây dựng và triển khai website diễn đàn sinh viên trên môi trường thử nghiệm. Trong quá trình thực nghiệm, hệ thống được kiểm tra các chức năng chính như đăng ký - đăng nhập, đăng bài viết, bình luận và tìm kiếm nội dung.

Thông qua quá trình thực nghiệm, đề tài đánh giá được tính ổn định, hiệu năng xử lý và khả năng đáp ứng của hệ thống.

2.4.5. Phương pháp đánh giá và hoàn thiện

Sau khi hoàn thành quá trình thực nghiệm, hệ thống được đánh giá dựa trên các tiêu chí:

- Mức độ đáp ứng yêu cầu đề ra.

- Tính ổn định và bảo mật của hệ thống.

- Trải nghiệm người dùng và mức độ thân thiện của giao diện.

Dựa trên kết quả đánh giá, hệ thống được điều chỉnh và hoàn thiện nhằm nâng cao chất lượng và hiệu quả sử dụng.

2.4.6. Kết luận phương pháp nghiên cứu

Việc kết hợp các phương pháp nghiên cứu trên giúp đề tài đảm bảo tính khoa học, tính thực tiễn và khả năng ứng dụng cao. Đây là cơ sở quan trọng để triển khai thành công website diễn đàn sinh viên và làm nền tảng cho các nghiên cứu mở rộng trong tương lai.

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1. Mô tả hệ thống

Website diễn đàn sinh viên là một hệ thống thông tin trực tuyến được xây dựng nhằm phục vụ nhu cầu trao đổi học tập, chia sẻ kiến thức và kết nối cộng đồng sinh viên trong môi trường số. Hệ thống cho phép sinh viên tham gia thảo luận, đặt câu hỏi, chia sẻ tài liệu học tập và cập nhật các thông tin liên quan đến đời sống học đường thông qua một nền tảng tập trung, có tổ chức.

Hệ thống cung cấp các chức năng cơ bản như đăng ký và đăng nhập tài khoản, quản lý thông tin cá nhân, đăng bài viết, bình luận, tìm kiếm nội dung theo chủ đề, cũng như các chức năng tương tác như thích bài viết, nhận thông báo và trò chuyện. Thông qua các chức năng này, sinh viên có thể dễ dàng trao đổi kiến thức, hỗ trợ lẫn nhau trong học tập và tăng cường sự kết nối trong cộng đồng.

Website diễn đàn sinh viên được xây dựng theo mô hình client - server, trong đó frontend chịu trách nhiệm hiển thị giao diện và xử lý tương tác với người dùng, backend đảm nhiệm xử lý nghiệp vụ, xác thực người dùng và quản lý dữ liệu. Cơ sở dữ liệu được sử dụng để lưu trữ thông tin người dùng, bài viết, bình luận và các dữ liệu liên quan. Ngoài ra, hệ thống tích hợp cơ chế giao tiếp thời gian thực nhằm hỗ trợ các chức năng như thông báo và trò chuyện, góp phần nâng cao trải nghiệm người dùng.

Hệ thống được thiết kế theo hướng thân thiện, dễ sử dụng, đảm bảo tính ổn định, bảo mật và khả năng mở rộng. Trong tương lai, website diễn đàn sinh viên có thể tiếp tục được phát triển và bổ sung thêm các chức năng nâng cao nhằm đáp ứng tốt hơn nhu cầu học tập và giao lưu của sinh viên.

3.2. Phân tích yêu cầu hệ thống

3.2.1. Yêu cầu chức năng

Chức năng cho người dùng

Người dùng là sinh viên tham gia sử dụng website diễn đàn. Các chức năng chính dành cho người dùng bao gồm:

Đăng ký tài khoản: Người dùng có thể tạo tài khoản mới bằng cách cung cấp các thông tin cần thiết như tên đăng nhập, mật khẩu và email.

Đăng nhập hệ thống: Người dùng đăng nhập để sử dụng các chức năng của diễn đàn theo quyền hạn được cấp.

Quản lý thông tin cá nhân: Người dùng có thể xem và chỉnh sửa thông tin cá nhân như ảnh đại diện, email hoặc mô tả cá nhân.

Xem thông tin người dùng khác: Người dùng có thể xem thông tin công khai của các thành viên khác trên diễn đàn như tên, ảnh đại diện và các bài viết đã đăng.

Xem bài viết: Người dùng được phép xem danh sách bài viết theo từng chủ đề và xem chi tiết nội dung bài viết.

Đăng bài viết: Người dùng có thể tạo bài viết mới để chia sẻ kiến thức, tài liệu học tập hoặc đặt câu hỏi thảo luận.

Thích và bình luận bài viết: Người dùng có thể bày tỏ sự quan tâm bằng cách thích (like) bài viết và tham gia bình luận, trao đổi ý kiến dưới các bài viết.

Tìm kiếm nội dung và tài liệu: Người dùng có thể tìm kiếm bài viết hoặc tài liệu theo từ khóa, chủ đề hoặc nội dung liên quan.

Nhận thông báo: Người dùng nhận thông báo khi có bài viết mới, lượt thích hoặc bình luận liên quan đến nội dung đã tham gia.

Báo cáo vi phạm: Người dùng có thể gửi báo cáo đối với các bài viết, bình luận hoặc tài khoản có nội dung vi phạm quy định của diễn đàn.

Chat toàn diễn đàn và chat riêng: Người dùng có thể tham gia trò chuyện chung trong diễn đàn hoặc trò chuyện riêng với người dùng khác nhằm tăng cường khả năng giao tiếp và trao đổi thông tin.

Quyền của người điều hành (MOD): Ngoài các chức năng của người dùng thông thường, người điều hành (MOD) còn có quyền duyệt bài viết, hỗ trợ quản lý nội dung trong phạm vi được phân công.

Chức năng cho quản trị viên

Quản trị viên là người chịu trách nhiệm quản lý và vận hành toàn bộ hệ thống website diễn đàn sinh viên. Các chức năng dành cho quản trị viên bao gồm:

Quản lý tài khoản người dùng: Xem danh sách người dùng, khóa hoặc xóa các tài khoản vi phạm.

Quản lý bài viết: Duyệt, chỉnh sửa hoặc xóa các bài viết không phù hợp với nội quy diễn đàn.

Quản lý bình luận: Kiểm soát và xử lý các bình luận vi phạm.

Quản lý báo cáo vi phạm: Xem và xử lý các báo cáo từ người dùng đối với bài viết, bình luận hoặc tài khoản.

Quản lý chủ đề và danh mục: Tạo, chỉnh sửa và xóa các chủ đề, danh mục trên diễn đàn.

Phân quyền người dùng: Thiết lập và thay đổi quyền hạn cho người dùng, người điều hành (MOD) và quản trị viên.

Giám sát hoạt động hệ thống: Theo dõi hoạt động tổng thể của hệ thống nhằm đảm bảo diễn đàn hoạt động ổn định và hiệu quả.

3.2.2. Yêu cầu phi chức năng

Hệ thống website diễn đàn sinh viên, ngoài việc đáp ứng các yêu cầu chức năng, còn phải đảm bảo các yêu cầu phi chức năng nhằm nâng cao chất lượng, hiệu năng và khả năng vận hành ổn định trong thực tế.

Hệ thống cần đảm bảo tính bảo mật, trong đó thông tin tài khoản người dùng phải được bảo vệ an toàn, mật khẩu được mã hóa trước khi lưu trữ và quyền truy cập được kiểm soát theo từng vai trò người dùng. Dữ liệu trao đổi giữa client và server cần được đảm bảo an toàn, hạn chế các truy cập trái phép và các hành vi vi phạm an ninh.

Về hiệu năng, hệ thống phải có khả năng xử lý nhanh các yêu cầu từ người dùng, đảm bảo thời gian phản hồi ổn định khi truy cập, đăng nhập, xem bài viết và bình luận. Hệ thống cũng cần đáp ứng tốt khi có nhiều người dùng truy cập đồng thời, đặc biệt đối với các chức năng thời gian thực như thông báo và trò chuyện.

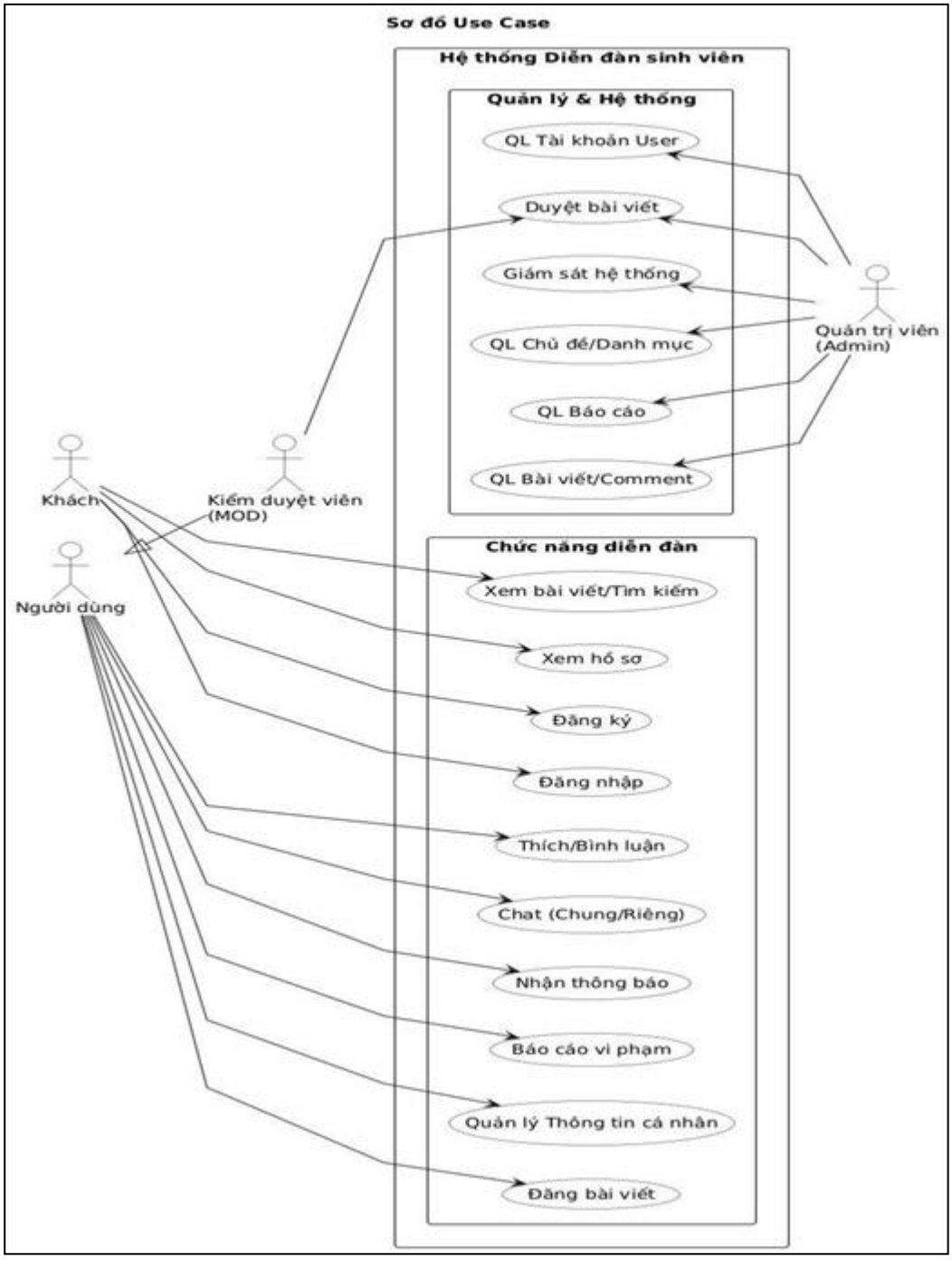
Hệ thống phải đảm bảo tính ổn định trong quá trình vận hành, hạn chế tối đa lỗi phát sinh, có khả năng hoạt động liên tục trong thời gian dài và phục hồi khi xảy ra sự cố. Dữ liệu lưu trữ cần được đảm bảo tính toàn vẹn và không bị mất mát.

Bên cạnh đó, hệ thống cần có khả năng triển khai và bảo trì thuận tiện. Việc cài đặt, vận hành và nâng cấp hệ thống cần được thực hiện dễ dàng. Mã nguồn được tổ chức rõ ràng, giúp việc sửa lỗi và mở rộng chức năng trong tương lai trở nên thuận lợi.

Hệ thống cũng cần đảm bảo khả năng tương thích tốt, hoạt động ổn định trên các trình duyệt phổ biến và tương thích với nhiều thiết bị khác nhau như máy tính và thiết bị di động, không phụ thuộc vào nền tảng phần cứng cụ thể.

Cuối cùng, trải nghiệm người dùng là một yếu tố quan trọng. Giao diện hệ thống cần trực quan, dễ sử dụng, các chức năng được bố trí hợp lý và thao tác mượt mà, giúp sinh viên dễ dàng tiếp cận và sử dụng hệ thống một cách hiệu quả.

3.2.3. Sơ đồ Use Case



Hình 3.1. Sơ đồ Use Case

3.3. Thiết kế hệ thống

3.3.1. Thiết kế kiến trúc tổng thể hệ thống

Hệ thống website diễn đàn sinh viên được thiết kế theo mô hình client - server, nhằm đảm bảo tính phân tách rõ ràng giữa giao diện người dùng, xử lý nghiệp vụ và lưu trữ dữ liệu. Kiến trúc này giúp hệ thống dễ dàng triển khai, bảo trì và mở rộng trong tương lai.

Ở phía client, hệ thống sử dụng ReactJS để xây dựng giao diện người dùng. Client đảm nhiệm việc hiển thị dữ liệu, xử lý tương tác của người dùng và gửi các yêu cầu đến server thông qua giao thức HTTP/HTTPS và các kết nối thời gian thực. Việc sử dụng ReactJS giúp giao diện hoạt động mượt mà, giảm số lần tải lại trang và nâng cao trải nghiệm người dùng.

