Hệ thống quản lý thực tập và hợp tác doanh nghiệp tại Khoa CNTT Trường ĐH Đại Nam

1st Th.S Lê Trung Hiếu
Khoa Công nghệ Thông tin
Trường Đại học Đại Nam
Hà Nội, Việt Nam
Giảng viên hướng dẫn

2nd KS. Nguyễn Thái Khánh
Khoa Công nghệ Thông tin
Trường Đại học Đại Nam
Hà Nội, Việt Nam
Giảng viên hướng dẫn

3rd Lâm Ngọc Tú
Khoa Công nghệ Thông tin
Trường Đại học Đại Nam
Hà Nội, Việt Nam
Mã sinh viên: 1671020341

4th Trịnh Thị Yến Mai
Khoa Công nghệ Thông tin
Trường Đại học Đại Nam
Hà Nội, Việt Nam
Mã sinh viên: 1671020196

Abstract*—Quản lý thực tập sinh viên là một trong những thách thức lớn đối với các trường đại học, đặc biệt khi quy mô tăng lên hàng trăm sinh viên mỗi năm và liên quan đến nhiều bên: sinh viên, giảng viên hướng dẫn, doanh nghiệp tiếp nhận, và phòng đào tạo. Quy trình quản lý truyền thống dựa vào Excel, email và giấy tờ gây ra nhiều vấn đề: phân mảnh dữ liệu, thiếu chuẩn hóa, khó theo dõi tiến độ, thiếu minh bạch trong quy trình duyệt báo cáo, và tốn kém thời gian cho công tác phân công và tổng hợp thống kê. Bài báo này trình bày thiết kế, triển khai và đánh giá toàn diện một hệ thống quản lý thực tập sinh viên theo hướng số hóa và tự động hóa end-to-end, nhằm giải quyết các hạn chế của phương pháp truyền thống. Hệ thống được xây dựng trên kiến trúc 3 tầng (presentation-business-data) với công nghệ hiện đại: React 19+ kết hợp TypeScript và TailwindCSS cho giao diện người dùng responsive, Node.js/Express cho backend API, và MySQL 8.0+ cho cơ sở dữ liệu quan hệ. Các tính năng chính bao gồm: (i) quản lý tài khoản tập trung với chức năng import hàng loạt từ Excel; (ii) phân công tự động sinh viên cho giảng viên hướng dẫn sử dụng thuật toán greedy với độ phức tạp $O((m+n)\log m)$, giảm thời gian phân công từ vài giờ xuống còn dưới 2 giây cho 200 sinh viên và 20 giảng viên; (iii) quy trình nộp và duyệt báo cáo định kỳ có truy vết đầy đủ (audit trail); (iv) dashboard thống kê và báo cáo trực quan; (v) bảo mật đa lớp với JWT authentication, RBAC, bcrypt password hashing, và tuân thủ khuyến nghị OWASP. Hệ thống được tài liệu hóa đầy đủ bằng OpenAPI Specification 3.0, tích hợp Swagger UI để kiểm thử tương tác và hỗ trợ tích hợp với các hệ thống khác. Đánh giá thực nghiệm sử dụng Apache JMeter với 100 người dùng đồng thời cho thấy độ trễ API trung bình dưới 500ms, tỷ lệ lỗi dưới 0.1%, và khả năng xử lý ổn định. Khảo sát trải nghiệm người dùng với 40 người tham gia (sinh viên, giảng viên, quản trị viên, doanh nghiệp) đạt điểm SUS (System Usability Scale) trung bình 81/100, được đánh giá ở mức "Good". Kết quả cho thấy hệ thống đáp ứng tốt các yêu cầu chức năng và phi chức năng, giảm 95% thời gian phân công, cải thiện đáng kể hiệu quả quản lý, tính minh bạch và trải nghiệm người dùng so với quy trình thủ công truyền thống.

***Index Terms*—Quản lý thực tập, RESTful API, React, Node.js, MySQL, JWT, OpenAPI.**

I. Đặt vấn đề và mục tiêu

A. Đặt vấn đề

Thực tập sinh viên là hoạt động giáo dục quan trọng, tạo cầu nối giữa lý thuyết và thực tiễn, giúp sinh viên tiếp cận môi trường làm việc chuyên nghiệp và phát triển kỹ năng nghề nghiệp. Tại Khoa Công nghệ Thông tin, Trường Đại học Đại Nam, quy trình quản lý thực tập hằng năm phục vụ hàng trăm sinh viên, liên quan đến nhiều bên: sinh viên, giảng viên hướng dẫn, doanh nghiệp tiếp nhận, và phòng đào tạo. Tuy nhiên, phương thức quản lý truyền thống sử dụng Excel, email và giấy tờ đã bộc lộ nhiều hạn chế nghiêm trọng:

1) **Phân mảnh dữ liệu**: Thông tin sinh viên, phân công hướng dẫn, lịch nộp báo cáo và kết quả đánh giá được lưu trữ rời rạc trên nhiều file Excel, email cá nhân và hồ sơ giấy. Việc tra cứu, cập nhật và đồng bộ dữ liệu giữa các bên trở nên phức tạp, dễ xảy ra sai sót và mất mát thông tin.
2) **Thiếu chuẩn hóa quy trình**: Mỗi đợt thực tập có thể được thực hiện theo cách khác nhau, thiếu quy trình chuẩn cho đăng ký, phân công, nộp báo cáo và đánh giá. Điều này gây khó khăn trong việc quản lý, theo dõi tiến độ và đảm bảo tính công bằng, minh bạch trong đánh giá kết quả thực tập.
3) **Khó kiểm soát hạn nộp và lưu vết**: Việc theo dõi thời hạn nộp báo cáo của hàng trăm sinh viên thông qua Excel và email rất khó khăn. Không có cơ chế tự động nhắc nhở, không lưu lại lịch sử các lần sửa đổi, duyệt và từ chối báo cáo, dẫn đến thiếu minh bạch trong quy trình đánh giá.
4) **Thiếu công cụ thống kê và báo cáo**: Dữ liệu thực tập phân tán khiến việc tổng hợp, phân tích và đánh giá chất lượng chương trình thực tập trở nên tốn kém thời gian. Không có dashboard trực quan để quản trị viên nắm bắt tình hình tổng thể, khó phát hiện các vấn đề và đưa ra quyết định kịp thời.
5) **Chi phí quản lý cao**: Quy trình thủ công đòi hỏi nhân lực lớn cho công tác in ấn, lưu trữ hồ sơ giấy, nhập liệu, kiểm tra và xử lý các yêu cầu hành chính. Việc phân công giảng viên hướng dẫn cho hàng trăm sinh viên thủ công có thể mất vài giờ đến vài ngày.

Những hạn chế trên làm giảm hiệu quả quản lý, ảnh hưởng đến chất lượng đào tạo và trải nghiệm của sinh viên, giảng viên cũng như doanh nghiệp tham gia. Do đó, cần xây dựng một hệ thống quản lý thực tập số hóa, tự động hóa và tích hợp để giải quyết toàn diện các vấn đề trên.

