TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH**

**HỌC KỲ I, NĂM HỌC 2025-2026**

# WEBSITE TUYÊN TRUYỀN VÀ CHIA SẺ MẸO SỐNG XANH

*Giảng viên hướng dẫn:*

ThS. Khấu Văn Nhật

*Sinh viên thực hiện:*

Họ tên: Kiên Thị Bé Hai

MSSV:110122218

Lớp: DA22TTA

TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH**

**HỌC KỲ I, NĂM HỌC 2025-2026**

**WEBSITE TUYÊN TRUYỀN**

**VÀ CHIA SẺ MẸO SỐNG XANH**

*Giảng viên hướng dẫn:*

ThS. Khấu Văn Nhật

*Sinh viên thực hiện:*

Họ tên: Kiên Thị Bé Hai

MSSV:110122218

Lớp: DA22TTA

**NHẬN XÉT CỦA GIẢNG 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**

Em xin bày tỏ lòng biết ơn sâu sắc đến giảng viên hướng dẫn thầy Khấu Văn Nhựt người đã dành thời gian quý báu để hướng dẫn, góp ý và hỗ trợ em trong suốt quá trình thực hiện đồ án chuyên ngành. Sự tận tâm và kiến thức chuyên sâu của thầy đã giúp em vượt qua nhiều khó khăn trong việc nghiên cứu và phát triển dự án. Em cũng xin chân thành cảm ơn các thầy cô trong Khoa Công nghệ Thông tin, Trường Kỹ thuật và Công nghệ, đã truyền đạt những kiến thức nền tảng vững chắc, tạo điều kiện cho em áp dụng vào thực tế. Ngoài ra, em xin gửi lời tri ân đến gia đình và bạn bè, những người luôn động viên, khích lệ em trong giai đoạn thực hiện đồ án.

Cuối cùng, em xin cảm ơn các nguồn tài liệu tham khảo và cộng đồng lập trình trực tuyến đã cung cấp thông tin hữu ích cho nghiên cứu này.

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN 15](#_Toc3220)

[1.1. Thực trạng vấn đề môi trường và lối sống xanh tại Việt Nam 15](#_Toc16196)

[1.2. Tổng quan các website/cộng đồng về sống xanh hiện nay 15](#_Toc32141)

[1.3. Khoảng trống và hướng giải quyết 16](#_Toc15507)

[1.4. Mục tiêu cụ thể của đồ án 17](#_Toc23313)

[CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT 18](#_Toc20828)

[2.1. Tổng quan về hệ thống Website tuyên truyền và chia sẻ mẹo sống xanh 18](#_Toc3036)

[2.1.1. Giới thiệu hệ thống 18](#_Toc3112)

[2.1.2. Các phân hệ chức năng 18](#_Toc9231)

[2.2. Kiến trúc MEAN Stack 18](#_Toc22889)

[2.2.1. Tổng quan kiến trúc MERN Stack 18](#_Toc11230)

[2.2.2 Mô hình MVC trong kiến trúc MERN Stack 18](#_Toc24548)

[2.2.3 Luồng xử lý dữ liệu trong hệ thống MERN 19](#_Toc30933)

[2.2.4 Đánh giá kiến trúc MERN Stack 19](#_Toc10599)

[2.3. Công nghệ và framework front-end 20](#_Toc5684)

[2.3.1. React.js 20](#_Toc16949)

[2.3.1.1. Giới thiệu React.js (hay React) 20](#_Toc23851)

*[2.3.1.2. Khái niệm React.js](#_Toc3278)* [20](#_Toc3278)

*[2.3.1.3 Cấu trúc](#_Toc9217)* [21](#_Toc9217)

*[2.3.1.4 Cách thức hoạt động](#_Toc13568)* [22](#_Toc13568)

*[2.3.1.5. Vai trò của React.js](#_Toc25452)* [22](#_Toc25452)

*[2.3.1.5. Ưu và nhược điểm của React.js](#_Toc17501)* [23](#_Toc17501)

*[2.3.1.6. Các ứng dụng của React.js](#_Toc27551)* [23](#_Toc27551)

[2.3.2. Vite.js 23](#_Toc11147)

[2.3.2.1. Giới thiệu Vite.js 23](#_Toc20189)

[2.3.2.2. Khái niệm và cách hoạt động 24](#_Toc28438)

[2.3.3. Công nghệ TailwindCSS 24](#_Toc5521)

[2.3.3.1. Giới thiệu TailwindCSS 24](#_Toc23451)

[2.3.3.2. Khái niệm 24](#_Toc12349)

[2.3.3.3 Cấu trúc 24](#_Toc1162)

[2.3.3.4 Cách thức hoạt động 24](#_Toc32185)

[2.3.3.5. Vai trò của TailwindCSS 24](#_Toc10713)

[2.3.3.6. Ưu và nhược điểm của TailwindCSS 25](#_Toc12445)

*[2.3.3.7. Các ứng dụng của TailwindCSS](#_Toc17494)* [25](#_Toc17494)

[2.4 Công nghệ và framework back-end 25](#_Toc29899)

[2.4.1. Node.js 25](#_Toc13690)

[2.4.1.1. Giới thiệu Node.js 25](#_Toc32370)

[2.4.1.2. Khái niệm 25](#_Toc16110)

[2.4.1.3. Cấu trúc: 25](#_Toc13826)

[2.4.1.4 Cách thức hoạt động 25](#_Toc18207)

[2.4.1.5. Vai trò của Node.js 26](#_Toc4329)

[2.4.1.6. Ưu và nhược điểm của Node.js 26](#_Toc23449)

[2.4.1.7. Các ứng dụng của Node.js 27](#_Toc25874)

[2.4.2. Công nghệ Express.js 27](#_Toc9978)

[2.4.2.1. Giới thiệu Express.js (hay gọi tắt là Express) 27](#_Toc32701)

[2.4.2.2. Khái niệm 27](#_Toc10333)

[2.4.2.3. Tại sao Developer lại sử dụng Express.js 27](#_Toc7731)

[2.4.2.3.1. Cấu trúc 28](#_Toc7104)

[2.4.2.3.2. Cách thức hoạt động 28](#_Toc15197)

[2.4.2.4. Vai trò của 28](#_Toc13161)

[2.4.2.5. Ưu và nhược điểm của Express.js 28](#_Toc10122)

[2.4.2.6. Các ứng dụng của Express.js 29](#_Toc21384)

[2.5 Hệ quản trị cơ sở dữ liệu 29](#_Toc7904)

[2.5.1. Cơ sở dữ liệu MongoDB 29](#_Toc21906)

*[2.5.1.1. Giới thiệu](#_Toc4925)* [29](#_Toc4925)

*[2.5.1.2. Khái niệm](#_Toc28968)* [29](#_Toc28968)

[2.5.1.4. Cách thức hoạt động: 30](#_Toc7700)

[2.5.1.5. Vai trò của MongoDB 30](#_Toc24828)

[2.5.1.6. Ưu và nhược điểm của MongoDB 30](#_Toc8748)

*[2.5.1.7. Các ứng dụng của MongoDB](#_Toc19474)* [31](#_Toc19474)

[2.5. Mô hình kiến trúc phần mềm MVC 31](#_Toc9330)

[2.5.1. Giới thiệu Mô hình MVC (Model - View - Controller) 31](#_Toc32204)

[2.5.2. Ứng dụng trong đồ án 31](#_Toc23611)

[2.6. Tiêu chuẩn giao tiếp RESTful API 32](#_Toc2984)

[2.6.1. Khái niệm REST (Representational State Transfer) 32](#_Toc16036)

[2.6.2. Cách thức hoạt động trong hệ thống 32](#_Toc15768)

[2.7. Cơ chế bảo mật và xác thực (Security & Authentication) 33](#_Toc9720)

[2.7.1. Mã hóa mật khẩu với Bcrypt 33](#_Toc30290)

[2.7.2. Xác thực JWT (JSON Web Token) 33](#_Toc32068)

[2.8. Công cụ hỗ trợ phát triển và triển khai 34](#_Toc795)

[2.8.1. Postman 34](#_Toc22910)

[2.8.2. GitHub và GitHub Actions 35](#_Toc14030)

[2.8.3. Docker 35](#_Toc29927)

[2.9. Cơ sở lý thuyết về môi trường và hành vi người dùng 35](#_Toc4654)

[2.9.1. Chỉ số chất lượng không khí (AQI) 35](#_Toc26265)

[2.9.2. Lý thuyết thay đổi hành vi qua tương tác xã hội 36](#_Toc3016)

[2.10. Lý luận khoa học của đề tài 36](#_Toc32602)

[2.11. Giả thuyết nghiên cứu 36](#_Toc23540)

[2.12. Phương pháp nghiên cứu 37](#_Toc15914)

[CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU 38](#_Toc5547)

[3.1 Mô tả bài toán 38](#_Toc8747)

[3.1.1 Mục tiêu hệ thống 38](#_Toc23380)

[3.1.2 Quy trình nghiệp vụ tổng quát 38](#_Toc3277)

[3.1.3 Đối tượng sử dụng 39](#_Toc14006)

[3.2 Đặc tả yêu cầu hệ thống 39](#_Toc15979)

[3.2.1 Yêu cầu chức năng 39](#_Toc5548)

[3.2.2. Yêu cầu phi chức năng (Non-Functional Requirements) 43](#_Toc2234)

[3.3. Phân tích và thiết kế hệ thống 43](#_Toc14674)

[3.3.1. Kiến trúc tổng thể 43](#_Toc28083)

[3.3.2. Thiết kế cơ sở dữ liệu (Database Design) 44](#_Toc5059)

[3.3.3. Sơ đồ Use Case 50](#_Toc22249)

[3.3.4 Sơ đồ luồng dữ liệu DFD 53](#_Toc12451)

[3.3.4.1 Sơ đồ luồng dữ liệu DFD mức 0 53](#_Toc12389)

[3.3.4.2 Sơ đồ luồng dữ liệu DFD mức 1 53](#_Toc26368)

[3.3.4.3 Sơ đồ luồng dữ liệu DFD mức 2 55](#_Toc19322)

[3.3.5 Phác thảo giao diện 56](#_Toc3802)

[3.4. Cài đặt và Hiện thực hóa chương trình 64](#_Toc2387)

[3.4.1. Môi trường phát triển và Công cụ 64](#_Toc28889)

[3.4.2. Cấu trúc tổ chức mã nguồn Backend 64](#_Toc8679)

[3.4.3. Hiện thực hóa các chức năng cốt lõi 65](#_Toc21136)

[CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU 66](#_Toc17675)

[4.1 Tổng quan kết quả đạt được 66](#_Toc12392)

[4.2 Đánh giá hiệu năng hệ thống 66](#_Toc30041)

[4.2.1 Hiệu năng phía máy chủ (Backend Performance) 66](#_Toc25370)

[4.2.2 Hiệu năng phía máy khách (Frontend Performance) 66](#_Toc12039)

[4.2.3 Hiệu năng tích hợp API bên thứ ba 67](#_Toc27367)

[4.3 Đánh giá trải nghiệm người dùng (User Experience) 67](#_Toc31429)

[4.3.1 Giao diện và tính khả dụng 67](#_Toc22879)

[4.3.2 Tính tương tác và phản hồi 67](#_Toc18658)

[4.3.3 Khả năng tiếp cận (Accessibility) 68](#_Toc30355)

[4.4 Trình bày giao diện và chức năng hệ thống 68](#_Toc24128)

[4.4.1 Giao diện trang chủ (Homepage Interface) 68](#_Toc25768)

[4.4.2 Giao diện quản lý nội dung (Content Management Interface) 70](#_Toc10770)

[4.4.3 Giao diện tương tác cộng đồng (Community Interaction Interface 73](#_Toc23839)

[4.4.4 Giao diện quản trị hệ thống (Administrative Interface) 75](#_Toc21127)

[4.5 Đánh giá tổng thể và hạn chế 83](#_Toc15701)

[4.5.1 Điểm mạnh của hệ thống 83](#_Toc30665)

[4.5.2 Hạn chế và cơ hội cải thiện 84](#_Toc1028)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 85](#_Toc26619)

[5.1 Kết luận 85](#_Toc12239)

[5.1.1 Kết quả đạt được 85](#_Toc5146)

[5.1.2 Đóng góp mới 85](#_Toc12057)

[5.1.3 Đề xuất mới 85](#_Toc21738)

[5.2 Hướng phát triển 86](#_Toc21437)

[5.2.1 Nâng cao tính năng cốt lõi 86](#_Toc18525)

**DANH MỤC HÌNH ẢNH**

[Hình 2.2. Cách thức hoạt động của Node.js 44](#_Toc3414)

[Hình 2.3 Mô hình MVC của hệ thống 50](#_Toc18018)

[Hình 2.4. Sơ đồ trình tự xác thực người dùng bằng JWT 52](#_Toc13527)

[Hình 3.1 Sơ đồ kiến trúc hệ thống 62](#_Toc13911)

[Hình 3.2 Sơ đồ cấu trúc dữ liệu MongoDB 64](#_Toc15912)

[Hình 3.3 Use Case người dùng 69](#_Toc14471)

[Hình 3.4 Use Case quản trị viên 70](#_Toc15659)

[Hình 3.5 Sơ đồ luồng dữ liệu DFD mức ngữ cảnh 71](#_Toc3542)

[Hình 3.6 Sơ đồ luồng dữ liệu DFD mức 1 của người dùng 72](#_Toc31364)

[Hình 3.7 Sơ đồ luồng dữ liệu DFD mức 1 của quản trị viên 73](#_Toc8545)

[Hình 3.8 Sơ đồ luồng dữ liệu DFD mức 2 đăng nhập 73](#_Toc25494)

[Hình 3.9 Sơ đồ luồng dữ liệu DFD mức 2 đăng bài 73](#_Toc30502)

[Hình 3.7 Sơ đồ luồng dữ liệu DFD mức 2 thích, bình luận 74](#_Toc14838)

[Hình 3.8 Phác thảo giao diện đăng ký 74](#_Toc4224)

[Hình 3.9 Phác thảo giao diện đăng nhập 75](#_Toc2308)

[Hình 3.10 Phác thảo giao diện trang chủ 76](#_Toc24848)

[Hình 3.10 Phác thảo giao diện trang tin tức 77](#_Toc30896)

[Hình 3.11 Phác thảo giao diện trang kiến thức 78](#_Toc4767)

[Hình 3.12 Phác thảo giao diện trang diễn đàn 79](#_Toc7281)

[Hình 3.13 Phác thảo giao diện trang xem chi tiết bài viết 80](#_Toc19674)

[Hình 3.14 Phác thảo giao diện trang đang bài 81](#_Toc29044)

[Hình 3.15 Phác thảo giao diện trang hồ sơ 82](#_Toc20339)

[Hình 4.1 Giao diện trang chủ 87](#_Toc31042)

[Hình 4.2 Giao diện trang tin tức 88](#_Toc22303)

[Hình 4.3 Giao diện trang kiến thức 89](#_Toc27420)

[Hình 4.4 Giao diện trang diễn đàn 91](#_Toc16706)

[Hình 4.5 Giao diện trang xem chi tiết bài viết 93](#_Toc5282)

[Hình 4.6 Giao diện trang quản trị 94](#_Toc20556)

[Hình 4.7 Giao diện trang quản lý người dùng 95](#_Toc22435)

[Hình 4.8 Giao diện trang quản lý diễn đàn 96](#_Toc30090)

[Hình 4.9 Giao diện trang quản lý tin tức 97](#_Toc17451)

[Hình 4.10 Giao diện trang quản lý kiến thức 98](#_Toc13483)

[Hình 4.11 Giao diện trang quản lý topic 99](#_Toc23877)

[Hình 4.12 Giao diện trang quản lý danh mục 100](#_Toc17439)

[Hình 4.13 Giao diện trang quản lý báo cáo 101](#_Toc28486)

**DANH MỤC BẢNG BIỂU**

[Bảng 2.1: So sánh React với các framework tương tự 23](#_Toc30850)

**TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH**

Đồ án tập trung nghiên cứu và phát triển một website tuyên truyền ý thức bảo vệ môi trường và chia sẻ các mẹo thực hành lối sống xanh trong cộng đồng. Trước bối cảnh biến đổi khí hậu và ô nhiễm môi trường ngày càng nghiêm trọng tại Việt Nam cũng như trên toàn cầu, việc nâng cao nhận thức và thúc đẩy hành vi sống bền vững là nhiệm vụ cấp thiết. Website được xây dựng nhằm cung cấp một nền tảng trực tuyến thân thiện, dễ tiếp cận, cho phép người dùng tìm hiểu kiến thức, chia sẻ kinh nghiệm thực tiễn và tham gia thảo luận về các giải pháp sống xanh.

Đồ án sử dụng công nghệ hiện đại MERN Stack (MongoDB, Express.js, React.js, Node.js) kết hợp với TailwindCSS để phát triển một ứng dụng web động, đáp ứng tốt trên nhiều thiết bị. Các tính năng chính bao gồm: trang chủ hiển thị chỉ số chất lượng không khí theo thời gian thực, chuyên mục bài viết tuyên truyền, diễn đàn chia sẻ mẹo sống xanh, hệ thống đăng nhập/đăng ký, bình luận, thích bài viết, tìm kiếm nâng cao và module quản trị dành cho admin. Hệ thống được thiết kế theo kiến trúc MVC, đảm bảo tính module hóa, dễ bảo trì và mở rộng.

Kết quả đạt được là một website hoàn chỉnh, đã được triển khai trực tuyến tại địa chỉ [https://green-living-tips.onrender.com](https://green-living-tips.onrender.com" \t "_blank) (hoặc tên miền thực tế), hoạt động ổn định, giao diện thân thiện, đáp ứng đầy đủ các yêu cầu chức năng và phi chức năng đề ra. Đồ án không chỉ là sản phẩm công nghệ mà còn góp phần lan tỏa thông điệp bảo vệ môi trường đến cộng đồng người dùng internet tại Việt Nam.

**MỞ ĐẦU**

**1. Lý do chọn đề tài**

Trong bối cảnh toàn cầu hóa và công nghiệp hóa diễn ra mạnh mẽ, biến đổi khí hậu và ô nhiễm môi trường đã và đang trở thành những thách thức cấp bách nhất của nhân loại. Việt Nam, với đặc thù địa lý và tốc độ phát triển kinh tế nhanh, được đánh giá là một trong những quốc gia chịu ảnh hưởng nặng nề nhất từ các hiện tượng thời tiết cực đoan và nước biển dâng. Các báo cáo gần đây từ Ngân hàng Thế giới (World Bank, 2023) đã chỉ ra rằng chất lượng không khí tại các đô thị lớn như Hà Nội và Thành phố Hồ Chí Minh thường xuyên ở mức kém và nguy hại, với chỉ số AQI nhiều thời điểm vượt ngưỡng an toàn, gây ảnh hưởng trực tiếp đến sức khỏe cộng đồng.

Bên cạnh ô nhiễm không khí, vấn đề rác thải cũng đang ở mức báo động. Ước tính mỗi năm Việt Nam phát sinh khoảng 1,8 triệu tấn rác thải nhựa, tuy nhiên tỷ lệ tái chế chỉ đạt khoảng 27%, dẫn đến tình trạng quá tải tại các bãi chôn lấp và nguy cơ ô nhiễm nguồn nước nghiêm trọng. Mặc dù Chính phủ đã ban hành nhiều chỉ thị và nghị quyết, như Chỉ thị số 20/CT-TTg hay Nghị quyết Đại hội XIII về xử lý cơ sở gây ô nhiễm, nhưng việc hiện thực hóa các mục tiêu này vẫn gặp nhiều trở ngại do hạn chế về nguồn lực và hạ tầng kỹ thuật.

Tuy nhiên, một trong những rào cản lớn nhất hiện nay nằm ở nhận thức và hành vi của cộng đồng. Mặc dù khái niệm "sống xanh" đã bắt đầu xuất hiện, nhưng việc chuyển đổi từ nhận thức sang hành động cụ thể vẫn còn hạn chế. Nguyên nhân chủ yếu xuất phát từ sự thiếu hụt các nguồn thông tin chính thống, có hệ thống và dễ tiếp cận. Các nền tảng mạng xã hội hiện nay tuy phổ biến nhưng thông tin về môi trường thường bị phân mảnh, thiếu kiểm chứng hoặc bị pha loãng bởi các nội dung giải trí khác. Trong khi đó, các trang web chuyên ngành của cơ quan nhà nước thường tập trung vào dữ liệu quan trắc khô khan, thiếu tính tương tác và không tạo được môi trường giao lưu cho người dùng.

Xuất phát từ thực tiễn trên, việc xây dựng một nền tảng công nghệ chuyên biệt, kết hợp giữa tuyên truyền kiến thức khoa học và tạo không gian tương tác cộng đồng là vô cùng cần thiết. Với chuyên ngành Công nghệ thông tin, tôi quyết định thực hiện đề tài "Website tuyên truyền và chia sẻ mẹo sống xanh". Đề tài không chỉ nhằm mục đích ứng dụng các công nghệ lập trình web hiện đại (MERN Stack) vào thực tế mà còn mong muốn góp phần xây dựng một cộng đồng trực tuyến văn minh, nơi mọi người có thể dễ dàng tiếp cận thông tin, chia sẻ kinh nghiệm và cùng nhau lan tỏa lối sống bền vững.

**2. Mục đích nghiên cứu**

Đề tài tập trung vào việc nghiên cứu và phát triển một hệ thống website hoàn chỉnh với các mục tiêu cụ thể sau:

Về mặt xã hội: Tuyên truyền, nâng cao nhận thức của cộng đồng, đặc biệt là giới trẻ, về các vấn đề môi trường cấp bách như biến đổi khí hậu, ô nhiễm không khí và rác thải nhựa. Đồng thời, tạo ra một không gian kết nối để người dùng chia sẻ các mẹo sống xanh thực tiễn (tái chế, tiết kiệm năng lượng, trồng cây...), từ đó thúc đẩy thay đổi hành vi theo hướng tích cực.

Về mặt kỹ thuật: Ứng dụng thành công các công nghệ phát triển web hiện đại theo mô hình MERN Stack (MongoDB, Express.js, React.js, Node.js) để xây dựng một ứng dụng web có hiệu năng cao, giao diện thân thiện và hoạt động ổn định trên đa nền tảng. Hệ thống phải đảm bảo các tính năng tương tác thời gian thực như hiển thị chỉ số AQI, bình luận và hệ thống quản trị nội dung chặt chẽ.

**3. Đối tượng và phạm vi nghiên cứu**

Đối tượng nghiên cứu:

- Các công nghệ lập trình web Full-stack, cụ thể là thư viện React.js cho giao diện người dùng (Frontend) và Node.js/Express.js cho hệ thống máy chủ (Backend), cùng cơ sở dữ liệu NoSQL MongoDB.

- Hành vi, nhu cầu tìm kiếm thông tin và thói quen tương tác của người dùng internet tại Việt Nam đối với các chủ đề về môi trường.

- Các giải pháp kỹ thuật để tối ưu hóa trải nghiệm người dùng trên diễn đàn trực tuyến và tích hợp dữ liệu thời gian thực (API thời tiết/không khí

Phạm vi nghiên cứu:

- Về nội dung: Tập trung vào các chủ đề sống xanh phù hợp với ngữ cảnh Việt Nam như: phân loại rác tại nguồn, tái chế sáng tạo, tiết kiệm điện nước và tiêu dùng bền vững.

- Về công nghệ: Giới hạn trong khuôn khổ phát triển ứng dụng web (Web Application) sử dụng JavaScript làm ngôn ngữ chủ đạo (MERN Stack) và TailwindCSS cho thiết kế giao diện.

- Về người dùng: Hướng tới đối tượng chính là học sinh, sinh viên và những người trẻ quan tâm đến môi trường, có thói quen sử dụng internet thường xuyên.

- Về không gian và thời gian: Đề tài được thực hiện và triển khai thử nghiệm trong phạm vi đồ án chuyên ngành từ tháng 10/2025 đến tháng 12/2025

**4. Ý nghĩa khoa học và thực tiễn**

Ý nghĩa khoa học: Đề tài là cơ hội để sinh viên hệ thống hóa và vận dụng tổng hợp các kiến thức chuyên ngành đã được đào tạo (Lập trình Web, Cơ sở dữ liệu, Kiến trúc phần mềm, An toàn thông tin...) vào giải quyết một bài toán thực tế. Việc xây dựng hệ thống theo kiến trúc MVC và Client-Server giúp kiểm chứng khả năng tổ chức mã nguồn, quản lý dữ liệu và thiết kế API chuẩn mực.

Ý nghĩa thực tiễn: Sản phẩm sau khi hoàn thiện có khả năng hoạt động độc lập và triển khai thực tế, cung cấp một công cụ hữu ích cho cộng đồng. Website không chỉ lấp đầy khoảng trống về thông tin môi trường mà còn đóng vai trò là cầu nối, khuyến khích các hành động nhỏ nhưng thiết thực. Về lâu dài, hệ thống có tiềm năng phát triển thành một mạng xã hội thu nhỏ chuyên về môi trường, góp phần lan tỏa thông điệp bảo vệ hành tinh xanh đến đông đảo người dân.

**CHƯƠNG 1: TỔNG QUAN**

**1.1. Thực trạng vấn đề môi trường và lối sống xanh tại Việt Nam**

Việt Nam hiện đang nằm trong nhóm các quốc gia chịu ảnh hưởng nặng nề nhất của biến đổi khí hậu và các hiện tượng thời tiết cực đoan. Tình trạng ô nhiễm môi trường tại các đô thị lớn đang ở mức báo động, đặc biệt là ô nhiễm không khí. Theo các số liệu quan trắc, chỉ số chất lượng không khí (AQI) tại Hà Nội và Thành phố Hồ Chí Minh thường xuyên vượt ngưỡng 150, thậm chí đạt mức nguy hại trên 250 vào những thời điểm cuối năm 2024 và đầu năm 2025, phản ánh sự gia tăng phát thải từ giao thông và công nghiệp. Bên cạnh đó, vấn đề quản lý chất thải rắn cũng đang gặp nhiều bất cập. Mỗi năm, Việt Nam phát sinh khoảng 1,8 triệu tấn rác thải nhựa, nhưng tỷ lệ tái chế chỉ dừng lại ở mức khiêm tốn khoảng 27%, dẫn đến quá tải tại các bãi chôn lấp và gây ô nhiễm nghiêm trọng cho nguồn nước và hệ sinh thái biển.