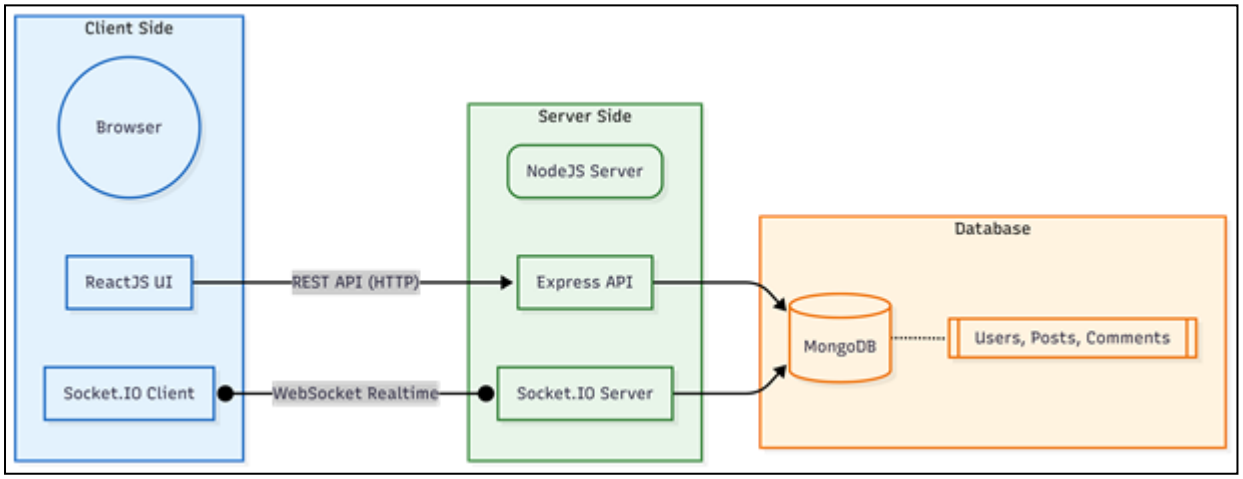
Ở phía server, hệ thống sử dụng NodeJS kết hợp với Express.js để xử lý các yêu cầu từ client. Server chịu trách nhiệm xử lý logic nghiệp vụ, xác thực và phân quyền người dùng, quản lý bài viết, bình luận và cung cấp các API cho frontend. Ngoài ra, server còn tích hợp Socket.IO để hỗ trợ các chức năng giao tiếp thời gian thực như thông báo và trò chuyện.

Cơ sở dữ liệu MongoDB được sử dụng để lưu trữ toàn bộ dữ liệu của hệ thống, bao gồm thông tin người dùng, bài viết, bình luận và các dữ liệu liên quan khác. MongoDB lưu trữ dữ liệu theo mô hình document, phù hợp với đặc thù dữ liệu linh hoạt của website diễn đàn và giúp hệ thống dễ dàng mở rộng.

Quá trình giao tiếp giữa các thành phần trong hệ thống được thực hiện như sau: client gửi yêu cầu đến server thông qua các API, server xử lý yêu cầu, tương tác với cơ sở dữ liệu để truy xuất hoặc cập nhật dữ liệu, sau đó trả kết quả về cho client. Đối với các chức năng cần cập nhật theo thời gian thực, server sử dụng Socket.IO để gửi dữ liệu trực tiếp đến client mà không cần chờ yêu cầu mới.

Đối với việc quản lý và lưu trữ các tệp đa phương tiện như hình ảnh bài viết, hình ảnh bình luận và ảnh đại diện người dùng, hệ thống sử dụng dịch vụ Cloudinary. Việc sử dụng Cloudinary giúp giảm tải cho server, tối ưu tốc độ tải tệp và đảm bảo khả năng mở rộng cũng như tính ổn định của hệ thống.

Kiến trúc tổng thể của hệ thống được trình bày trong Hình 3.2, thể hiện rõ mối quan hệ giữa client, server, cơ sở dữ liệu và cơ chế giao tiếp thời gian thực. Thiết kế kiến trúc này đảm bảo hệ thống hoạt động ổn định, đáp ứng tốt nhu cầu trao đổi thông tin của sinh viên và tạo nền tảng cho việc phát triển các chức năng nâng cao trong tương lai.

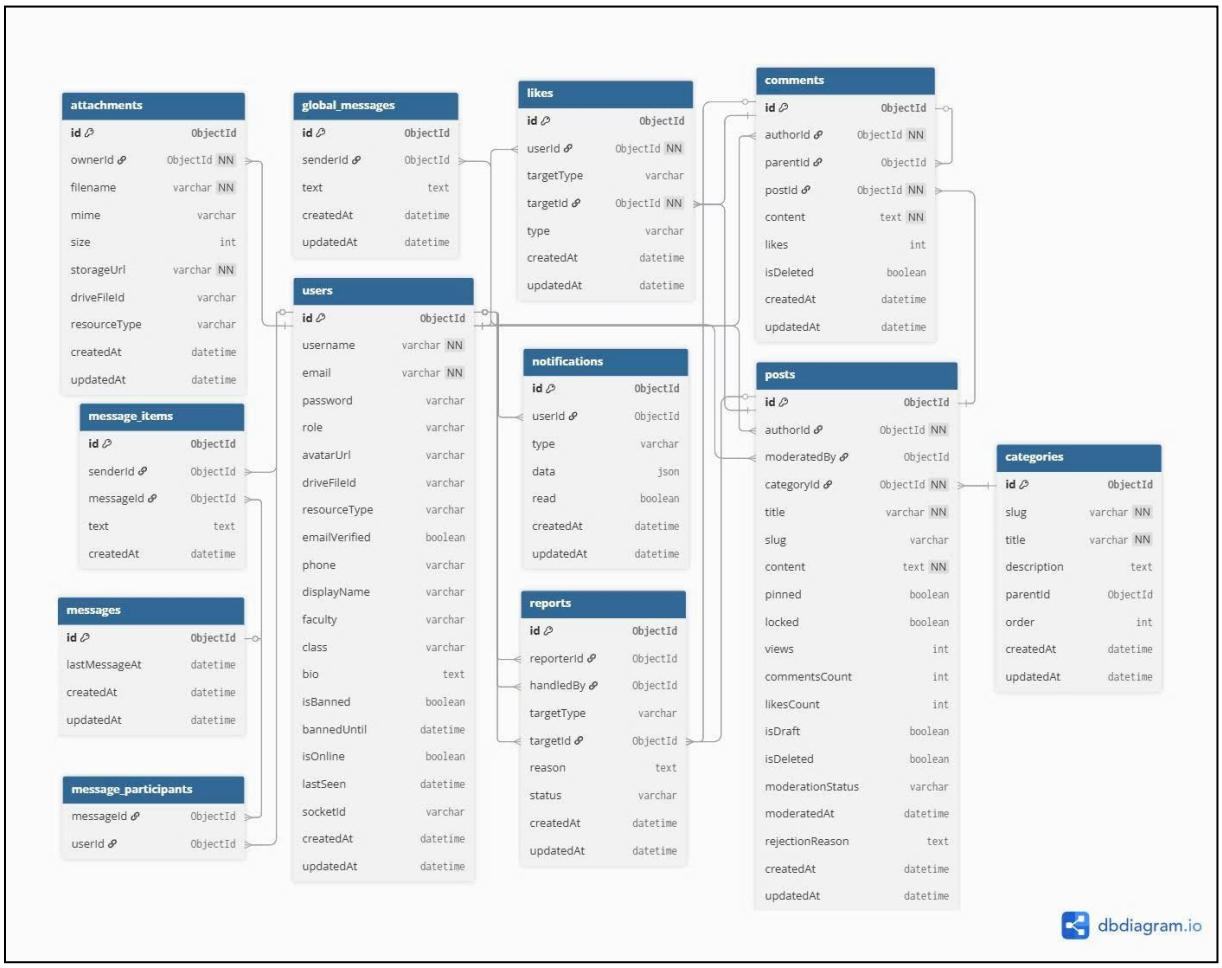


Hình 3.2. Sơ đồ kiến trúc tổng thể hệ thống

3.3.2. Thiết kế cơ sở dữ liệu

Hệ thống website diễn đàn sinh viên sử dụng MongoDB làm hệ quản trị cơ sở dữ liệu chính. MongoDB là cơ sở dữ liệu NoSQL, lưu trữ dữ liệu dưới dạng document, phù hợp với các hệ thống có dữ liệu linh hoạt và thường xuyên thay đổi như website diễn đàn.

Dữ liệu trong hệ thống được tổ chức thành các collection, mỗi collection đại diện cho một nhóm đối tượng trong hệ thống. Mối quan hệ giữa các collection được thiết lập thông qua các khóa tham chiếu (ObjectId), đảm bảo khả năng liên kết dữ liệu trong khi vẫn giữ được tính linh hoạt của mô hình NoSQL.



Hình 3.3. Mô hình dữ liệu MongoDB của hệ thống

Bảng 3.1. Chi tiết các thực thể User

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	ObjectId	Khóa chính
2	username	String	Tên đăng nhập
3	email	String	Email người dùng
4	password	String	Mật khẩu đã mã hóa
5	role	String	Vai trò (student/admin/mod)
6	avatarUrl	String	Ảnh đại diện
7	driveFileId	String	ID file avatar
8	resourceType	String	Loại file
9	faculty	String	Khoa
10	class	String	Lớp
11	bio	String	Giới thiệu

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
12	isBanned	Boolean	Trạng thái cấm
13	bannedUntil	Date	Thời gian hết hạn cấm
14	isOnline	Boolean	Trạng thái online
15	lastSeen	Date	Lần hoạt động cuối
16	socketId	String	Socket realtime
17	createdAt	Date	Ngày tạo
18	updatedAt	Date	Ngày cập nhật

Bảng 3.2. Chi tiết các thực thể Category

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	ObjectId	Khóa chính
2	slug	String	Đường dẫn
3	title	String	Tên danh mục
4	description	String	Mô tả
5	parentId	ObjectId	Danh mục cha
6	order	Number	Thứ tự hiển thị
7	createdAt	Date	Ngày tạo
8	updatedAt	Date	Ngày cập nhật

Bảng 3.3. Chi tiết các thực thể Post

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	ObjectId	Khóa chính
2	authorId	ObjectId	Tác giả
3	categoryId	ObjectId	Danh mục
4	title	String	Tiêu đề
5	slug	String	Đường dẫn
6	content	String	Nội dung
7	tags	Array	Thẻ
8	attachments	Array<ObjectId>	File đính kèm
9	pinned	Boolean	Ghim

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
10	locked	Boolean	Khóa
11	views	Number	Lượt xem
12	commentsCount	Number	Số bình luận
13	likesCount	Number	Lượt thích
14	isDraft	Boolean	Bản nháp
15	isDeleted	Boolean	Đã xóa
16	moderationStatus	String	Trạng thái duyệt
17	moderatedBy	ObjectId	Người duyệt
18	moderatedAt	Date	Thời gian duyệt
19	rejectionReason	String	Lý do từ chối
20	createdAt	Date	Ngày tạo
21	updatedAt	Date	Ngày cập nhật

Bảng 3.4. Chi tiết các thực thể Comment

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	ObjectId	Khóa chính
2	postId	ObjectId	Bài viết
3	authorId	ObjectId	Người bình luận
4	parentId	ObjectId	Bình luận cha
5	content	String	Nội dung
6	attachments	Array<ObjectId>	File đính kèm
7	likes	Number	Lượt thích
8	isDeleted	Boolean	Đã xóa
9	createdAt	Date	Ngày tạo
10	updatedAt	Date	Ngày cập nhật

Bảng 3.5. Chi tiết các thực thể Attachment

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	ObjectId	Khóa chính
2	ownerId	ObjectId	Người sở hữu
3	filename	String	Tên file
4	mime	String	Loại MIME
5	size	Number	Kích thước
6	storageUrl	String	URL lưu trữ
7	driveFileId	String	ID Cloudinary
8	resourceType	String	Loại file
9	createdAt	Date	Ngày tạo
10	updatedAt	Date	Ngày cập nhật

Bảng 3.6. Chi tiết các thực thể Like

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	ObjectId	Khóa chính
2	userId	ObjectId	Người thực hiện
3	targetType	String	post / comment
4	targetId	ObjectId	ID đối tượng
5	type	String	like / dislike
6	createdAt	Date	Ngày tạo
7	updatedAt	Date	Ngày cập nhật

Bảng 3.7. Chi tiết các thực thể Message

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	ObjectId	Khóa chính
2	participants	Array<ObjectId>	Người tham gia
3	messages	Array<Object>	Tin nhắn
4	messages.senderId	ObjectId	Người gửi
5	messages.text	String	Nội dung

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
6	messages.attachments	Array<ObjectId>	File đính kèm
7	messages.createdAt	Date	Thời gian gửi
8	lastMessageAt	Date	Tin nhắn cuối
9	readMarks	Map	Trạng thái đã đọc
10	createdAt	Date	Ngày tạo
11	updatedAt	Date	Ngày cập nhật

Bảng 3.8. Chi tiết các thực thể GlobalMessage

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	ObjectId	Khóa chính
2	senderId	ObjectId	Người gửi tin nhắn
3	text	String	Nội dung tin nhắn
4	attachments	Array<ObjectId>	Tệp đính kèm
5	createdAt	Date	Thời gian gửi
6	updatedAt	Date	Thời gian cập nhật

Bảng 3.9. Chi tiết các thực thể Notification

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	ObjectId	Khóa chính
2	userId	ObjectId	Người nhận thông báo
3	type	String	Loại thông báo (comment, like, system)
4	data	Object	Dữ liệu chi tiết của thông báo
5	read	Boolean	Trạng thái đã đọc
6	createdAt	Date	Thời gian tạo
7	updatedAt	Date	Thời gian cập nhật

Bảng 3.10. Chi tiết các thực thể Report

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	_id	ObjectId	Khóa chính
2	reporterId	ObjectId	Người gửi báo cáo
3	targetType	String	Đối tượng bị báo cáo (post, comment, user)
4	targetId	ObjectId	ID đối tượng bị báo cáo
5	reason	String	Lý do báo cáo
6	status	String	Trạng thái xử lý (open, reviewed, closed)
7	handledBy	ObjectId	Người xử lý báo cáo
8	createdAt	Date	Thời gian tạo
9	updatedAt	Date	Thời gian cập nhật

3.3.3. Thiết kế giao diện người dùng

Giao diện người dùng của hệ thống diễn đàn sinh viên được thiết kế theo hướng đơn giản, trực quan và thân thiện, nhằm giúp sinh viên dễ dàng tiếp cận và sử dụng các chức năng của hệ thống. Quá trình thiết kế giao diện được thực hiện dựa trên các prototype (bản mẫu giao diện), giúp mô phỏng cách bố trí các thành phần và luồng tương tác của người dùng trước khi triển khai chính thức.