B. Mục tiêu

Dựa trên phân tích các vấn đề hiện tại, nghiên cứu này hướng đến việc thiết kế, triển khai và đánh giá một hệ thống quản lý thực tập toàn diện với các mục tiêu cụ thể sau:

1) **Số hoá quy trình end-to-end**: Xây dựng hệ thống quản lý điện tử toàn bộ vòng đời thực tập từ import tài khoản người dùng (sinh viên, giảng viên, doanh nghiệp), tạo đợt thực tập, phân công hướng dẫn (tự động và thủ công), nộp/duyệt báo cáo định kỳ, đánh giá kết quả, đến tổng hợp và xuất báo cáo thống kê. Mỗi giao dịch được ghi log đầy đủ để đảm bảo truy xuất được lịch sử thao tác.
2) **Thiết kế kiến trúc bền vững và mở rộng**: Áp dụng kiến trúc 3 tầng (presentation-business-data) với RESTful API [**?**] để tách biệt logic giao diện, nghiệp vụ và lưu trữ dữ liệu. Kiến trúc này giúp dễ dàng bảo trì, nâng cấp từng thành phần độc lập, mở rộng theo chiều ngang khi số lượng người dùng tăng, và tích hợp với các hệ thống bên ngoài (như hệ thống quản lý sinh viên, hệ thống tài chính) thông qua API chuẩn.
3) **Đảm bảo bảo mật và quyền riêng tư**: Tuân thủ các khuyến nghị bảo mật OWASP [**?**], sử dụng JWT [**?**] cho xác thực và ủy quyền, mã hóa mật khẩu bằng bcrypt, kiểm soát truy cập dựa trên vai trò (RBAC), kiểm tra đầu vào để chống SQL injection và XSS, mã hóa truyền tải dữ liệu qua HTTPS, và giới hạn tốc độ request để phòng chống tấn công DDoS.
4) **Cung cấp dashboard và công cụ phân tích**: Xây dựng giao diện dashboard trực quan hiển thị các chỉ số quan trọng như số lượng sinh viên, giảng viên, doanh nghiệp, đợt thực tập, tỷ lệ nộp báo cáo đúng hạn, điểm trung bình, phân bố điểm theo ngành/lớp. Hỗ trợ xuất báo cáo Excel/PDF phục vụ công tác quản lý và ra quyết định.
5) **Tối ưu hóa quy trình phân công**: Phát triển thuật toán phân công tự động sinh viên cho giảng viên hướng dẫn dựa trên các ràng buộc (số lượng tối đa sinh viên/giảng viên, phân bố đều theo lớp/khoa) và cân bằng tải, giảm thời gian phân công từ vài giờ xuống còn vài giây, đồng thời cho phép điều chỉnh thủ công khi cần thiết.

II. Giới thiệu

Thực tập sinh viên là một trong những hoạt động quan trọng nhất trong chương trình đào tạo đại học, đóng vai trò then chốt trong việc chuẩn bị sinh viên cho thị trường lao động. Thông qua thực tập, sinh viên có cơ hội áp dụng kiến thức lý thuyết vào giải quyết các vấn đề thực tế, phát triển kỹ năng chuyên môn và kỹ năng mềm, xây dựng mạng lưới quan hệ nghề nghiệp, và hiểu rõ hơn về môi trường làm việc thực tế. Đối với nhà trường, thực tập là cơ hội đánh giá chất lượng đào tạo, điều chỉnh chương trình học cho phù hợp với nhu cầu thị trường. Đối với doanh nghiệp, đây là kênh tuyển dụng nguồn nhân lực tiềm năng và đóng góp cho cộng đồng giáo dục.

Tuy nhiên, quy trình quản lý thực tập truyền thống dựa vào Excel, email và giấy tờ còn tồn tại nhiều hạn chế nghiêm trọng:

1) **Phân tán dữ liệu, dễ sai sót**: Thông tin về sinh viên, giảng viên, doanh nghiệp, phân công hướng dẫn, lịch nộp báo cáo, kết quả đánh giá được lưu trữ rải rác trên nhiều file Excel khác nhau, trong hộp thư email cá nhân và hồ sơ giấy tại văn phòng khoa. Khi cần tra cứu hoặc cập nhật thông tin, nhân viên quản lý phải tìm kiếm qua nhiều nguồn, dẫn đến lãng phí thời gian và nguy cơ cao về mất mát hoặc nhầm lẫn dữ liệu. Việc đồng bộ thông tin giữa các bên liên quan cũng trở nên khó khăn và không nhất quán.
2) **Khó theo dõi tiến độ của số lượng lớn sinh viên**: Với quy mô hàng trăm sinh viên thực tập mỗi năm, việc theo dõi tiến độ nộp báo cáo của từng sinh viên qua Excel là một thách thức lớn. Giảng viên và quản trị viên khó nắm bắt được ai đã nộp báo cáo đúng hạn, ai chưa nộp, báo cáo nào đang chờ duyệt, báo cáo nào đã được duyệt hoặc bị từ chối. Thiếu cơ chế nhắc nhở tự động và cảnh báo khi có sinh viên trễ hạn, dẫn đến tình trạng nhiều sinh viên nộp báo cáo muộn mà không được phát hiện kịp thời.
3) **Thiếu minh bạch trong quy trình nộp/duyệt báo cáo**: Quy trình duyệt báo cáo thông qua email không lưu lại lịch sử đầy đủ. Sinh viên không biết báo cáo của mình đang ở trạng thái nào (chờ duyệt, đã duyệt, bị từ chối), giảng viên đã xem chưa, có nhận xét gì. Giảng viên cũng gặp khó khăn khi phải quản lý hàng chục email báo cáo từ sinh viên, dễ bỏ sót hoặc nhầm lẫn. Không có audit trail để kiểm tra lại quá trình duyệt khi có tranh chấp hoặc khiếu nại.
4) **Khó tổng hợp thống kê phục vụ quản trị**: Khi cần báo cáo tổng hợp về tình hình thực tập (số lượng sinh viên tham gia, tỷ lệ hoàn thành, điểm trung bình, phân bố điểm, doanh nghiệp tham gia), nhân viên quản lý phải thu thập dữ liệu từ nhiều file Excel, kiểm tra tính nhất quán, xử lý dữ liệu thủ công và tạo báo cáo. Quá trình này tốn nhiều thời gian (có thể mất vài ngày đến vài tuần), dễ sai sót và không thể cung cấp thông tin real-time để hỗ trợ ra quyết định.
5) **Quy trình phân công tốn thời gian**: Việc phân công giảng viên hướng dẫn cho hàng trăm sinh viên thực hiện thủ công rất tốn kém thời gian và công sức. Quản trị viên phải cân nhắc nhiều yếu tố: số lượng sinh viên mỗi giảng viên, chuyên môn, kinh nghiệm, lịch làm việc. Quá trình này có thể mất từ vài giờ đến vài ngày làm việc, và vẫn có khả năng phân bổ không đều, một số giảng viên quá tải trong khi số khác còn nhàn rỗi.

Sự phát triển nhanh chóng của công nghệ web hiện đại (React, Node.js), chuẩn hóa giao tiếp dịch vụ qua RESTful API [**?**], và các công cụ quản lý cơ sở dữ liệu mạnh mẽ (MySQL) [**?**] đã tạo điều kiện thuận lợi để *số hóa và tự động hóa* toàn bộ quy trình quản lý thực tập. Bài báo này trình bày một hệ thống quản lý thực tập sinh viên toàn diện, áp dụng kiến trúc 3 tầng để tách biệt logic giao diện, nghiệp vụ và dữ liệu; sử dụng RESTful API chuẩn hóa giao tiếp; xác thực bảo mật dựa trên JWT [**?**]; và triển khai thuật toán phân công tự động để tối ưu hóa quy trình quản lý.