Về mặt chính sách, Chính phủ đã có những động thái quyết liệt như ban hành Chỉ thị số 20/CT-TTg hay đặt mục tiêu xử lý 100% cơ sở gây ô nhiễm nghiêm trọng vào năm 2025. Tuy nhiên, việc chuyển hóa các chính sách này thành hành động thực tiễn trong cộng đồng vẫn còn nhiều hạn chế. Lối sống xanh, bao gồm các hành vi như phân loại rác, tiết kiệm năng lượng hay sử dụng sản phẩm thân thiện với môi trường, vẫn chưa trở thành thói quen phổ biến. Các phong trào hiện tại thường mang tính tự phát, thiếu sự hỗ trợ của các hệ thống công nghệ để duy trì và nhân rộng, dẫn đến hiệu quả chưa cao và thiếu tính bền vững.

**1.2. Tổng quan các website/cộng đồng về sống xanh hiện nay**

Trên bình diện quốc tế, các nền tảng như Treehugger.com, EcoWatch.com và TheGoodTrade.com đã cung cấp nội dung phong phú về lối sống xanh, bao gồm bài viết, video và diễn đàn thảo luận về bảo vệ môi trường. Tuy nhiên, nội dung chủ yu bằng tiếng Anh và chưa được tùy chỉếnh phù hợp với ngữ cảnh văn hóa, kinh tế của các quốc gia đang phát triển như Việt Nam, dẫn đến hạn chế về tính tiếp cận đối với người dùng địa phương.

Tại Việt Nam, một số website và cộng đồng đã hình thành nhằm thúc đẩy ý thức bảo vệ môi trường. Ví dụ, GreenViet.org tập trung vào bảo tồn quần thể động vật hoang dã như Chà vá chân nâu và các hệ sinh thái quan trọng, với các hoạt động nghiên cứu và giáo dục cộng đồng. Trang thông tin của Cục Môi trường (vea.mae.gov.vn) cung cấp dữ liệu chính thức về kiểm soát ô nhiễm, quản lý chất thải và quan trắc môi trường, phục vụ cho mục đích tham khảo chuyên sâu. Ngoài ra, Moitruongxanh.org, một dự án do học sinh khởi xướng, đã tổ chức các chiến dịch dọn rác và giáo dục môi trường qua sáu mùa hoạt động.

Các tổ chức phi chính phủ như CHANGE (Trung tâm Hành động và Liên kết vì Môi trường và Phát triển), WildAct (Trung tâm Hành động vì Động vật hoang dã Việt Nam) và GreenHub (Trung tâm Hỗ trợ Phát triển Xanh) cũng duy trì các trang web và chương trình hỗ trợ, tập trung vào giảm thiểu rác thải nhựa và bảo vệ động vật hoang dã. Bên cạnh đó, các báo điện tử như Moitruong.net.vn (Môi trường & Cuộc sống) và Nongnghiepmoitruong.vn cung cấp tin tức cập nhật về biến đổi khí hậu và lối sống xanh, với các bài viết về các dự án địa phương như Green Da Nang. Trang Tainguyenmoitruong.gov.vn của Trung tâm Truyền thông Tài nguyên và Môi trường cũng đóng vai trò quan trọng trong việc cập nhật sự kiện và tin tức liên quan đến môi trường.

Ngoài ra, một số ứng dụng di động như EcoHero hỗ trợ người dùng theo dõi hành trình sống xanh cá nhân, nhưng chúng chủ yếu tập trung vào ghi chép cá nhân mà thiếu tính năng tương tác cộng đồng rộng rãi. Các blog như GreenInVietnam.org thảo luận về giải pháp lối sống xanh để bảo vệ môi trường toàn cầu, nhấn mạnh vai trò của hành động giảm ô nhiễm và bảo vệ động vật hoang dã.

**1.3. Khoảng trống và hướng giải quyết**

Mặc dù đã có nhiều nền tảng tồn tại, nhưng tại Việt Nam vẫn còn khoảng trống đáng kể trong việc kết hợp tuyên truyền kiến thức môi trường với không gian chia sẻ kinh nghiệm thực tiễn. Hầu hết các website hiện tại tập trung vào cung cấp thông tin một chiều hoặc hoạt động của tổ chức cụ thể, thiếu tính năng tương tác như diễn đàn đăng bài, bình luận và tìm kiếm nâng cao theo chủ đề. Ví dụ, các trang chính thức như Cục Môi trường cung cấp dữ liệu quan trắc nhưng không hỗ trợ người dùng đóng góp nội dung cá nhân, trong khi các diễn đàn trên mạng xã hội như Facebook thường bị pha loãng bởi thông tin không kiểm chứng.

Khoảng trống này càng rõ nét khi xem xét nhu cầu của người dùng trẻ tuổi, những người mong muốn một nền tảng hiện đại, thân thiện với thiết bị di động, tích hợp quản lý nội dung chuyên nghiệp và khả năng phân loại theo chủ đề cụ thể như năng lượng tái tạo, tái chế rác thải hay giao thông xanh. Hơn nữa, trong bối cảnh ô nhiễm môi trường ngày càng nghiêm trọng, cần có các công cụ tích hợp dữ liệu thời gian thực như chỉ số chất lượng không khí để nâng cao tính thực tiễn và cấp bách của thông tin.

Hướng giải quyết mà đồ án đề xuất là phát triển một website chuyên biệt, sử dụng công nghệ MERN Stack để đảm bảo tính động và khả năng mở rộng. Website sẽ lấp đầy khoảng trống bằng cách kết hợp tuyên truyền khoa học, diễn đàn chia sẻ mẹo sống xanh và hệ thống quản trị nội dung, nhằm tạo ra một cộng đồng bền vững và dễ tiếp cận cho người dùng Việt Nam.

**1.4. Mục tiêu cụ thể của đồ án**

Xây dựng một website hoạt động ổn định, đáp ứng tốt trên các thiết bị desktop với giao diện thân thiện và hiện đại sử dụng TailwindCSS.

Phát triển ít nhất năm chuyên mục chính về các chủ đề môi trường, bao gồm bài viết tuyên truyền về biến đổi khí hậu, ô nhiễm và lợi ích sống xanh, cùng với diễn đàn chia sẻ mẹo thực tiễn như tái chế rác thải, tiết kiệm năng lượng và trồng cây tại nhà.

Tích hợp hệ thống đăng nhập/đăng ký, phân quyền người dùng và admin, cho phép đăng bài, bình luận, thích nội dung và quản lý bài viết với chức năng duyệt và thống kê.

Triển khai các tính năng nâng cao như tìm kiếm nội dung theo chủ đề, hiển thị chỉ số chất lượng không khí thời gian thực và phân tích dữ liệu sử dụng, nhằm nâng cao trải nghiệm người dùng và khả năng vận hành lâu dài sau khi hoàn thành đồ án.

# CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1. Tổng quan về hệ thống Website tuyên truyền và chia sẻ mẹo sống xanh

2.1.1. Giới thiệu hệ thống

Hệ thống được xây dựng là một nền tảng web trực tuyến tích hợp giữa cổng thông tin tuyên truyền và mạng xã hội thu nhỏ. Mục tiêu chính là nâng cao nhận thức cộng đồng về các vấn đề môi trường cấp bách như biến đổi khí hậu, ô nhiễm không khí và rác thải nhựa. Đồng thời, hệ thống cung cấp không gian để người dùng chia sẻ các kinh nghiệm thực tiễn (mẹo sống xanh) như tái chế, tiết kiệm năng lượng, từ đó thúc đẩy thay đổi hành vi theo hướng bền vững.

2.1.2. Các phân hệ chức năng

Hệ thống được chia thành hai phân hệ chính dựa trên quyền hạn người dùng, được kiểm soát chặt chẽ bởi cơ chế phân quyền (Role-based Authorization):

Phân hệ Người dùng (Client-side):

- Tiếp nhận thông tin: Xem các bài viết tuyên truyền, tin tức môi trường và theo dõi chỉ số chất lượng không khí (AQI) theo thời gian thực.

- Tương tác cộng đồng: Đăng ký/đăng nhập, chia sẻ mẹo sống xanh trên diễn đàn, bình luận, thích bài viết và lưu bài viết quan tâm.

Phân hệ Quản trị (Admin-side):

- Quản trị nội dung: Duyệt bài viết người dùng đóng góp (Approve), ghim các bài quan trọng (Pin) và quản lý danh mục tin tức/chủ đề diễn đàn.

- Quản trị người dùng: Xem danh sách thành viên, thực hiện khóa (Ban) hoặc mở khóa tài khoản vi phạm và phân quyền quản trị.

- Thống kê: Theo dõi tổng quan hoạt động của hệ thống thông qua Dashboard.

2.2. Kiến trúc MEAN Stack

2.2.1. Tổng quan kiến trúc MERN Stack

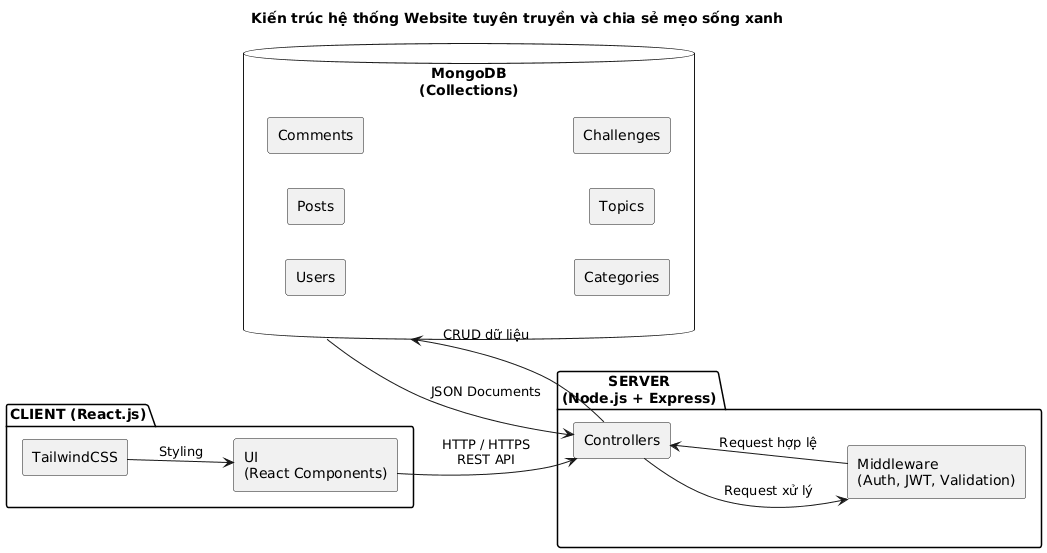
MERN Stack là một kiến trúc phát triển ứng dụng web full-stack hiện đại dựa hoàn toàn trên ngôn ngữ JavaScript. Tên gọi MERN là chữ viết tắt của bốn công nghệ chính gồm MongoDB, Express.js, React.js và Node.js. Kiến trúc này cho phép xây dựng các ứng dụng web theo mô hình Client–Server, hỗ trợ mạnh mẽ cho các ứng dụng Single Page Application (SPA), đồng thời đảm bảo khả năng mở rộng và hiệu năng cao

2.2.2 Mô hình MVC trong kiến trúc MERN Stack

Kiến trúc MERN Stack thường được triển khai theo mô hình MVC phân tán. Trong đó, React.js đảm nhiệm vai trò View để hiển thị giao diện và tiếp nhận tương tác người dùng; Express.js đóng vai trò Controller để xử lý nghiệp vụ và điều phối dữ liệu; MongoDB kết hợp với Mongoose Schema đóng vai trò Model để lưu trữ và quản lý dữ liệu. Việc áp dụng mô hình MVC giúp hệ thống có cấu trúc rõ ràng, dễ bảo trì và mở rộng.

2.2.3 Luồng xử lý dữ liệu trong hệ thống MERN

Khi người dùng thực hiện thao tác trên giao diện React, yêu cầu được gửi đến server thông qua các RESTful API. Express.js tiếp nhận và xử lý yêu cầu, sau đó tương tác với MongoDB để truy xuất hoặc cập nhật dữ liệu. Kết quả xử lý được trả về client dưới dạng JSON và React cập nhật lại giao diện người dùng tương ứng. Luồng xử lý này đảm bảo tính nhất quán và hiệu quả trong toàn bộ hệ thống.



Hình 2.1 Sơ đồ kiến trúc tổng thể hệ thống

Hệ thống được xây dựng theo mô hình Client–Server. Phía Client sử dụng React.js kết hợp TailwindCSS để xây dựng giao diện người dùng. Phía Server sử dụng Node.js và Express để cung cấp các RESTful API, xử lý nghiệp vụ thông qua Controllers và Middleware. Dữ liệu được lưu trữ trong cơ sở dữ liệu MongoDB dưới dạng các collection và document JSON.

2.2.4 Đánh giá kiến trúc MERN Stack

Kiến trúc MERN Stack mang lại nhiều ưu điểm như sử dụng một ngôn ngữ duy nhất cho cả frontend và backend, tốc độ phát triển nhanh, hiệu năng cao và khả năng mở rộng tốt. Tuy nhiên, việc quản lý bất đồng bộ và trạng thái phía client đòi hỏi người phát triển phải có kiến thức vững về JavaScript và kiến trúc web. Nhìn chung, MERN Stack là lựa chọn phù hợp cho các ứng dụng web hiện đại, đặc biệt là các hệ thống SPA.

2.3. Công nghệ và framework front-end

2.3.1. React.js

2.3.1.1. Giới thiệu React.js (hay React)

React.js (hay React) là một thư viện mã nguồn JavaScript được phát triển và duy trì bởi Facebook (nay là Meta Platforms, Inc.) cùng với một cộng đồng cho các nhà phát triển cá nhân và doanh nghiệp trên toàn thế giới. Được giới thiệu lần đầu vào tháng 5 năm 2013 tại hội nghị CTonf US, React nhanh chóng trở thành một trong những công nghệ phổ biến nhất để xây dựng giao diện người dùng (Giao diện người dùng - UI), đặc biệt là cho các ứng dụng đơn trang (Single Page Application - SPA).

React ra đời nhằm giải quyết các vấn đề về hiệu quả và khả năng bảo trì của các ứng dụng web có giao diện phức tạp và cần cập nhật liên tục dữ liệu. Tính đến năm 2024, React đã được sử dụng bởi hơn 10 triệu trang web trên toàn cầu, bao gồm nhiều ứng dụng như Facebook, Instagram, Netflix, Airbnb và Discord.

*2.3.1.2. Khái niệm React.js*

React không phải là một framework toàn diện mà là một thư viện tập trung vào tầng "View" (Giao diện) trong mô hình MVC (Model-View-Controller). Khái niệm cốt lõi của React là chia nhỏ giao diện người dùng thành các phần nhỏ, độc lập và có thể tái sử dụng được gọi là các "Component" (Thành phần). Các khái niệm quan trọng bao gồm:

Component: Là cơ sở xây dựng khối, có thể là một nút ấn, nhập liệu hoặc hoàn thành một trang.

JSX (JavaScript XML): Cú pháp cho phép viết HTML trong JavaScript:

|  |
| --- |
| // Ví dụ JSX  const Header = () => {  return (  <div className="header">  <h1>Website Sống Xanh</h1>  </div>  );  }; |

Props: Data tryền từ thành phần cha xuống thành phần con, không thể thay đổi:

|  |
| --- |
| <PostCard title="Tiết kiệm điện" author="Admin" /> |

Trạng thái: Nội dung dữ liệu của thành phần, có thể thay đổi và kích hoạt lại giao diện kết xuất:

|  |
| --- |
| const [likes, setLikes] = useState(0); |

*2.3.1.3 Cấu trúc*

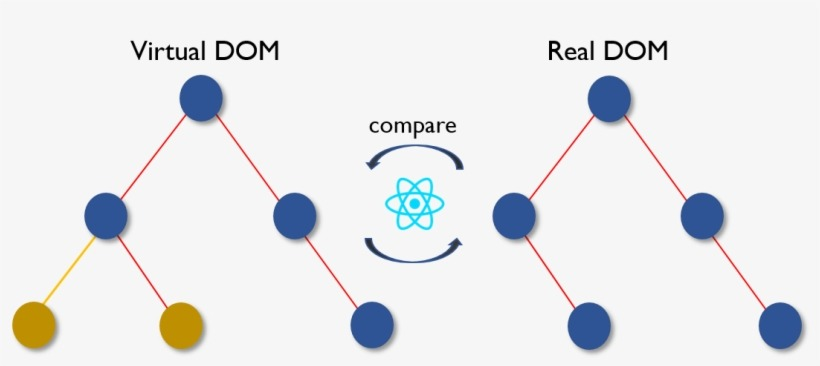
Cấu trúc của một ứng dụng React được xây dựng dựa trên cây phân cấp Component. Mỗi Component quản lý trạng thái riêng của nó (State) và nhận dữ liệu từ Component cha thông qua các thuộc tính (Props). React sử dụng cú pháp JSX (JavaScript XML), cho phép viết mã HTML pha trộn với logic JavaScript, giúp mã nguồn trở nên trực quan và dễ hiểu hơn. Trong đồ án, cấu trúc thư mục được thiết kế như sau:

|  |
| --- |
| client/  └── src/  ├── components/  │ └── Footer.jsx  │  ├── pages/  │ ├── Admin.jsx  │ ├── Articles.jsx  │ ├── CreatePost.jsx  │ ├── EditPost.jsx  │ ├── ForgotPassword.jsx  │ ├── Forum.jsx  │ ├── Home.jsx  │ ├── Login.jsx  │ ├── News.jsx  │ ├── PostDetail.jsx  │ ├── Profile.jsx  │ └── Register.jsx  │  ├── apiConfig.js  ├── App.css  ├── App.jsx  ├── index.css  └── main.jsx |

Mỗi component có thể chứa các component con, tạo thành cây phân cấp giúp quản lý dễ dàng.

*2.3.1.4 Cách thức hoạt động*

React hoạt động dựa trên cơ chế Virtual DOM (DOM ảo). Thay vì thao tác trực tiếp lên DOM thật của trình duyệt (vốn có tốc độ chậm), React tạo ra một bản sao của DOM trong bộ nhớ. Khi trạng thái (State) của ứng dụng thay đổi, React sẽ so sánh (diffing) phiên bản Virtual DOM mới với phiên bản cũ, tính toán những điểm khác biệt tối thiểu và chỉ cập nhật những phần cần thiết lên DOM thật.



Ví dụ, khi người dùng bình luận vào khung comment vào bất kỳ bài Blog nào trên website và nhấn “Enter”. Dĩ nhiên, người dùng của bạn sẽ cần phải thấy được bình luận của mình đã được thêm vào danh sách bình luận. Giả sử trong trường hợp không sử dụng ReactJS, toàn bộ cây DOM sẽ phải cập nhật để báo hiệu sự thay đổi mới này. Còn khi sử dụng React, nó sẽ giúp bạn scan qua Virtual DOM để xem những gì đã thay đổi sau khi người dùng thực hiện hành động trên (trong trường hợp này, thêm mới bình luận) và lựa chọn đúng nơi đúng chỗ cần cập nhật sự thay đổi mà thôi.

*2.3.1.5. Vai trò của React.js*

Trong đồ án này, React.js đóng vai trò xây dựng toàn bộ giao diện phía Client (Frontend) nằm trong thư mục client/. Nó chịu trách nhiệm hiển thị dữ liệu cho người dùng, xử lý các tương tác (như click, nhập liệu) và giao tiếp với Server thông qua API để lấy hoặc gửi dữ liệu mà không cần tải lại toàn bộ trang.

Hiển thị nội dung: Danh sách bài viết, diễn đàn, chỉ số AQI,...

Xử lý tương tác: Đăng nhập, đăng bài, bình luận, like

Giao tiếp với Backend: Gọi API để lấy/gửi dữ liệu

Quản lý trạng thái: Lưu thông tin người dùng, danh sách bài viết

*2.3.1.5. Ưu và nhược điểm của React.js*

A. Ưu điểm

Hiệu năng cao: Nhờ cơ chế Virtual DOM, React giảm thiểu các thao tác vẽ lại (repaint) không cần thiết trên trình duyệt.

Khả năng tái sử dụng: Các Component (Header, Footer, PostCard) có thể được tái sử dụng ở nhiều nơi, giúp giảm dư thừa mã nguồn và dễ bảo trì.

Hệ sinh thái mạnh mẽ: Cộng đồng lớn cung cấp nhiều thư viện hỗ trợ như Redux, React Router, Axios.

1. Nhược điểm:

Chỉ là View Library: React chỉ lo phần giao diện, lập trình viên cần tự tìm kiếm các thư viện khác để xử lý định tuyến (Routing) hay quản lý trạng thái (State Management).

SEO (Search Engine Optimization) không tối ưu: Mặc định React render phía client (CSR), điều này có thể gây khó khăn cho các công cụ tìm kiếm trong việc thu thập dữ liệu nội dung so với web tĩnh truyền thống (tuy nhiên có thể khắc phục bằng Next.js hoặc Server-Side Rendering).

Bảng 2.1: So sánh React với các framework tương tự

|  |  |  |  |
| --- | --- | --- | --- |
| Tiêu chí | React | Angular | Vue.js |
| Độ khó học | Trung bình | Khó | Dễ |
| Kích thước | 40KB | 500KB+ | 20KB |
| Hiệu năng | Cao | Trung bình | Cao |
| Cộng đồng | Rất lớn | Lớn | Lớn |

*2.3.1.6. Các ứng dụng của React.js*

React được sử dụng rộng rãi để xây dựng các nền tảng mạng xã hội (Facebook, Instagram), các trang thương mại điện tử, bảng điều khiển quản trị (Dashboard) và các ứng dụng giải trí trực tuyến (Netflix), nơi yêu cầu tính tương tác cao và cập nhật dữ liệu liên tục.

2.3.2. Vite.js

2.3.2.1. Giới thiệu Vite.js

Vite là công cụ build tool hiện đại được Evan You (tác giả Vue.js) phát triển từ năm 2020. Vite được thiết kế để giải quyết vấn đề tốc độ khởi động chậm của các build tool truyền thống như Webpack, Create React App.

Trong đồ án này, Vite được sử dụng để khởi tạo và phát triển ứng dụng React với tốc độ build nhanh hơn đáng kể so với các công cụ cũ.

2.3.2.2. Khái niệm và cách hoạt động

Vite hoạt động dựa trên hai nguyên lý chính:

A. Native ES Modules trong development

- Không bundle toàn bộ code khi khởi động dev server

- Chỉ biên dịch file đang được request

- Khởi động server gần như tức thì (dưới 1 giây)

B. Pre-bundling với esbuild:

- esbuild được viết bằng Go, nhanh hơn JavaScript 10-100 lần

- Pre-bundle các dependencies từ node\_modules

- Hot Module Replacement (HMR) cực nhanh

2.3.3. Công nghệ TailwindCSS

2.3.3.1. Giới thiệu TailwindCSS

TailwindCSS là một framework CSS ưu tiên tiện ích (utility-first) đang ngày càng phổ biến trong thiết kế web hiện đại. Khác với Bootstrap hay Foundation cung cấp các thành phần dựng sẵn (như Button, Navbar), Tailwind cung cấp các lớp (class) cấp thấp để xây dựng giao diện tùy chỉnh.

2.3.3.2. Khái niệm

Khái niệm cốt lõi của TailwindCSS là Utility-First. Thay vì viết CSS trong một file riêng biệt (.css) với các tên class ngữ nghĩa (như .btn-primary), lập trình viên sử dụng trực tiếp các class tiện ích có sẵn (như bg-blue-500, text-white, p-4) ngay trong thẻ HTML để định dạng phần tử.

2.3.3.3 Cấu trúc

Tailwind được cấu hình thông qua tệp tailwind.config.js, cho phép định nghĩa các biến màu sắc, phông chữ, khoảng cách... để đảm bảo tính nhất quán (Consistency) cho toàn bộ dự án.

2.3.3.4 Cách thức hoạt động

Khi biên dịch, Tailwind sẽ quét toàn bộ mã nguồn HTML/JS để tìm các class được sử dụng và chỉ tạo ra các đoạn mã CSS tương ứng. Các class không dùng đến sẽ bị loại bỏ (PurgeCSS), giúp file CSS cuối cùng cực kỳ nhẹ.

2.3.3.5. Vai trò của TailwindCSS

Trong đồ án, TailwindCSS chịu trách nhiệm toàn bộ phần trình bày giao diện (Styling) cho Frontend. Nó giúp xây dựng giao diện đáp ứng (Responsive) nhanh chóng, đảm bảo website hiển thị tốt trên cả máy tính và thiết bị di động.

2.3.3.6. Ưu và nhược điểm của TailwindCSS

A. Ưu điểm

- Phát triển nhanh: Không cần chuyển đổi qua lại giữa file HTML và CSS.

- Tùy biến cao: Không bị rập khuôn theo thiết kế mặc định như Bootstrap.

- Tối ưu hiệu năng: File CSS đầu ra có dung lượng nhỏ.

B. Nhược điểm

- HTML trở nên dài dòng: Việc gắn quá nhiều class vào thẻ HTML có thể làm mã nguồn trông rối mắt và khó đọc đối với người mới.

- Cần thời gian làm quen: Lập trình viên cần nhớ tên các class tiện ích của Tailwind.

*2.3.3.7. Các ứng dụng của TailwindCSS*

TailwindCSS phù hợp cho hầu hết các dự án web hiện đại, từ các Landing Page đơn giản đến các hệ thống Design System phức tạp của doanh nghiệp, đặc biệt hiệu quả khi kết hợp với các Component-based Framework như React.js hay Vue.js.

## 2.4 Công nghệ và framework back-end

2.4.1. Node.js

2.4.1.1. Giới thiệu Node.js

Node.js là một môi trường thực thi (Runtime Environment) mã JavaScript đa nền tảng, mã nguồn mở, được xây dựng trên nền tảng V8 JavaScript Engine của Google Chrome. Node.js cho phép các nhà phát triển sử dụng JavaScript để viết mã cho phía máy chủ (Server-side), mở ra kỷ nguyên "JavaScript Everywhere" (JavaScript ở mọi nơi).