Hình 3.4. Wireframe của trang chủ

Hình 3.4 minh họa Wireframe của trang chủ hệ thống. Trang chủ được thiết kế để hiển thị các nội dung chính như danh sách bài viết mới nhất, bài viết nổi bật, danh mục bài viết và các chức năng tìm kiếm. Bố cục giao diện được sắp xếp khoa học, giúp người dùng nhanh chóng nắm bắt thông tin và dễ dàng điều hướng đến các khu vực mong muốn. Ngoài ra, các nút chức năng như đăng bài, đăng nhập và thông báo được đặt ở vị trí dễ nhận biết nhằm nâng cao trải nghiệm người dùng.

Hình 3.5. Wireframe của tin nhắn riêng

Hình 3.5 thể hiện Wireframe của chức năng tin nhắn riêng giữa các người dùng. Giao diện tin nhắn được thiết kế theo dạng hộp thoại (chat), cho phép hiển thị danh sách các cuộc trò chuyện, nội dung tin nhắn và trạng thái người dùng. Thiết kế này giúp người dùng theo dõi và trao đổi thông tin một cách thuận tiện, đồng thời hỗ trợ giao tiếp thời gian thực thông qua cơ chế Socket.IO. Việc bố trí rõ ràng các khu vực gửi và nhận tin nhắn giúp tăng tính trực quan và dễ sử dụng.

3.4. Thiết kế, kiến trúc ứng dụng

3.4.1. Xây dựng backend

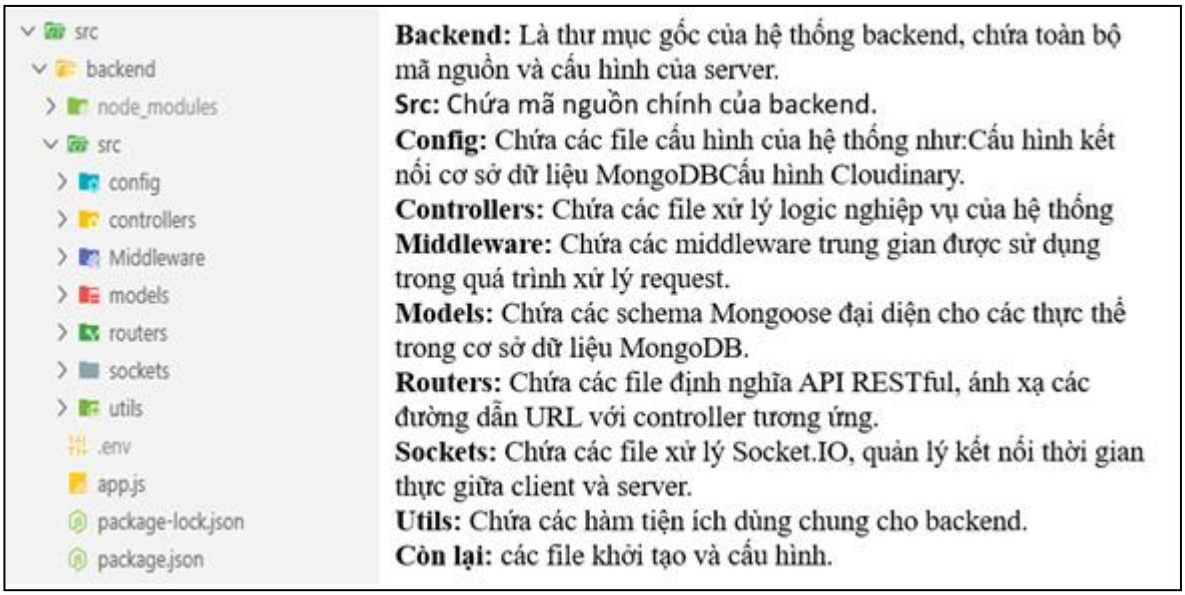
Backend của hệ thống diễn đàn sinh viên được xây dựng trên nền tảng NodeJS kết hợp với Express.js, đóng vai trò xử lý toàn bộ logic nghiệp vụ và cung cấp các API cho frontend. Backend đảm nhiệm các chức năng chính như xác thực người dùng, phân quyền, quản lý bài viết, bình luận, tin nhắn, thông báo và báo cáo vi phạm.

Cấu trúc thư mục backend được tổ chức theo mô hình phân lớp, bao gồm các thành phần như cấu hình hệ thống, router, controller, middleware, model và xử lý giao tiếp thời gian thực. Cách tổ chức này giúp mã nguồn rõ ràng, dễ quản lý và thuận tiện cho việc mở rộng các chức năng trong tương lai.

Cụ thể, backend sử dụng MongoDB làm cơ sở dữ liệu để lưu trữ dữ liệu theo mô hình document, JWT (JSON Web Token) để xác thực và phân quyền người dùng,

Cloudinary để lưu trữ các tệp đa phương tiện và Socket.IO để hỗ trợ các chức năng giao tiếp thời gian thực.

Cấu trúc thư mục backend của hệ thống được trình bày trong Hình 3.6.



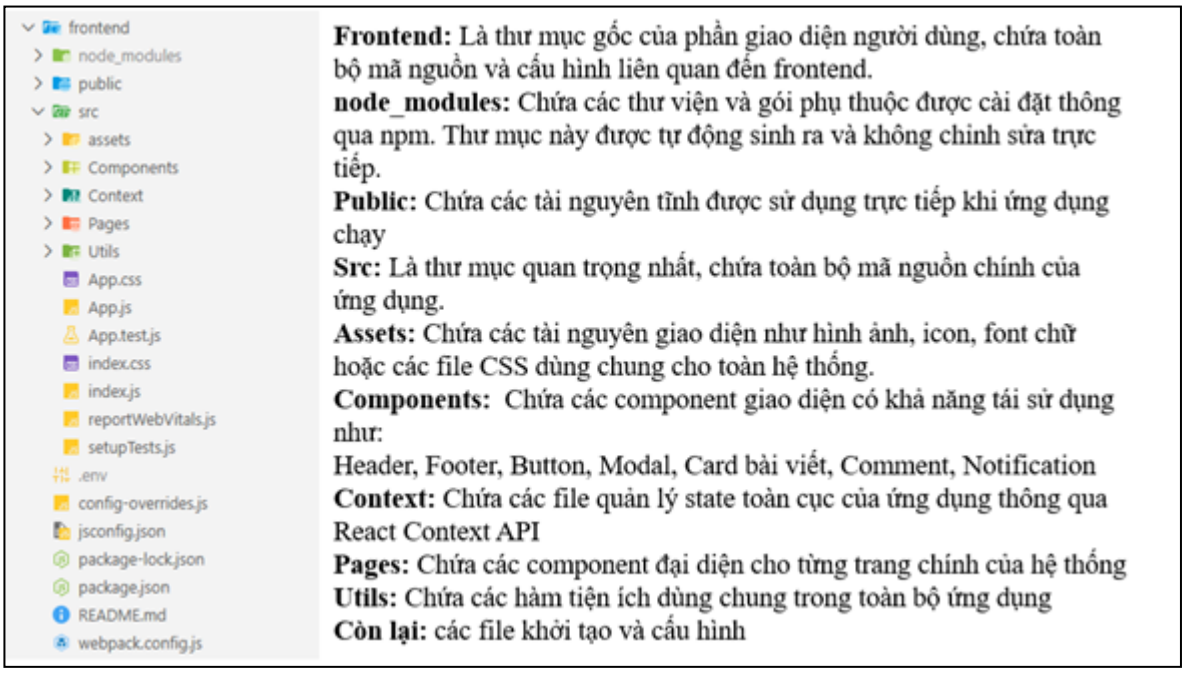
Hình 3.6. Cấu trúc thư mục backend

3.4.2. Xây dựng frontend

Frontend của hệ thống được xây dựng bằng ReactJS, chịu trách nhiệm hiển thị giao diện người dùng và xử lý các tương tác phía client. Giao diện được phát triển theo hướng component - based, trong đó mỗi component đại diện cho một phần chức năng cụ thể như trang chủ, đăng nhập - đăng ký, danh sách bài viết, chi tiết bài viết, bình luận và nhắn tin.

Frontend sử dụng React Router để quản lý điều hướng giữa các trang trong ứng dụng web đơn trang (SPA), giúp người dùng chuyển đổi giữa các chức năng mà không cần tải lại trang. Dữ liệu được frontend truy xuất từ backend thông qua các API RESTful và được quản lý bằng state, props cũng như Context API để đảm bảo tính đồng bộ của ứng dụng.

Cấu trúc thư mục frontend được tổ chức rõ ràng, phân chia giữa các thành phần giao diện, trang chức năng, tài nguyên và các tiện ích dùng chung. Điều này giúp việc phát triển giao diện trở nên linh hoạt, dễ bảo trì và nâng cao trải nghiệm người dùng.



Hình 3.7. Cấu trúc thư mục frontend

3.4.3. Xây dựng chức năng giao tiếp thời gian thực (Socket.IO)

Nhằm nâng cao tính tương tác và trải nghiệm người dùng, hệ thống diễn đàn sinh viên được tích hợp chức năng giao tiếp thời gian thực (real - time) thông qua thư viện Socket.IO. Công nghệ này cho phép client và server trao đổi dữ liệu hai chiều liên tục mà không cần gửi các yêu cầu HTTP lặp lại, giúp giảm độ trễ và tăng hiệu quả truyền tải dữ liệu.

Socket.IO được tích hợp vào backend NodeJS để quản lý các kết nối thời gian thực từ phía client. Sau khi người dùng đăng nhập thành công, frontend sẽ thiết lập kết nối Socket.IO với server. Kết nối này được duy trì trong suốt quá trình người dùng sử dụng hệ thống, cho phép server chủ động gửi dữ liệu đến client ngay khi có sự kiện xảy ra.

Các chức năng thời gian thực được triển khai trong hệ thống bao gồm:

- Nhắn tin riêng giữa người dùng, cho phép trao đổi thông tin nhanh chóng.

- Chat toàn cục, nơi người dùng có thể tham gia thảo luận chung.

- Thông báo tức thời, khi có các hoạt động mới như bình luận bài viết, lượt thích hoặc báo cáo vi phạm.

- Cập nhật trạng thái người dùng, như trạng thái online/offline.

Quá trình giao tiếp thời gian thực được thực hiện dựa trên mô hình sự kiện (event - driven). Khi client gửi một sự kiện (ví dụ: gửi tin nhắn), server sẽ tiếp nhận, xử lý và phát sự kiện tương ứng đến các client liên quan. Socket.IO cũng hỗ trợ các cơ chế như broadcast và room, cho phép gửi dữ liệu đến một nhóm người dùng cụ thể hoặc toàn bộ hệ thống.

Ngoài ra, Socket.IO còn hỗ trợ các tính năng tự động kết nối lại khi mất kết nối và sử dụng các cơ chế truyền tải dự phòng trong trường hợp WebSocket không khả dụng. Điều này giúp đảm bảo hệ thống giao tiếp thời gian thực hoạt động ổn định ngay cả khi điều kiện mạng không thuận lợi.

Việc tích hợp Socket.IO giúp hệ thống diễn đàn sinh viên trở nên sinh động hơn, tăng khả năng tương tác giữa người dùng và đáp ứng tốt các yêu cầu về cập nhật dữ liệu tức thời. Chức năng thời gian thực đóng vai trò quan trọng trong việc nâng cao chất lượng và tính hiện đại của website diễn đàn.

3.5. Cài đặt và triển khai hệ thống

Sau khi hoàn thành quá trình thiết kế và xây dựng hệ thống, bước tiếp theo là tiến hành cài đặt và triển khai hệ thống lên môi trường thực tế. Việc triển khai giúp đánh giá khả năng vận hành của hệ thống, kiểm tra tính ổn định cũng như mức độ đáp ứng yêu cầu của người dùng. Hệ thống được triển khai theo mô hình phân tách backend và frontend, sử dụng các nền tảng triển khai hiện đại.

3.5.1. Môi trường và công cụ phát triển

Hệ thống website diễn đàn sinh viên được phát triển và triển khai trong môi trường sau:

- Hệ điều hành: Windows 11
- Ngôn ngữ lập trình: JavaScript (ES6+)
- Frontend: ReactJS
- Backend: NodeJS, Express.js
- Cơ sở dữ liệu: MongoDB (MongoDB Atlas)
- Xác thực và phân quyền: JWT (JSON Web Token)
- Giao tiếp thời gian thực: Socket.IO

- Lưu trữ tệp đa phương tiện: Cloudinary

Các công cụ hỗ trợ phát triển:

- Visual Studio Code: soạn thảo mã nguồn

- Git & GitHub: quản lý mã nguồn

- Postman: kiểm thử API

- MongoDB Compass: quản lý dữ liệu

- Trình duyệt Google Chrome: kiểm thử giao diện

- Render: triển khai backend

- Vercel: triển khai frontend

Việc lựa chọn các công nghệ và công cụ trên giúp hệ thống đảm bảo hiệu năng, tính ổn định và khả năng mở rộng trong tương lai.