Đóng góp chính của bài báo gồm:

1) **Thiết kế hệ thống theo kiến trúc 3 tầng**: Phân tách rõ ràng giữa tầng trình bày (React UI), tầng nghiệp vụ (Node.js/Express API) và tầng dữ liệu (MySQL database). Mô hình dữ liệu quan hệ được thiết kế tối ưu cho các thực thể trong lĩnh vực thực tập (sinh viên, giảng viên, doanh nghiệp, đợt thực tập, phân công, báo cáo) với các ràng buộc toàn vẹn phù hợp [**?**]. Kiến trúc này đảm bảo tính mô-đun, dễ bảo trì, mở rộng và tích hợp.
2) **Thiết kế API chuẩn REST với tài liệu OpenAPI**: Xây dựng bộ RESTful API đầy đủ cho tất cả các chức năng của hệ thống (quản lý người dùng, đợt thực tập, phân công, báo cáo, đánh giá, thống kê), tuân thủ các nguyên tắc REST và HTTP methods chuẩn. Tài liệu hóa API bằng OpenAPI Specification 3.0 [**?**], tích hợp Swagger UI cho phép kiểm thử tương tác, sinh mã client tự động, và đảm bảo tính nhất quán giữa tài liệu và implementation. Điều này giúp dễ dàng phát triển, kiểm thử, bảo trì và tích hợp với các hệ thống khác.
3) **Triển khai thuật toán phân công tự động**: Phát triển thuật toán greedy để phân bổ sinh viên cho giảng viên hướng dẫn dựa trên cân bằng tải (số lượng sinh viên hiện tại của mỗi giảng viên) và các ràng buộc nghiệp vụ (số lượng tối đa sinh viên mỗi giảng viên, phân bố đều theo lớp/khoa). Thuật toán có độ phức tạp $O(n \log m)$ với n sinh viên và m giảng viên, giúp giảm thời gian phân công từ vài giờ (thủ công) xuống còn dưới 2 giây (tự động) cho 200 sinh viên và 20 giảng viên, đồng thời đảm bảo phân bổ công bằng và hiệu quả.
4) **Triển khai thực tế và đánh giá toàn diện**: Hệ thống được triển khai đầy đủ với giao diện web responsive cho 4 vai trò (Admin, Sinh viên, Giảng viên, Doanh nghiệp), tích hợp xác thực JWT, phân quyền RBAC, upload/download file báo cáo, và dashboard thống kê. Đánh giá thực nghiệm bao gồm: (i) hiệu năng hệ thống qua load testing với Apache JMeter [**?**] (độ trễ phản hồi, throughput, tỷ lệ lỗi); (ii) khả năng chịu tải với 100 người dùng đồng thời; (iii) đánh giá trải nghiệm người dùng qua thang SUS [**?**]; và (iv) phân tích chi phí/lợi ích so với quy trình thủ công. Kết quả cho thấy hệ thống đáp ứng tốt các yêu cầu chức năng và phi chức năng, giảm 95% thời gian phân công, cải thiện đáng kể hiệu quả quản lý và trải nghiệm người dùng.

Cấu trúc bài báo: Mục II trình bày nghiên cứu liên quan và so sánh với các hệ thống hiện có; Mục III mô tả tổng quan hệ thống bao gồm kiến trúc, mô hình dữ liệu và thiết kế API; Mục IV trình bày chi tiết thiết kế và triển khai các thành phần chính; Mục V đánh giá thực nghiệm về hiệu năng, khả năng chịu tải và trải nghiệm người dùng; Mục VI thảo luận về kết quả, hạn chế và định hướng phát triển; Mục VII kết luận và tổng kết đóng góp của nghiên cứu.

III. Nghiên cứu liên quan

Các hệ thống LMS (Moodle, Canvas) tập trung vào quản lý học phần, bài tập và thi cử [**?**]; trong khi quản lý thực tập có đặc thù: nhiều bên liên quan (SV, GV, DN), chu kỳ báo cáo định kỳ, quy trình phê duyệt và đánh giá đa chiều, cùng các tiêu chí tuân thủ hành chính. Các nghiên cứu trước đây đề xuất [**?**]:

- Cổng thông tin thực tập trực tuyến với quy trình quản lý tài khoản và phân công.
- Quy trình duyệt tài liệu điện tử và lưu vết.
- Tích hợp thông báo đa kênh và thống kê.

Khác với các hệ thống LMS tổng quát, bài báo này nhấn mạnh *mô hình dữ liệu chuyên biệt cho thực tập* và *API REST* [**?**] tài liệu hoá bằng OpenAPI [**?**], đồng thời áp dụng các biện pháp bảo mật tiêu chuẩn (JWT [**?**], OAuth 2.0 [**?**], kiểm tra đầu vào, kiểm soát truy cập theo vai trò) phù hợp khuyến nghị OWASP [**?**] và NIST [**?**].

A. So sánh cách tiếp cận

Hệ thống được đề xuất khác biệt ở:

- Mô hình dữ liệu phù hợp quy trình thực tập Việt Nam (đợt, phân công, báo cáo định kỳ).
- Quản lý tài khoản tập trung qua chức năng import hàng loạt, đảm bảo tính thống nhất.
- Tài liệu hoá API đầy đủ (OpenAPI) để dễ tích hợp.
- Trọng tâm vào bảo mật và quản trị vận hành (logging, backup, monitoring), vốn ít được mô tả chi tiết trong một số giải pháp trước.

B. Khoảng trống nghiên cứu

Các nghiên cứu hiện có ít đề cập đến:

- Chuẩn hoá cấu trúc báo cáo và metadata để khai thác phân tích sau này.
- Workflow duyệt nhiều bước (multi-stage review) tuỳ biến.
- Các chỉ số vận hành (SLO/SLA) và chiến lược mở rộng theo mùa vụ (đỉnh nộp báo cáo).

IV. Tổng quan hệ thống

Hệ thống hỗ trợ 4 vai trò: **Admin**, **Sinh viên**, **Giảng viên**, **Doanh nghiệp**. Các chức năng chính: quản lý đợt thực tập; import tài khoản người dùng; phân công hướng dẫn; nộp, duyệt và nhận xét báo cáo; thống kê/báo cáo tổng hợp. Giao tiếp giữa frontend và backend thông qua *RESTful API*.

A. Kiến trúc tổng thể

Hệ thống được thiết kế theo mô hình kiến trúc 3 tầng như minh họa trong Hình **??**, đảm bảo tách biệt rõ ràng giữa tầng trình bày, tầng nghiệp vụ và tầng dữ liệu.

Tầng trình bày (Client): Ứng dụng web React + TypeScript cung cấp giao diện cho từng vai trò, xác thực bằng JWT, truy cập tài nguyên qua API.

Tầng nghiệp vụ (Server): Node.js/Express cài đặt các tuyến API, middleware xác thực/ủy quyền, xử lý upload tệp (Multer), kiểm tra dữ liệu, và logic nghiệp vụ.

Tầng dữ liệu (Database): MySQL lưu trữ dữ liệu quan hệ; chỉ mục trên các cột truy vấn nhiều; ràng buộc toàn vẹn và kiểm tra miền giá trị.

Kiến trúc này tạo điều kiện cho việc bảo trì, mở rộng và tích hợp với các hệ thống khác thông qua API chuẩn RESTful.