2.4.1.2. Khái niệm

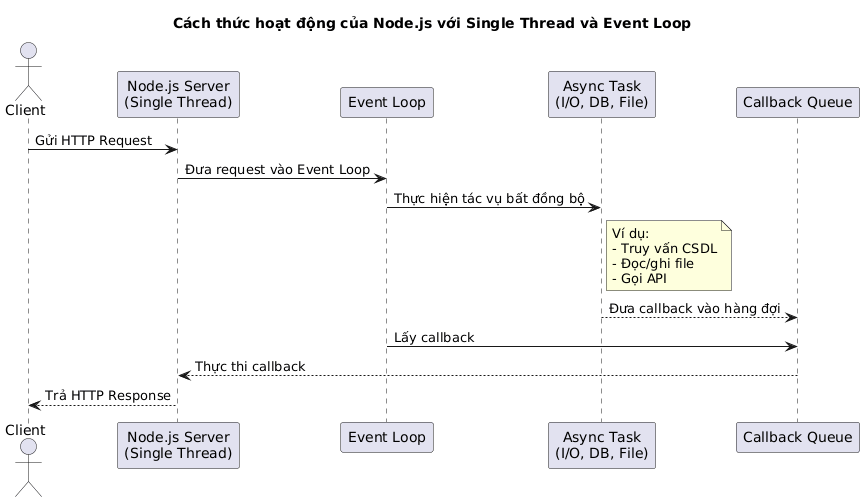
Khác với các nền tảng máy chủ truyền thống dựa trên luồng (thread-based) như PHP hay Java, Node.js hoạt động dựa trên mô hình hướng sự kiện (Event-driven) và cơ chế I/O bất đồng bộ, không chặn (Non-blocking I/O).

2.4.1.3. Cấu trúc:

Node.js được thiết kế với kiến trúc mô-đun (Module). Ứng dụng Node.js thường bao gồm các module cốt lõi (http, fs, path) và các thư viện từ bên thứ ba được quản lý thông qua NPM (Node Package Manager). Trong đồ án, cấu trúc backend được chia thành các thư mục routes, controllers, models để quản lý logic.

2.4.1.4 Cách thức hoạt động

Node.js hoạt động trên một luồng đơn (Single Thread) kết hợp với Event Loop (Vòng lặp sự kiện). Khi có yêu cầu (request) đến, thay vì tạo ra một luồng mới cho mỗi yêu cầu (gây tốn tài nguyên), Node.js tiếp nhận và xử lý chúng trong nền. Khi tác vụ hoàn thành, nó sẽ kích hoạt một sự kiện (callback) để trả kết quả về, cho phép server xử lý hàng ngàn kết nối đồng thời.



Hình 2.2. Cách thức hoạt động của Node.js

Node.js hoạt động trên một luồng đơn (Single Thread) kết hợp với cơ chế Event Loop. Khi có yêu cầu từ client, Node.js tiếp nhận và đưa vào vòng lặp sự kiện để xử lý. Các tác vụ tốn thời gian như truy vấn cơ sở dữ liệu hoặc I/O được xử lý bất đồng bộ. Khi tác vụ hoàn thành, callback được đưa vào hàng đợi và thực thi để trả kết quả về cho client. Cơ chế này cho phép Node.js xử lý hàng ngàn kết nối đồng thời với hiệu năng cao.

2.4.1.5. Vai trò của Node.js

Trong đồ án, Node.js kết hợp với framework Express.js đóng vai trò là Web Server và API Server. Nó tiếp nhận các yêu cầu HTTP từ Frontend (React), thực hiện các logic nghiệp vụ (đăng nhập, đăng bài, bình luận), truy vấn cơ sở dữ liệu MongoDB và trả về kết quả dưới dạng JSON.

2.4.1.6. Ưu và nhược điểm của Node.js

A. Ưu điểm:

Tốc độ xử lý nhanh: Nhờ động cơ V8 và cơ chế non-blocking I/O, Node.js xử lý rất nhanh các tác vụ vào/ra dữ liệu (I/O bound).

Khả năng mở rộng: Dễ dàng mở rộng theo chiều ngang và xử lý lượng lớn kết nối đồng thời.

Cộng đồng NPM: Kho thư viện khổng lồ giúp giải quyết hầu hết các vấn đề lập trình.

B. Nhược điểm:

Xử lý tính toán kém: Do chạy đơn luồng, Node.js không phù hợp với các tác vụ yêu cầu tính toán CPU nặng (CPU bound) như xử lý video hay hình ảnh phức tạp, vì có thể gây nghẽn Event Loop.

Callback Hell: Nếu không quản lý tốt mã nguồn, việc lồng ghép quá nhiều hàm callback có thể làm mã trở nên khó đọc (tuy nhiên đã được khắc phục bằng Async/Await).

2.4.1.7. Các ứng dụng của Node.js

Node.js là lựa chọn hàng đầu cho các ứng dụng thời gian thực (Chat, Streaming), ứng dụng trang đơn (SPA), API Server (RESTful/GraphQL) và các ứng dụng Internet of Things (IoT).

2.4.2. Công nghệ Express.js

2.4.2.1. Giới thiệu Express.js (hay gọi tắt là Express)

Là một khung ứng dụng web (Web Application Framework) mã nguồn mở, tối giản và linh hoạt dành cho môi trường Node.js. Được phát triển bởi TJ Holowaychuk và ra mắt lần đầu vào năm 2010, Express hiện là framework phổ biến nhất trong hệ sinh thái Node.js và là thành phần chữ "E" quan trọng trong bộ công nghệ MERN Stack (MongoDB, Express, React, Node).

2.4.2.2. Khái niệm

Khái niệm cốt lõi của Express.js là Middleware (Phần mềm trung gian) và Routing (Định tuyến). Express không áp đặt một cấu trúc thư mục hay mô hình thiết kế cứng nhắc nào, mà cung cấp một lớp mỏng các tính năng cơ bản giúp quản lý máy chủ và luồng dữ liệu HTTP một cách hiệu quả. Nó hoạt động như một lớp trung gian giúp đơn giản hóa việc giao tiếp giữa Client và Node.js Server.

2.4.2.3. Tại sao Developer lại sử dụng Express.js

Trong khi Node.js cung cấp các module cốt lõi như http để tạo máy chủ, việc xây dựng một ứng dụng web phức tạp chỉ bằng Node.js thuần (Vanilla Node.js) đòi hỏi rất nhiều mã nguồn lặp lại để xử lý các tác vụ như phân tích cú pháp URL, quản lý session hay xử lý dữ liệu POST. Express.js giải quyết vấn đề này bằng cách cung cấp các hàm tiện ích mạnh mẽ, giúp rút ngắn thời gian phát triển và tối ưu hóa mã nguồn.

2.4.2.3.1. Cấu trúc

Cấu trúc của một ứng dụng Express xoay quanh đối tượng app và các Router. Trong đồ án, cấu trúc này được thể hiện rõ qua việc phân chia các tuyến đường API vào thư mục routes/. Mỗi tệp route (như authRoutes.js, postRoutes.js) sử dụng express.Router() để định nghĩa các điểm cuối (endpoints) riêng biệt, giúp mã nguồn được tổ chức khoa học và dễ bảo trì.

2.4.2.3.2. Cách thức hoạt động

Express hoạt động dựa trên mô hình Pipeline (Đường ống) xử lý Request-Response. Khi một yêu cầu HTTP gửi đến, nó sẽ đi qua một chuỗi các hàm Middleware. Mỗi Middleware có thể thực hiện các thao tác như ghi log, kiểm tra xác thực, hoặc phân tích dữ liệu, sau đó chuyển quyền kiểm soát cho Middleware tiếp theo bằng hàm next(). Ví dụ thực tế trong đồ án: Khi người dùng gọi API đăng bài (POST /api/posts), request sẽ đi qua middleware xác thực verifyToken để kiểm tra danh tính, sau đó qua middleware upload để xử lý file ảnh và cuối cùng mới đến Controller để lưu vào cơ sở dữ liệu.

2.4.2.4. Vai trò của

Express.js trong đồ án Express.js đóng vai trò là xương sống của hệ thống Backend, chịu trách nhiệm:

Xây dựng RESTful API: Định nghĩa các phương thức HTTP (GET, POST, PUT, DELETE) để Frontend (React) có thể tương tác với dữ liệu. Ví dụ: API router.post('/login', ...) xử lý đăng nhập, API router.get('/posts', ...) lấy danh sách bài viết.

Quản lý Middleware: Tích hợp các thư viện xử lý trung gian quan trọng như cors (cho phép truy cập chéo miền), body-parser (đọc dữ liệu JSON) và multer (quản lý tải lên tập tin).

Xử lý lỗi tập trung: Cung cấp cơ chế bắt lỗi (Error Handling) để đảm bảo server không bị dừng hoạt động khi gặp ngoại lệ trong quá trình xử lý yêu cầu.

2.4.2.5. Ưu và nhược điểm của Express.js

A. Ưu điểm:

Tối giản và Linh hoạt: Không áp đặt quy tắc, cho phép lập trình viên tự do thiết kế kiến trúc hệ thống theo nhu cầu cụ thể (như việc áp dụng mô hình MVC trong đồ án này).

Hiệu suất cao: Là một lớp mỏng trên nền Node.js, Express không làm giảm đáng kể tốc độ xử lý vốn có của Node.js.

Hỗ trợ Middleware mạnh mẽ: Dễ dàng tích hợp hàng ngàn middleware có sẵn từ cộng đồng NPM để mở rộng tính năng.

B. Nhược điểm:

Thiếu cấu trúc chuẩn: Sự tự do quá mức đôi khi dẫn đến việc mã nguồn thiếu nhất quán nếu không được quản lý tốt (Spaghetti code).

Phải cấu hình thủ công: Khác với các framework "pin đi kèm" (batteries-included) như NestJS hay Django, lập trình viên Express phải tự cài đặt và cấu hình từng thư viện (ORM, Validation, Security) một cách thủ công.

2.4.2.6. Các ứng dụng của Express.js

Express.js là lựa chọn tiêu chuẩn cho hầu hết các ứng dụng Node.js hiện nay, từ các ứng dụng trang đơn (SPA Backend), ứng dụng render phía máy chủ (SSR), đến các hệ thống Microservices phức tạp và các nền tảng API công cộng.

## 2.5 Hệ quản trị cơ sở dữ liệu

**2.5.1. Cơ sở dữ liệu MongoDB**

*2.5.1.1. Giới thiệu*

MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở đa nền tảng, được phát triển bởi MongoDB Inc. và giới thiệu lần đầu vào năm 2009. Khác với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) truyền thống như MySQL hay SQL Server sử dụng ngôn ngữ SQL, MongoDB thuộc nhóm cơ sở dữ liệu NoSQL (Not Only SQL), được thiết kế để xử lý khối lượng dữ liệu lớn, phân tán và có cấu trúc linh hoạt.

*2.5.1.2. Khái niệm*

Khái niệm cốt lõi của MongoDB là Document-oriented (Hướng tài liệu). Thay vì tổ chức dữ liệu dưới dạng các bảng (Table) và hàng (Row) cứng nhắc, MongoDB lưu trữ dữ liệu dưới dạng các Document (Tài liệu). Các Document này sử dụng định dạng JSON (JavaScript Object Notation) hoặc chính xác hơn là BSON (Binary JSON) để lưu trữ thông tin. Tập hợp các Document có cùng chủ đề được gọi là một Collection (Bộ sưu tập).

Document: Đơn vị lưu trữ cơ bản, tương tự như một object JSON:

|  |
| --- |
| {  "\_id": "1",  "username": "admin01",  "password": "$2b$10$hashedadmin",  "fullName": "Quản trị viên",  "email": "admin@gmail.com",  "role": "admin",  "isBanned": false  } |

Collection: Tập hợp các Document, tương tự như Table trong SQL

2.5.1.4. Cách thức hoạt động:

MongoDB hoạt động dựa trên cơ chế lưu trữ và truy xuất dữ liệu dạng Key-Value. Khi ứng dụng Node.js gửi yêu cầu, MongoDB sẽ tìm kiếm trong các Collection tương ứng. Các mối quan hệ giữa các dữ liệu không được liên kết bằng Khóa ngoại (Foreign Key) phức tạp như SQL, mà thường được tham chiếu thông qua ObjectId. Ví dụ, trong Collection Comment, trường user và post lưu trữ ObjectId tham chiếu đến User và Post tương ứng, giúp việc truy xuất dữ liệu liên quan trở nên nhanh chóng thông qua thao tác populate.

2.5.1.5. Vai trò của MongoDB

Trong hệ thống website này, MongoDB đóng vai trò là tầng Persistence (Lưu trữ bền vững). Nhiệm vụ cụ thể bao gồm:

- Lưu trữ dữ liệu người dùng: Quản lý thông tin đăng nhập, phân quyền (Admin/User) và trạng thái tài khoản.

- Quản lý nội dung: Lưu trữ hàng nghìn bài viết tin tức, bài thảo luận diễn đàn và các bình luận phân cấp.

- Lưu trữ danh mục và chủ đề: Quản lý các phân loại động như Category cho tin tức và Topic cho diễn đàn.

- Xử lý tương tác: Ghi nhận các hành động thời gian thực như lượt thích (likes), lượt xem (views) và báo cáo vi phạm (reports).

2.5.1.6. Ưu và nhược điểm của MongoDB

A. Ưu điểm

Linh hoạt (Schema-less): Cho phép thay đổi cấu trúc dữ liệu (thêm/bớt trường) mà không cần phải thay đổi toàn bộ cơ sở dữ liệu hay downtime hệ thống, rất phù hợp cho các dự án phát triển theo mô hình Agile.

Hiệu năng cao: Tốc độ đọc/ghi dữ liệu rất nhanh do cấu trúc đơn giản và khả năng tối ưu hóa bộ nhớ đệm (In-memory computing).

Khả năng mở rộng (Scalability): Hỗ trợ mạnh mẽ việc mở rộng theo chiều ngang (Sharding) để xử lý Big Data.

B.Nhược điểm

Không hỗ trợ Joins mạnh mẽ: MongoDB không hỗ trợ các câu lệnh JOIN phức tạp như SQL. Việc kết nối dữ liệu thường phải xử lý ở tầng ứng dụng hoặc dùng $lookup, điều này có thể gây khó khăn khi cần tổng hợp báo cáo phức tạp.

Tính nhất quán dữ liệu (Consistency): Để đổi lấy tốc độ và tính sẵn sàng (Availability), MongoDB áp dụng mô hình "Eventual Consistency" (Nhất quán cuối cùng), đôi khi dữ liệu chưa được đồng bộ tức thì trên toàn bộ hệ thống phân tán.

*2.5.1.7. Các ứng dụng của MongoDB*

MongoDB hiện được ứng dụng rộng rãi trong các hệ thống Big Data, Hệ thống quản trị nội dung (CMS), Mạng xã hội, các ứng dụng Internet of Things (IoT) và các nền tảng thương mại điện tử cần xử lý danh mục sản phẩm đa dạng, phức tạp.

2.5. Mô hình kiến trúc phần mềm MVC

2.5.1. Giới thiệu Mô hình MVC (Model - View - Controller)

Là một mẫu kiến trúc phần mềm kinh điển, được sử dụng để tách biệt phần logic xử lý dữ liệu, giao diện hiển thị và luồng điều khiển của ứng dụng.

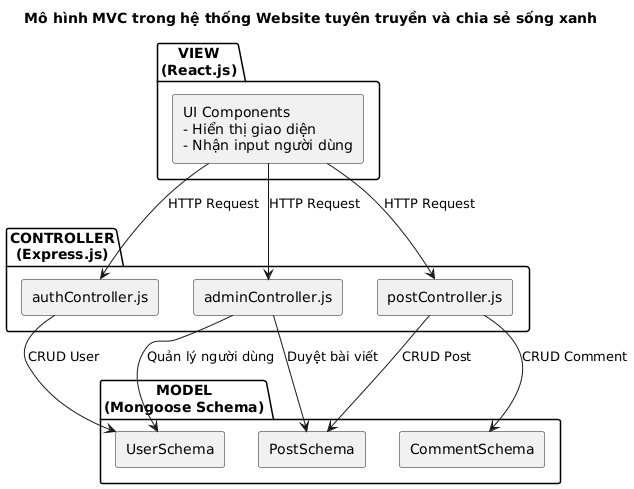
2.5.2. Ứng dụng trong đồ án

Mặc dù đồ án được xây dựng theo kiến trúc Client-Server tách biệt (React.js riêng và Node.js riêng), nhưng cấu trúc bên trong của Backend (Node.js) vẫn tuân thủ chặt chẽ mô hình MVC để đảm bảo tính tổ chức và dễ bảo trì.

Model (Mô hình dữ liệu): Tương ứng với các Schema được định nghĩa bằng thư viện Mongoose trong thư mục models/. Ví dụ, PostSchema quy định cấu trúc của một bài viết, đảm bảo tính toàn vẹn của dữ liệu trước khi lưu vào MongoDB.

View (Giao diện): Trong ngữ cảnh API Backend, "View" chính là dữ liệu JSON được trả về cho Client. Phía React.js sẽ đóng vai trò là tầng hiển thị cuối cùng, nhận dữ liệu này và render ra màn hình.

Controller (Bộ điều khiển): Được đặt trong thư mục controllers/. Đây là nơi chứa logic nghiệp vụ chính. Ví dụ: postController.js tiếp nhận yêu cầu đăng bài từ Router, gọi đến Model để lưu dữ liệu, sau đó trả về kết quả JSON cho người dùng.



Hình 2.3 Mô hình MVC của hệ thống

Hệ thống được thiết kế theo mô hình MVC (Model–View–Controller). View được xây dựng bằng React.js để hiển thị giao diện và tiếp nhận dữ liệu từ người dùng. Controller sử dụng Express.js để xử lý các yêu cầu HTTP và điều phối nghiệp vụ. Model sử dụng Mongoose Schema để định nghĩa cấu trúc dữ liệu và thao tác với cơ sở dữ liệu MongoDB.

2.6. Tiêu chuẩn giao tiếp RESTful API

2.6.1. Khái niệm REST (Representational State Transfer)

Là một kiểu kiến trúc phần mềm quy định các ràng buộc khi thiết kế Web Service. Một hệ thống tuân thủ các nguyên tắc này được gọi là RESTful API.

2.6.2. Cách thức hoạt động trong hệ thống

Hệ thống sử dụng RESTful API làm phương thức giao tiếp duy nhất giữa Frontend (React.js) và Backend (Node.js). Các tài nguyên (Resources) được định danh qua URL và thao tác thông qua các phương thức HTTP chuẩn:

GET: Sử dụng để lấy dữ liệu (Ví dụ: GET /api/posts lấy danh sách bài viết).

POST: Sử dụng để tạo mới dữ liệu (Ví dụ: POST /api/auth/register để đăng ký tài khoản).

PUT: Sử dụng để cập nhật dữ liệu (Ví dụ: PUT /api/users/profile để cập nhật hồ sơ).

DELETE: Sử dụng để xóa dữ liệu (Ví dụ: DELETE /api/posts/:id để xóa bài viết).

2.7. Cơ chế bảo mật và xác thực (Security & Authentication)

Để đảm bảo an toàn cho hệ thống mạng xã hội nơi có sự tương tác giữa nhiều người dùng, đồ án áp dụng hai kỹ thuật bảo mật quan trọng: Mã hóa mật khẩu và Xác thực không trạng thái.

2.7.1. Mã hóa mật khẩu với Bcrypt

Việc lưu trữ mật khẩu dưới dạng văn bản thuần (plain text) là một lỗ hổng bảo mật nghiêm trọng. Đồ án sử dụng thư viện bcryptjs để băm (hash) mật khẩu trước khi lưu vào cơ sở dữ liệu MongoDB.

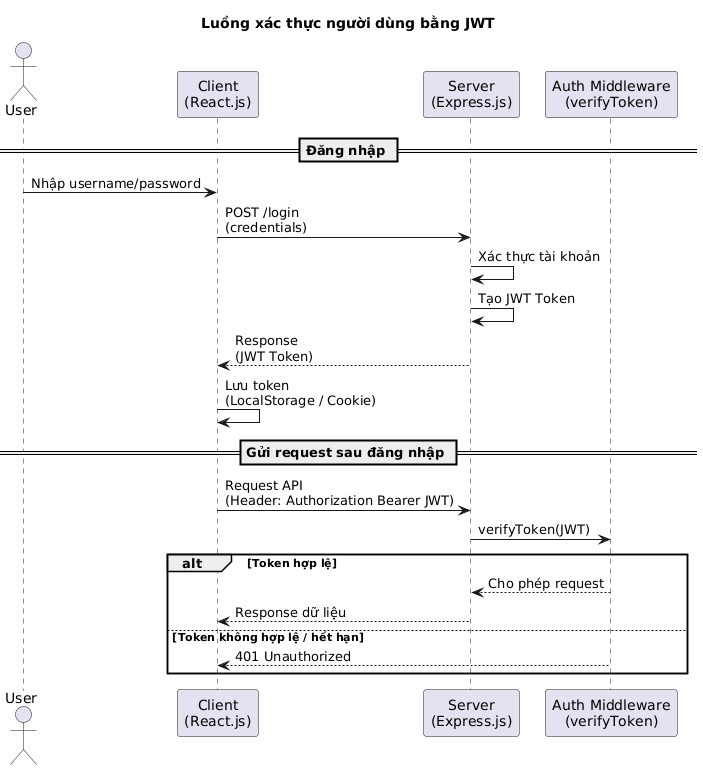
Cơ chế: Khi người dùng đăng ký, mật khẩu sẽ được cộng thêm một chuỗi ngẫu nhiên (salt) và băm nhiều lần. Khi đăng nhập, hệ thống sẽ băm mật khẩu nhập vào và so sánh với chuỗi băm trong cơ sở dữ liệu. Điều này đảm bảo rằng ngay cả khi quản trị viên truy cập vào database cũng không thể đọc được mật khẩu của người dùng.

2.7.2. Xác thực JWT (JSON Web Token)

Hệ thống sử dụng JWT để quản lý phiên làm việc theo cơ chế Stateless (Không lưu trạng thái session trên server).

Cấu trúc: Một JWT bao gồm 3 phần: Header, Payload (chứa thông tin User ID, Role) và Signature.

Quy trình: Khi đăng nhập thành công, Server tạo ra một token ký bằng SECRET\_KEY và gửi về Client. Trong các yêu cầu tiếp theo (như đăng bài, bình luận), Client gửi kèm token này trong Header. Middleware verifyToken sẽ giải mã token để xác định danh tính và quyền hạn (req.userRole) của người dùng.



Hình 2.4. Sơ đồ trình tự xác thực người dùng bằng JWT

Sơ đồ mô tả luồng xác thực người dùng bằng JWT. Sau khi người dùng đăng nhập thành công, Server tạo và trả về một JWT Token cho Client. Token được lưu trữ phía Client và được gửi kèm trong Header của các request tiếp theo. Middleware trên Server chịu trách nhiệm kiểm tra tính hợp lệ của token để quyết định cho phép hoặc từ chối request.

2.8. Công cụ hỗ trợ phát triển và triển khai

2.8.1. Postman

Postman là công cụ hỗ trợ kiểm thử và tương tác với các API thông qua giao thức HTTP/HTTPS. Trong hệ thống, Postman được sử dụng để kiểm tra các RESTful API do server Node.js và Express.js cung cấp, bao gồm các chức năng như đăng ký, đăng nhập, quản lý bài viết, bình luận và xác thực người dùng bằng JWT.

Postman cho phép gửi các yêu cầu với nhiều phương thức khác nhau như GET, POST, PUT và DELETE, đồng thời hỗ trợ truyền dữ liệu dưới dạng JSON và thêm các HTTP header cần thiết. Nhờ đó, quá trình phát triển và kiểm thử API được thực hiện độc lập với giao diện người dùng, giúp phát hiện lỗi sớm và đảm bảo API hoạt động đúng theo thiết kế.

2.8.2. GitHub và GitHub Actions

GitHub là nền tảng quản lý mã nguồn được sử dụng để lưu trữ, theo dõi và quản lý các phiên bản của hệ thống. Việc sử dụng GitHub giúp nhóm phát triển dễ dàng quản lý lịch sử thay đổi mã nguồn, phối hợp làm việc và khôi phục phiên bản khi cần thiết.

Bên cạnh đó, GitHub Actions được sử dụng để tự động hóa quy trình tích hợp và triển khai liên tục (CI/CD). Thông qua các workflow được cấu hình sẵn, hệ thống có thể tự động thực hiện các bước như kiểm tra mã nguồn, build ứng dụng và triển khai lên môi trường server mỗi khi có thay đổi được đẩy lên repository. Điều này giúp giảm thiểu sai sót do thao tác thủ công và nâng cao độ tin cậy của quá trình triển khai.

2.8.3. Docker

Docker là nền tảng container hóa được sử dụng để đóng gói ứng dụng cùng với các thư viện và cấu hình cần thiết vào các container độc lập. Trong hệ thống, Docker được áp dụng để triển khai cả backend (Node.js + Express) và frontend (React.js), giúp đảm bảo môi trường chạy ứng dụng nhất quán giữa các máy phát triển và môi trường triển khai thực tế.

Việc sử dụng Docker giúp hệ thống dễ dàng mở rộng, triển khai nhanh chóng và giảm thiểu các lỗi phát sinh do khác biệt môi trường. Ngoài ra, Docker còn hỗ trợ quản lý các service thông qua Docker Compose, giúp đơn giản hóa việc khởi chạy và vận hành toàn bộ hệ thống.

2.9. Cơ sở lý thuyết về môi trường và hành vi người dùng

Để website đạt được mục tiêu "Tuyên truyền và chia sẻ mẹo sống xanh", nghiên cứu cần dựa trên các lý thuyết về khoa học môi trường và tâm lý học hành vi.

2.9.1. Chỉ số chất lượng không khí (AQI)

AQI (Air Quality Index) là thước đo định lượng mức độ ô nhiễm không khí. Việc tích hợp hiển thị AQI thời gian thực trên website dựa trên cơ sở khoa học rằng: sự nhận thức về mức độ nguy hại của môi trường xung quanh (như nồng độ bụi mịn PM2.5) là động lực trực tiếp thúc đẩy con người thực hiện các hành động bảo vệ môi trường.

2.9.2. Lý thuyết thay đổi hành vi qua tương tác xã hội

Dựa trên Lý thuyết học tập xã hội (Social Learning Theory), hành vi sống xanh có thể được lan truyền thông qua sự quan sát và tương tác.

Cơ chế: Khi người dùng thấy các thành viên khác chia sẻ mẹo tái chế hoặc tham gia "Diễn đàn sống xanh", họ có xu hướng bắt chước và thực hiện theo.