3.5.2. Cài đặt hệ thống backend

Backend của hệ thống được xây dựng bằng NodeJS và Express.js, sau đó được triển khai trên nền tảng Render - một dịch vụ cloud hỗ trợ triển khai ứng dụng web nhanh chóng và ổn định.

Quá trình triển khai backend trên Render được thực hiện theo các bước sau:

Chuẩn bị mã nguồn backend

- Backend được đóng gói đầy đủ, đảm bảo có file package.json, cấu hình cổng chạy server và kết nối cơ sở dữ liệu MongoDB Atlas.

Cấu hình biến môi trường

- Các biến môi trường được khai báo trên Render, bao gồm: Chuỗi kết nối MongoDB, JWT secret key, Thông tin cấu hình Cloudinary, Port chạy ứng dụng

Triển khai backend trên Render: Render tự động build và khởi chạy server từ repository GitHub. Sau khi triển khai thành công, backend được cung cấp một URL công khai để frontend có thể gửi yêu cầu API và kết nối Socket.IO.

Kiểm tra backend sau triển khai

- Các API được kiểm tra bằng Postman để đảm bảo hoạt động chính xác, đồng thời kiểm tra khả năng kết nối cơ sở dữ liệu và xử lý các chức năng nghiệp vụ.

Việc triển khai backend trên Render giúp hệ thống hoạt động ổn định, dễ dàng quản lý và thuận tiện trong việc mở rộng khi số lượng người dùng tăng lên.

3.5.3. Cài đặt hệ thống frontend

Frontend của hệ thống được xây dựng bằng ReactJS và được triển khai trên nền tảng Vercel, dịch vụ chuyên hỗ trợ triển khai các ứng dụng frontend với hiệu năng cao.

Quá trình triển khai frontend được thực hiện như sau:

- Chuẩn bị mã nguồn frontend: Ứng dụng ReactJS được cấu hình hoàn chỉnh, đảm bảo kết nối đúng với backend thông qua URL API.

- Cấu hình biến môi trường frontend: URL của backend được khai báo trong biến môi trường của Vercel để frontend có thể gọi API và sử dụng Socket.IO.

- Triển khai frontend trên Vercel: Vercel tự động build và deploy ứng dụng từ repository GitHub. Sau khi hoàn tất, hệ thống frontend được cung cấp một tên miền truy cập công khai.

- Kiểm tra giao diện người dùng: Giao diện được kiểm tra trên nhiều trình duyệt để đảm bảo khả năng hiển thị tốt, các chức năng hoạt động ổn định và trải nghiệm người dùng mượt mà.

Việc triển khai frontend trên Vercel giúp hệ thống có tốc độ tải nhanh, khả năng mở rộng tốt và dễ dàng bảo trì.

3.5.4. Triển khai và chạy thử nghiệm

Sau khi hoàn tất triển khai backend trên Render và frontend trên Vercel, hệ thống được tiến hành chạy thử nghiệm tổng thể. Quá trình thử nghiệm bao gồm:

- Kiểm tra chức năng đăng ký, đăng nhập và xác thực người dùng

- Kiểm tra các chức năng đăng bài, bình luận, lượt thích

- Kiểm tra chức năng nhắn tin và thông báo thời gian thực bằng Socket.IO

- Kiểm tra chức năng tải lên và hiển thị hình ảnh thông qua Cloudinary

- Kiểm tra khả năng kết nối giữa frontend và backend trong môi trường triển khai thực tế

Kết quả thử nghiệm cho thấy hệ thống hoạt động ổn định, các chức năng đáp ứng đúng yêu cầu đề ra và không phát sinh lỗi nghiêm trọng. Việc triển khai và chạy thử nghiệm thành công khẳng định tính khả thi của hệ thống và tạo tiền đề cho việc mở rộng, nâng cấp trong tương lai.

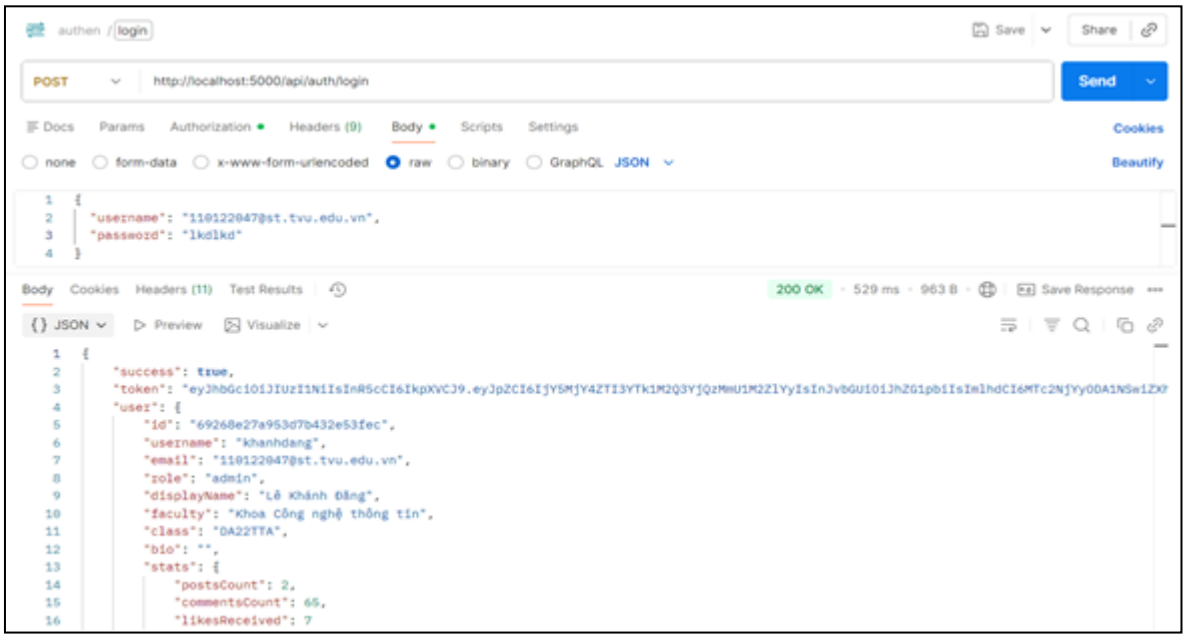
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1. Kiểm thử API với Postman

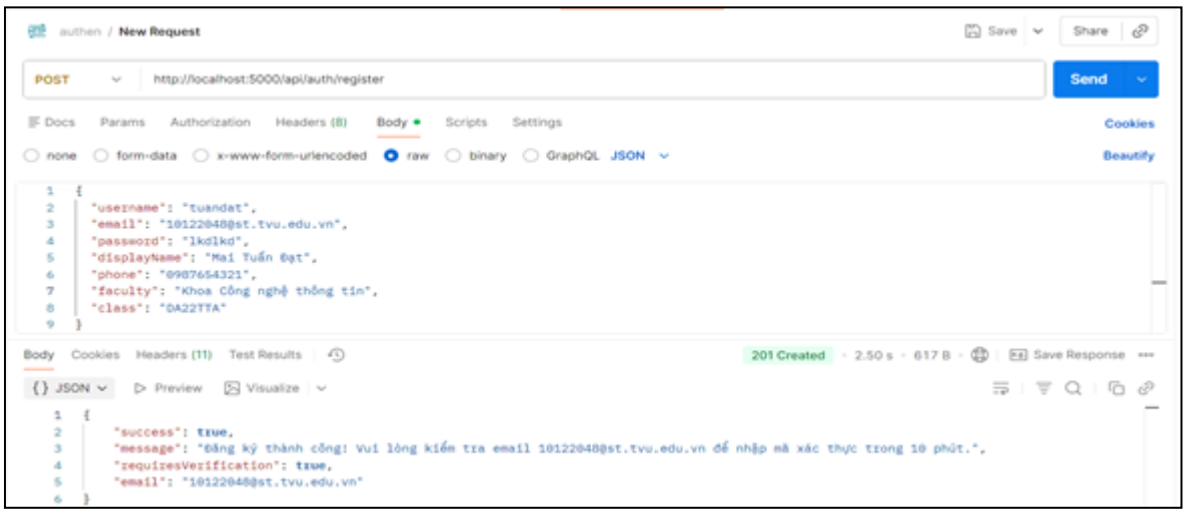
Trong quá trình phát triển hệ thống backend, các API được kiểm thử bằng công cụ Postman nhằm đảm bảo tính chính xác, ổn định và khả năng đáp ứng yêu cầu của hệ thống. Việc kiểm thử API giúp xác minh các chức năng hoạt động đúng theo thiết kế trước khi tích hợp với giao diện frontend.

Các nhóm API chính đã được kiểm thử bao gồm:

API xác thực và quản lý người dùng: đăng ký, đăng nhập

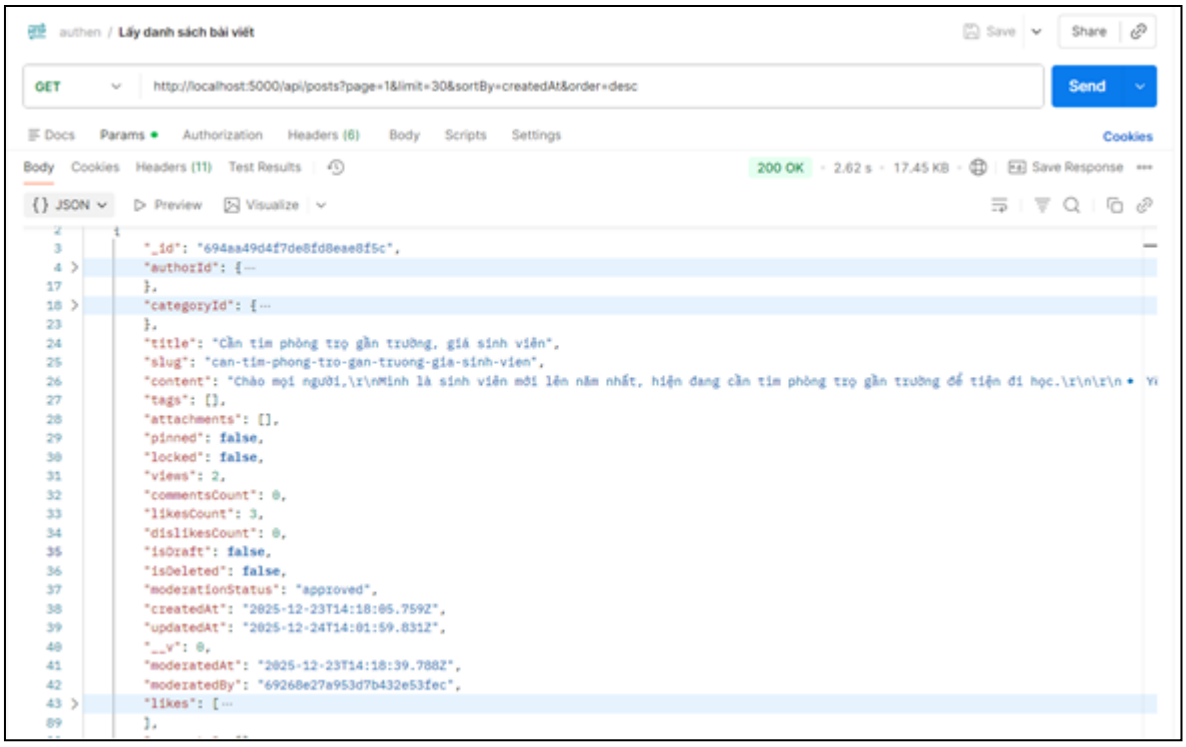


Hình 4.1. Kiểm thử API đăng nhập

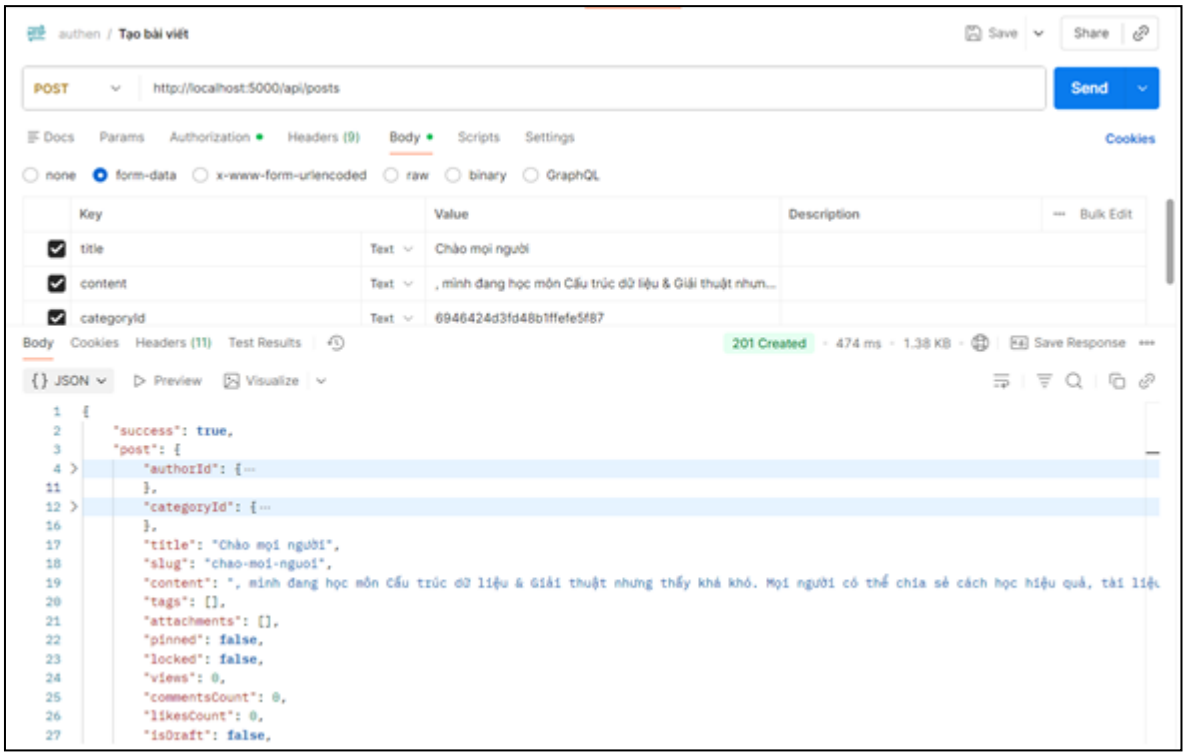


Hình 4.2. Kiểm thử API đăng ký

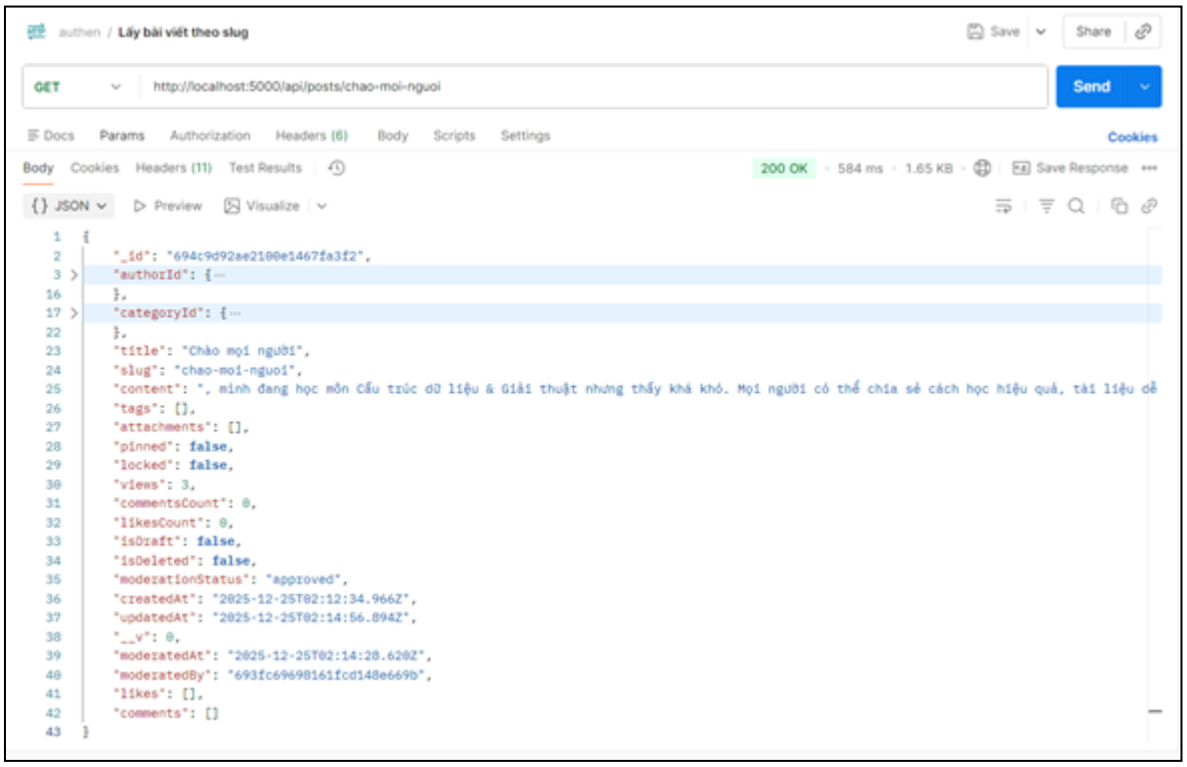
API quản lý bài viết: tạo bài viết, lấy danh sách bài viết, xem chi tiết bài viết theo slug, thích bài viết.



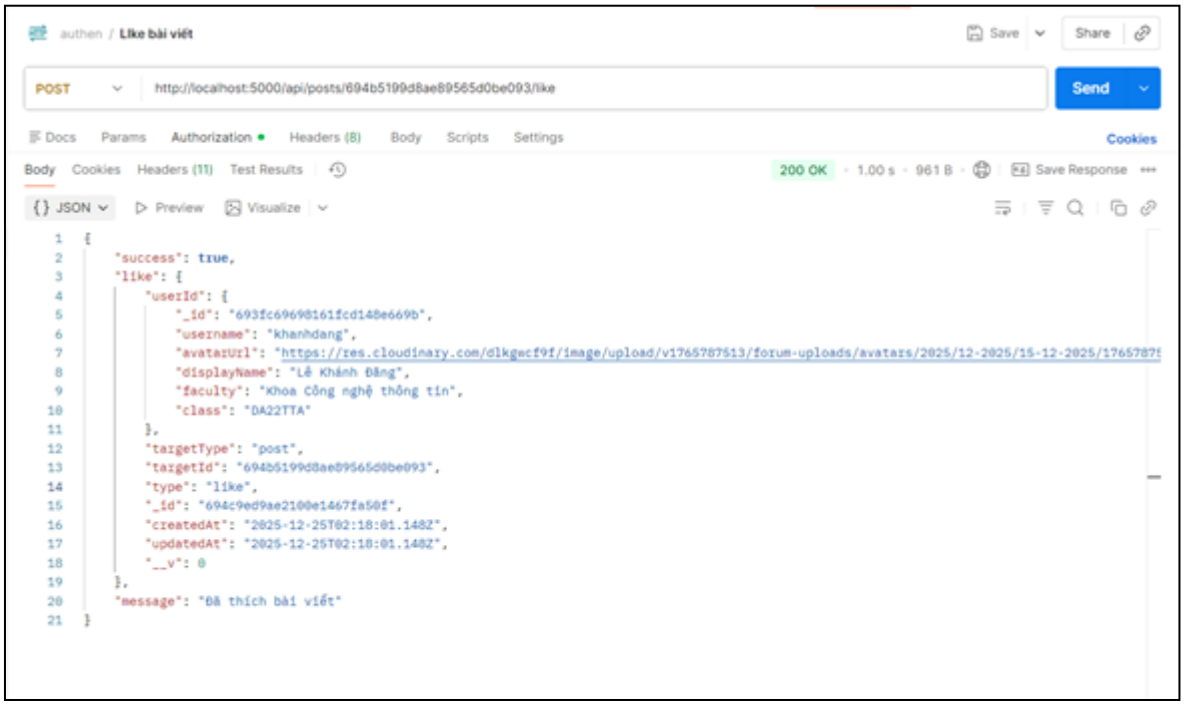
Hình 4.3. Kiểm thử API lấy danh sách bài viết



Hình 4.4. Kiểm thử API tạo bài viết

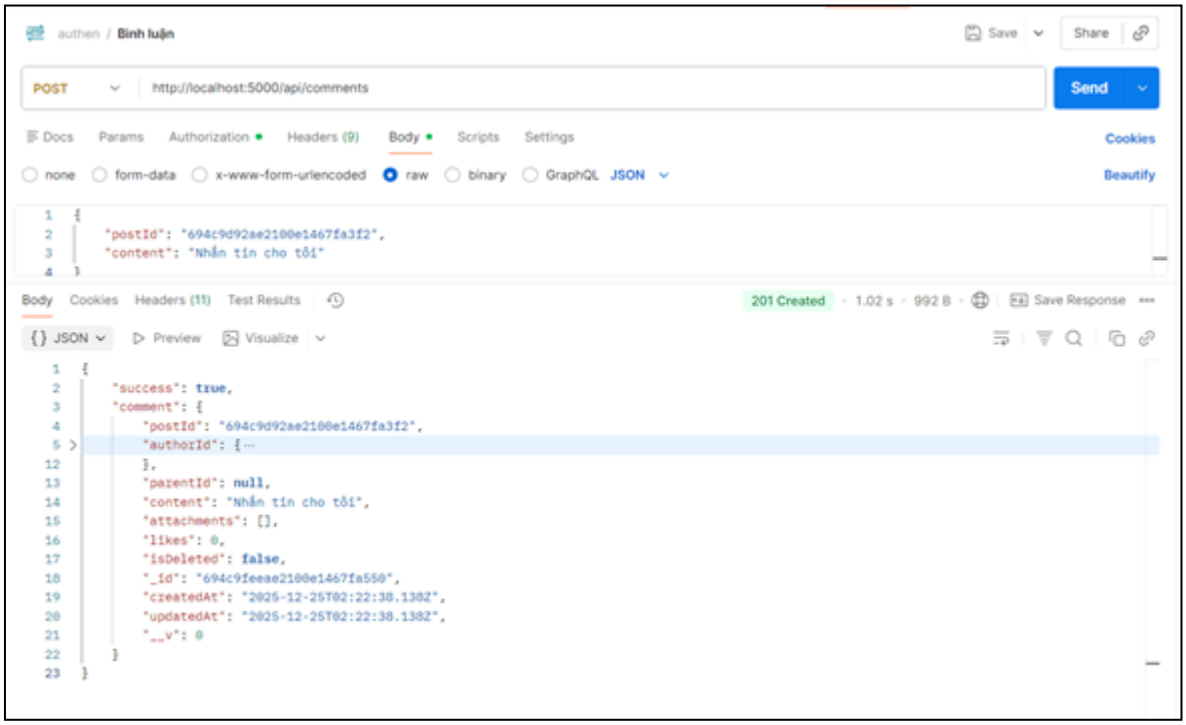


Hình 4.5. Kiểm thử API xem chi tiết bài viết theo slug

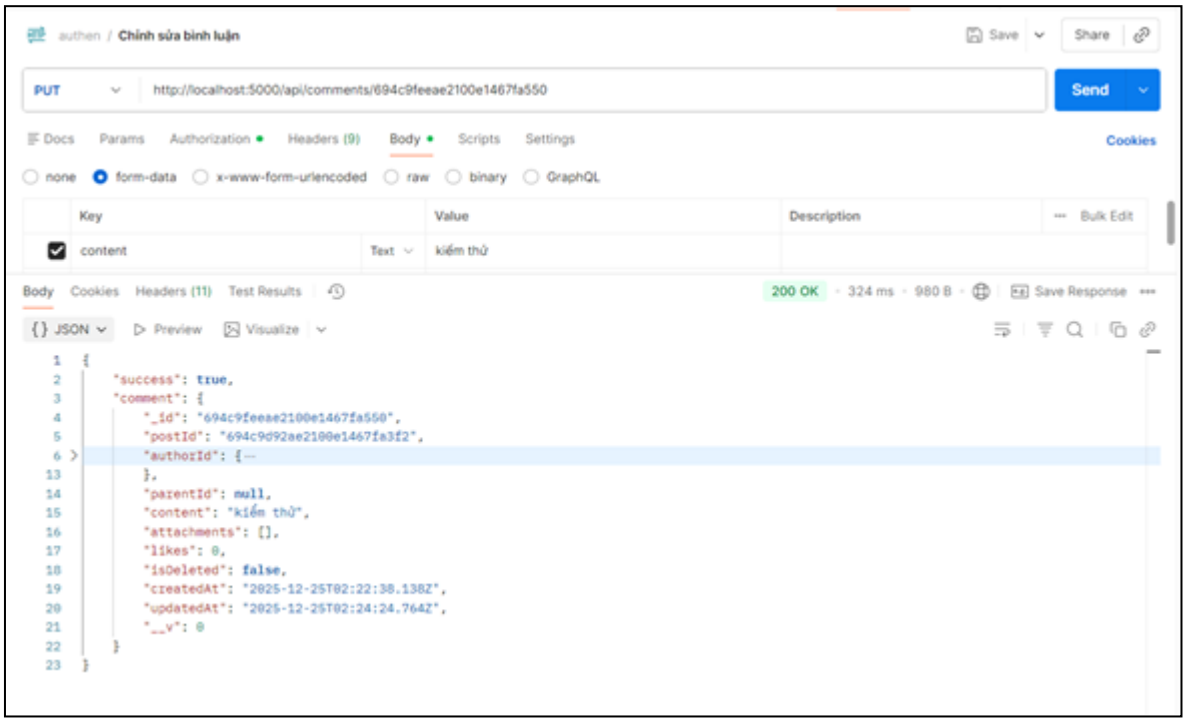


Hình 4.6. Kiểm thử API like bài viết

API quản lý bình luận: tạo bình luận, chỉnh sửa bình luận.