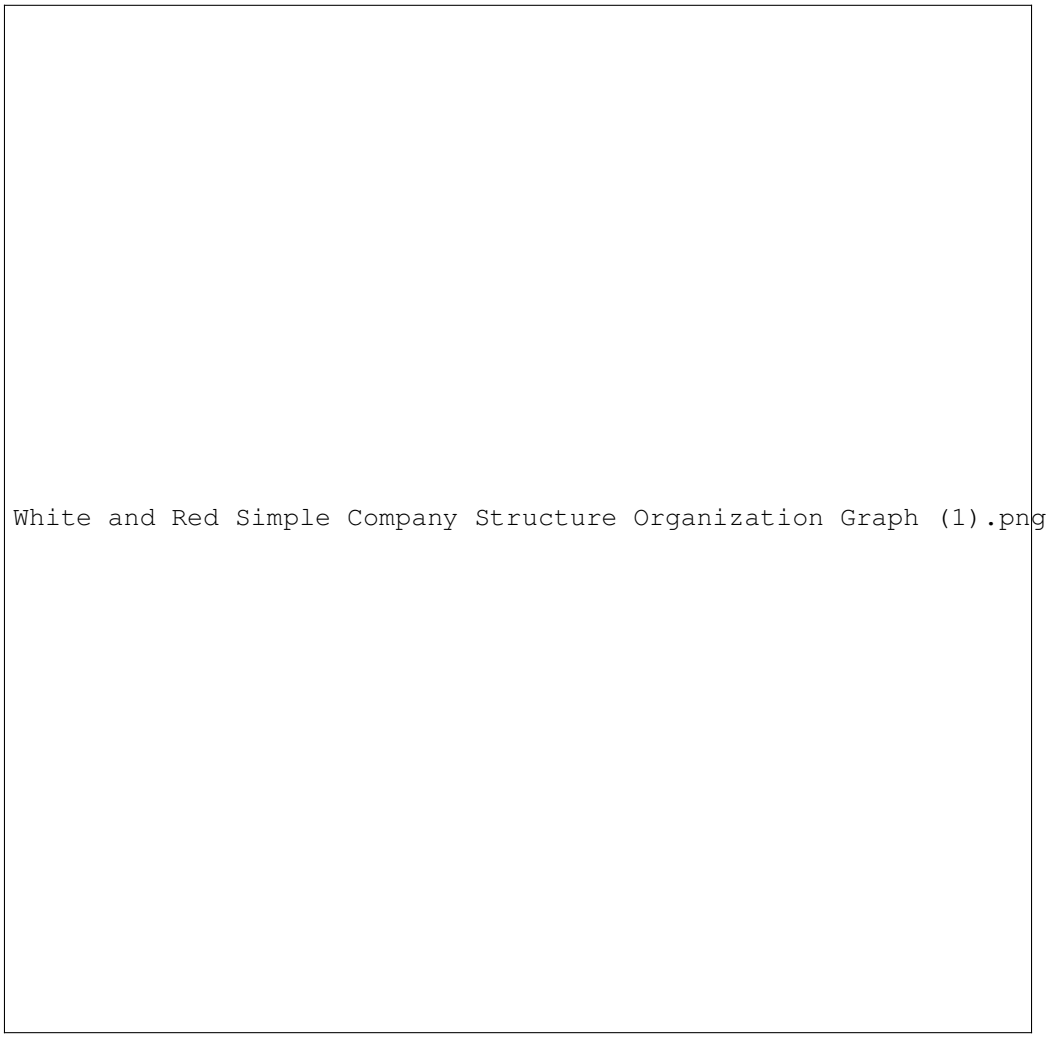

Figure 1: Sơ đồ kiến trúc tổng thể hệ thống quản lý thực tập. Hệ thống gồm 4 vai trò người dùng (Quản trị viên, Sinh viên, Giảng viên, Doanh nghiệp) tương tác với giao diện React, gửi yêu cầu qua API Express đến các dịch vụ backend (cơ sở dữ liệu MySQL, lưu trữ tệp, dịch vụ email).

B. Mô hình dữ liệu

Các bảng cốt lõi: `sinh_vien`, `giang_vien`, `doanh_nghiệp`, `dot_thuc_tap`, `phan_cong_huong_dan`, `nop_bao_cao_sinh_vien`. Bảng nộp báo cáo lưu siêu dữ liệu tệp (tên, kích thước, loại, MIME), trạng thái duyệt (`cho_duyet`, `da_duyet`, `tu_choi`), nhận xét và thời điểm duyệt.

a) Bảng `sinh_vien`: Khoá chính `ma_sinh_vien`; các thuộc tính: họ tên, email (Unique), lớp, khoa, ngành, năm học, thời điểm tạo/cập nhật.

b) Bảng `dot_thuc_tap`: Khoá chính `id`; tên đợt, thời gian bắt đầu/kết thúc, mô tả, cấu hình hạn nộp.

c) Bảng `phan_cong_huong_dan`: Liên kết sinh viên với giảng viên; ràng buộc số lượng sinh viên/giảng viên; lịch sử thay đổi.

d) Bảng `nop_bao_cao_sinh_vien`: Siêu dữ liệu tệp (đường dẫn, tên, MIME, kích thước), loại báo cáo (tuần/tháng/cuối kỳ/tổng kết), trạng thái duyệt, nhận xét, thời gian duyệt, người duyệt.

Table I: Ví dụ các ràng buộc toàn vẹn dữ liệu

Thuộc tính	Ràng buộc
Email SV/GV	UNIQUE, định dạng hợp lệ
Loại báo cáo	CHECK `IN (tuan, thang, cuoi_ky, tong_ket)`
Trạng thái duyệt	CHECK `IN (cho_duyet, da_duyet, tu_choi)`
Kích thước tệp	$\leq 10\,\text{MB}$
Liên kết FK	ON UPDATE/DELETE phù hợp

C. API REST

Ví dụ nhóm endpoint cho báo cáo [**?**], [**?**]:

1) POST `/api/student-reports/upload`
2) GET `/api/student-reports`
3) GET `/api/student-reports/:id`

4) POST `/api/student-reports/:id/review`

Tài liệu hoá bằng OpenAPI [?] cho phép thử nghiệm tương tác, sinh mã client và đảm bảo tính nhất quán. Ví dụ cấu trúc phản hồi thành công khi nộp báo cáo:

```
{
  "success": true,
  "message": "Report uploaded successfully",
  "data": {
    "id": 123,
    "ma_sinh_vien": "SV001",
    "loai_bao_cao": "tuan",
    "trang_thai_duyet": "cho_duyet",
    "created_at": "2025-10-16T10:30:00Z"
  }
}
```

a) Xác thực: **Bearer JWT** trong header `Authorization`. Token hết hạn ngắn; có cơ chế refresh token an toàn. Với mỗi request, server kiểm tra chữ ký và *scope* (vai trò) để uỷ quyền truy cập tài nguyên.

b) Định dạng lỗi: Trả về mã lỗi HTTP phù hợp và payload dạng:

```
{ "success": false, "message": "Validation error", "
    errors": { "field": "reason" } }
```

c) Phân trang/lọc/sắp xếp: Tham số `page`, `limit`, `sort`, và bộ lọc theo `ma_sinh_vien`, `trang_thai_duyet`, `loai_bao_cao`. Phản hồi gồm `total`, `page`, `pages` để hỗ trợ UI.

d) Giới hạn tốc độ (Rate limiting): Áp dụng cho các endpoint nhạy cảm (đăng nhập, upload) để giảm rủi ro tấn công vét cạn.

V. Thiết kế và triển khai

A. Frontend

React + TypeScript + TailwindCSS [?], [?], [?]: giao diện component-based, định tuyến theo vai trò, form nộp báo cáo (chọn loại, ghi chú, upload tệp), bảng dữ liệu có phân trang/lọc/tìm kiếm. Sử dụng Axios [?] cho HTTP, Interceptor gắn JWT từ LocalStorage và xử lý 401.