Ứng dụng: Tính năng Diễn đàn (Forum) và Bình luận (Comment) được thiết kế để tối ưu hóa hiệu ứng này, tạo ra một cộng đồng hỗ trợ lẫn nhau thay vì chỉ cung cấp thông tin một chiều.

2.10. Lý luận khoa học của đề tài

Nghiên cứu được xây dựng dựa trên sự kết hợp giữa Sức mạnh công nghệ (MERN Stack) và Tâm lý học cộng đồng.

Lý luận cốt lõi là: Thông tin đơn thuần không đủ để thay đổi hành vi. Các trang web truyền thống thường chỉ cung cấp tin tức khô khan. Để thực sự thúc đẩy lối sống xanh, cần một môi trường có tính tương tác cao (Social Engagement).

Công nghệ React.js và Node.js đảm bảo trải nghiệm người dùng mượt mà, tốc độ phản hồi nhanh, giữ chân người dùng ở lại trang lâu hơn.

Cấu trúc dữ liệu linh hoạt của MongoDB cho phép lưu trữ đa dạng các loại nội dung từ bài viết chuyên sâu đến các thảo luận ngắn, phục vụ nhiều đối tượng độc giả khác nhau.

Cơ chế phân quyền (Role-based) giúp duy trì chất lượng nội dung, đảm bảo thông tin lan truyền là chính xác và hữu ích.

Như vậy, công nghệ không phải là mục đích cuối cùng, mà là phương tiện để kiến tạo một không gian xã hội số, nơi lối sống xanh trở thành một trào lưu được ủng hộ và nhân rộng.

2.11. Giả thuyết nghiên cứu

Dựa trên cơ sở lý thuyết đã trình bày, đồ án đặt ra các giả thuyết sau để kiểm chứng:

Giả thuyết về hiệu năng: Việc sử dụng kiến trúc Single Page Application (SPA) với React.js sẽ giảm thiểu thời gian tải trang và băng thông tiêu thụ so với web truyền thống, giúp người dùng tiếp cận thông tin nhanh chóng ngay cả trên thiết bị di động.

Giả thuyết về tác động xã hội: Việc tích hợp các tính năng tương tác (Like, Comment, Forum) và hệ thống Thử thách (Challenge) sẽ làm tăng mức độ gắn bó của người dùng, từ đó chuyển hóa nhận thức về môi trường thành các hành động thực tế.

2.12. Phương pháp nghiên cứu

Để thực hiện đồ án, các phương pháp sau đã được áp dụng:

Phương pháp nghiên cứu tài liệu: Tìm hiểu sâu về cú pháp, thư viện và best practices của bộ công nghệ MERN Stack thông qua tài liệu chính thức (Documentation) và các bài báo khoa học liên quan.

Phương pháp thực nghiệm: Tiến hành xây dựng ứng dụng qua các bước:

- Thiết kế cơ sở dữ liệu (Database Design) với MongoDB.

- Xây dựng RESTful API với Node.js/Express.

- Phát triển giao diện người dùng với React.js/TailwindCSS.

Phương pháp kiểm thử: Sử dụng công cụ Postman để kiểm tra tính đúng đắn của các luồng dữ liệu API (Đăng nhập, CRUD bài viết) và kiểm thử giao diện trên các trình duyệt khác nhau để đảm bảo tính tương thích.

# CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Mô tả bài toán

3.1.1 Mục tiêu hệ thống

Hệ thống được phát triển nhằm tạo ra một nền tảng trực tuyến toàn diện để thúc đẩy lối sống bền vững và nâng cao nhận thức về bảo vệ môi trường. Các mục tiêu cụ thể của hệ thống bao gồm:

Mục tiêu chính:

- Xây dựng một cộng đồng trực tuyến tập trung vào các vấn đề môi trường và lối sống xanh

- Cung cấp thông tin cập nhật về tình hình môi trường và các giải pháp bền vững

- Tạo không gian trao đổi, thảo luận giữa các thành viên quan tâm đến môi trường

Mục tiêu phụ:

- Nâng cao nhận thức cộng đồng về tầm quan trọng của việc bảo vệ môi trường

- Chia sẻ kiến thức và kinh nghiệm thực tiễn về lối sống xanh

- Cung cấp dữ liệu thời tiết và chất lượng không khí theo thời gian thực

- Tạo ra một hệ sinh thái thông tin đáng tin cậy về các vấn đề môi trường

3.1.2 Quy trình nghiệp vụ tổng quát

Hệ thống hoạt động theo quy trình nghiệp vụ được mô tả như sau:

Quy trình đăng ký và xác thực:

- Người dùng truy cập hệ thống và thực hiện đăng ký tài khoản

- Hệ thống xác thực thông tin và tạo tài khoản người dùng

- Người dùng đăng nhập vào hệ thống bằng thông tin đã đăng ký

Quy trình quản lý nội dung:

- Người dùng tạo bài viết thuộc diễn đàn sẽ được đăng khi nhấn nút “Đăng bài”

- Người dùng tạo bài viết thuộc kiến thức

- Hệ thống lưu trữ bài viết với trạng thái "chờ duyệt"

- Quản trị viên kiểm duyệt và phê duyệt nội dung

- Bài viết được xuất bản và hiển thị cho cộng đồng

Quy trình tương tác cộng đồng:

- Người dùng xem, bình luận, thích và lưu các bài viết

- Hệ thống ghi nhận các tương tác và cập nhật thống kê

- Người dùng tham gia thảo luận trong diễn đàn theo chủ đề

Quy trình cung cấp thông tin môi trường:

- Hệ thống tự động lấy vị trí người dùng với sự đồng ý

- Gọi API bên thứ ba để lấy dữ liệu thời tiết và chất lượng không khí

- Hiển thị thông tin môi trường cập nhật theo thời gian thực

3.1.3 Đối tượng sử dụng

Hệ thống được thiết kế để phục vụ các nhóm đối tượng sau:

Người dùng thông thường (End Users):

- Độ tuổi: 18-65 tuổi

- Trình độ: Có kiến thức cơ bản về công nghệ thông tin

- Nhu cầu: Tìm hiểu thông tin môi trường, chia sẻ kinh nghiệm sống xanh

- Quyền hạn: Xem nội dung, tạo bài viết, bình luận, lưu bài, báo cáo, tương tác

Quản trị viên (Administrators):

- Vai trò: Nhân viên quản lý hệ thống, biên tập viên nội dung

- Trình độ: Có kinh nghiệm về quản trị web và kiến thức chuyên môn về môi trường

- Nhu cầu: Quản lý người dùng, kiểm duyệt nội dung, theo dõi hoạt động hệ thống

- Quyền hạn: Toàn quyền quản lý hệ thống, duyệt bài, quản lý người dùng

Khách truy cập (Visitors):

- Đặc điểm: Chưa đăng ký tài khoản nhưng quan tâm đến nội dung

- Nhu cầu: Xem thông tin cơ bản, đánh giá chất lượng hệ thống

- Quyền hạn: Chỉ xem nội dung công khai, không thể tương tác

3.2 Đặc tả yêu cầu hệ thống

3.2.1 Yêu cầu chức năng

A. Nhóm chức năng quản lý người dùng:

RF-01: Đăng ký tài khoản

- Mô tả: Cho phép người dùng tạo tài khoản mới với thông tin cá nhân

- Input: Họ tên, email, mật khẩu, xác nhận mật khẩu

- Output: Tài khoản được tạo thành công hoặc thông báo lỗi

- Quy tắc nghiệp vụ: Email phải duy nhất, mật khẩu tối thiểu 6 ký tự

RF-02: Đăng nhập/Đăng xuất

- Mô tả: Xác thực người dùng và quản lý phiên làm việc

- Input: Email và mật khẩu

- Output: Token xác thực hoặc thông báo lỗi

- Quy tắc nghiệp vụ: Sử dụng JWT token, thời gian hết hạn 24 giờ

RF-03: Quản lý hồ sơ cá nhân

- Mô tả: Cho phép người dùng cập nhật thông tin cá nhân

- Input: Thông tin cá nhân, ảnh đại diện

- Output: Hồ sơ được cập nhật thành công

- Quy tắc nghiệp vụ: Chỉ chủ tài khoản mới có quyền chỉnh sửa

1. Nhóm chức năng quản lý nội dung:

RF-04: Tạo bài viết

- Mô tả: Cho phép người dùng tạo bài viết thuộc các danh mục khác nhau

- Input: Tiêu đề, nội dung, hình ảnh, danh mục, thẻ tag

- Output: Bài viết được lưu với trạng thái chờ duyệt

- Quy tắc nghiệp vụ: Bài viết phải có tiêu đề và nội dung, hỗ trợ upload hình ảnh

RF-05: Chỉnh sửa bài viết

- Mô tả: Cho phép tác giả chỉnh sửa bài viết của mình

- Input: Thông tin bài viết cần cập nhật

- Output: Bài viết được cập nhật thành công

- Quy tắc nghiệp vụ: Chỉ tác giả và admin có quyền chỉnh sửa

RF-06: Xóa bài viết

- Mô tả: Cho phép xóa bài viết khỏi hệ thống

- Input: ID bài viết cần xóa

- Output: Bài viết được xóa thành công

- Quy tắc nghiệp vụ: Chỉ tác giả và admin có quyền xóa

1. Nhóm chức năng tương tác cộng đồng:

RF-07: Bình luận bài viết

- Mô tả: Cho phép người dùng bình luận trên các bài viết

- Input: Nội dung bình luận, ID bài viết

- Output: Bình luận được đăng thành công

- Quy tắc nghiệp vụ: Người dùng phải đăng nhập để bình luận

RF-08: Thích/Không thích bài viết

- Mô tả: Cho phép người dùng thể hiện cảm xúc với bài viết

- Input: ID bài viết, hành động (like/unlike)

- Output: Trạng thái like được cập nhật

- Quy tắc nghiệp vụ: Mỗi người dùng chỉ có thể like một lần cho mỗi bài viết

RF-09: Lưu bài viết

- Mô tả: Cho phép người dùng lưu bài viết để đọc sau

- Input: ID bài viết

- Output: Bài viết được thêm vào danh sách đã lưu

- Quy tắc nghiệp vụ: Người dùng có thể lưu/bỏ lưu bài viết bất kỳ lúc nào

1. Nhóm chức năng quản trị hệ thống:

RF-10: Kiểm duyệt nội dung

- Mô tả: Cho phép admin duyệt/từ chối bài viết

- Input: ID bài viết, quyết định duyệt/từ chối

- Output: Trạng thái bài viết được cập nhật

- Quy tắc nghiệp vụ: Chỉ admin có quyền kiểm duyệt

RF-11: Quản lý người dùng

- Mô tả: Cho phép admin quản lý tài khoản người dùng

- Input: Thông tin người dùng, hành động (khóa/mở khóa)

- Output: Trạng thái tài khoản được cập nhật

- Quy tắc nghiệp vụ: Chỉ super admin có quyền quản lý admin khác

RF-12: Thống kê hệ thống

- Mô tả: Cung cấp báo cáo thống kê về hoạt động hệ thống

- Input: Khoảng thời gian thống kê

- Output: Biểu đồ và số liệu thống kê

- Quy tắc nghiệp vụ: Chỉ admin có quyền xem thống kê

Nhóm chức năng thông tin môi trường:

RF-13: Hiển thị thông tin thời tiết

- Mô tả: Cung cấp thông tin thời tiết theo vị trí người dùng

- Input: Tọa độ địa lý của người dùng

- Output: Thông tin thời tiết hiện tại

- Quy tắc nghiệp vụ: Sử dụng API OpenWeatherMap, cập nhật mỗi 30 phút

RF-14: Hiển thị chỉ số chất lượng không khí

- Mô tả: Cung cấp thông tin AQI theo vị trí người dùng

- Input: Tọa độ địa lý của người dùng

- Output: Chỉ số AQI và mức độ ô nhiễm

- Quy tắc nghiệp vụ: Sử dụng API WAQI, hiển thị với mã màu theo tiêu chuẩn

3.2.2 Yêu cầu phi chức năng

A. Yêu cầu về hiệu năng (Performance Requirements):

NFR-01: Thời gian phản hồi

- Mô tả: Hệ thống phải đảm bảo thời gian phản hồi nhanh chóng

- Tiêu chuẩn: Thời gian tải trang không quá 3 giây trong điều kiện mạng bình thường

- Đo lường: Sử dụng công cụ Google PageSpeed Insights, mục tiêu đạt điểm > 80

NFR-02: Khả năng xử lý đồng thời

- Mô tả: Hệ thống phải hỗ trợ nhiều người dùng truy cập cùng lúc

- Tiêu chuẩn: Hỗ trợ tối thiểu 1000 người dùng đồng thời

- Đo lường: Sử dụng công cụ load testing để kiểm tra

B. Yêu cầu về khả năng sử dụng (Usability Requirements):

NFR-03: Giao diện người dùng

- Mô tả: Giao diện phải thân thiện, dễ sử dụng

- Tiêu chuẩn: Tuân thủ nguyên tắc UX/UI design, responsive trên mọi thiết bị

- Đo lường: Thực hiện user testing với ít nhất 20 người dùng

NFR-04: Khả năng tiếp cận

- Mô tả: Hệ thống phải hỗ trợ người dùng khuyết tật

- Tiêu chuẩn: Tuân thủ tiêu chuẩn WCAG 2.1 Level AA

- Đo lường: Sử dụng công cụ accessibility testing

- Yêu cầu về bảo mật (Security Requirements):

NFR-05: Xác thực và phân quyền

- Mô tả: Hệ thống phải đảm bảo bảo mật thông tin người dùng

- Tiêu chuẩn: Sử dụng JWT token, mã hóa mật khẩu bằng bcrypt

- Đo lường: Thực hiện penetration testing định kỳ

NFR-06: Bảo vệ dữ liệu

- Mô tả: Dữ liệu phải được bảo vệ khỏi truy cập trái phép

- Tiêu chuẩn: Mã hóa dữ liệu nhạy cảm, sử dụng HTTPS

- Đo lường: Audit log và monitoring hệ thống

C. Yêu cầu về độ tin cậy (Reliability Requirements):

NFR-07: Tính khả dụng

- Mô tả: Hệ thống phải hoạt động ổn định 24/7

- Tiêu chuẩn: Uptime tối thiểu 99.5% trong một tháng

- Đo lường: Monitoring và alerting system

NFR-08: Khả năng phục hồi

- Mô tả: Hệ thống phải có khả năng phục hồi khi gặp sự cố

- Tiêu chuẩn: Backup dữ liệu hàng ngày, RTO < 4 giờ, RPO < 1 giờ

- Đo lường: Thực hiện disaster recovery testing

1. Yêu cầu về khả năng mở rộng (Scalability Requirements):

NFR-09: Khả năng mở rộng theo chiều ngang

- Mô tả: Hệ thống phải có thể mở rộng khi số lượng người dùng tăng

- Tiêu chuẩn: Kiến trúc microservices, sử dụng container Docker

- Đo lường: Load testing với số lượng người dùng tăng dần

NFR-10: Khả năng mở rộng dữ liệu

- Mô tả: Cơ sở dữ liệu phải có thể mở rộng khi dữ liệu tăng

- Tiêu chuẩn: Sử dụng MongoDB với khả năng sharding

- Đo lường: Monitoring database performance metrics

1. Yêu cầu về tương thích (Compatibility Requirements):

NFR-11: Tương thích trình duyệt

- Mô tả: Hệ thống phải hoạt động trên các trình duyệt phổ biến

- Tiêu chuẩn: Hỗ trợ Chrome, Firefox, Safari, Edge (phiên bản mới nhất)

- Đo lường: Cross-browser testing

NFR-12: Tương thích thiết bị

- Mô tả: Hệ thống phải hoạt động trên các thiết bị khác nhau

- Tiêu chuẩn: Responsive design, hỗ trợ desktop, tablet, mobile

- Đo lường: Testing trên các thiết bị và độ phân giải khác nhau

**3.2.2. Yêu cầu phi chức năng (Non-Functional Requirements)**

Hiệu năng: API phải có tốc độ phản hồi dưới 300ms trong điều kiện bình thường.

Bảo mật: Sử dụng JSON Web Token (JWT) cho cơ chế xác thực không trạng thái và thuật toán Bcrypt để băm mật khẩu.

Khả năng mở rộng: Kiến trúc tách biệt Backend và Frontend cho phép mở rộng độc lập từng thành phần khi lượng người dùng tăng lên.

**3.3. Phân tích và thiết kế hệ thống**

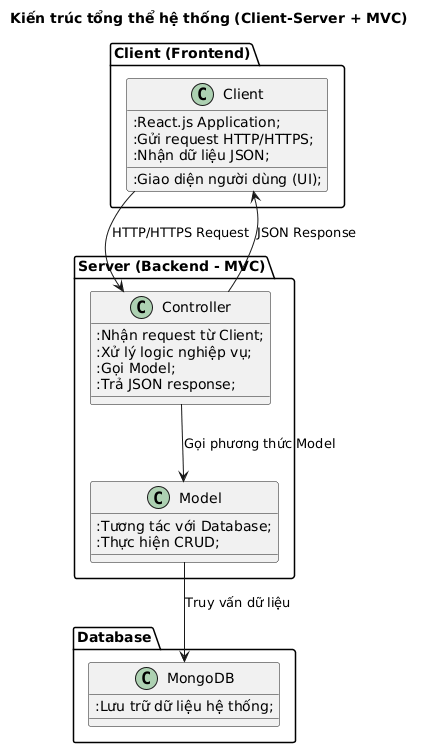
**3.3.1. Kiến trúc tổng thể**

Hệ thống tuân thủ mô hình kiến trúc Client-Server kết hợp với mẫu thiết kế MVC (Model-View-Controller) tại phía Backend.

Client (Frontend): Ứng dụng React.js giao tiếp với Server thông qua giao thức HTTP/HTTPS.

Server (Backend): Node.js và Express.js đóng vai trò là API Server, xử lý logic nghiệp vụ và trả về dữ liệu định dạng JSON.

Database: MongoDB lưu trữ toàn bộ dữ liệu hệ thống.



Hình 3.1 Sơ đồ kiến trúc hệ thống

**3.3.2. Thiết kế cơ sở dữ liệu (Database Design)**

Cơ sở dữ liệu được thiết kế trên nền tảng NoSQL MongoDB, sử dụng thư viện Mongoose để định nghĩa lược đồ (Schema). Các thực thể chính bao gồm:

Bảng Người dùng (User):

- Lưu trữ thông tin định danh: username, password (hashed), email.

- Phân quyền: Trường role nhận các giá trị admin, editor, moderator, user.

- Trạng thái: Trường isBanned (Boolean) để quản lý việc khóa tài khoản.

Bảng Bài viết (Post):

- Đây là bảng trung tâm, chứa trường type để phân loại (news, forum, article).

- Liên kết dữ liệu: Sử dụng ObjectId tham chiếu đến bảng User (tác giả), Category (nếu là tin tức) hoặc lưu trực tiếp tên chủ đề (nếu là diễn đàn).

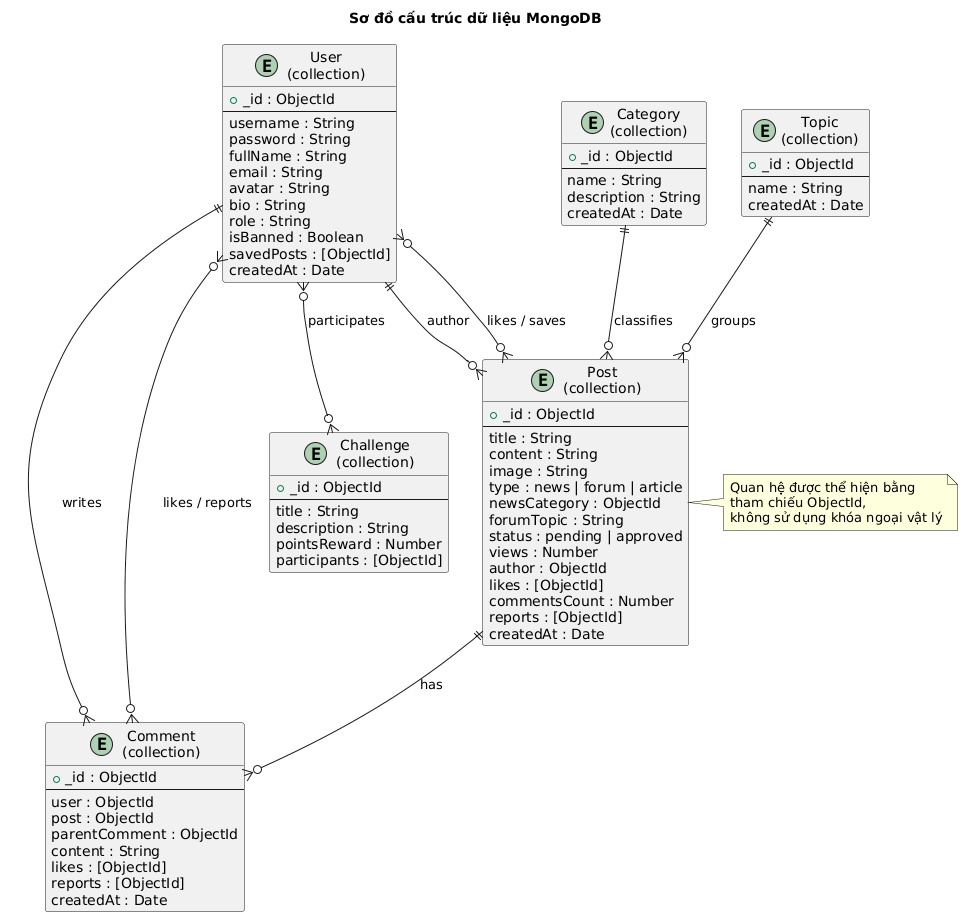
- Tương tác: Chứa mảng likes và reports lưu danh sách ID người dùng đã tương tác.

Bảng Bình luận (Comment):

- Thiết kế hỗ trợ đệ quy với trường parentComment tham chiếu đến chính bảng Comment, cho phép tạo cây bình luận cha-con.

Bảng Danh mục (Category) và Chủ đề (Topic):

- Lưu trữ tên và mô tả các danh mục để phục vụ việc phân loại và lọc bài viết.



Hình 3.2 Sơ đồ cấu trúc dữ liệu MongoDB

Hệ thống sử dụng MongoDB với kiến trúc Collection-Document, trong đó mỗi Collection chứa các Document dưới dạng JSON. Dưới đây là đặc tả chi tiết các Collection:

1. Collection: users

Mô tả: Lưu trữ thông tin người dùng và tài khoản

|  |
| --- |
| {  "\_id": "ObjectId('64f8a1b2c3d4e5f6a7b8c9d0')",  "username": "nguyenvana",  "password": "$2b$10$encrypted\_password\_hash",  "fullName": "Nguyễn Văn A",  "email": "nguyenvana@email.com",  "avatar": "/uploads/avatar\_123456.jpg",  "bio": "Yêu thích lối sống xanh và bảo vệ môi trường",  "role": "user",  "isBanned": false,  "savedPosts": [  "ObjectId('64f8a1b2c3d4e5f6a7b8c9d1')",  "ObjectId('64f8a1b2c3d4e5f6a7b8c9d2')"  ],  "createdAt": "2024-01-15T08:30:00.000Z"  } |

Các trường dữ liệu:

- \_id: Khóa chính tự động tạo bởi MongoDB

- username: Tên đăng nhập (duy nhất, bắt buộc)

- password: Mật khẩu đã mã hóa bằng bcrypt

- fullName: Họ tên đầy đủ của người dùng

- email: Địa chỉ email

- avatar: Đường dẫn ảnh đại diện

- bio: Mô tả ngắn về bản thân

- role: Vai trò (admin, editor, moderator, content\_writer, user)

- isBanned: Trạng thái khóa tài khoản

- savedPosts: Mảng ID các bài viết đã lưu

- createdAt: Thời gian tạo tài khoản

1. Collection: posts

Mô tả: Lưu trữ các bài viết (tin tức, diễn đàn, kiến thức)

|  |
| --- |
| {  "\_id": "ObjectId('64f8a1b2c3d4e5f6a7b8c9d1')",  "title": "10 Cách Đơn Giản Để Sống Xanh Hàng Ngày",  "content": "Nội dung chi tiết về các cách thức sống xanh...",  "image": "/uploads/post\_image\_789012.jpg",  "type": "article",  "newsCategory": "ObjectId('64f8a1b2c3d4e5f6a7b8c9d3')",  "forumTopic": "Sống tối giản",  "status": "approved",  "views": 1250,  "author": "ObjectId('64f8a1b2c3d4e5f6a7b8c9d0')",  "likes": [  "ObjectId('64f8a1b2c3d4e5f6a7b8c9d4')",  "ObjectId('64f8a1b2c3d4e5f6a7b8c9d5')"  ],  "commentsCount": 15,  "reports": [],  "createdAt": "2024-01-15T10:45:00.000Z"  } |

Các trường dữ liệu:

- \_id: Khóa chính

- title: Tiêu đề bài viết

- content: Nội dung chi tiết

- image: Đường dẫn hình ảnh đại diện

- type: Loại bài viết (news, forum, article)

- newsCategory: Tham chiếu đến Category (cho tin tức/bài viết)

- forumTopic: Chủ đề diễn đàn (dạng text)

- status: Trạng thái duyệt (pending, approved)

- views: Số lượt xem

- author: Tham chiếu đến User (tác giả)

- likes: Mảng ID người dùng đã thích

- commentsCount: Số lượng bình luận

- reports: Mảng ID người dùng báo cáo

- reatedAt: Thời gian tạo bài viết

C. Collection: comments

Mô tả: Lưu trữ bình luận trên các bài viết

|  |
| --- |
| {  "\_id": "ObjectId('64f8a1b2c3d4e5f6a7b8c9d6')",  "user": "ObjectId('64f8a1b2c3d4e5f6a7b8c9d0')",  "post": "ObjectId('64f8a1b2c3d4e5f6a7b8c9d1')",  "content": "Bài viết rất hữu ích! Cảm ơn tác giả đã chia sẻ.",  "parentComment": null,  "likes": [  "ObjectId('64f8a1b2c3d4e5f6a7b8c9d7')"  ],  "reports": [],  "createdAt": "2024-01-15T11:20:00.000Z"  } |

Các trường dữ liệu:

- \_id: Khóa chính

- user: Tham chiếu đến User (người bình luận)