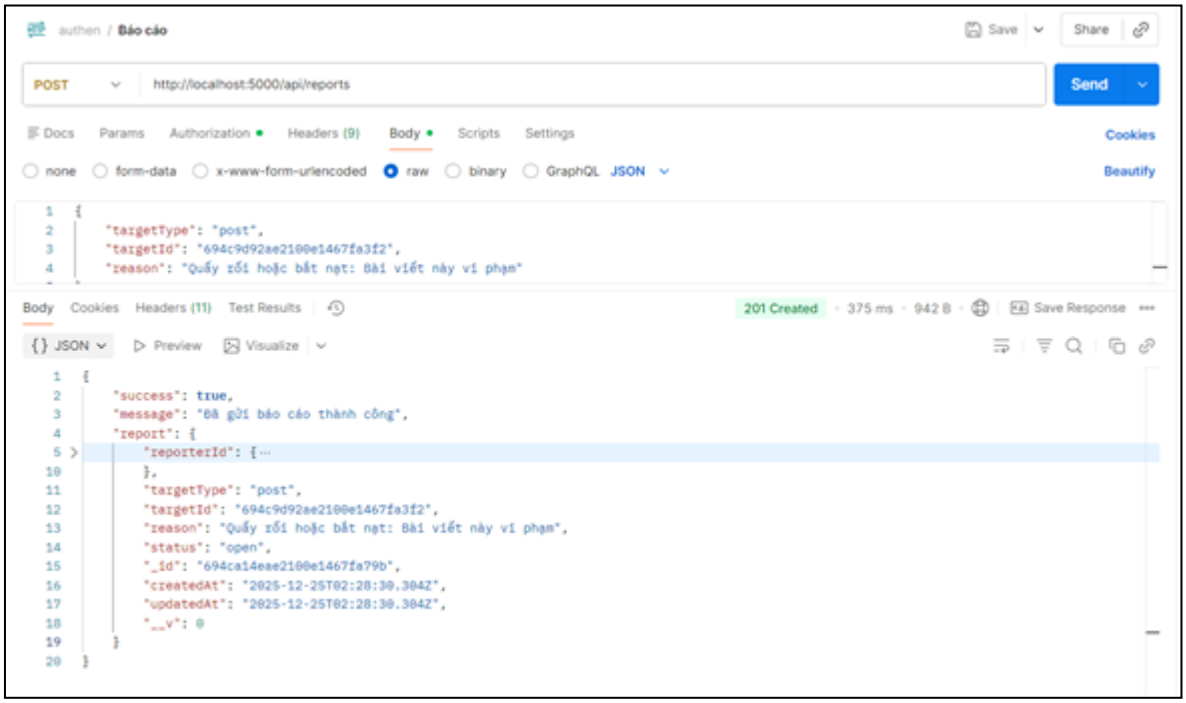


Hình 4.7. Kiểm thử API bình luận



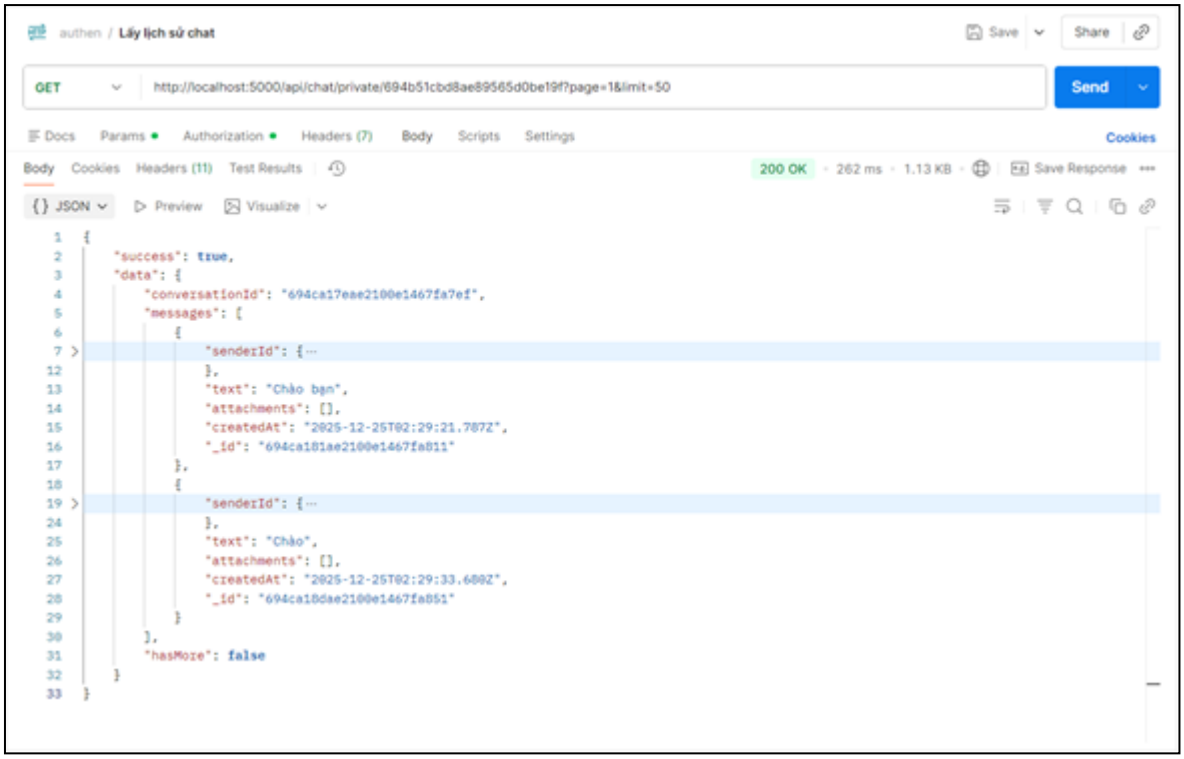
Hình 4.8. Kiểm thử API chỉnh sửa bình luận

API báo cáo vi phạm: gửi báo cáo

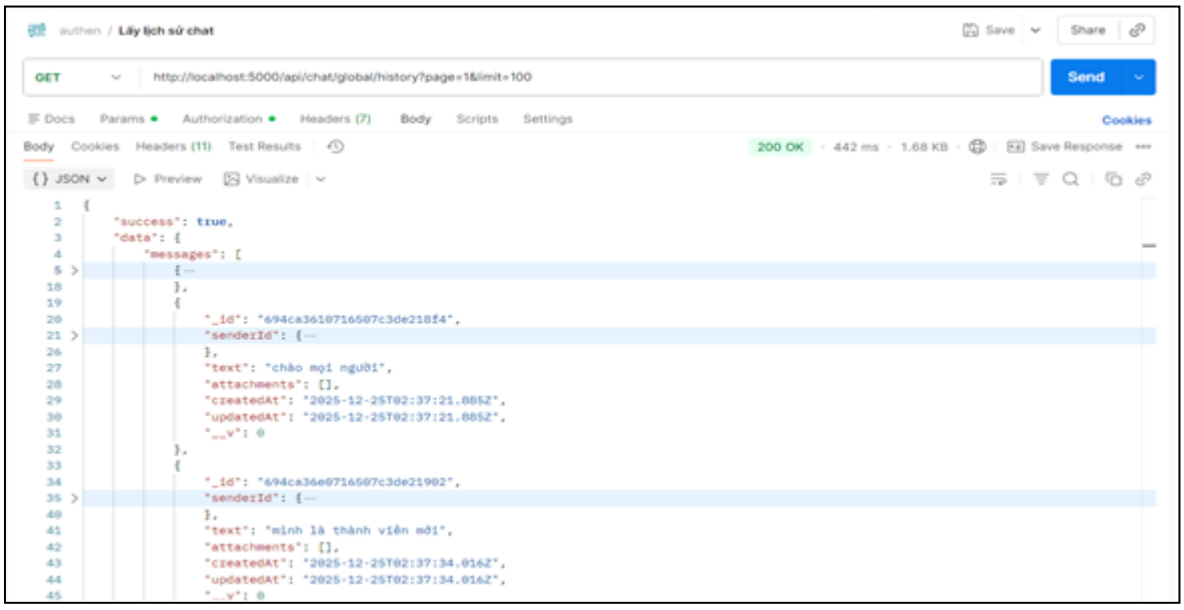


Hình 4.9. Kiểm thử API gửi báo cáo

API chat và giao tiếp thời gian thực: lấy lịch sử chat riêng, lấy lịch sử chat toàn diễn đàn.



Hình 4.10. Kiểm thử API lấy lịch sử chat riêng



Hình 4.11. Kiểm thử API lấy lịch sử chat toàn diễn đàn

Kết quả kiểm thử cho thấy các API phản hồi đúng định dạng, xử lý chính xác các yêu cầu hợp lệ và từ chối các yêu cầu không hợp lệ hoặc không đủ quyền truy cập. Điều này chứng minh hệ thống backend hoạt động ổn định và sẵn sàng cho việc tích hợp với giao diện người dùng.

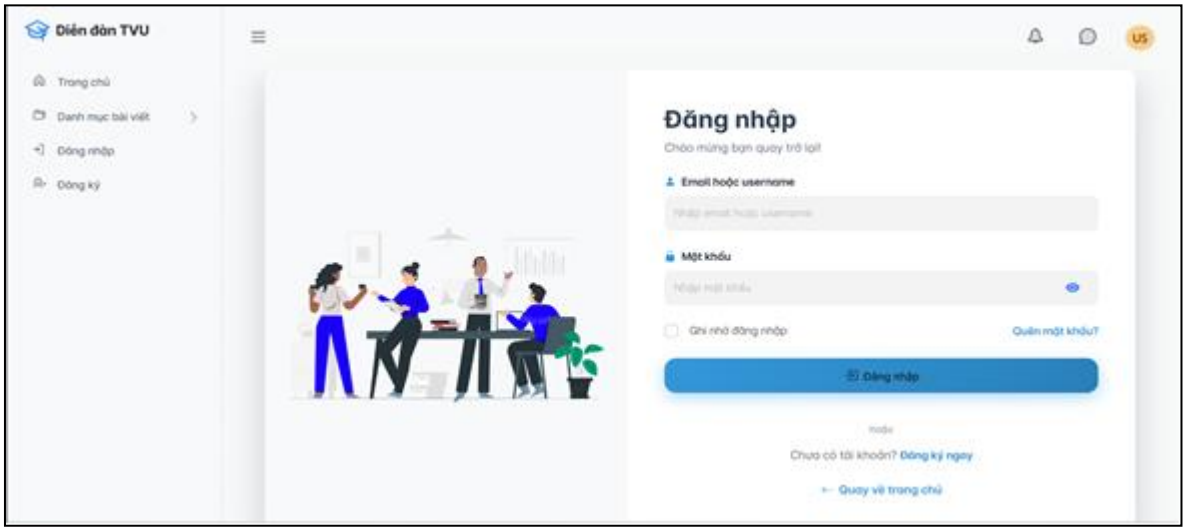
4.2. Giao diện người dùng

Giao diện người dùng của website diễn đàn sinh viên được xây dựng bằng ReactJS, hướng đến sự đơn giản, trực quan và dễ sử dụng. Giao diện được thiết kế theo hướng hiện đại, đảm bảo tính thân thiện với người dùng và dễ tiếp cận ngay cả với những sinh viên lần đầu sử dụng hệ thống.

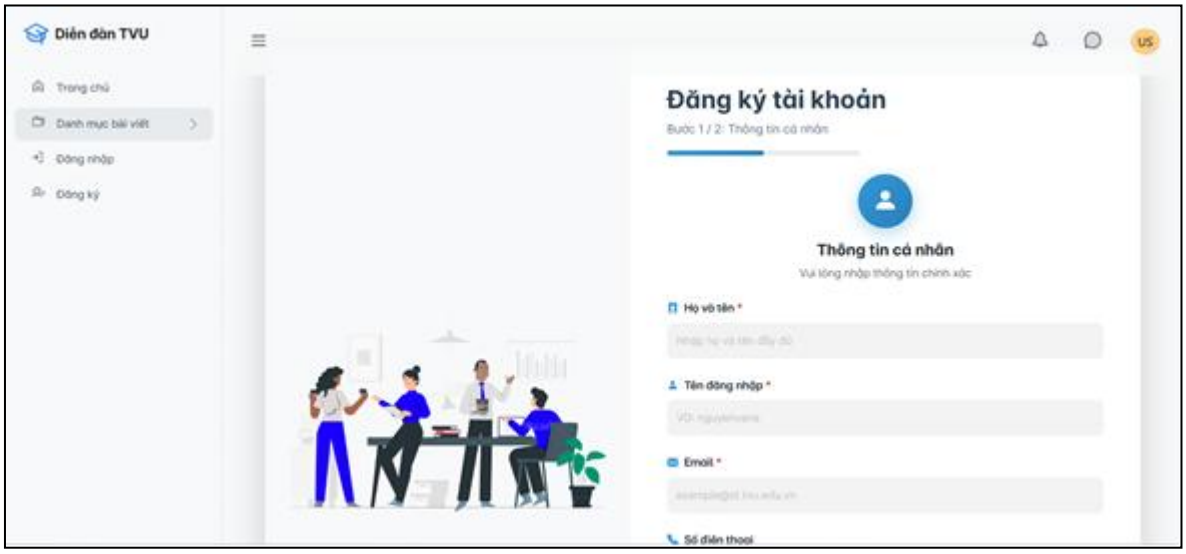
Các chức năng chính như đăng ký - đăng nhập, xem danh sách bài viết, đăng bài, bình luận và tìm kiếm được bố trí khoa học, logic, giúp người dùng dễ dàng thao tác và tiết kiệm thời gian trong quá trình sử dụng. Màu sắc và bố cục giao diện được lựa chọn phù hợp với môi trường học tập, tạo cảm giác thoải mái khi sử dụng lâu dài.

Bên cạnh đó, giao diện được xây dựng theo mô hình component của ReactJS, giúp việc quản lý và mở rộng các chức năng trở nên linh hoạt. Website cũng hỗ trợ hiển thị tốt trên nhiều thiết bị khác nhau như máy tính, máy tính bảng và điện thoại di động, góp phần nâng cao trải nghiệm người dùng và đáp ứng nhu cầu truy cập mọi lúc, mọi nơi của sinh viên.

Giao diện đăng ký và đăng nhập: Giao diện này hỗ trợ người dùng tạo tài khoản mới bằng cách nhập các thông tin cần thiết, đồng thời cho phép đăng nhập vào hệ thống thông qua tài khoản đã đăng ký. Chức năng này giúp quản lý người dùng hiệu quả, đảm bảo xác thực danh tính và kiểm soát quyền truy cập vào các chức năng của hệ thống.