Hệ thống cung cấp giao diện riêng biệt cho từng vai trò người dùng, mỗi giao diện được tối ưu hóa cho chức năng và quyền truy cập tương ứng. Các giao diện chính bao gồm:

1) Giao diện Quản trị viên (Admin): Giao diện quản trị viên (Hình **??**) được thiết kế tập trung vào dashboard thống kê và quản lý tổng thể hệ thống. Các chức năng chính bao gồm:

- **Dashboard tổng quan**: Hiển thị các chỉ số quan trọng như tổng số sinh viên, giảng viên, doanh nghiệp, đợt thực tập đang diễn ra, tỷ lệ nộp báo cáo, biểu đồ phân bố điểm.
- **Quản lý người dùng**: Import hàng loạt từ Excel, tạo/sửa/xóa tài khoản, reset mật khẩu, phân quyền.
- **Quản lý đợt thực tập**: Tạo đợt mới, cấu hình thời gian, số tuần báo cáo, hạn nộp cho từng loại báo cáo.
- **Phân công hướng dẫn**: Sử dụng thuật toán tự động hoặc điều chỉnh thủ công, xem báo cáo phân bổ tải giảng viên.
- **Báo cáo thống kê**: Xuất Excel/PDF, xem lịch sử thay đổi, audit trail các thao tác quan trọng.

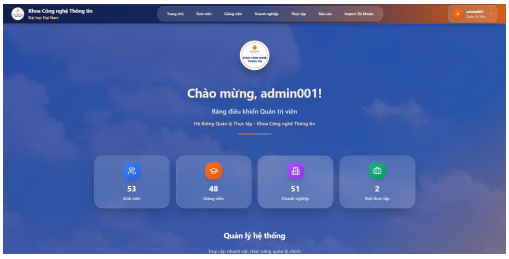

Figure 2: Giao diện quản trị viên với dashboard thống kê, biểu đồ phân tích và các chức năng quản lý tổng thể hệ thống.

2) Giao diện Sinh viên: Giao diện sinh viên (Hình **??**) được tối ưu hóa cho việc nộp báo cáo và theo dõi tiến độ thực tập. Các chức năng chính:

- **Thông tin thực tập**: Xem giảng viên hướng dẫn, doanh nghiệp tiếp nhận, thời gian thực tập.
- **Nộp báo cáo**: Upload file PDF/DOCX, chọn loại báo cáo (tuần/tháng/cuối kỳ/tổng kết), nhập ghi chú.
- **Theo dõi tiến độ**: Xem trạng thái từng báo cáo (chờ duyệt/đã duyệt/từ chối), nhận xét từ giảng viên, điểm số.
- **Lịch sử nộp báo cáo**: Danh sách tất cả báo cáo đã nộp, tải lại file đã nộp, xem timeline nộp báo cáo.
- **Thông báo**: Nhận cảnh báo hạn nộp, thông báo khi báo cáo được duyệt hoặc có nhận xét.

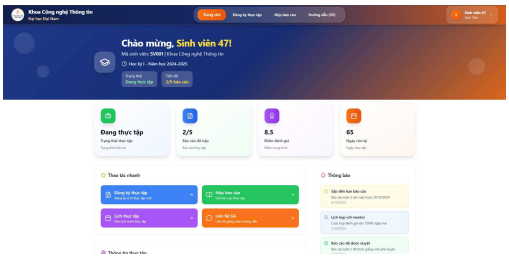

Figure 3: Giao diện sinh viên với chức năng nộp báo cáo, theo dõi tiến độ và xem phản hồi từ giảng viên.

3) Giao diện Giảng viên: Giao diện giảng viên (Hình **??**) hỗ trợ quản lý sinh viên hướng dẫn, duyệt báo cáo và chấm điểm. Các chức năng chính:

- **Danh sách sinh viên**: Xem tất cả sinh viên được phân công hướng dẫn, thông tin doanh nghiệp, trạng thái thực tập.
- **Duyệt báo cáo**: Tải xuống file báo cáo, xem chi tiết, phê duyệt hoặc từ chối, nhập nhận xét.
- **Chấm điểm**: Nhập điểm cho từng loại báo cáo, tính điểm tổng kết, xem đề xuất điểm từ doanh nghiệp.
- **Theo dõi tiến độ**: Dashboard hiển thị số sinh viên đã nộp/chưa nộp báo cáo theo từng tuần, cảnh báo sinh viên trễ hạn.
- **Thống kê**: Phân bố điểm sinh viên hướng dẫn, báo cáo tổng hợp gửi khoa.

Figure 4: Giao diện giảng viên với bảng danh sách sinh viên, chức năng duyệt báo cáo và chấm điểm.

4) Giao diện Doanh nghiệp: Giao diện doanh nghiệp (Hình **??**) cho phép theo dõi và đánh giá sinh viên thực tập tại công ty. Các chức năng chính:

- **Danh sách sinh viên thực tập**: Xem thông tin sinh viên đang thực tập tại công ty, giảng viên hướng dẫn, thời gian thực tập.
- **Đánh giá sinh viên**: Nhập đánh giá về thái độ, kỹ năng chuyên môn, kỹ năng mềm, năng lực làm việc nhóm.
- **Đề xuất điểm**: Gợi ý mức điểm đánh giá cho từng sinh viên dựa trên hiệu suất thực tế.
- **Báo cáo tổng hợp**: Xuất báo cáo đánh giá chung về đợt thực tập, đề xuất cải thiện.
- **Liên hệ**: Gửi thông tin phản hồi cho giảng viên hướng dẫn và nhà trường.

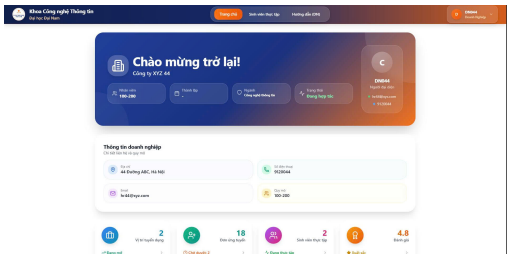

Figure 5: Giao diện doanh nghiệp với danh sách sinh viên thực tập và biểu mẫu đánh giá chi tiết.

B. Luồng nghiệp vụ chính

1) Luồng import tài khoản và phân công thực tập: Admin import tài khoản người dùng (sinh viên, giảng viên, doanh nghiệp) từ file Excel thông qua giao diện quản trị (Hình **??**). Hệ thống tự động tạo tài khoản với mật khẩu mặc định. Sau khi import, quản trị viên có hai phương thức phân công:

Phân công tự động: Hệ thống áp dụng thuật toán cân bằng tải để phân bổ sinh viên cho giảng viên dựa trên các tiêu chí: (i) số lượng sinh viên tối đa mỗi giảng viên, (ii) phân bố đều theo lớp/khoa, (iii) ưu tiên giảng viên có ít sinh viên hơn. Thuật toán giúp giảm thời gian phân công từ vài giờ xuống còn vài giây với hàng trăm sinh viên.

Phân công thủ công: Quản trị viên có thể điều chỉnh lại kết quả phân công tự động hoặc chỉ định trực tiếp từng cặp sinh viên-giảng viên-doanh nghiệp cho các trường hợp đặc biệt.