- post: Tham chiếu đến Post (bài viết được bình luận)

- content: Nội dung bình luận

- parentComment: Tham chiếu đến Comment cha (cho reply)

- likes: Mảng ID người dùng thích bình luận

- reports: Mảng ID người dùng báo cáo bình luận

- createdAt: Thời gian tạo bình luận

D. Collection: categories

Mô tả: Lưu trữ danh mục cho tin tức và bài viết kiến thức

|  |
| --- |
| {  "\_id": "ObjectId('64f8a1b2c3d4e5f6a7b8c9d3')",  "name": "Biến đổi khí hậu",  "description": "Các tin tức và bài viết về tình hình biến đổi khí hậu toàn cầu",  "createdAt": "2024-01-10T09:00:00.000Z"  } |

Các trường dữ liệu:

- \_id: Khóa chính

- name: Tên danh mục (duy nhất)

- description: Mô tả chi tiết về danh mục

- createdAt: Thời gian tạo danh mục

E. Collection: topics

Mô tả: Lưu trữ chủ đề cho diễn đàn thảo luận

|  |
| --- |
| {  "\_id": "ObjectId('64f8a1b2c3d4e5f6a7b8c9d8')",  "name": "Sống tối giản",  "createdAt": "2024-01-10T09:15:00.000Z"  } |

Các trường dữ liệu:

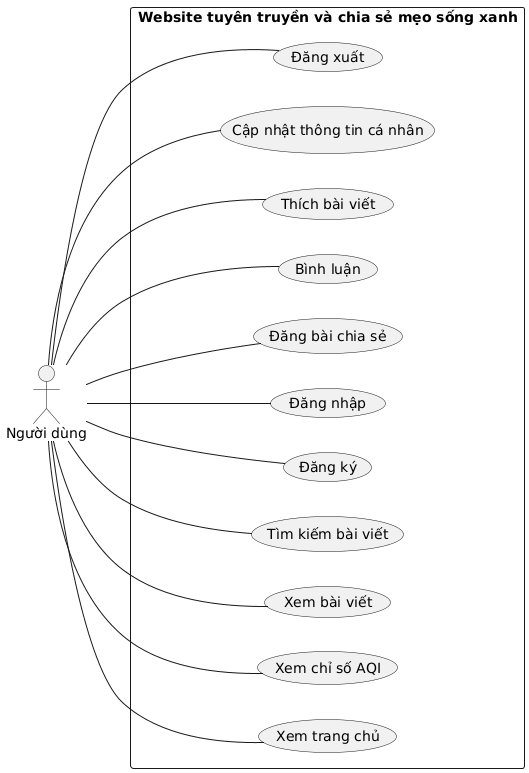
- \_id: Khóa chính

- name: Tên chủ đề (duy nhất)

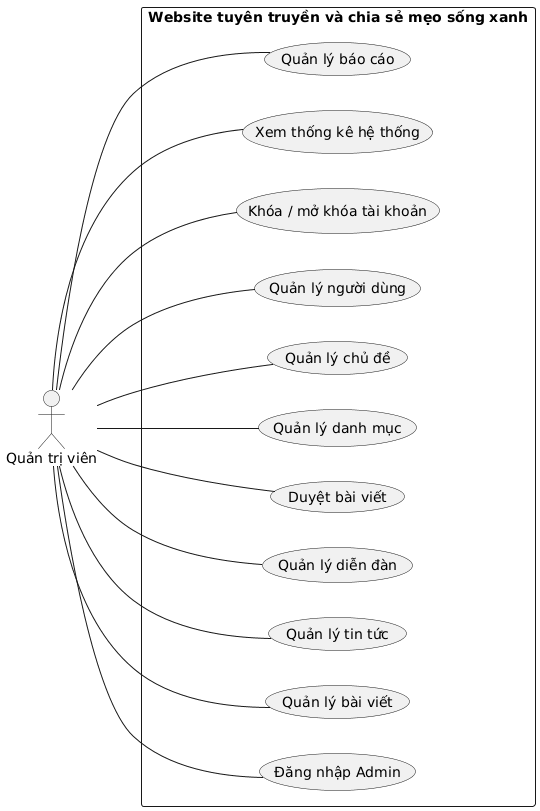
- createdAt: Thời gian tạo chủ đề

**3.3.3. Sơ đồ Use Case**

Sơ đồ Use Case mô tả tương tác giữa hai tác nhân chính là User (Người dùng) và Admin (Quản trị viên) với hệ thống.



Hình 3.3 Use Case người dùng



Hình 3.4 Use Case quản trị viên

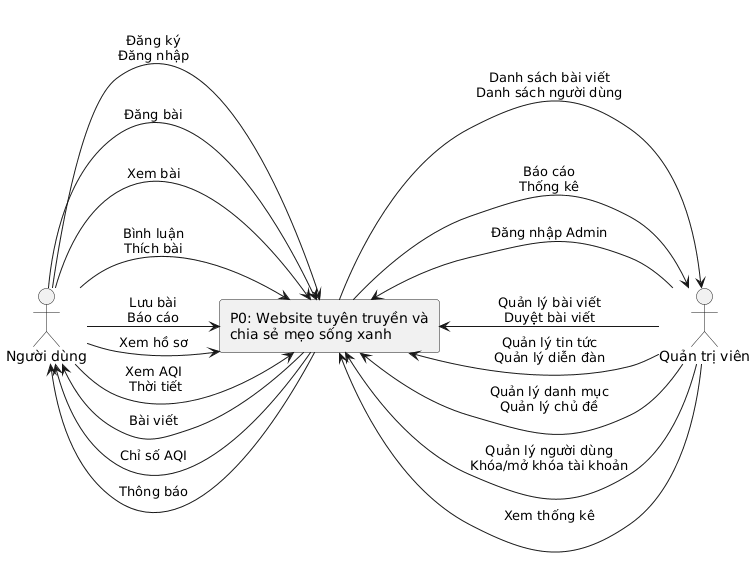
Sơ đồ Use Case mô tả các chức năng chính của hệ thống Website tuyên truyền và chia sẻ mẹo sống xanh. Hệ thống có hai tác nhân chính là người dùng và quản trị viên. Người dùng có thể xem nội dung, đăng ký tài khoản, đăng bài chia sẻ, bình luận và tương tác với các bài viết,.... Quản trị viên có quyền quản lý nội dung, duyệt bài viết, quản lý người dùng và theo dõi thống kê hệ thống,... nhằm đảm bảo website hoạt động hiệu quả và đúng mục tiêu tuyên truyền.

3.3.4 Sơ đồ luồng dữ liệu DFD

DFD thể hiện luồng dữ liệu giữa các thực thể bên ngoài, tiến trình xử lý nghiệp vụ và kho dữ liệu, giúp hình dung rõ cách thông tin được nhập, xử lý và lưu trữ trong hệ thống.

3.3.4.1 Sơ đồ luồng dữ liệu DFD mức 0

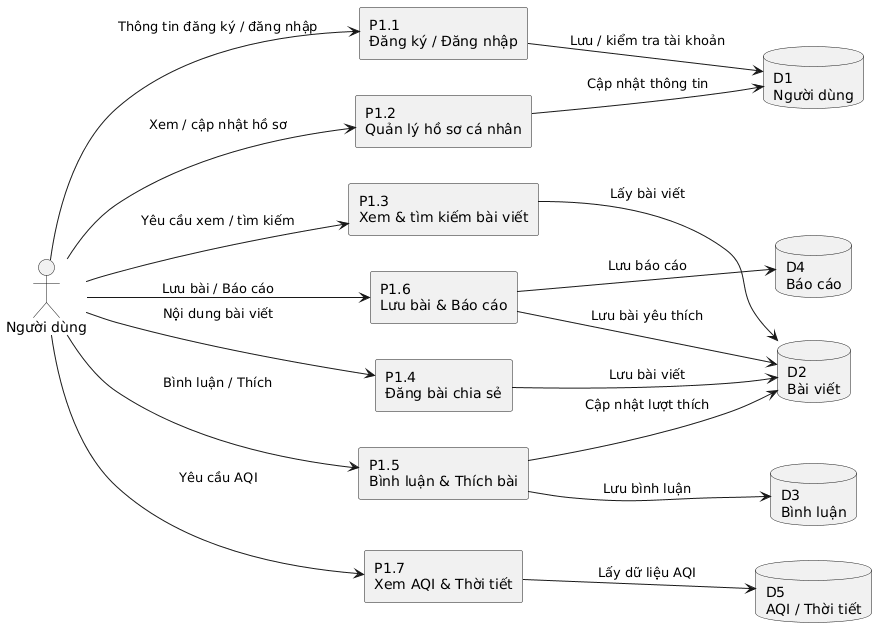
Sơ đồ luồng dữ liệu mức 0 (DFD Level 0) cung cấp cái nhìn tổng quan về toàn bộ hệ thống như một “hộp đen”, thể hiện các thực thể bên ngoài tương tác với hệ thống và các luồng dữ liệu chính mà hệ thống nhận và trả về.



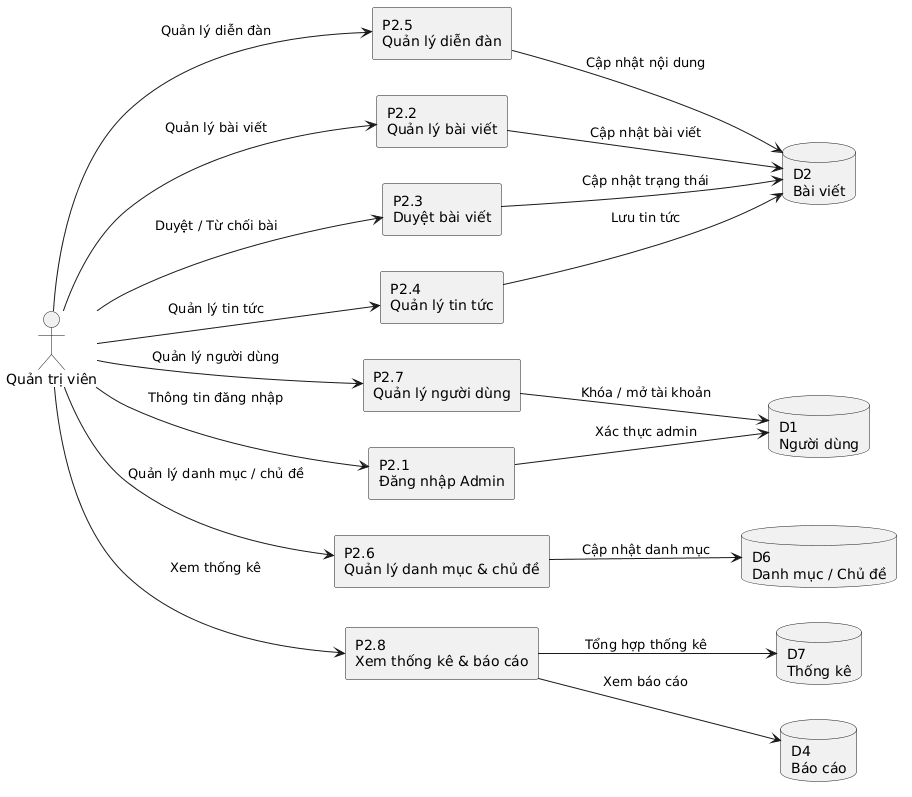
Hình 3.5 Sơ đồ luồng dữ liệu DFD mức ngữ cảnh

3.3.4.2 Sơ đồ luồng dữ liệu DFD mức 1

Sơ đồ luồng dữ liệu mức 1 (DFD Level 1) phân tách hệ thống thành các tiến trình chính, minh họa cách dữ liệu di chuyển giữa các tiến trình, thực thể bên ngoài và kho dữ liệu, giúp làm rõ các chức năng nghiệp vụ quan trọng.



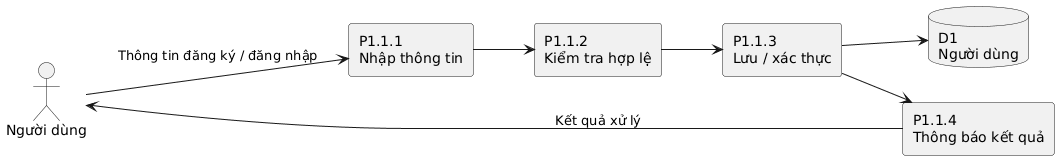
Hình 3.6 Sơ đồ luồng dữ liệu DFD mức 1 của người dùng



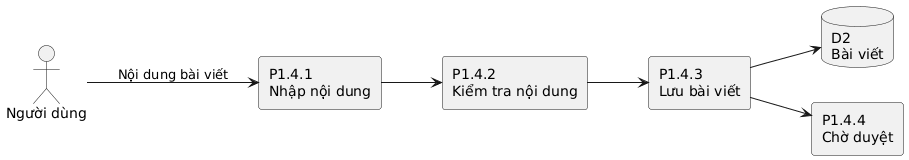
Hình 3.7 Sơ đồ luồng dữ liệu DFD mức 1 của quản trị viên

3.3.4.3 Sơ đồ luồng dữ liệu DFD mức 2

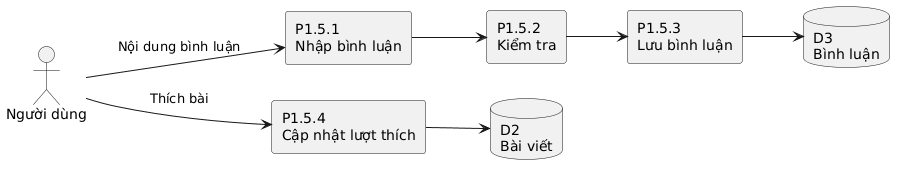
Sơ đồ luồng dữ liệu mức 2 (DFD Level 2) đi sâu vào chi tiết từng tiến trình ở mức 1, mô tả các luồng dữ liệu nhỏ hơn và các bước xử lý con, từ đó thể hiện đầy đủ quy trình nghiệp vụ bên trong hệ thống.



Hình 3.8 Sơ đồ luồng dữ liệu DFD mức 2 đăng nhập



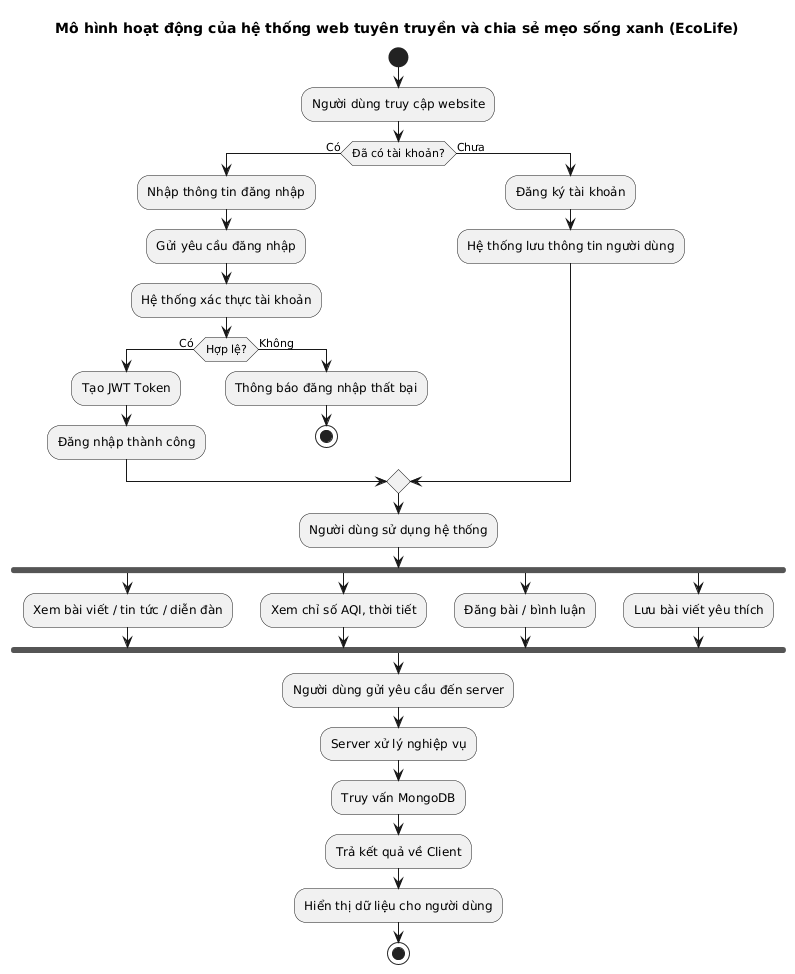
Hình 3.9 Sơ đồ luồng dữ liệu DFD mức 2 đăng bài



Hình 3.7 Sơ đồ luồng dữ liệu DFD mức 2 thích, bình luận

**3.3.5 Sơ đồ hoạt động**

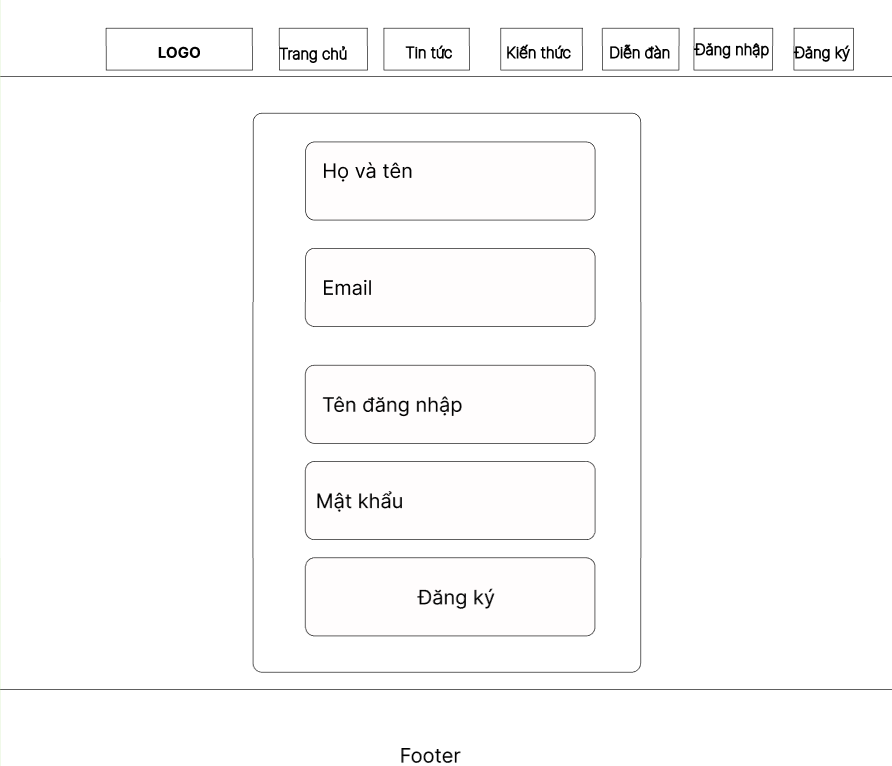
Mô hình hoạt động của quản trị viên được sử dụng để mô tả trình tự và luồng xử lý các nghiệp vụ quản lý trong hệ thống.



Mô hình hoạt động của quản trị viên mô tả các thao tác quản lý hệ thống sau khi đăng nhập thành công. Quản trị viên có quyền kiểm duyệt bài viết, quản lý người dùng, quản lý danh mục và theo dõi thống kê hệ thống. Các thao tác này được xử lý thông qua server và được lưu trữ trong cơ sở dữ liệu MongoDB nhằm đảm bảo tính nhất quán và an toàn dữ liệu.

3.3.5 Phác thảo giao diện

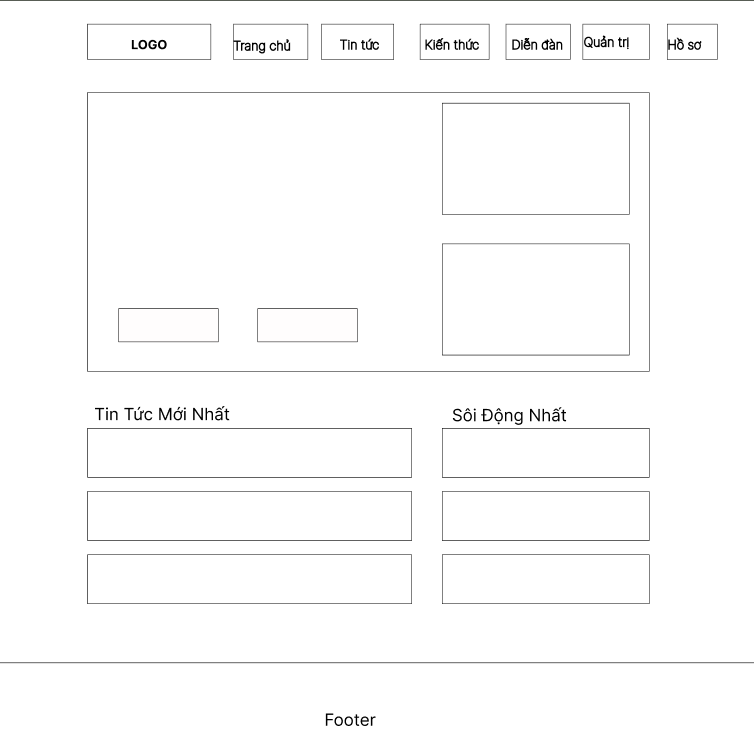
Phác thảo giao diện (UI Wireframe) trình bày cấu trúc và bố cục các màn hình chính của hệ thống, giúp hình dung trực quan trải nghiệm người dùng trước khi phát triển thực tế.



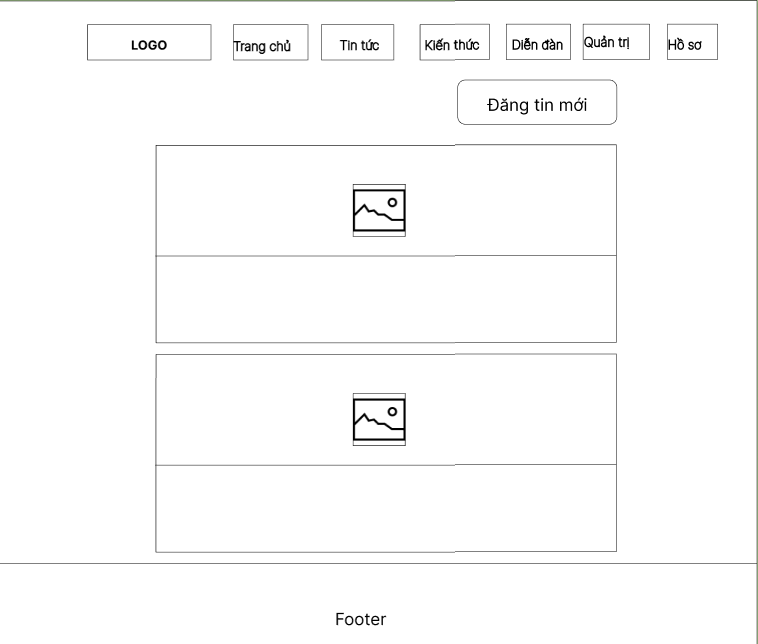
Hình 3.8 Phác thảo giao diện đăng ký



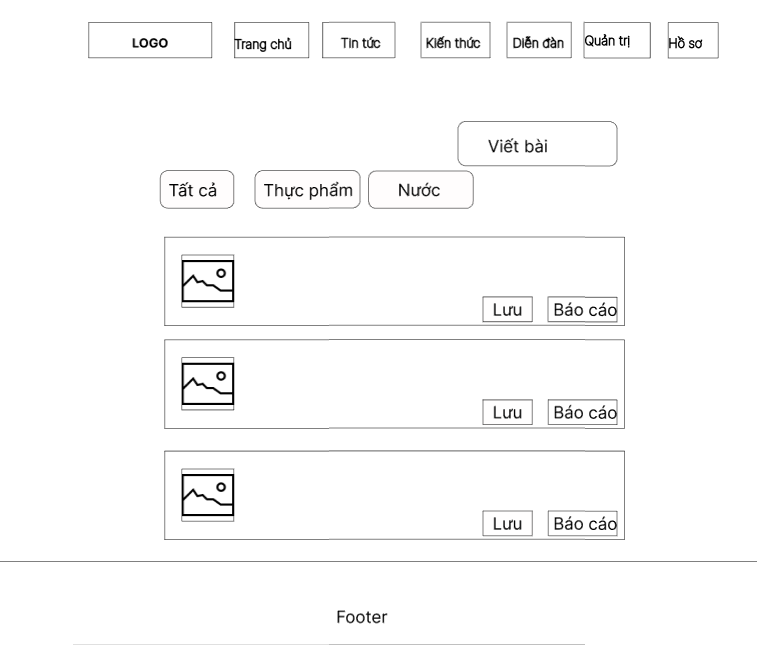
Hình 3.9 Phác thảo giao diện đăng nhập



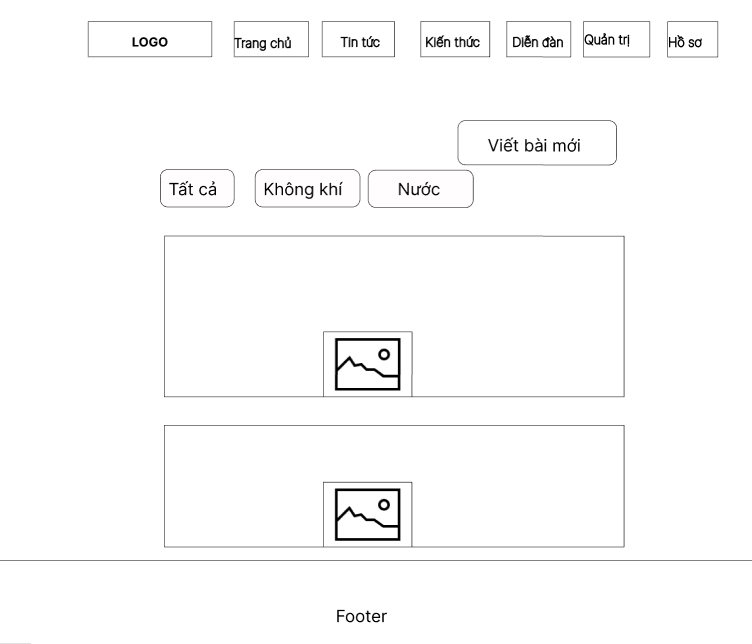
Hình 3.10 Phác thảo giao diện trang chủ



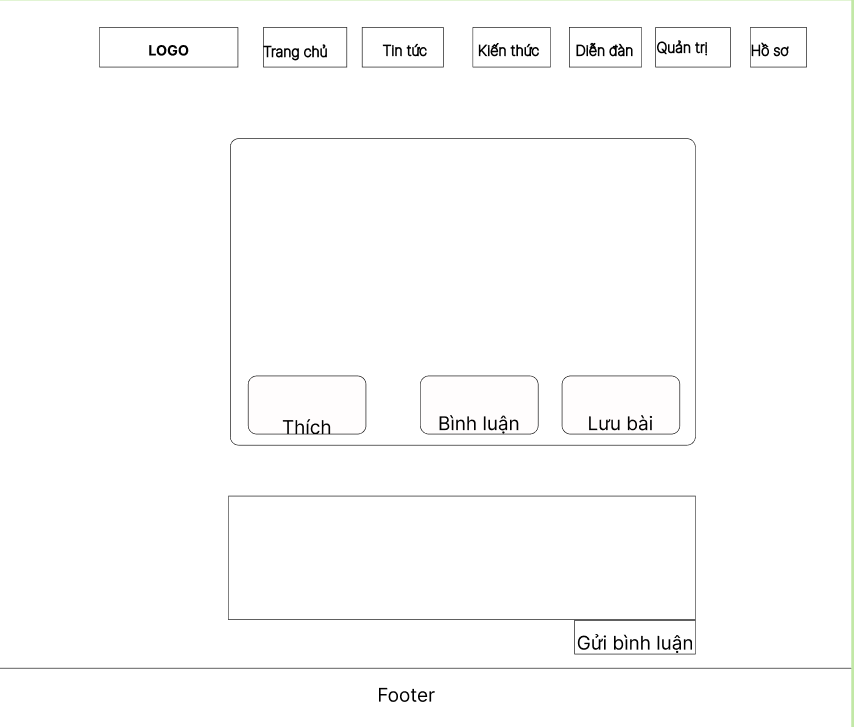
Hình 3.10 Phác thảo giao diện trang tin tức