Hình 4.12. Giao diện đăng nhập



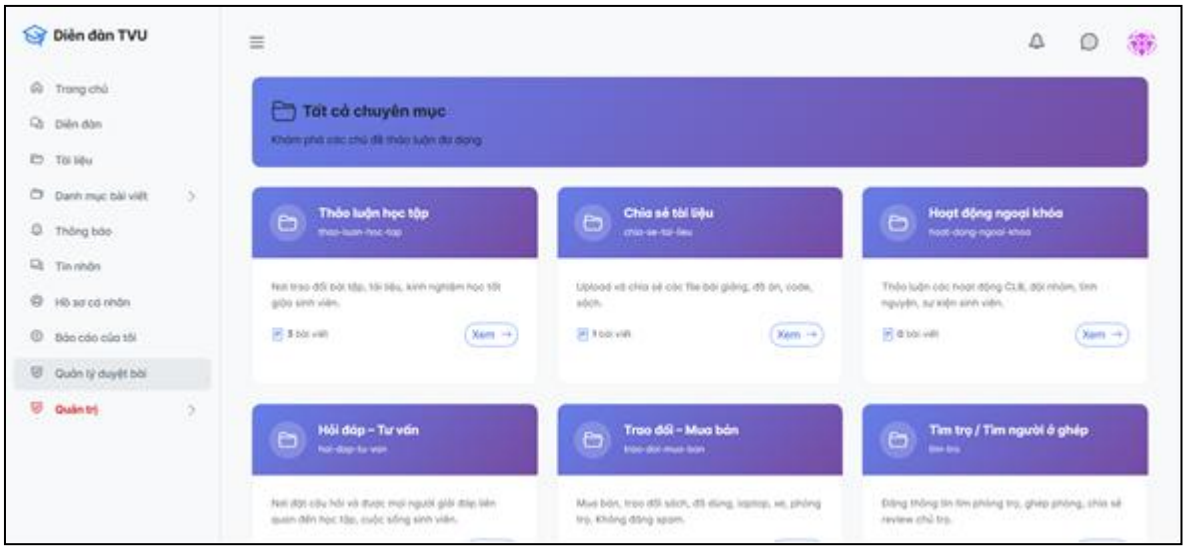
Hình 4.13. Giao diện đăng ký

Giao diện trang chủ diễn đàn: hiển thị các bài viết mới nhất cùng danh mục liên quan, giúp người dùng nhanh chóng tiếp cận nội dung đang được quan tâm, đồng thời cung cấp thông tin tổng quan để sinh viên dễ dàng theo dõi và tham gia thảo luận.

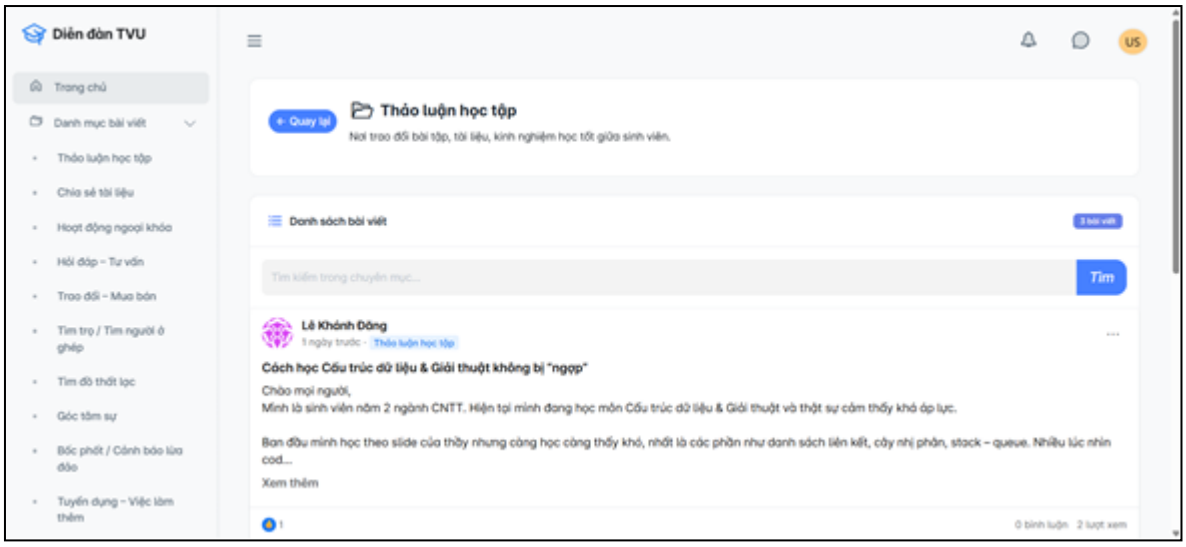
Hình 4.14. Giao diện trang chủ

Hình 4.15. Giao diện trang chủ hiển thị bài viết

Giao diện trang danh mục và bài viết: cho phép người dùng xem và lọc các bài viết theo từng chủ đề cụ thể, đồng thời hỗ trợ chức năng tìm kiếm theo từ khóa, giúp sinh viên nhanh chóng tìm được nội dung phù hợp với nhu cầu học tập và trao đổi.



Hình 4.16. Giao diện trang hiển thị tất cả danh mục

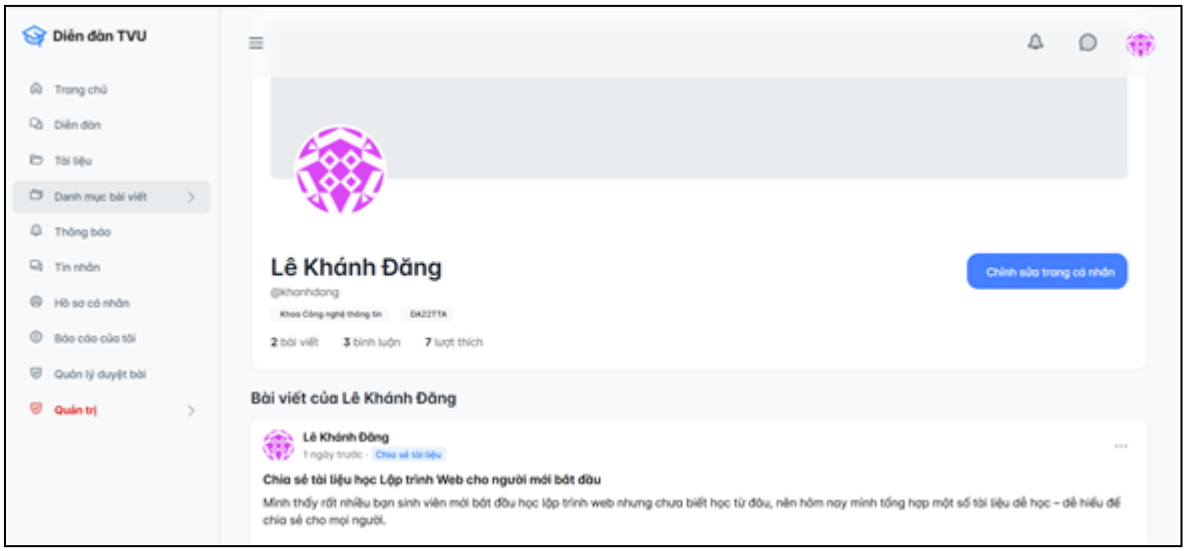


Hình 4.17. Giao diện trang bài viết theo danh mục

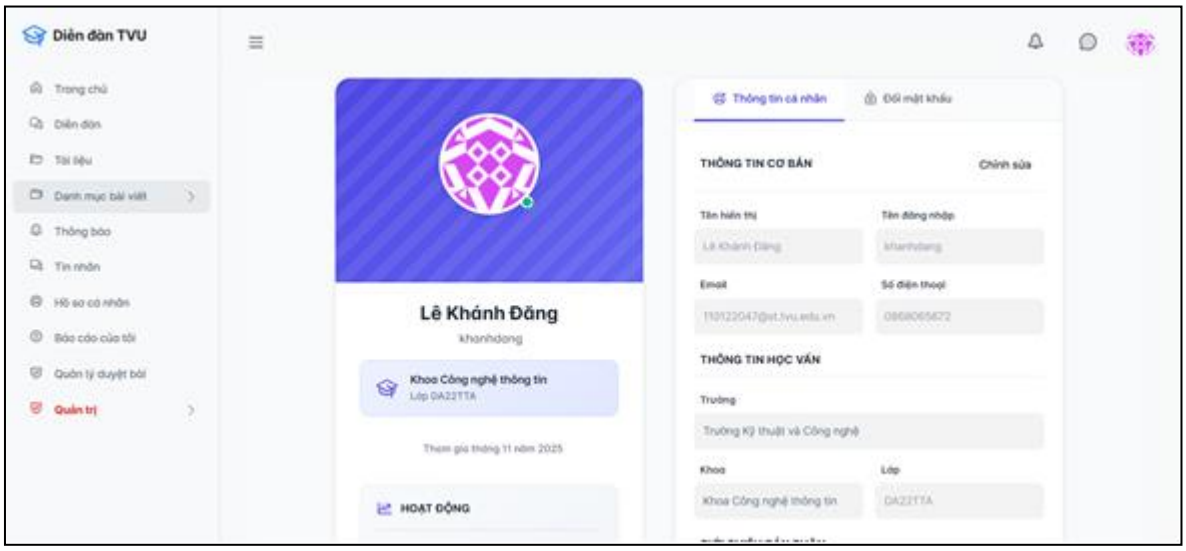
Giao diện trang chi tiết bài viết: hiển thị đầy đủ nội dung của bài viết, thông tin tác giả, thời gian đăng. Tại đây, người dùng có thể thực hiện các chức năng tương tác như thích bài viết, viết bình luận, chia sẽ và trao đổi ý kiến.

Hình 4.18. Giao diện trang chi tiết bài viết

Giao diện trang quản lý tài khoản cá nhân cho phép người dùng xem và chỉnh sửa các thông tin cá nhân như tên hiển thị, email, mô tả bản thân và ảnh đại diện. Giao diện được thiết kế đơn giản, dễ thao tác, giúp người dùng chủ động cập nhật hồ sơ, qua đó cá nhân hóa tài khoản và nâng cao trải nghiệm sử dụng diễn đàn.