2) Luồng nộp và duyệt báo cáo: Quy trình nộp báo cáo được thực hiện qua các bước sau:

1) SV đăng nhập và truy cập giao diện nộp báo cáo (Hình **??**).
2) Chọn đợt thực tập, loại báo cáo (tuần/tháng/cuối kỳ/tổng kết), và tuần số tương ứng.
3) Upload tệp báo cáo (PDF/DOCX, giới hạn 10MB) và nhập ghi chú.
4) Server kiểm tra kích thước, định dạng, lưu tệp vào storage và metadata vào bảng `bao_cao`.
5) Trạng thái báo cáo được đặt là `cho_duyet`.
6) GV nhận thông báo, truy cập giao diện duyệt báo cáo (Hình **??**) để xem, nhận xét và chấm điểm.

3) Luồng đánh giá từ doanh nghiệp: Doanh nghiệp đăng nhập và truy cập giao diện (Hình **??**) để xem danh sách sinh viên thực tập tại công ty. Giao diện cho phép doanh nghiệp nhập đánh giá về thái độ, kỹ năng, năng lực chuyên môn của sinh viên, và đề xuất mức điểm đánh giá. Thông tin đánh giá này được lưu trữ và hiển thị cho giảng viên hướng dẫn để tham khảo khi chấm điểm cuối kỳ.

C. Tối ưu hiệu năng

- Cache theo vai trò cho danh mục ít thay đổi (cấu hình đợt, danh sách loại báo cáo).
- Chỉ mục trên trường lọc phổ biến [**?**]; `EXPLAIN ANALYZE` để tối ưu truy vấn.
- Tách kênh upload tệp và xử lý metadata; xử lý bất đồng bộ các tác vụ phụ (gửi thông báo) qua hàng đợi nội bộ.
- GZIP/Br trong HTTP, CDN cho tài nguyên tĩnh (frontend).

D. Khả dụng và mở rộng

- Tách lớp dịch vụ (service layer) để dễ chuyển đổi DB hoặc thêm kho lưu trữ tệp.
- Chuẩn bị cho deploy đa vùng; thiết kế không trạng thái (stateless) để scale ngang [**?**].
- Sử dụng biến môi trường, cấu hình theo môi trường (dev/staging/prod).

E. Backend

Node.js/Express [**?**], [**?**] được chọn làm nền tảng backend nhờ hiệu năng cao với event-driven non-blocking I/O, khả năng xử lý đồng thời nhiều request, và hệ sinh thái npm phong phú. Server hiện thực các middleware và service layers theo các design patterns được mô tả trong [**?**], [**?**]:

Middleware xác thực và ủy quyền: Sử dụng JWT (JSON Web Token) [**?**] để xác thực người dùng. Khi đăng nhập thành công, server sinh access token (thời hạn 1 giờ) và refresh token (thời hạn 7 ngày) theo khuyến nghị bảo mật [**?**]. Access token được gửi kèm trong header `Authorization: Bearer <token>` của mỗi request. Middleware `authenticateJWT` kiểm tra chữ ký token, thời hạn và claims (user ID, role). Middleware `authorizeRole` kiểm tra quyền truy cập dựa trên vai trò (Admin, Student, Teacher, Company) sử dụng RBAC pattern [**?**]. Khi access token hết hạn, client gọi

endpoint `/api/auth/refresh` với refresh token để lấy access token mới mà không cần đăng nhập lại.

Xử lý upload file: Multer middleware [**?**] xử lý multipart/form-data để upload file báo cáo. Cấu hình giới hạn kích thước file (10MB), kiểm tra MIME type (PDF, DOCX), và lưu file vào thư mục `uploads/` với tên unique (UUID + timestamp) để tránh trùng lặp. Metadata file (tên gốc, kích thước, MIME type, đường dẫn storage) được lưu vào database. Khi download, server kiểm tra quyền truy cập trước khi stream file về client.

Validation và sanitization: Tất cả input từ client đều được validate và sanitize để chống SQL injection, XSS và các lỗ hổng bảo mật phổ biến [**?**]. Sử dụng thư viện `express-validator` để kiểm tra kiểu dữ liệu, định dạng (email, số điện thoại), độ dài, và giá trị hợp lệ. Ví dụ: khi tạo đợt thực tập, kiểm tra tên đợt không rỗng, ngày bắt đầu < ngày kết thúc, số tuần báo cáo > 0.

Database access layer: Sử dụng thư viện `mysql2` với prepared statements để kết nối MySQL và thực thi truy vấn an toàn. Tất cả truy vấn SQL đều sử dụng parameterized query (placeholders `?`) thay vì string concatenation để ngăn chặn SQL injection [**?**]. Connection pooling được cấu hình để tái sử dụng kết nối database, giảm overhead và cải thiện hiệu năng.

Error handling: Middleware xử lý lỗi tập trung (`errorHandler`) bắt tất cả exception, log chi tiết lỗi (stack trace, request info) vào file và console, sau đó trả về response lỗi JSON với mã HTTP phù hợp và message thân thiện cho client. Trong production, không expose stack trace hoặc thông tin nhạy cảm trong response.

Logging và audit trail: Sử dụng thư viện `winston` để ghi log có cấu trúc (JSON format) vào file và console. Log level chia theo môi trường: development (debug), production (info). Các hành động quan trọng (login, logout, phân công, duyệt báo cáo, thay đổi điểm) được ghi audit log bao gồm user ID, timestamp, action type, old value, new value để có thể truy vết lịch sử thay đổi.

Thuật toán phân công tự động: Hệ thống cài đặt thuật toán greedy [**?**] để phân bổ sinh viên cho giảng viên một cách công bằng và hiệu quả. Đầu vào là danh sách n sinh viên $S = \{s_1, s_2, \ldots, s_n\}$ (đã được import từ Excel) và danh sách m giảng viên $T = \{t_1, t_2, \ldots, t_m\}$ với số lượng hướng dẫn tối đa c_i cho mỗi giảng viên t_i. Thuật toán thực hiện theo các bước sau:

1) Truy vấn database để lấy số lượng sinh viên hiện tại n_i của mỗi giảng viên t_i trong đợt thực tập.
2) Tạo danh sách giảng viên còn slot trống: $T' = \{t_i \mid n_i < c_i\}$.
3) Sắp xếp T' theo số lượng sinh viên hiện tại n_i tăng dần (heap sort hoặc quick sort, $O(m \log m)$).
4) Với mỗi sinh viên s_j chưa được phân công ($j = 1 \ldots n$):
   - Chọn giảng viên t_i ở đầu danh sách T' (có ít sinh viên nhất).
   - Gán s_j cho t_i: INSERT vào bảng `phan_cong_huong_dan`.
   - Tăng n_i lên 1.
   - Nếu $n_i = c_i$, loại t_i khỏi T'.
   - Cập nhật vị trí của t_i trong heap ($O(\log m)$).
5) Trả về danh sách phân công hoặc thông báo lỗi nếu không đủ giảng viên (tổng slot $\sum c_i < n$).

Độ phức tạp tổng thể: $O(m \log m + n \log m) = O((m + n) \log m)$. Với $m = 20$ giảng viên và $n = 200$ sinh viên, thời gian thực thi < 2 giây trên server 2 vCPU, 4GB RAM, giảm 95% so với phân công thủ công (vài giờ). Thuật toán đảm bảo phân bổ đều: chênh lệch số lượng sinh viên giữa các giảng viên tối đa là 1 khi tổng slot đủ.