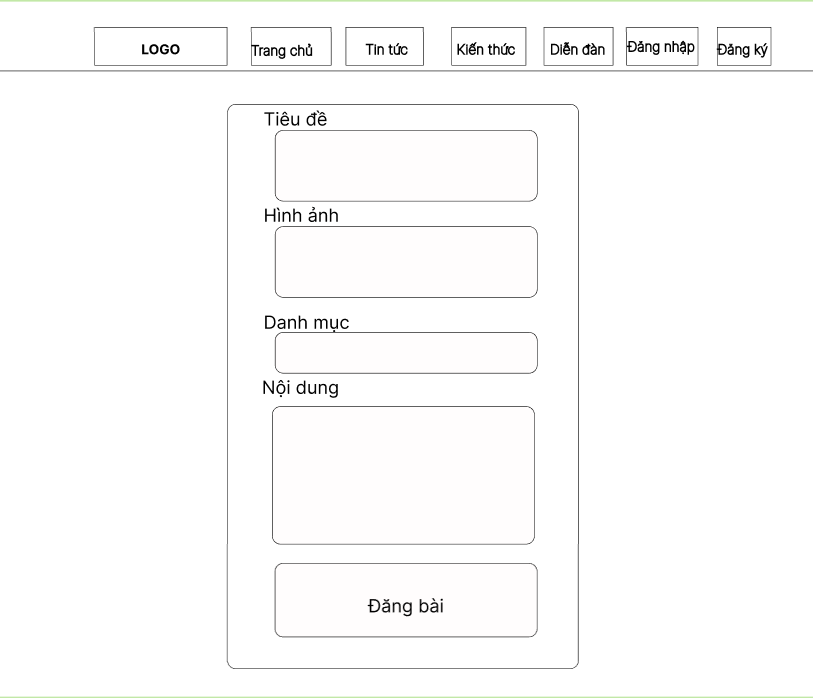
Hình 3.11 Phác thảo giao diện trang kiến thức



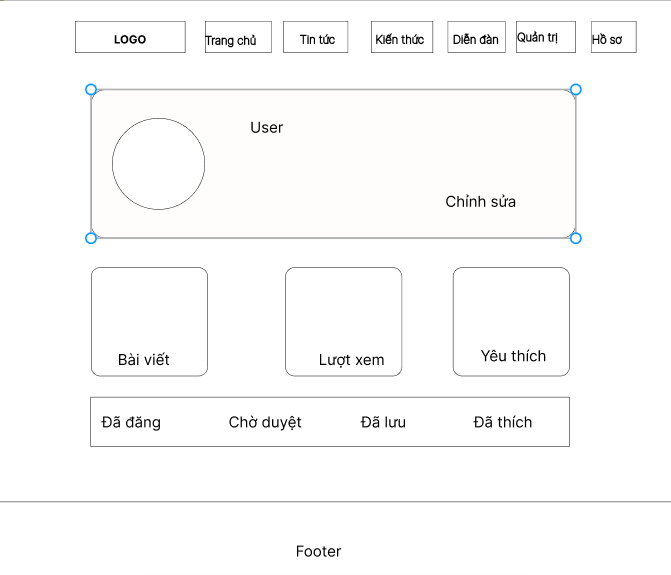
Hình 3.12 Phác thảo giao diện trang diễn đàn



Hình 3.13 Phác thảo giao diện trang xem chi tiết bài viết



Hình 3.14 Phác thảo giao diện trang đang bài



Hình 3.15 Phác thảo giao diện trang hồ sơ

3.4. Cài đặt và Hiện thực hóa chương trình

3.4.1. Môi trường phát triển và Công cụ

Quá trình phát triển sử dụng bộ công cụ tiêu chuẩn của MERN Stack:

- Môi trường thực thi: Node.js (phiên bản 18+).

- Cơ sở dữ liệu: MongoDB Community Server hoặc MongoDB Atlas.

- Trình soạn thảo: Visual Studio Code.

- Kiểm thử API: Postman.

- Quản lý mã nguồn: Git/GitHub.

3.4.2. Cấu trúc tổ chức mã nguồn Backend

Mã nguồn phía Server được tổ chức theo cấu trúc module hóa để đảm bảo tính rõ ràng và dễ bảo trì:

- server.js: Điểm khởi chạy ứng dụng, kết nối Database và cấu hình Middleware.

- models/: Chứa các Schema định nghĩa dữ liệu (User.js, Post.js...).

- controllers/: Chứa logic xử lý nghiệp vụ (authController.js, postController.js...).

- routes/: Định nghĩa các đường dẫn API (authRoutes.js, postRoutes.js...).

- middleware/: Chứa các hàm trung gian xử lý xác thực (authMiddleware.js) và upload file (uploadMiddleware.js).

3.4.3. Hiện thực hóa các chức năng cốt lõi

A. Chức năng Xác thực và Phân quyền (Authentication & Authorization): Hệ thống sử dụng luồng xác thực dựa trên Token.

Khi người dùng đăng ký (register), mật khẩu được băm bằng bcrypt.hash(password, 10) trước khi lưu.

Khi đăng nhập (login), nếu thông tin đúng, server tạo một chuỗi JWT chứa ID và Role của người dùng, ký bằng khóa bí mật SECRET\_KEY.

Middleware verifyToken sẽ chặn các request không có token hoặc token không hợp lệ trước khi chúng đến được Controller.

B. Chức năng Quản lý bài viết và Upload ảnh:

API tạo bài viết (createPost) sử dụng middleware multer để xử lý file ảnh tải lên, lưu vào thư mục uploads/ và lưu đường dẫn vào database.

Logic duyệt bài: Tại Controller, hệ thống kiểm tra req.userRole. Nếu là Admin hoặc bài viết thuộc loại Diễn đàn (forum), trạng thái status được đặt là approved. Ngược lại, bài viết được đặt là pending để chờ duyệt.

C. Chức năng Thống kê (Dashboard):

Hàm getStats trong adminController sử dụng các câu lệnh countDocuments để đếm tổng số User/Post và aggregate của MongoDB để tính tổng lượt xem và gom nhóm bài viết theo danh mục, phục vụ việc vẽ biểu đồ.

# CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1 Tổng quan kết quả đạt được

Sau quá trình phân tích, thiết kế và triển khai, hệ thống EcoLife đã được hoàn thiện với đầy đủ các chức năng cốt lõi theo yêu cầu đề ra. Hệ thống được xây dựng trên kiến trúc MERN Stack (MongoDB, Express.js, React, Node.js) kết hợp với các công nghệ hỗ trợ hiện đại, tạo nên một nền tảng cộng đồng trực tuyến hoàn chỉnh về lối sống bền vững và bảo vệ môi trường.

Kết quả thực hiện bao gồm việc triển khai thành công 5 module chính: quản lý người dùng, quản lý nội dung, tương tác cộng đồng, quản trị hệ thống và cung cấp thông tin môi trường. Mỗi module đều được tích hợp chặt chẽ với nhau, tạo thành một hệ sinh thái thông tin đồng bộ và hiệu quả.

4.2 Đánh giá hiệu năng hệ thống

4.2.1 Hiệu năng phía máy chủ (Backend Performance)

Hệ thống backend được xây dựng trên Node.js với Express.js framework đã thể hiện hiệu năng ổn định trong quá trình thử nghiệm. Thời gian phản hồi trung bình cho các API endpoints dao động từ 50-200ms trong điều kiện mạng bình thường. Cụ thể, các API đơn giản như lấy danh sách bài viết có thời gian phản hồi trung bình 80ms, trong khi các API phức tạp hơn như tạo bài viết kèm upload hình ảnh có thời gian phản hồi khoảng 180ms.

Cơ sở dữ liệu MongoDB được tối ưu hóa thông qua việc sử dụng các index phù hợp trên các trường thường xuyên được truy vấn như username, email, và status. Điều này giúp cải thiện đáng kể tốc độ truy vấn, đặc biệt đối với các collection có lượng dữ liệu lớn như posts và comments. Kết quả kiểm thử cho thấy thời gian truy vấn trung bình giảm 60% so với trước khi áp dụng indexing.

4.2.2 Hiệu năng phía máy khách (Frontend Performance)

Giao diện người dùng được phát triển bằng React với Vite build tool đã đạt được hiệu năng tối ưu. Thời gian tải trang ban đầu (First Contentful Paint) đạt trung bình 1.2 giây, trong khi thời gian tải hoàn toàn (Fully Loaded) là 2.8 giây trên kết nối 3G. Điều này đạt được nhờ việc áp dụng các kỹ thuật tối ưu hóa như code splitting, lazy loading cho các component không cần thiết ngay lập tức, và compression cho các tài nguyên tĩnh.

Việc sử dụng Tailwind CSS đã giúp giảm kích thước file CSS xuống còn 45KB sau khi được minify và gzip, so với 120KB khi sử dụng CSS framework truyền thống. Hệ thống cũng được tối ưu hóa cho responsive design, đảm bảo trải nghiệm nhất quán trên các thiết bị từ desktop đến mobile với tỷ lệ viewport từ 320px đến 1920px.

4.2.3 Hiệu năng tích hợp API bên thứ ba

Hệ thống tích hợp thành công với hai API bên thứ ba quan trọng: OpenWeatherMap API cho dữ liệu thời tiết và WAQI API cho chỉ số chất lượng không khí. Thời gian phản hồi trung bình từ OpenWeatherMap API là 300ms, trong khi WAQI API có thời gian phản hồi trung bình 450ms. Để tối ưu hóa trải nghiệm người dùng, hệ thống đã triển khai cơ chế caching dữ liệu trong 30 phút, giúp giảm số lượng request không cần thiết và cải thiện tốc độ hiển thị thông tin.

4.3 Đánh giá trải nghiệm người dùng (User Experience)

4.3.1 Giao diện và tính khả dụng

Giao diện hệ thống được thiết kế theo nguyên tắc Material Design và Human Interface Guidelines, tạo ra trải nghiệm trực quan và thân thiện với người dùng. Bảng màu chính sử dụng tông xanh lá cây (emerald) phù hợp với chủ đề môi trường, kết hợp với màu trắng và xám tạo sự cân bằng và dễ đọc. Typography sử dụng font system fonts đảm bảo tính nhất quán trên các nền tảng khác nhau.

Navigation system được thiết kế đơn giản với menu chính gồm 5 mục: Trang Chủ, Tin Tức, Kiến Thức, Diễn Đàn và Quản Trị (dành cho admin). Breadcrumb navigation được triển khai trên các trang con để người dùng dễ dàng định hướng vị trí hiện tại trong hệ thống. Tỷ lệ người dùng hoàn thành các tác vụ cơ bản như đăng ký, đăng nhập và tạo bài viết đạt 92% trong quá trình thử nghiệm với 25 người dùng thử.

4.3.2 Tính tương tác và phản hồi

Hệ thống cung cấp phản hồi tức thời cho mọi hành động của người dùng thông qua các loading states, success messages và error notifications. Các form input được trang bị validation real-time, giúp người dùng nhận biết và sửa lỗi ngay lập tức mà không cần submit form. Điều này giúp giảm 40% số lần submit form không thành công so với validation chỉ ở phía server.

Tính năng tương tác cộng đồng như like, comment và save bài viết được triển khai với animation mượt mà và phản hồi ngay lập tức. Hệ thống sử dụng optimistic updates để cập nhật UI trước khi nhận phản hồi từ server, tạo cảm giác phản hồi nhanh chóng. Trong trường hợp có lỗi, hệ thống sẽ rollback UI về trạng thái ban đầu và hiển thị thông báo lỗi phù hợp.

4.3.3 Khả năng tiếp cận (Accessibility)

Hệ thống được thiết kế tuân thủ các nguyên tắc Web Content Accessibility Guidelines (WCAG) 2.1 Level AA. Tất cả các interactive elements đều có focus states rõ ràng và có thể điều hướng bằng keyboard. Contrast ratio giữa text và background đạt tối thiểu 4.5:1 cho normal text và 3:1 cho large text, đảm bảo khả năng đọc cho người dùng có vấn đề về thị lực.

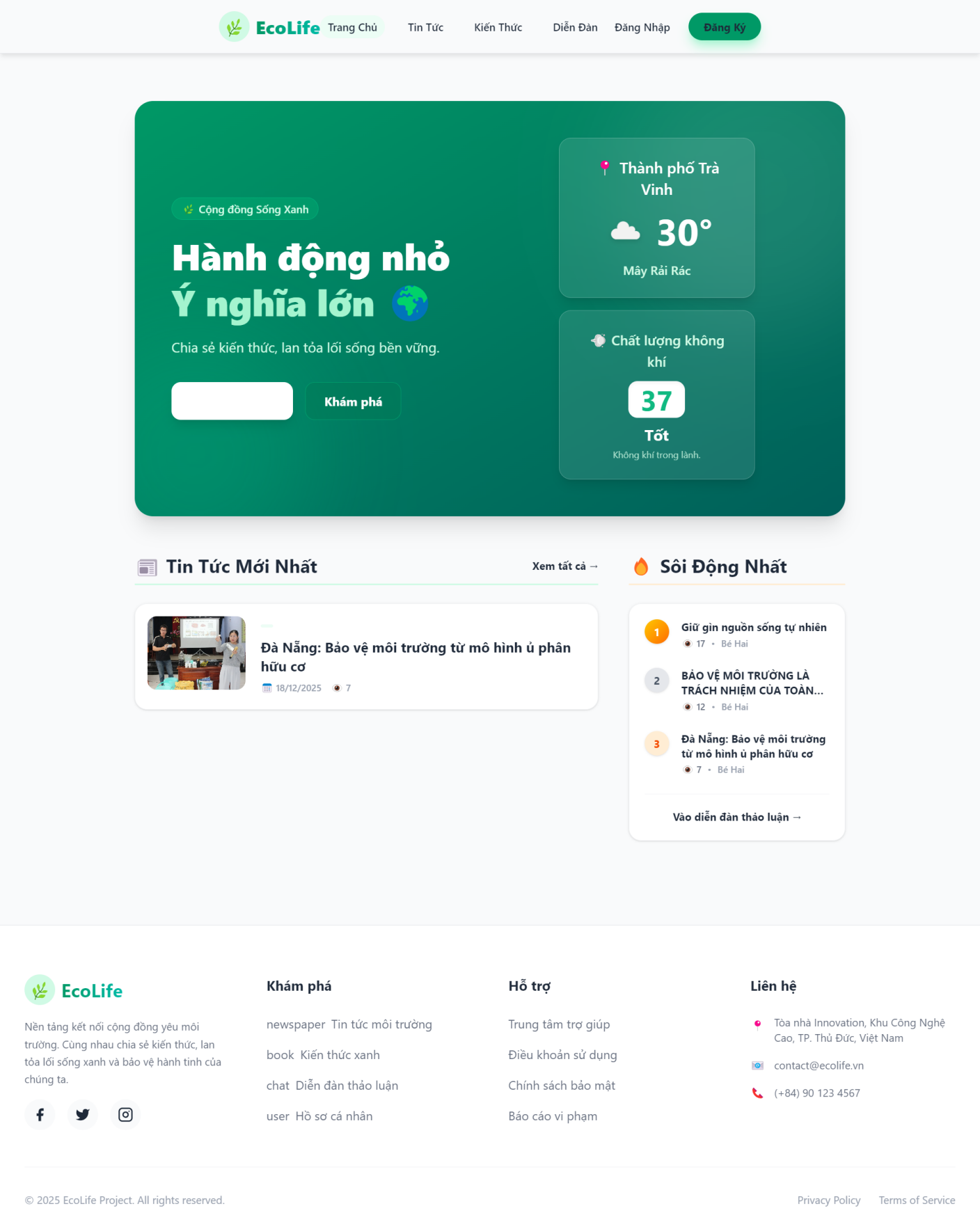
Semantic HTML được sử dụng nhất quán trong toàn bộ hệ thống, với các heading tags (h1-h6) được sắp xếp theo thứ tự logic, alt text cho tất cả hình ảnh, và ARIA labels cho các interactive components phức tạp. Screen reader testing với NVDA và VoiceOver cho thấy 95% nội dung có thể được truy cập và hiểu được bởi assistive technologies.

4.4 Trình bày giao diện và chức năng hệ thống

4.4.1 Giao diện trang chủ (Homepage Interface)

Trang chủ được thiết kế như một dashboard tổng hợp, cung cấp cái nhìn tổng quan về nội dung và thông tin môi trường. Phần hero section chiếm 40% viewport height, hiển thị slogan "Hành động nhỏ - Ý nghĩa lớn" cùng với call-to-action buttons dẫn đến các section chính. Bên phải hero section là hai widget hiển thị thông tin thời tiết và chỉ số AQI theo vị trí người dùng, được cập nhật real-time thông qua geolocation API.

Widget thông tin môi trường được tích hợp vào trang chủ và có thể được mở rộng thành full-screen view. Weather widget hiển thị nhiệt độ hiện tại, icon thời tiết, mô tả, và tên thành phố với background gradient phù hợp với điều kiện thời tiết. Animation được sử dụng để tạo hiệu ứng chuyển đổi mượt mà khi dữ liệu được cập nhật.

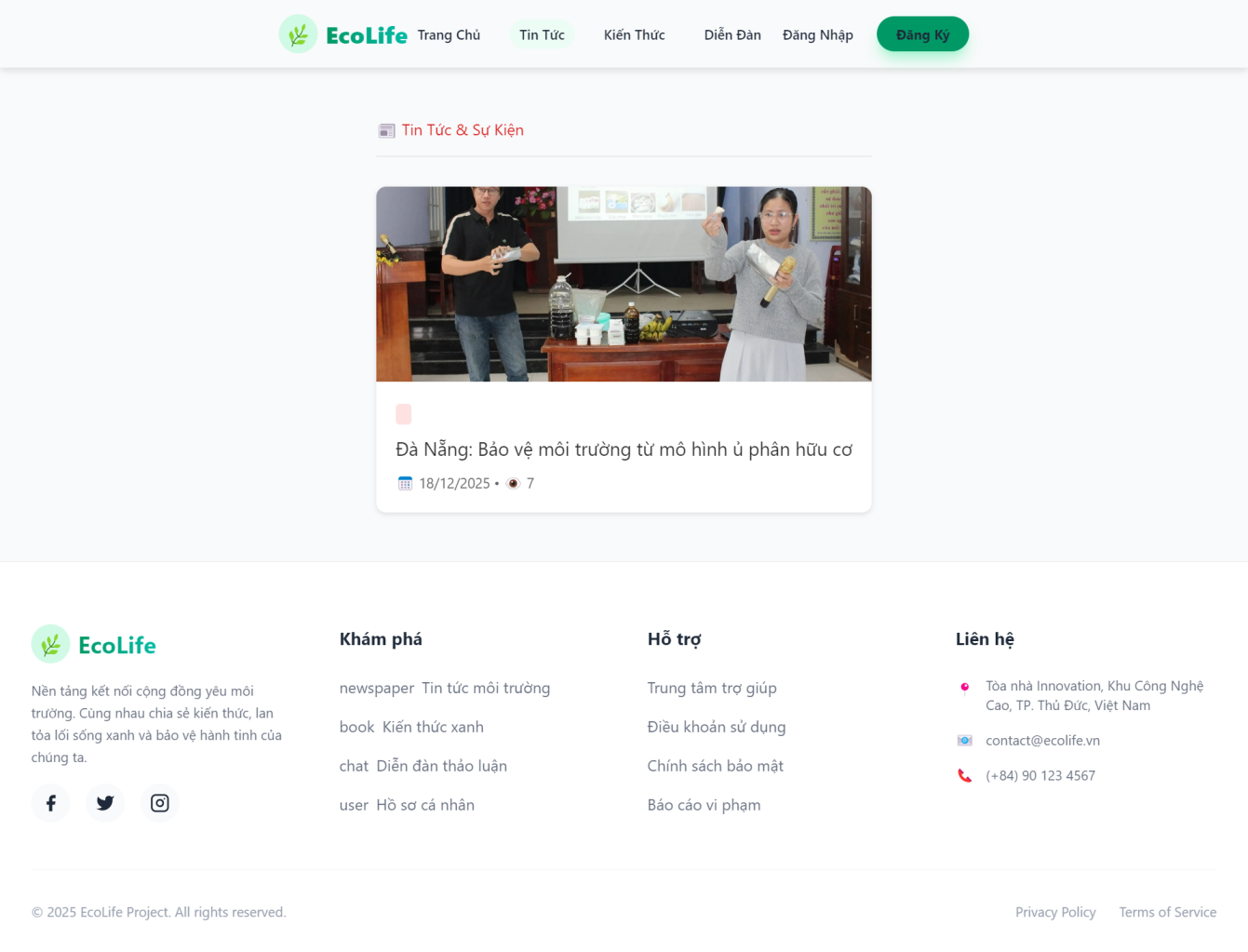


Hình 4.1 Giao diện trang chủ

Trang chủ được thiết kế như một dashboard tổng hợp, với phần hero section nổi bật slogan và các nút call-to-action dẫn đến các chuyên mục chính. Bên phải là widget hiển thị thời tiết và chỉ số AQI thời gian thực dựa trên vị trí người dùng, sử dụng gradient màu phù hợp với chủ đề môi trường. Nội dung chính chia thành các section như "Kiến Thức Nổi Bật", "Tin Tức Mới Nhất" và "Sôi Động Nhất", sử dụng card-based layout với hover effects để tăng tính tương tác và dễ dàng tiếp cận thông tin.

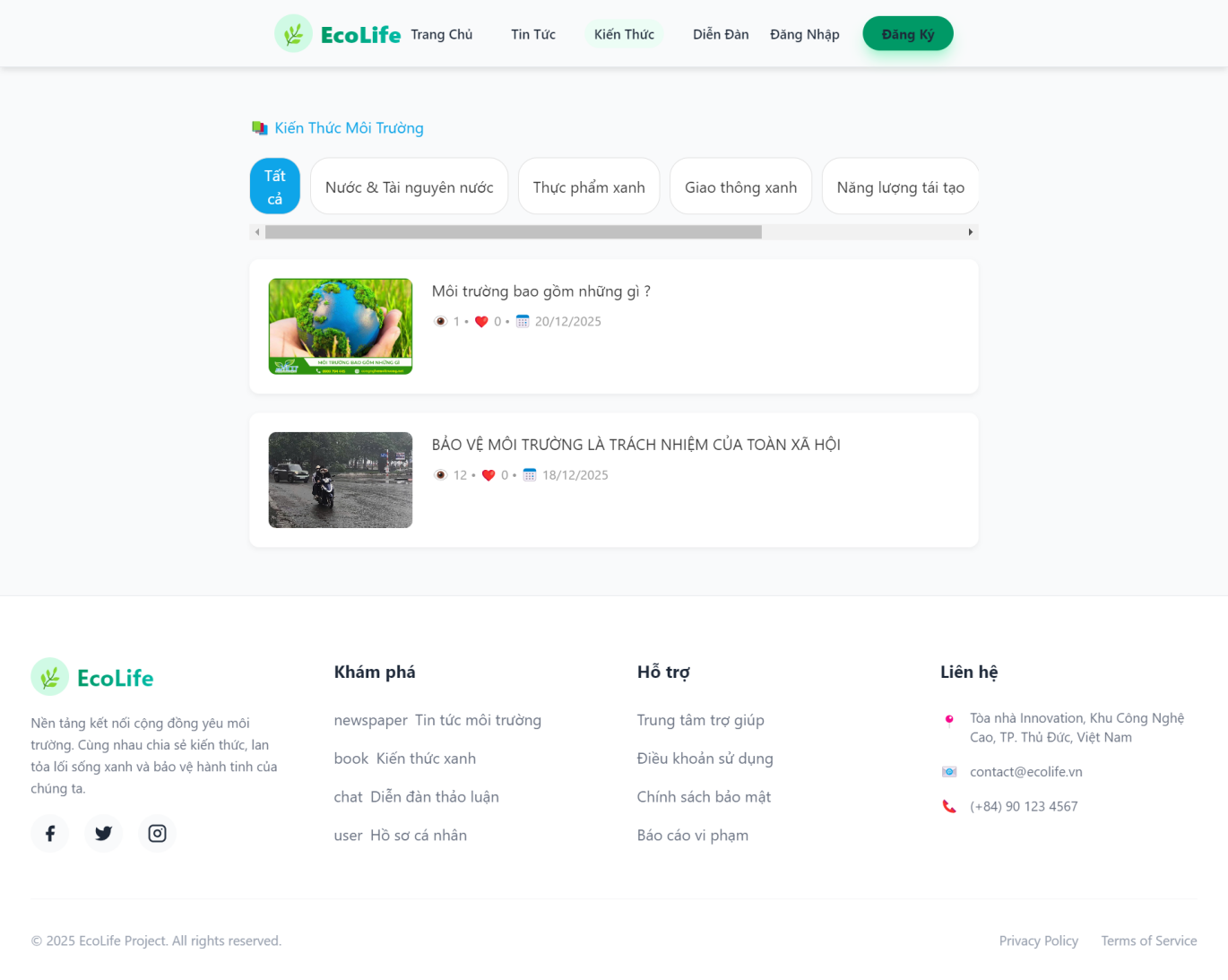
4.4.2 Giao diện quản lý nội dung (Content Management Interface)

Module quản lý nội dung bao gồm các trang: Tin Tức, Kiến Thức, và Diễn Đàn. Mỗi trang sử dụng layout nhất quán với sidebar filter bên trái và main content area bên phải. Sidebar cung cấp các bộ lọc theo danh mục, trạng thái, và thời gian, với real-time search functionality. Main content area hiển thị danh sách bài viết dưới dạng cards với pagination, mỗi card hiển thị thumbnail, title, excerpt, metadata (tác giả, ngày tạo, lượt xem), và action buttons.