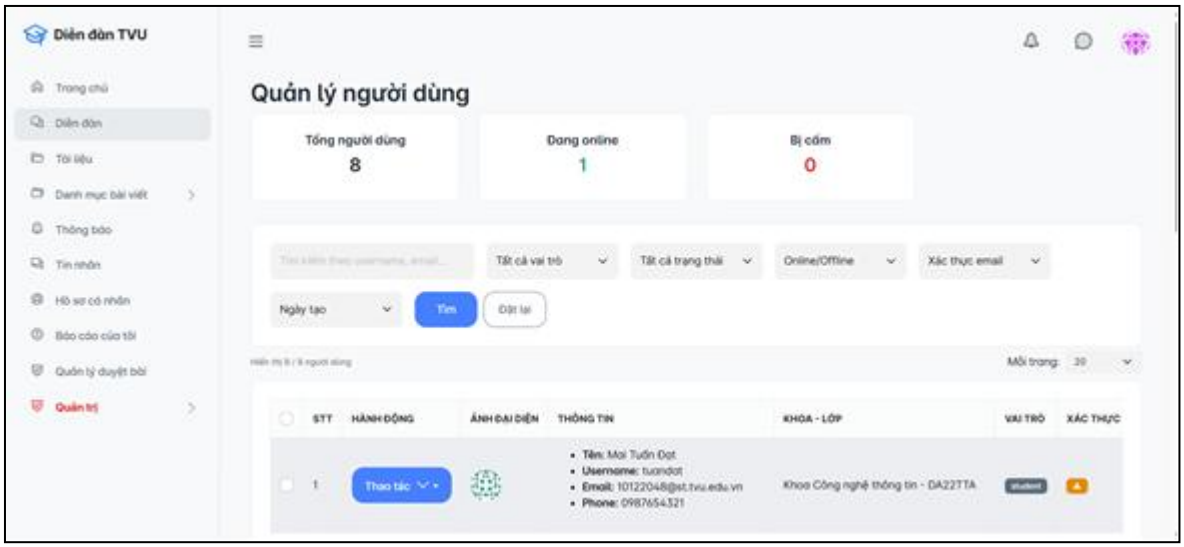


Hình 4.19. Giao diện trang hồ sơ cá nhân

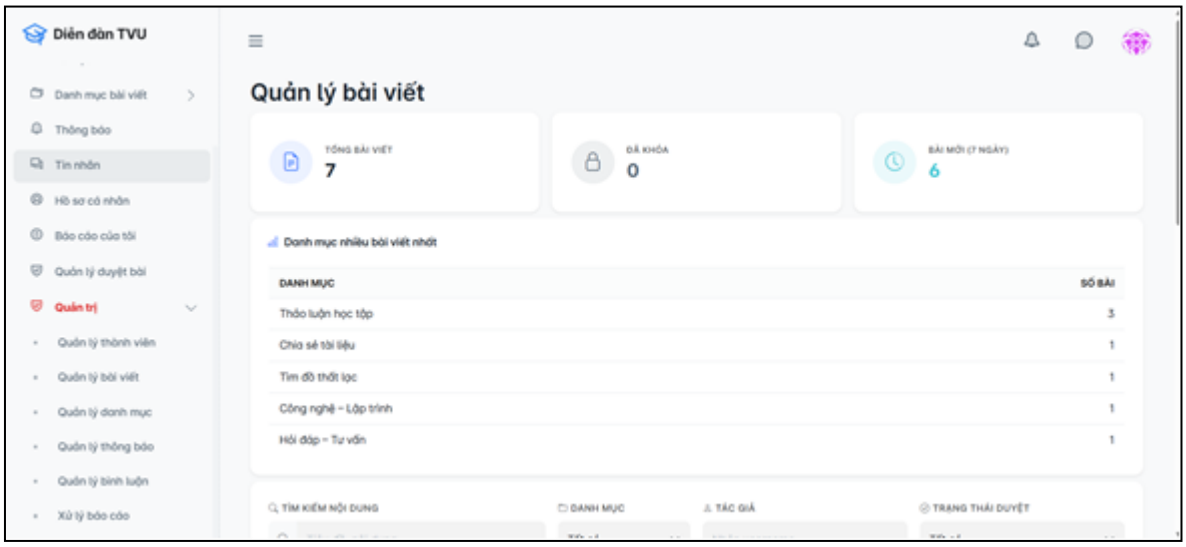


Hình 4.20. Giao diện trang cập nhật hồ sơ cá nhân

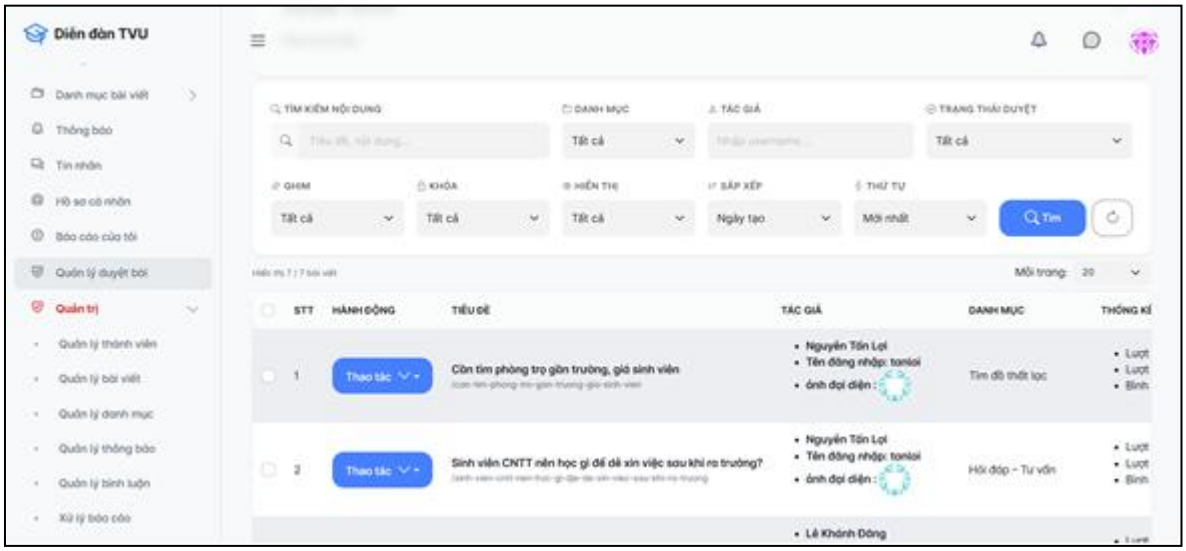
Giao diện trang quản trị được xây dựng nhằm hỗ trợ quản trị viên và người điều hành theo dõi, kiểm soát và vận hành toàn bộ hệ thống diễn đàn. Thông qua giao diện này, người quản trị có thể quản lý danh sách người dùng, giám sát và xử lý bài viết, bình luận, xem thống kê hoạt động cũng như tiếp nhận và xử lý các báo cáo vi phạm. Các chức năng được bố trí khoa học, rõ ràng, giúp công tác quản lý diễn đàn diễn ra hiệu quả, đảm bảo môi trường trao đổi học tập lành mạnh và ổn định.



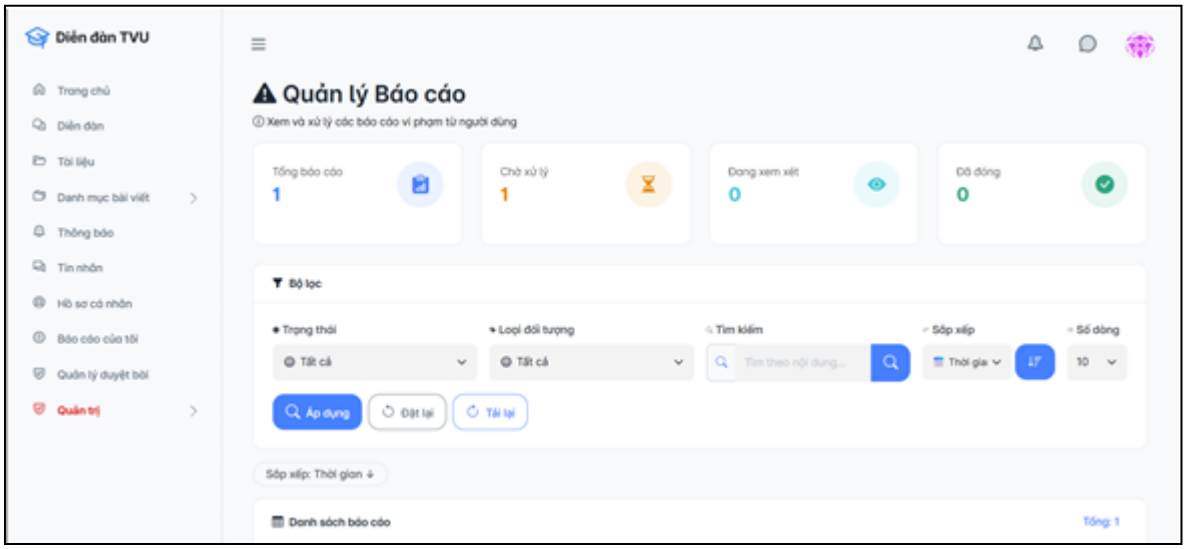
Hình 4.21. Giao diện trang quản lý người dùng



Hình 4.22. Giao diện trang quản lý thống kê bài viết

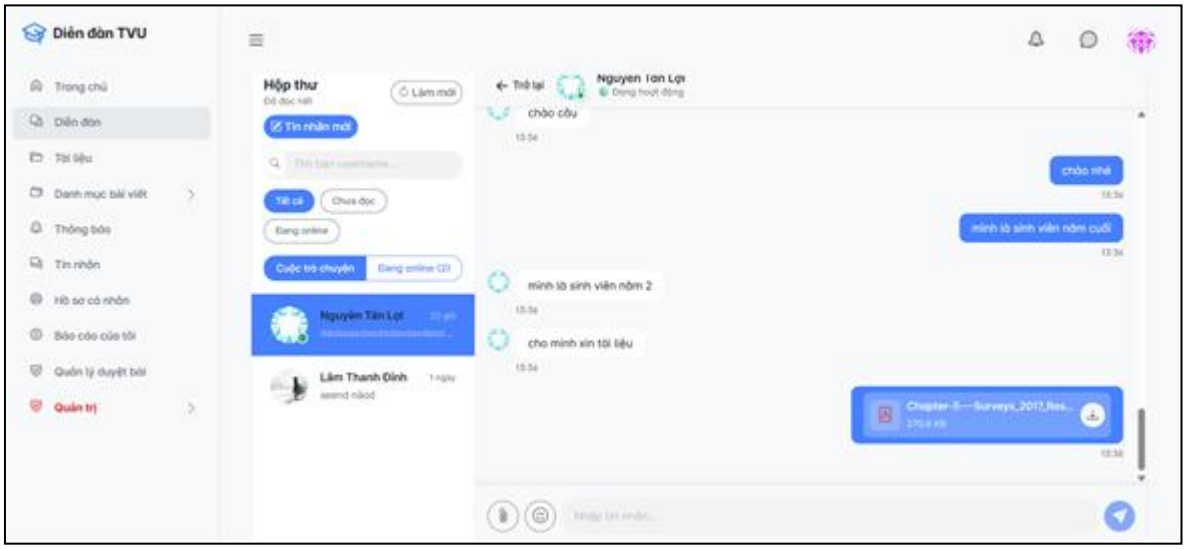


Hình 4.23. Giao diện trang quản lý bài viết

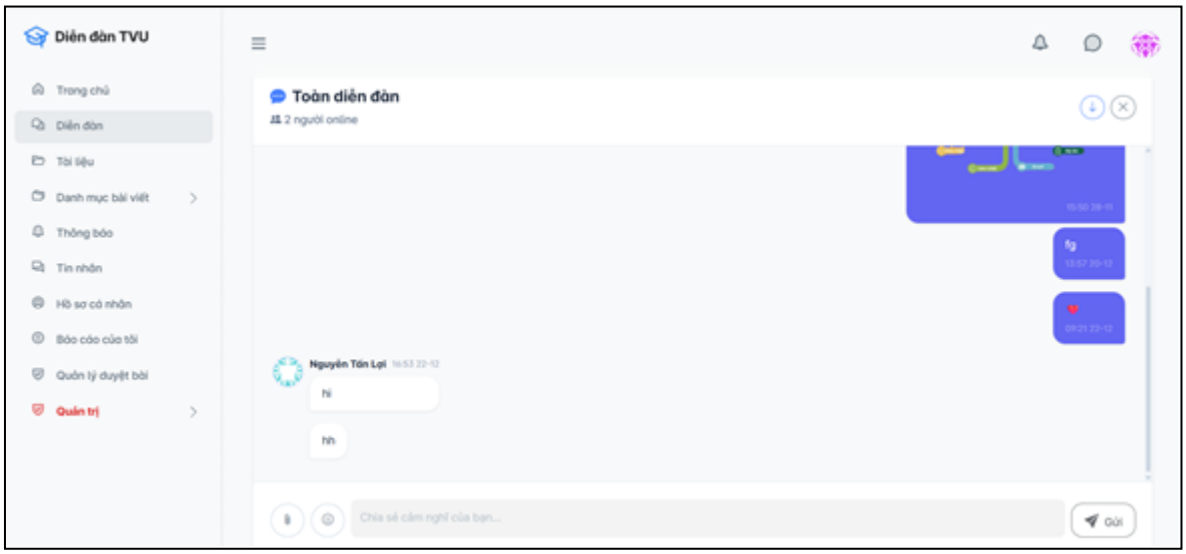


Hình 4.24. Giao diện quản lý các báo cáo

Giao diện trang chat: cho phép người dùng tham gia trò chuyện theo thời gian thực, bao gồm cả hình thức chat riêng giữa các người dùng và chat chung toàn diễn đàn. Chức năng này giúp sinh viên dễ dàng trao đổi thông tin, thảo luận nhanh các vấn đề học tập và tăng cường sự tương tác, kết nối giữa các thành viên trong diễn đàn.



Hình 4.25. Giao diện chat riêng



Hình 4.26. Giao diện chat toàn diễn đàn

Qua quá trình sử dụng thử nghiệm, giao diện hoạt động mượt mà, phản hồi nhanh và dễ thao tác. Các chức năng thời gian thực như thông báo và chat giúp nâng cao trải nghiệm người dùng, tạo cảm giác tương tác liên tục khi sử dụng hệ thống.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Đồ án "Xây dựng Website diễn đàn sinh viên" được thực hiện với mục tiêu xây dựng một hệ thống thông tin trực tuyến phục vụ nhu cầu trao đổi học tập, chia sẻ kiến thức và kết nối cộng đồng sinh viên trong môi trường số. Thông qua quá trình nghiên cứu, phân tích yêu cầu, thiết kế và hiện thực hệ thống, đồ án đã hoàn thành các mục tiêu đặt ra ban đầu.

Kết quả của đồ án là xây dựng được một website diễn đàn sinh viên hoạt động ổn định, đáp ứng đầy đủ các chức năng cơ bản như quản lý tài khoản người dùng, đăng và quản lý bài viết, bình luận, tìm kiếm nội dung, thông báo và trò chuyện. Hệ thống được thiết kế theo mô hình client - server với frontend sử dụng ReactJS, backend sử dụng NodeJS kết hợp Express.js và cơ sở dữ liệu MongoDB, đảm bảo tính linh hoạt và khả năng mở rộng.

Đồ án đã vận dụng hiệu quả các kiến thức đã học về lập trình web, thiết kế kiến trúc hệ thống, cơ sở dữ liệu NoSQL và giao tiếp thời gian thực vào việc xây dựng một sản phẩm phần mềm hoàn chỉnh. Kết quả đạt được cho thấy hệ thống đáp ứng tốt các yêu cầu chức năng và phi chức năng đã đề ra, có thể ứng dụng trong thực tế với vai trò là một nền tảng trao đổi thông tin dành cho sinh viên.

5.2. Hướng phát triển

Mặc dù hệ thống đã hoàn thành các chức năng cơ bản, nhưng do giới hạn về thời gian và phạm vi đồ án, hệ thống vẫn còn nhiều tiềm năng để tiếp tục phát triển và hoàn thiện trong tương lai. Một số hướng phát triển có thể được đề xuất như sau:

- Mở rộng thêm các chức năng nâng cao như gợi ý bài viết theo sở thích người dùng, đánh dấu bài viết yêu thích hoặc theo dõi chủ đề quan tâm.

- Cải thiện hiệu năng hệ thống khi số lượng người dùng và dữ liệu tăng cao, tối ưu truy vấn cơ sở dữ liệu và cơ chế xử lý đồng thời.

- Tăng cường các giải pháp bảo mật như xác thực đa yếu tố, kiểm soát truy cập nâng cao và giám sát các hành vi bất thường.

- Phát triển phiên bản ứng dụng trên thiết bị di động nhằm nâng cao khả năng tiếp cận và tiện lợi cho người dùng.

- Bổ sung các công cụ hỗ trợ quản trị và thống kê nhằm giúp quản trị viên dễ dàng theo dõi và quản lý hoạt động của diễn đàn.

Các hướng phát triển trên sẽ góp phần hoàn thiện hệ thống website diễn đàn sinh viên, nâng cao hiệu quả sử dụng và mở rộng khả năng ứng dụng trong môi trường học tập và trao đổi trực tuyến.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Meta Platforms, Inc., React - A JavaScript library for building user interfaces. https://react.dev/

[2] M. Banks, Learning React, O'Reilly Media, 2020.

[3] OpenJS Foundation, Node.js Documentation. https://nodejs.org/en/docs/

[4] Express.js Foundation, Express.js Documentation. https://expressjs.com/

[5] Socket.IO, Socket.IO Documentation. https://socket.io/docs/

[6] MongoDB Inc., MongoDB Manual. https://www.mongodb.com/docs/

[7] Postman Inc., Postman API Testing Documentation. [Online]. Available: https://learning.postman.com/