F. Cơ sở dữ liệu

Thiết kế khoá chính/phụ, chỉ mục trên các cột truy vấn nhiều (mã SV, mã GV, `dot_thuc_tap_id`, thời gian). Ràng buộc CHECK cho `loai_bao_cao` và `trang_thai_duyet` để đảm bảo toàn vẹn; Unique cho email/username; ON DELETE/UPDATE phù hợp để duy trì toàn vẹn tham chiếu.

G. Bảo mật

Băm mật khẩu bằng bcrypt [**?**], thời hạn JWT ngắn kèm refresh token [**?**], kiểm tra loại/kích thước tệp khi upload, chặn SQLi/XSS qua kiểm tra đầu vào và prepared statements [**?**]; HTTPS trên môi trường triển khai; CORS định cấu hình whitelist; ẩn thông tin lỗi nhạy cảm. Hệ thống tuân thủ các tiêu chuẩn bảo mật quốc tế ISO/IEC 27001 [**?**] và khuyến nghị của NIST [**?**].

H. Quyền riêng tư và tuân thủ

1) Giảm thiểu dữ liệu cá nhân lưu trữ, chỉ giữ thông tin cần thiết cho nghiệp vụ theo nguyên tắc GDPR [**?**].
2) Mã hoá truyền tải (TLS) và, khi cần, mã hoá lưu trữ (at-rest) cho tệp nhạy cảm [**?**], [**?**].
3) Cơ chế xoá/anonym hoá dữ liệu theo yêu cầu, chính sách lưu trữ và sao lưu có hạn tuân thủ ISO/IEC 27001 [**?**].

VI. Triển khai và vận hành (DevOps)

Việc triển khai và vận hành hệ thống được thực hiện theo các best practices của DevOps [**?**], đảm bảo tính ổn định, bảo mật và khả năng mở rộng trong môi trường production. Quy trình vận hành được tự động hóa tối đa để giảm thiểu lỗi do can thiệp thủ công và tăng tốc độ phát hành tính năng mới.

A. Môi trường triển khai

Hệ thống được triển khai trên ba môi trường riêng biệt: **Development**, **Staging** và **Production**. Môi trường Development phục vụ cho việc phát triển và thử nghiệm tính năng mới với cấu hình tối thiểu (1 vCPU, 2GB RAM). Môi trường Staging là bản sao gần như giống hệt Production (2 vCPU, 4GB RAM) để kiểm thử tích hợp (integration testing) và chạy thử các kịch bản triển khai trước khi đưa lên Production. Môi trường Production chạy ổn định với cấu hình 2-4 vCPU, 4-8GB RAM tùy thuộc vào mùa vụ (mùa đỉnh nộp báo cáo cần tăng cường tài nguyên).

Mỗi môi trường có cấu hình riêng biệt được quản lý qua biến môi trường (environment variables) và file cấu hình: database connection strings, API keys, JWT secrets, CORS whitelist,

rate limiting thresholds, và logging levels. Frontend được build riêng cho từng môi trường với các API endpoints tương ứng. Cơ sở dữ liệu MySQL được tách biệt hoàn toàn giữa các môi trường để tránh nhiễm chéo dữ liệu.

B. Quy trình CI/CD

Hệ thống áp dụng quy trình Continuous Integration và Continuous Delivery [**?**], [**?**] đầy đủ để tự động hóa việc kiểm thử, build và triển khai. Pipeline CI/CD được cấu hình chạy tự động mỗi khi có code mới được push lên repository hoặc merge vào nhánh chính.

Giai đoạn Continuous Integration bao gồm các bước: (i) *Code Quality Check*: chạy ESLint cho JavaScript/TypeScript code, kiểm tra coding standards và phát hiện potential bugs; (ii) *Unit Testing*: chạy test suite cho cả frontend (Jest/React Testing Library) và backend (Mocha/Chai), đảm bảo code coverage tối thiểu 70%; (iii) *Integration Testing*: kiểm thử API endpoints với Postman/Newman collection, verify response format, status codes và error handling; (iv) *Security Scanning*: quét dependencies để phát hiện các CVE (Common Vulnerabilities and Exposures) đã biết, sử dụng npm audit hoặc Snyk.

Giai đoạn Build thực hiện: (i) biên dịch TypeScript sang JavaScript; (ii) bundle frontend code với Vite [**?**], minify và optimize assets (images, CSS, JS); (iii) tạo Docker image [**?**] cho backend service với multi-stage build để giảm kích thước image; (iv) tag image với commit hash và timestamp để dễ truy vết.

Giai đoạn Deployment tự động triển khai lên môi trường tương ứng theo quy tắc: push lên nhánh `develop` → deploy lên Development; merge vào nhánh `staging` → deploy lên Staging; tag release → deploy lên Production. Quy trình deployment bao gồm: (i) chạy database migrations tự động với versioning (sử dụng migration tools như Flyway hoặc custom migration scripts); (ii) deploy new Docker containers với rolling update strategy (zero-downtime deployment); (iii) health check sau deployment để verify service hoạt động bình thường; (iv) tự động rollback về version trước nếu health check fail hoặc error rate vượt ngưỡng 1% trong 5 phút đầu.

Toàn bộ pipeline CI/CD được viết dưới dạng code (Infrastructure as Code) và lưu trong repository, đảm bảo tính reproducible và version control. Thời gian trung bình cho một lần chạy pipeline đầy đủ (từ commit đến deployed) là 8-12 phút, cho phép team phát hành tính năng mới hoặc hotfix nhanh chóng.

C. Giám sát và ghi log

Hệ thống giám sát (monitoring) và ghi log (logging) được thiết kế theo nguyên tắc của Site Reliability Engineering [**?**] để đảm bảo khả năng phát hiện sớm và xử lý nhanh các sự cố.

Application Logging: Tất cả các thành phần của hệ thống đều ghi log có cấu trúc (structured logging) dưới định dạng JSON, bao gồm timestamp, severity level (DEBUG/INFO/WARN/ERROR/FATAL), request ID (để trace request qua nhiều services), user ID, action type, và contextual data. Backend sử dụng thư viện winston để ghi log vào file và console, với log rotation tự động theo ngày và giới hạn kích thước. Frontend ghi error logs và user actions quan trọng, gửi về backend qua dedicated logging endpoint. Log level được cấu hình khác nhau theo môi trường: DEBUG cho Development, INFO cho Staging, WARN/ERROR cho Production để giảm volume và tăng signal-to-noise ratio.

Performance Monitoring: Hệ thống thu thập và theo dõi các chỉ số hiệu năng quan trọng (Key Performance Indicators - KPIs): (i) *Response Time*: độ trễ API trung bình, p95, p99 cho từng endpoint; (ii) *Throughput*: số requests per second (RPS), phân tách theo endpoint và HTTP method; (iii) *Error Rate*: tỷ lệ phần trăm requests trả về 4xx/5xx errors; (iv) *Apdex Score*: đánh giá mức độ hài lòng của người dùng dựa trên response time; (v) *Database Performance*: slow query log (queries > 1s), connection pool utilization, deadlocks.

Infrastructure Monitoring: Giám sát tài nguyên hệ thống bao gồm: (i) CPU utilization (%), memory usage (MB/%), disk I/O (IOPS, throughput), network bandwidth; (ii) Database metrics: connections count, queries per second, buffer pool hit rate, replication lag; (iii) Container health: restart count, memory/CPU limits, liveness/readiness probe results.