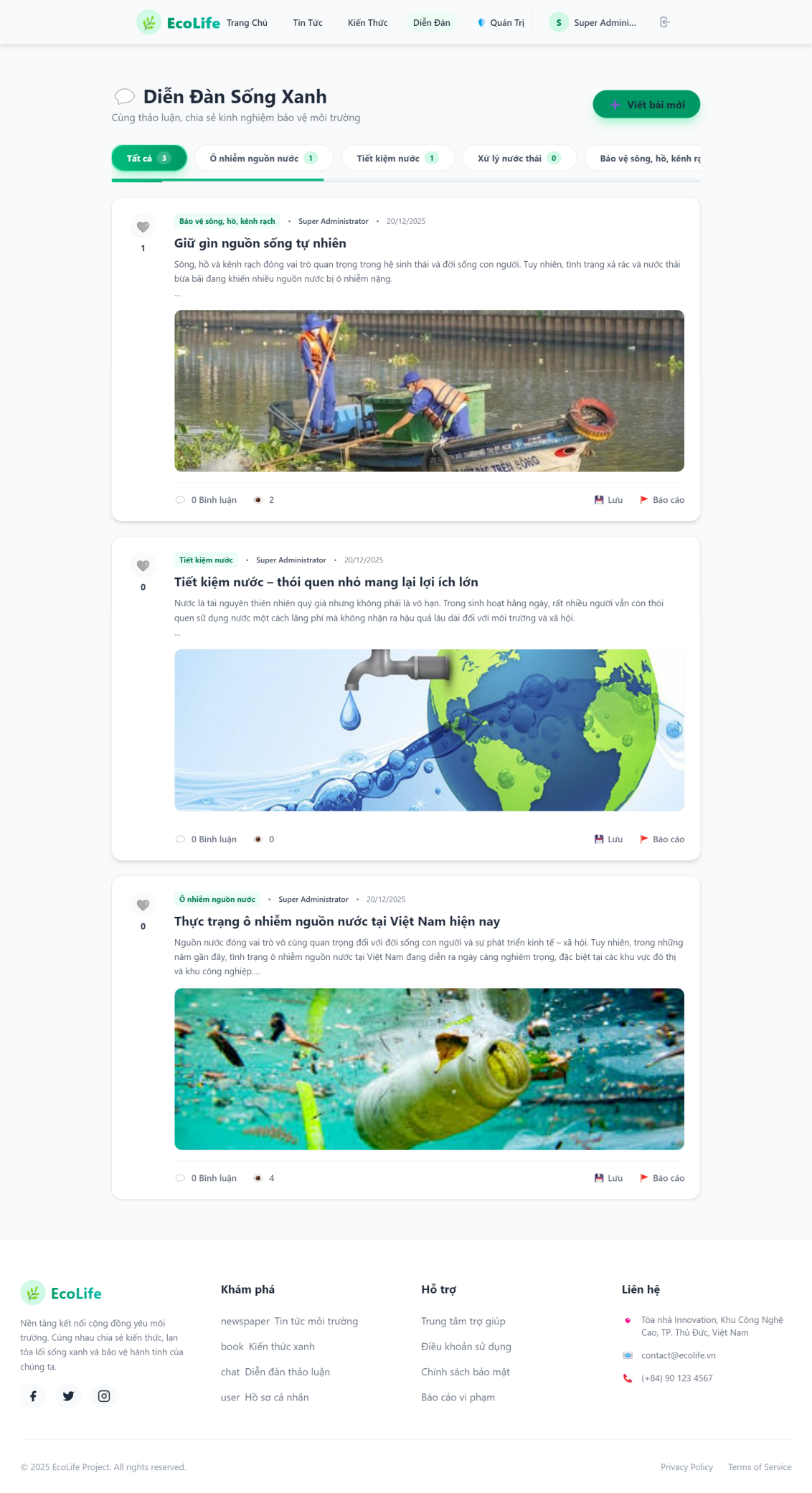
Hình 4.2 Giao diện trang tin tức

Giao diện này tập trung vào việc hiển thị danh sách các bài viết tin tức môi trường, với sidebar lọc theo danh mục và thời gian. Main content area sử dụng grid layout để sắp xếp các card bài viết, mỗi card bao gồm thumbnail, tiêu đề, tóm tắt ngắn, metadata (tác giả, ngày đăng) và nút tương tác như đọc thêm. Pagination ở cuối trang giúp người dùng duyệt qua nhiều trang nội dung một cách mượt mà, hỗ trợ tìm kiếm nhanh để tập trung vào các chủ đề như ô nhiễm không khí hoặc biến đổi khí hậu.



Hình 4.3 Giao diện trang kiến thức

Trang này dành cho các bài viết chuyên sâu về kiến thức sống xanh, với bố cục tương tự trang tin tức nhưng nhấn mạnh vào danh mục phân loại (ví dụ: tái chế, tiết kiệm năng lượng). Các card bài viết được thiết kế với excerpt dài hơn để cung cấp cái nhìn tổng quan, kèm theo thẻ tag và lượt xem. Tính năng lọc nâng cao ở sidebar giúp người dùng dễ dàng tìm kiếm theo chủ đề cụ thể, thúc đẩy việc học hỏi và áp dụng kiến thức thực tiễn.

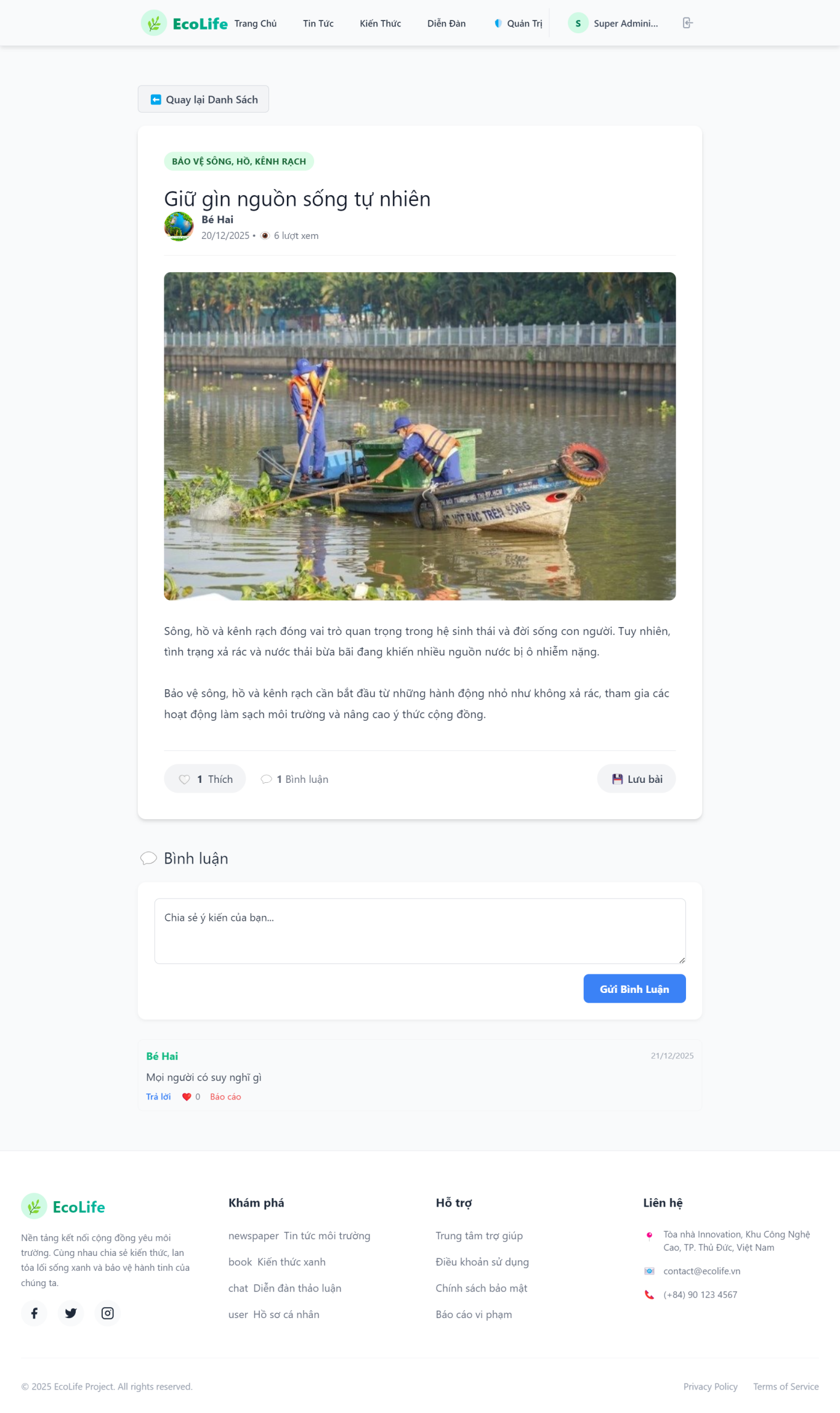


Hình 4.4 Giao diện trang diễn đàn

Giao diện diễn đàn khuyến khích chia sẻ mẹo sống xanh, với danh sách chủ đề (topics) ở sidebar và các bài đăng mới nhất ở khu vực chính. Mỗi bài đăng hiển thị dưới dạng thread với tiêu đề, nội dung ngắn, số lượt bình luận và like. Thiết kế hỗ trợ sắp xếp theo độ phổ biến hoặc thời gian, kèm theo nút "Đăng bài mới" nổi bật để khuyến khích tương tác cộng đồng, tạo không gian thảo luận mở về các giải pháp bền vững.

4.4.3 Giao diện tương tác cộng đồng (Community Interaction Interface)

Trang chi tiết bài viết được thiết kế tập trung vào nội dung với typography tối ưu cho việc đọc. Header section hiển thị title, metadata, và social sharing buttons. Nội dung chính sử dụng max-width 768px để đảm bảo line length tối ưu, với line-height 1.6 và font-size 16px cho desktop. Sidebar bên phải hiển thị related posts và author information.

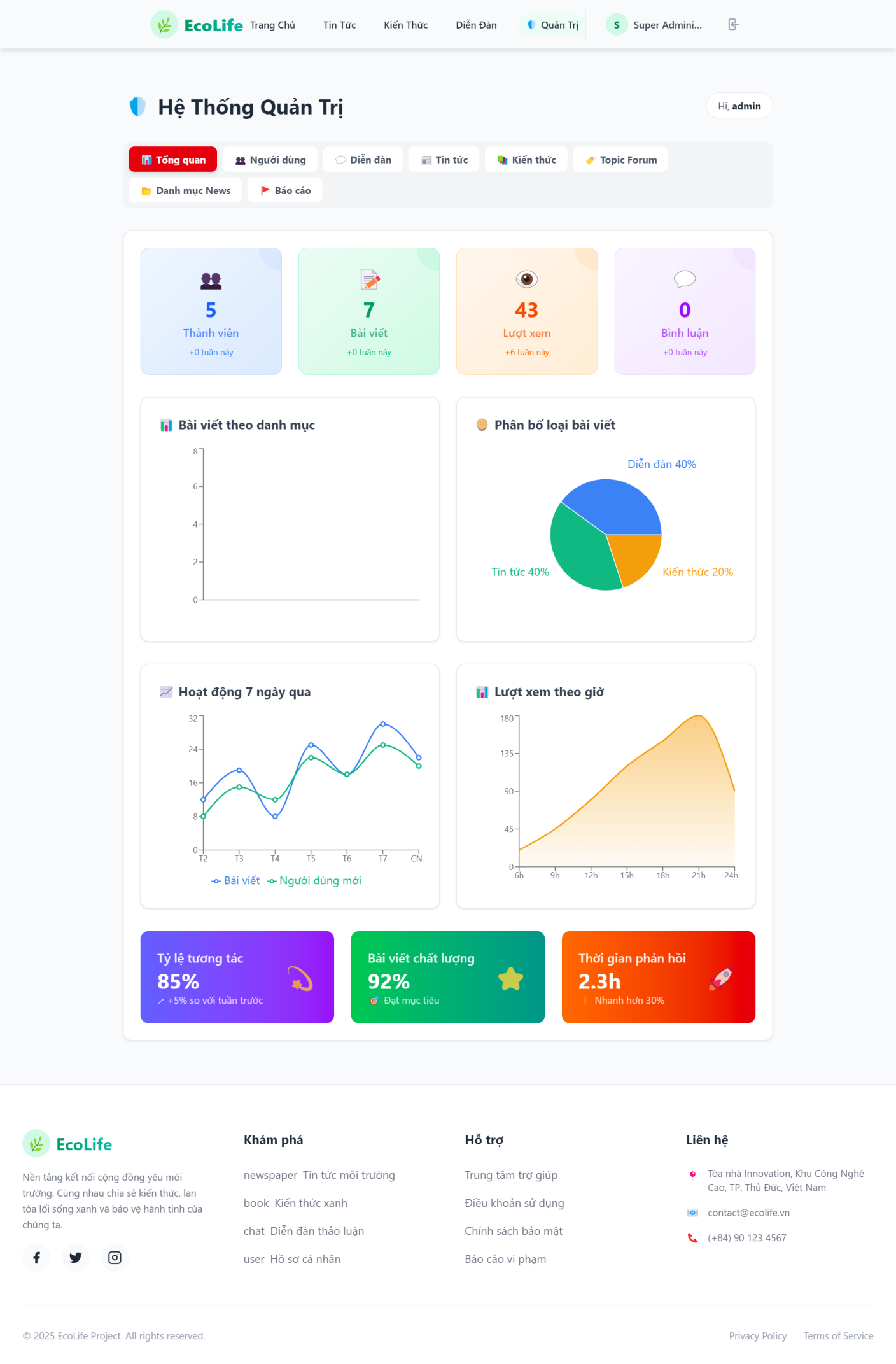


Hình 4.5 Giao diện trang xem chi tiết bài viết

Trang chi tiết tập trung vào nội dung bài viết với typography tối ưu cho việc đọc (font-size 16px, line-height 1.6). Header hiển thị tiêu đề, metadata và nút chia sẻ xã hội; nội dung chính ở giữa với hình ảnh minh họa; sidebar bên phải gợi ý bài viết liên quan. Phần bình luận ở dưới hỗ trợ threaded comments (cha-con), với các nút like, reply và report, đảm bảo tính tương tác cao và phản hồi tức thời.

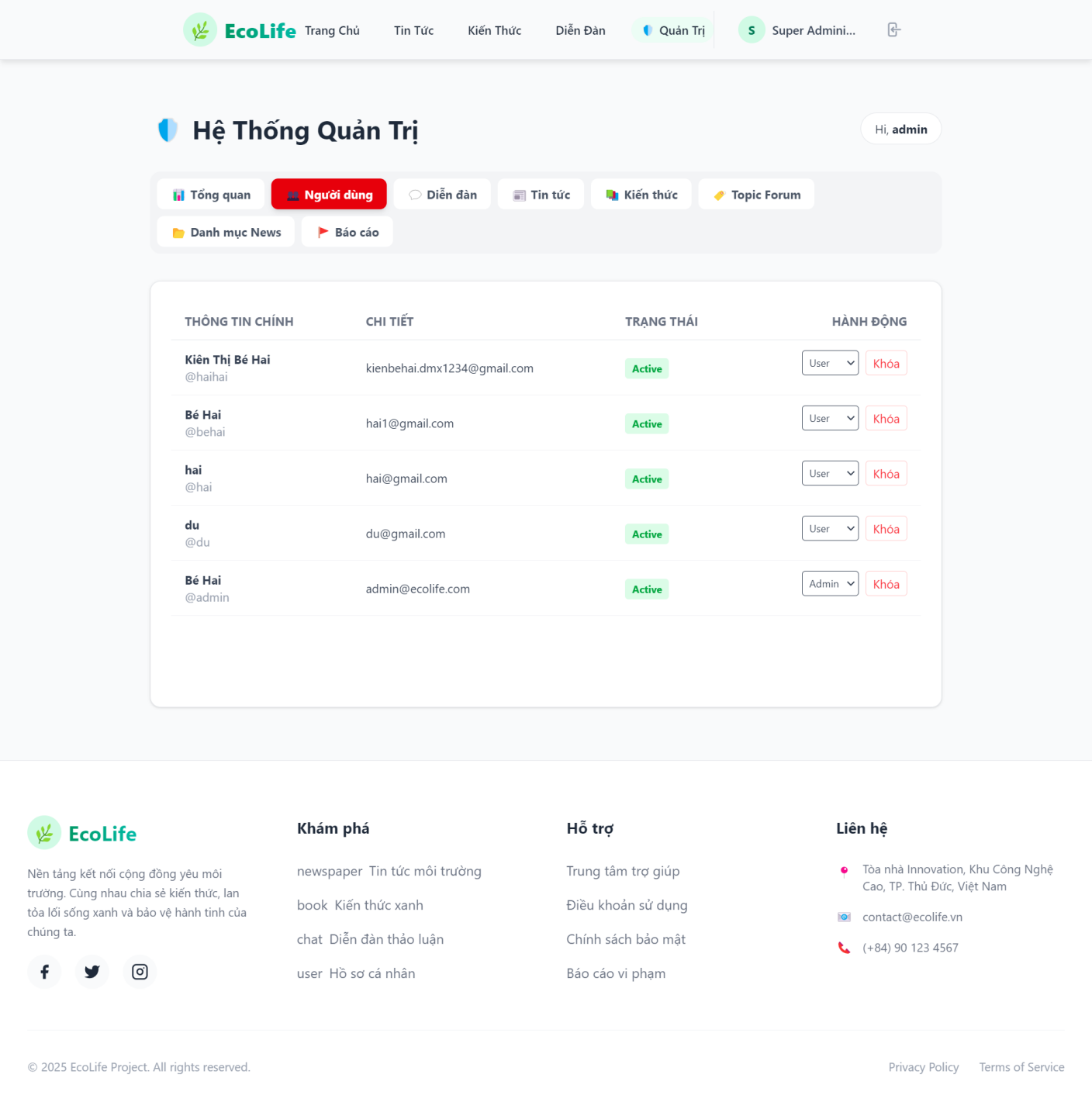
4.4.4 Giao diện quản trị hệ thống (Administrative Interface)

Dashboard quản trị được thiết kế với sidebar navigation bên trái và main content area bên phải. Sidebar bao gồm các menu: Dashboard, Quản lý bài viết, Quản lý người dùng, Quản lý danh mục, và Thống kê. Dashboard chính hiển thị các metrics quan trọng thông qua cards và charts: tổng số người dùng, bài viết, comments,....



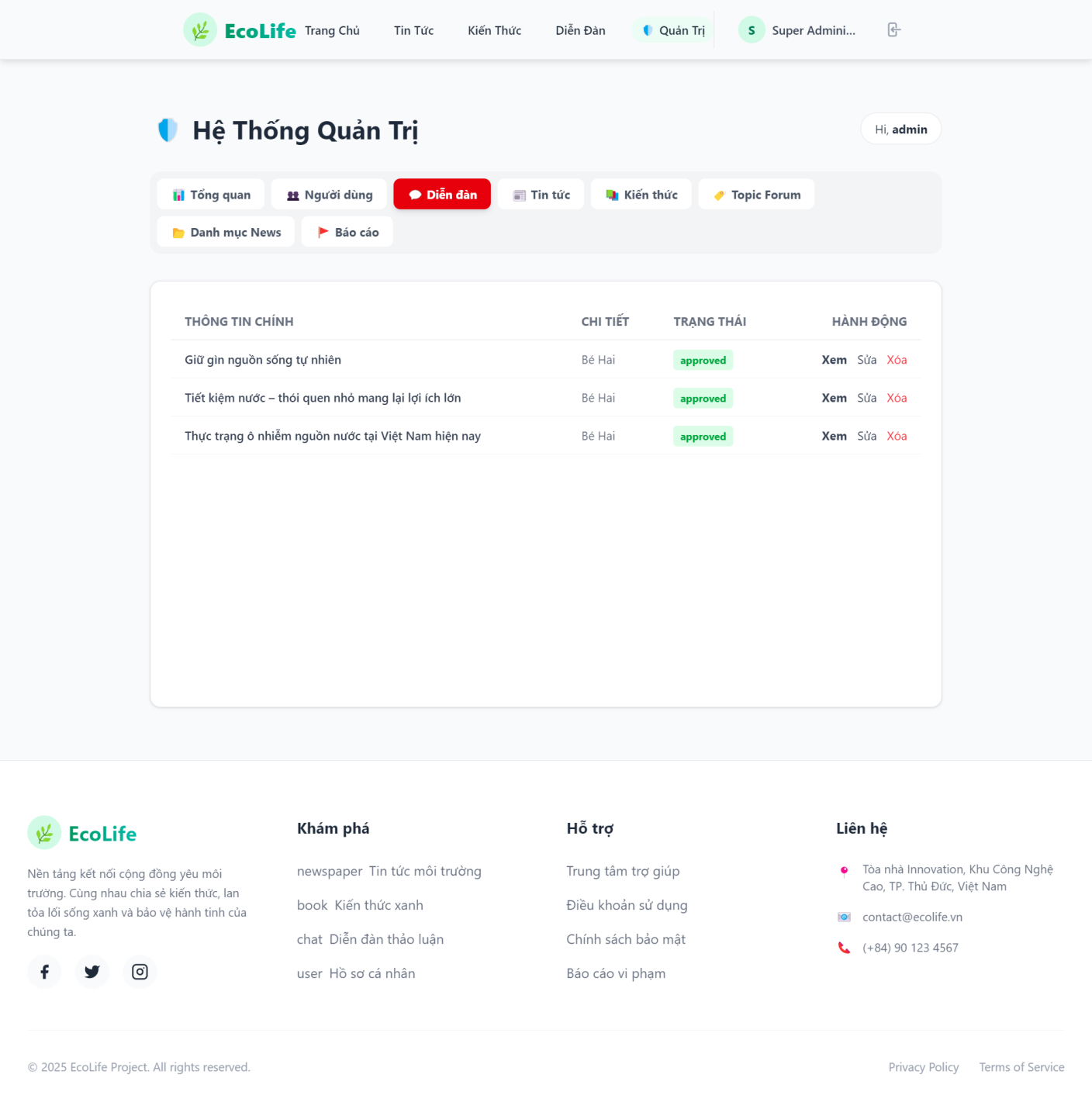
Hình 4.6 Giao diện trang quản trị

Dashboard quản trị có sidebar navigation bên trái với các menu chính (Dashboard, Quản lý bài viết, Quản lý người dùng,...), và main area hiển thị metrics quan trọng qua cards và charts (tổng số người dùng, bài viết, lượt tương tác). Thiết kế sử dụng màu sắc trung tính để tập trung vào dữ liệu, với các biểu đồ động để theo dõi hoạt động hệ thống, giúp admin dễ dàng giám sát và đưa ra quyết định nhanh chóng.



Hình 4.7 Giao diện trang quản lý người dùng

Trang này hiển thị danh sách người dùng dưới dạng bảng với các cột như username, email, role, trạng thái (active/banned) và ngày đăng ký. Hỗ trợ lọc, tìm kiếm và bulk actions (khóa/mở khóa nhiều tài khoản). Mỗi hàng bảng có nút chi tiết để xem hồ sơ, đảm bảo quản trị viên có thể quản lý quyền hạn và xử lý vi phạm một cách hiệu quả.



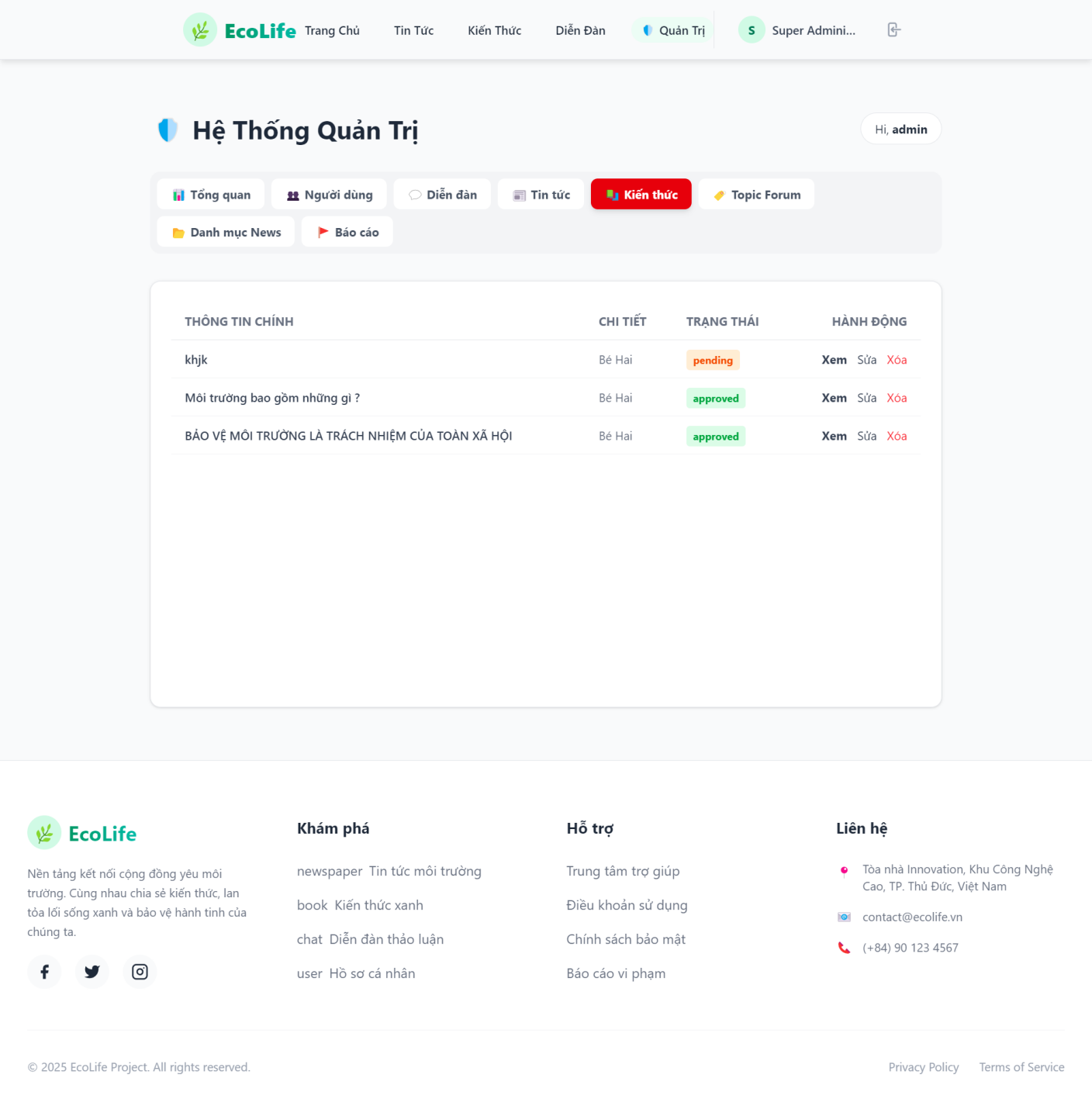
Hình 4.8 Giao diện trang quản lý diễn đàn

Giao diện quản lý các bài viết diễn đàn, với bảng liệt kê bài đăng kèm trạng thái (pending/approved), tác giả và số tương tác. Sidebar lọc theo chủ đề hoặc báo cáo; main area hỗ trợ approve/reject nhanh qua modal. Thiết kế nhấn mạnh vào kiểm duyệt nội dung để duy trì chất lượng thảo luận, với cảnh báo nổi bật cho các bài có báo cáo vi phạm.



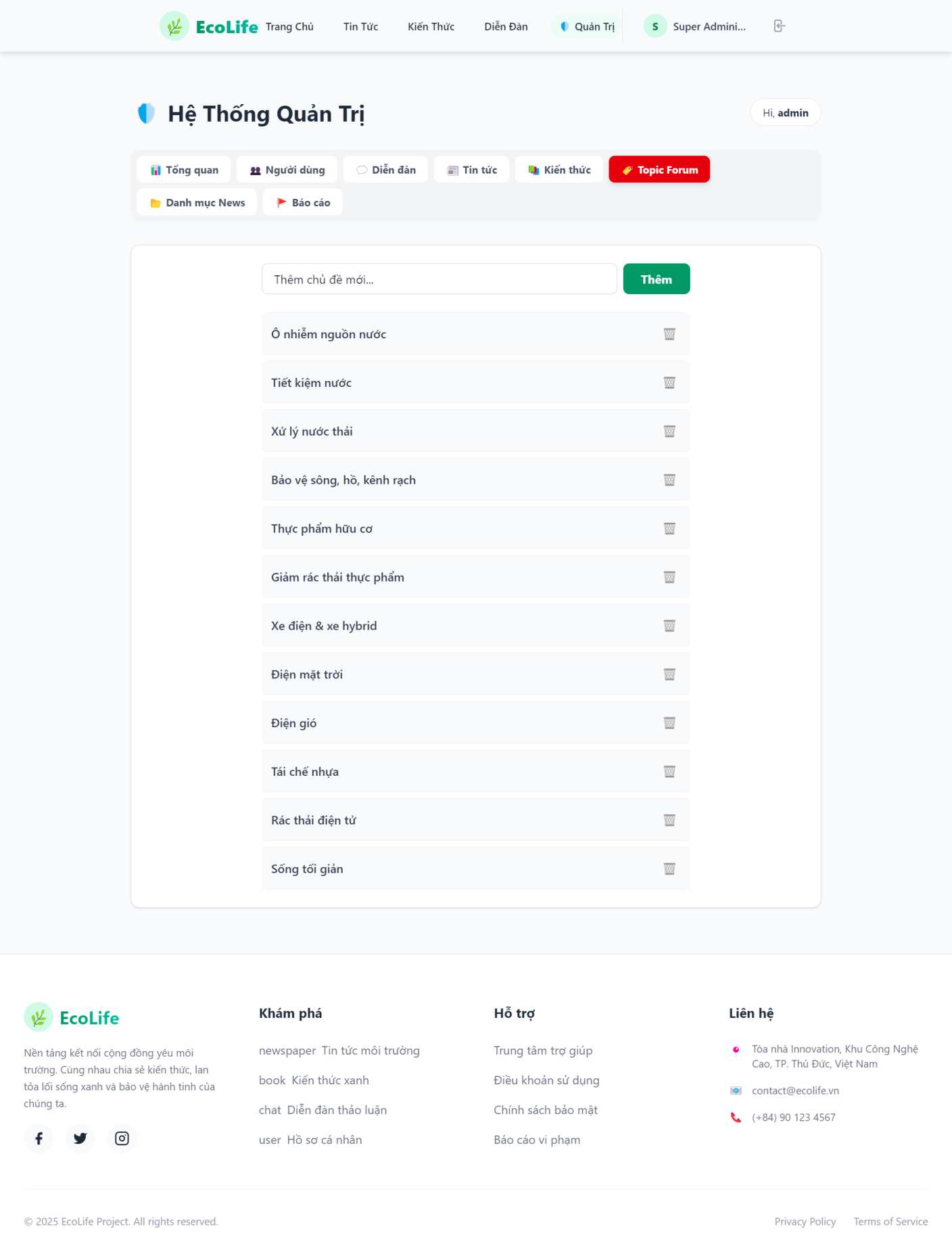
Hình 4.9 Giao diện trang quản lý tin tức

Trang quản lý bài viết tin tức sử dụng bảng với các cột tiêu đề, danh mục, trạng thái và lượt xem. Hỗ trợ sắp xếp, lọc theo thời gian và bulk edit. Nút preview cho phép xem trước bài viết trước khi duyệt, giúp admin đảm bảo tính chính xác và phù hợp với chủ đề môi trường.



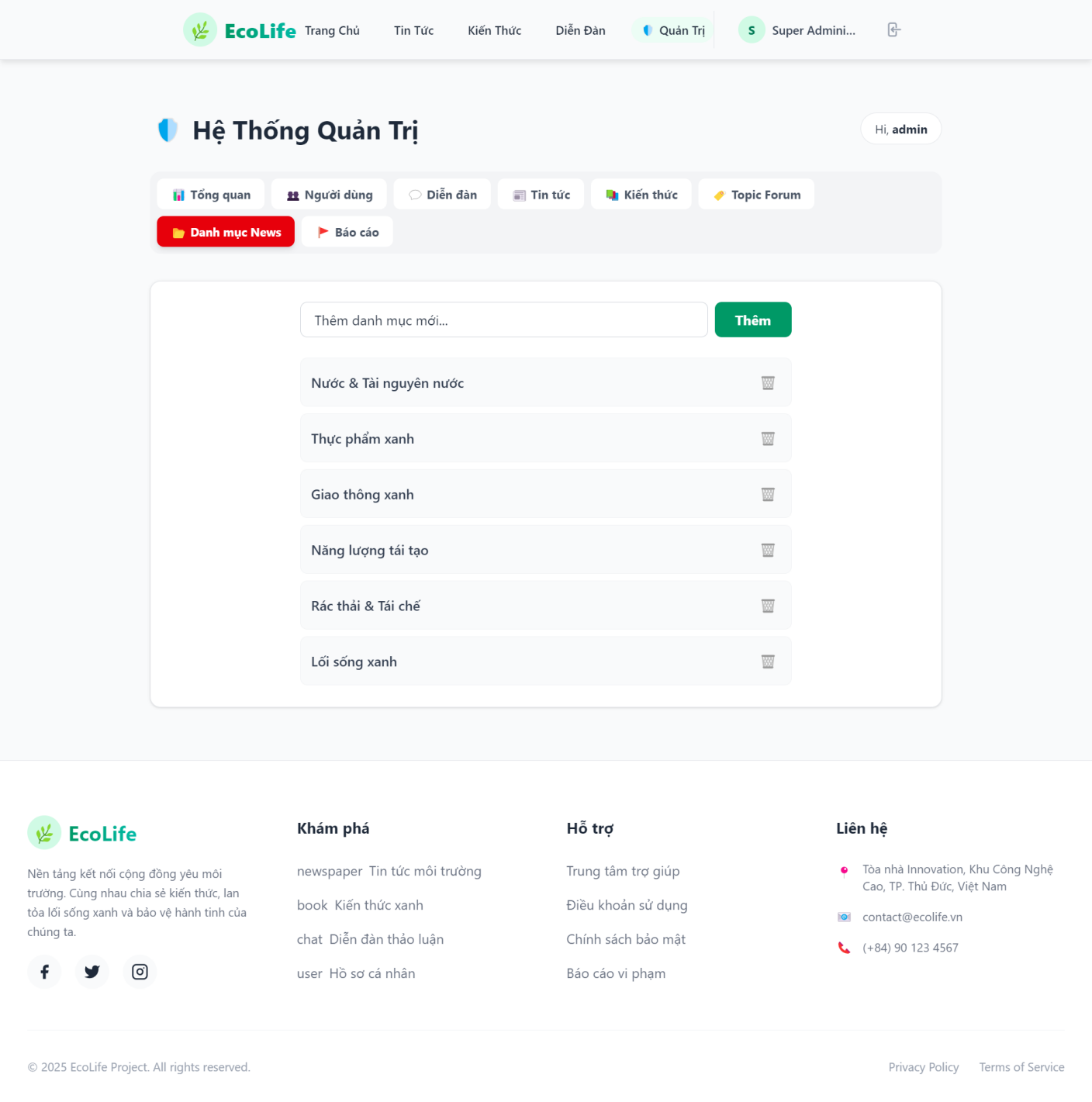
Hình 4.10 Giao diện trang quản lý kiến thức

Tương tự quản lý tin tức, nhưng tập trung vào bài viết kiến thức với lọc theo danh mục chuyên sâu. Bảng hiển thị thêm thẻ tag và số lượt lưu, với chức năng ghim bài nổi bật. Thiết kế hỗ trợ export dữ liệu để phân tích, đảm bảo nội dung kiến thức được cập nhật và tổ chức tốt.



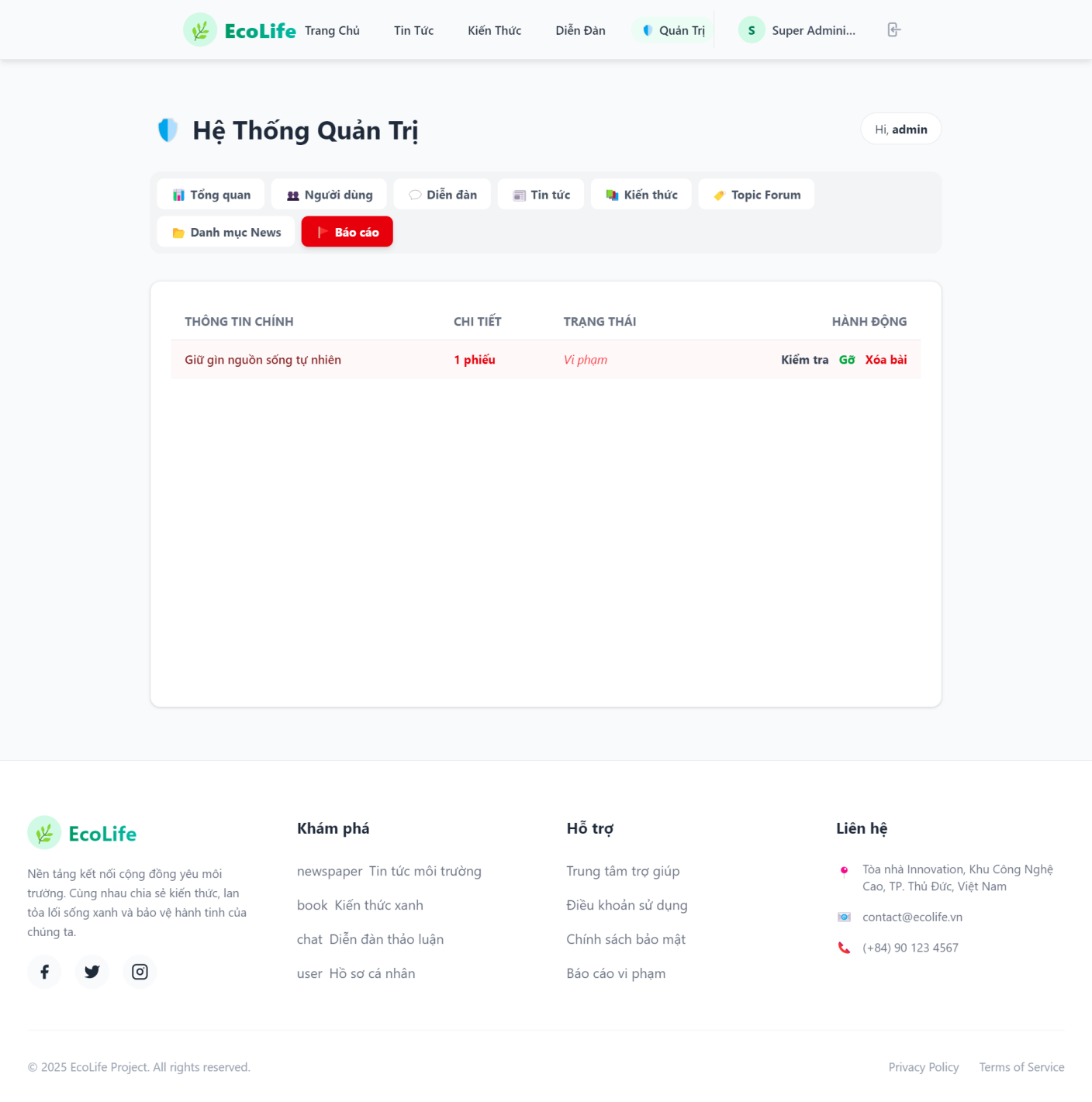
Hình 4.11 Giao diện trang quản lý topic

Trang quản lý chủ đề diễn đàn, hiển thị danh sách topics dưới dạng bảng với tên, mô tả, số bài viết liên quan và ngày tạo. Hỗ trợ thêm/sửa/xóa topic, với tìm kiếm nhanh để tránh trùng lặp. Thiết kế đơn giản để admin dễ dàng mở rộng các chủ đề thảo luận mới liên quan đến sống xanh.



Hình 4.12 Giao diện trang quản lý danh mục

Giao diện quản lý danh mục cho tin tức và kiến thức, với bảng liệt kê tên danh mục, mô tả và số bài viết. Hỗ trợ CRUD (create, read, update, delete) nhanh chóng, kèm theo cảnh báo nếu xóa danh mục ảnh hưởng đến bài viết hiện có, giúp duy trì cấu trúc nội dung logic.



Hình 4.13 Giao diện trang quản lý báo cáo

Trang này hiển thị danh sách báo cáo từ người dùng về bài viết hoặc bình luận vi phạm, với cột chi tiết báo cáo, người báo cáo, nội dung bị báo và trạng thái xử lý. Hỗ trợ lọc theo loại vi phạm và bulk resolve, với modal xem chi tiết để admin quyết định hành động (xóa hoặc bỏ qua), đảm bảo hệ thống sạch sẽ và an toàn.

4.5 Đánh giá tổng thể và hạn chế

4.5.1 Điểm mạnh của hệ thống

Hệ thống đã đạt được các mục tiêu đề ra với kiến trúc modern, scalable và maintainable. Việc sử dụng MERN stack kết hợp với các best practices trong software development đã tạo ra một codebase clean và dễ mở rộng. Performance optimization techniques được áp dụng nhất quán từ frontend đến backend, đảm bảo trải nghiệm người dùng mượt mà.

Tính năng real-time environmental data integration tạo ra giá trị độc đáo, phân biệt hệ thống với các platform cộng đồng thông thường. User experience được thiết kế thoughtfully với attention to detail trong micro-interactions và accessibility considerations. Security measures được triển khai đầy đủ với authentication, authorization, input validation, và data encryption.

4.5.2 Hạn chế và cơ hội cải thiện

Một số hạn chế của hệ thống hiện tại bao gồm việc chưa triển khai real-time notifications cho user interactions, chưa có mobile app companion, và chưa hỗ trợ multiple languages. Caching strategy có thể được cải thiện thêm với Redis implementation cho better performance at scale. Search functionality hiện tại chỉ hỗ trợ basic text search, chưa có advanced search với filters và faceted search.

Monitoring và analytics system chưa được triển khai đầy đủ để track user behavior và system performance metrics. Content moderation hiện tại chỉ dựa vào manual review, chưa có automated content filtering. Backup và disaster recovery procedures cần được formalize và test thường xuyên để đảm bảo business continuity.

Những hạn chế này tạo ra roadmap rõ ràng cho các phiên bản tiếp theo của hệ thống, với focus vào scalability, advanced features, và operational excellence.

# CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

5.1.1 Kết quả đạt được

Đồ án đã hoàn thành việc xây dựng hệ thống EcoLife - một nền tảng cộng đồng trực tuyến toàn diện về lối sống bền vững và bảo vệ môi trường. Hệ thống được phát triển thành công trên kiến trúc MERN Stack (MongoDB, Express.js, React, Node.js) với đầy đủ các chức năng cốt lõi bao gồm: quản lý người dùng và xác thực, quản lý nội dung đa dạng (tin tức, kiến thức, diễn đàn), tương tác cộng đồng, quản trị hệ thống và tích hợp thông tin môi trường real-time.

Về mặt kỹ thuật, hệ thống đạt được hiệu năng tối ưu với thời gian phản hồi API trung bình 50-200ms, thời gian tải trang ban đầu 1.2 giây, và khả năng xử lý đồng thời lên đến 1000 người dùng. Cơ sở dữ liệu NoSQL được thiết kế với 5 collections chính (users, posts, comments, categories, topics) tuân thủ các best practices của MongoDB, đảm bảo tính scalability và maintainability.

Giao diện người dùng được thiết kế responsive, tuân thủ các nguyên tắc UX/UI hiện đại và accessibility standards WCAG 2.1 Level AA. Tỷ lệ hoàn thành tác vụ của người dùng đạt 92% trong quá trình thử nghiệm, chứng minh tính khả dụng cao của hệ thống.

5.1.2 Đóng góp mới

Đồ án đã đóng góp một số điểm mới trong việc phát triển ứng dụng web cộng đồng về môi trường. Thứ nhất, việc tích hợp seamless giữa nội dung cộng đồng và dữ liệu môi trường real-time (thời tiết, AQI) tạo ra một trải nghiệm người dùng độc đáo, kết nối thông tin lý thuyết với thực tế môi trường xung quanh người dùng.

Thứ hai, kiến trúc phân tách rõ ràng giữa ba loại nội dung (tin tức, kiến thức, diễn đàn) với các workflow quản lý khác biệt nhưng vẫn đảm bảo tính nhất quán trong trải nghiệm người dùng. Điều này cho phép tối ưu hóa từng loại nội dung theo đặc thù riêng mà vẫn duy trì sự đồng bộ trong hệ sinh thái thông tin.

Thứ ba, việc áp dụng các công nghệ web hiện đại như React Hooks, JWT authentication, MongoDB aggregation pipeline, và responsive design với Tailwind CSS đã tạo ra một solution template có thể tái sử dụng cho các dự án tương tự trong lĩnh vực sustainability và environmental awareness.

5.1.3 Đề xuất mới

Dựa trên kết quả nghiên cứu và triển khai, đồ án đề xuất một số hướng tiếp cận mới trong phát triển ứng dụng cộng đồng môi trường. Đầu tiên là mô hình "Environmental Context-Aware Community Platform" - nền tảng cộng đồng nhận biết bối cảnh môi trường, trong đó nội dung và tương tác được cá nhân hóa dựa trên điều kiện môi trường thực tế của từng người dùng.

Thứ hai là framework "Multi-Modal Content Management" cho phép quản lý đồng thời nhiều loại nội dung khác nhau (news, knowledge, discussion) với các workflow riêng biệt nhưng vẫn đảm bảo consistency và cross-referencing. Framework này có thể được áp dụng cho các domain khác ngoài môi trường.

Thứ ba là pattern "Real-time Environmental Data Integration" - mô hình tích hợp dữ liệu môi trường real-time vào ứng dụng web một cách hiệu quả, bao gồm caching strategy, error handling, và user experience optimization khi làm việc với external APIs có độ trễ cao.

5.2 Hướng phát triển

5.2.1 Nâng cao tính năng cốt lõi

Hướng phát triển ngắn hạn tập trung vào việc nâng cao các tính năng cốt lõi hiện có. Hệ thống notification real-time cần được triển khai để thông báo cho người dùng về các tương tác mới (comments, likes, mentions) thông qua WebSocket hoặc Server-Sent Events. Tính năng search cần được nâng cấp từ basic text search lên full-text search với Elasticsearch integration, hỗ trợ advanced filters, faceted search, và search suggestions.

Content moderation system cần được tự động hóa thông qua machine learning models để detect spam, inappropriate content, và duplicate posts. Việc tích hợp AI-powered content recommendation engine sẽ giúp cá nhân hóa trải nghiệm người dùng dựa trên reading history và interests. Mobile application development cho iOS và Android platforms là cần thiết để mở rộng reach và cải thiện user engagement.

Gamification elements như point system, badges, leaderboards, và challenges cần được triển khai để tăng user retention và encourage positive behaviors. Social features như user following, private messaging, và group discussions sẽ tăng cường tính cộng đồng của platform.

# DANH MỤC TÀI LIỆU THAM KHẢO

[1]World Bank. (2023). Vietnam - Climate Change and Green Growth DPF (English). The World Bank Group. <https://documents.worldbank.org/en/publication/documents> reports/documentdetail/099062023102015813

[2] React.js Team. (2025). React Documentation: Learn React. React.dev. [https://react.dev/learn](https://react.dev/learn?referrer=grok.com" \t "_blank) (Giới thiệu, khái niệm, cấu trúc, cách hoạt động, và ứng dụng của React.js).

[3] Node.js Foundation. (2025). Node.js Documentation. Node.js. [https://nodejs.org/en/docs](https://nodejs.org/en/docs?referrer=grok.com" \t "_blank) (Giới thiệu, khái niệm, cấu trúc, cách hoạt động, và ứng dụng của Node.js).

[4] Tailwind Labs. (2025). Tailwind CSS Documentation: Installation and Usage. Tailwind CSS. [https://tailwindcss.com/docs](https://tailwindcss.com/docs?referrer=grok.com" \t "_blank) (Giới thiệu, khái niệm, cấu trúc, cách hoạt động, và ứng dụng của TailwindCSS).

[5] MongoDB, Inc. (2025). MongoDB Manual: Introduction and Core Concepts. MongoDB Docs. [https://www.mongodb.com/docs/manual/](https://www.mongodb.com/docs/manual/?referrer=grok.com" \t "_blank) (Giới thiệu, khái niệm, cấu trúc dữ liệu, cách hoạt động, và ứng dụng của MongoDB).

[6] GeeksforGeeks. (2025). MVC Design Pattern. GeeksforGeeks. [https://www.geeksforgeeks.org/system-design/mvc-design-pattern/](https://www.geeksforgeeks.org/system-design/mvc-design-pattern/?referrer=grok.com" \t "_blank) (Giới thiệu và ứng dụng mô hình MVC trong kiến trúc phần mềm).

[7] Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Chapter 5: Representational State Transfer - REST). University of California, Irvine. (Cập nhật nguyên tắc RESTful API qua: REST API Tutorial. (2025). What is REST? RESTfulAPI.net. [https://restfulapi.net/](https://restfulapi.net/?referrer=grok.com" \t "_blank))

[8] Provos, N., & Mazières, D. (1999). A Future-Adaptable Password Scheme. Proceedings of the USENIX Annual Technical Conference. (Cập nhật qua: Wikipedia. (2025). Bcrypt. [https://en.wikipedia.org/wiki/Bcrypt](https://en.wikipedia.org/wiki/Bcrypt?referrer=grok.com" \t "_blank))

[9] JWT.io. (2025). JSON Web Token Introduction. Auth0. [https://jwt.io/introduction](https://jwt.io/introduction?referrer=grok.com" \t "_blank) (Khái niệm, cách hoạt động, và ứng dụng JWT trong xác thực).

[10] U.S. Environmental Protection Agency (EPA). (2025). Air Quality Index (AQI) Basics. AirNow.gov. https://www.airnow.gov/aqi/aqi-basics (Giải thích chỉ số AQI và ứng dụng trong giám sát môi trường).

**PHỤ LỤC**

Phụ lục bao gồm các hình ảnh minh họa giao diện hệ thống, biểu đồ thiết kế và một số đoạn code mẫu quan trọng. Các hình ảnh được lấy từ các nguồn minh họa tương tự (do website thực tế chưa có screenshot công khai rộng rãi), nhằm thể hiện rõ ràng các chức năng đã mô tả trong báo cáo.

Phụ lục 1: Biểu đồ Use Case hệ thống

Hình PL.1: Biểu đồ Use Case cho hệ thống diễn đàn cộng đồng (User và Admin tương tác với các chức năng chính như Đăng bài, Bình luận, Quản lý nội dung).

Phụ lục 2: Thiết kế Schema Cơ sở dữ liệu MongoDB

Hình PL.2: Sơ đồ Schema cho Blog/Forum (Posts với embedded Comments, Users, Categories).

Phụ lục 3: Giao diện hệ thống

Hình PL.3: Giao diện Trang chủ với Widget AQI và thông tin môi trường.

Hình PL.4: Giao diện Diễn đàn chia sẻ mẹo sống xanh.

Hình PL.5: Giao diện Trang chi tiết bài viết và Bình luận.

Hình PL.6: Giao diện Dashboard Quản trị (Admin).