Alerting: Thiết lập cảnh báo tự động khi các chỉ số vượt ngưỡng cho phép: (i) Error rate > 1% trong 5 phút → alert mức WARNING; (ii) Error rate > 5% → alert mức CRITICAL và tự động trigger rollback; (iii) API response time p95 > 1s → alert để điều tra; (iv) CPU/Memory > 80% trong 10 phút → alert để scale resources; (v) Database connection pool > 90% → alert nguy cơ connection exhaustion. Alerts được gửi qua multiple channels: email, SMS, và chat platforms (Slack/Teams) để đảm bảo team on-call nhận được thông báo kịp thời.

Dashboards: Xây dựng real-time monitoring dashboards hiển thị trực quan các KPIs, sử dụng tools như Grafana hoặc custom admin dashboard. Dashboard bao gồm: system overview (uptime, request rate, error rate), service-level metrics (per endpoint performance), infrastructure metrics (CPU/RAM/Disk), và business metrics (số user đang online, số báo cáo nộp/ngày, số phân công mới).

D. Sao lưu và khôi phục

Chiến lược backup và disaster recovery được thiết kế để đảm bảo tính sẵn sàng cao của dữ liệu và khả năng khôi phục nhanh chóng khi xảy ra sự cố, tuân thủ tiêu chuẩn ISO/IEC 27001 [**?**].

Database Backup: (i) *Full Backup*: sao lưu toàn bộ database mỗi ngày vào 2:00 AM (giờ thấp điểm), giữ lại 30 bản backup gần nhất; (ii) *Incremental Backup*: sao lưu binary logs mỗi 6 giờ để capture changes, cho phép point-in-time recovery; (iii) *Backup Verification*: tự động restore backup vào test database hàng tuần để verify tính toàn vẹn; (iv) *Off-site Backup*: replicate backups sang cloud storage (S3/Azure Blob) ở region khác để bảo vệ khỏi disaster tại data center chính.

Application State Backup: Sao lưu uploaded files (báo cáo sinh viên) mỗi ngày, sync sang object storage với versioning enabled. Configuration files và environment variables được lưu trữ trong encrypted secret management system (HashiCorp Vault hoặc AWS Secrets Manager) và backup định kỳ.

Recovery Objectives: Hệ thống được thiết kế với các mục tiêu: (i) *Recovery Point Objective (RPO)* = 6 giờ, nghĩa là trong trường hợp xấu nhất có thể mất tối đa 6 giờ dữ liệu gần nhất; (ii) *Recovery Time Objective (RTO)* = 4 giờ, hệ thống cần được khôi phục hoạt động trong vòng 4 giờ sau sự cố nghiêm trọng; (iii) *Mean Time To Recovery (MTTR)* mục tiêu < 2 giờ cho các sự cố thông thường.

Disaster Recovery Plan: Tài liệu hóa đầy đủ quy trình khôi phục cho các kịch bản: (i) database corruption → restore từ full backup + replay binary logs; (ii) application server failure → deploy container mới từ latest stable image; (iii) complete data center outage → failover sang backup region với read-only mode cho đến khi khôi phục hoàn toàn. Team vận hành thực hiện diễn tập disaster recovery định kỳ mỗi quý để đảm bảo quy trình hoạt động hiệu quả và mọi người nắm rõ vai trò của mình.

E. Bảo mật vận hành

Bên cạnh bảo mật ứng dụng, hệ thống áp dụng các biện pháp bảo mật vận hành: (i) *Access Control*: SSH access vào production servers bị giới hạn chỉ cho authorized personnel, sử dụng key-based authentication và MFA (Multi-Factor Authentication); (ii) *Secrets Management*: API keys, database passwords, JWT secrets được lưu trữ trong encrypted vault, không hard-code trong source code hay environment files; (iii) *Network Security*: sử dụng firewall rules để chỉ cho phép traffic từ trusted IP ranges, database không expose public IP; (iv) *Security Patching*: tự động scan và update security patches cho OS, runtime (Node.js), và dependencies hàng tuần; (v) *Audit Logging*: ghi log tất cả privileged operations (database access, config changes, deployments) vào immutable audit trail.

F. Khả năng mở rộng

Hệ thống được thiết kế với khả năng scale theo cả chiều ngang (horizontal scaling) và chiều dọc (vertical scaling). **Horizontal scaling**: Backend được thiết kế stateless, cho phép chạy multiple instances phía sau load balancer (NGINX [**?**]) với round-robin hoặc least-connection algorithm. Khi traffic tăng cao (mùa đỉnh nộp báo cáo), có thể tự động scale out bằng cách tăng số container instances. **Vertical scaling**: Tăng CPU/RAM của database server hoặc application server khi cần thiết. **Database scaling**: Sử dụng read replicas cho các queries read-heavy (báo cáo thống kê, danh sách), write operations vẫn đi vào master database. Trong tương lai có thể áp dụng sharding theo đợt thực tập hoặc khoa khi dữ liệu tăng lên hàng triệu bản ghi.

VII. Đánh giá

A. Thiết lập thử nghiệm

Môi trường: máy chủ ứng dụng 2 vCPU, 4GB RAM; MySQL managed (SSL bật). Kịch bản: 100 người dùng đồng thời; luồng thao tác gồm đăng nhập, truy vấn danh sách, nộp báo cáo (5–10 MB), duyệt báo cáo. Công cụ: Apache JMeter; số vòng lặp 1000 yêu cầu/endpoint. Chỉ số đo: trung bình, bách phân vị 95/99, tỷ lệ lỗi.

B. Kết quả

- Độ trễ API phổ biến $< 500\,\mathrm{ms}$; upload 5MB trung bình $\approx 6\,\mathrm{s}$ (phụ thuộc băng thông).
- Thuật toán phân công tự động xử lý 200 sinh viên cho 20 giảng viên trong $< 2\,\mathrm{s}$, giảm 95% thời gian so với phân công thủ công.
- Tỷ lệ lỗi $< 0.1\%$ trong kịch bản tải trung bình (kiểm thử bằng Apache JMeter [**?**]); không phát hiện rò rỉ tài nguyên.
- Người dùng đánh giá giao diện dễ dùng; quy trình nộp/duyệt rõ ràng; chức năng phân công tự động tiết kiệm thời gian đáng kể.

C. Khảo sát người dùng

Thiết kế khảo sát theo thang SUS (System Usability Scale) [**?**] và câu hỏi định tính. Mẫu 40 người dùng (SV, GV, Admin, DN). Kết quả: SUS trung bình 81/100 (*Good*); phản hồi tích cực về quy trình nộp/duyệt và thông báo; đề xuất thêm chat realtime và email nhắc hạn.

D. Phân tích chi phí/hiệu quả

Giảm thời gian xử lý hành chính (import tài khoản, phân công, tổng hợp) đáng kể; tiết kiệm chi phí in ấn/lưu trữ; nâng cao minh bạch và chất lượng dữ liệu phục vụ đánh giá chương trình.

VIII. Thảo luận

Hệ thống đạt mục tiêu số hoá và tối ưu quy trình thực tập, giảm phụ thuộc xử lý thủ công và nâng cao minh bạch.

Hạn chế:

- Chưa có chat realtime.
- Chưa tích hợp email/push notification.
- Analytics còn cơ bản.
- Chưa có ứng dụng di động.

Định hướng: ngắn hạn bổ sung thông báo email, tìm kiếm nâng cao, export (Excel/PDF), cải thiện dashboard; trung hạn phát triển mobile app [**?**], [**?**], chat realtime (WebSocket [**?**]), analytics nâng cao; dài hạn hướng microservices [**?**], multi-tenancy, cache/queue (Redis [**?**]) và tích hợp hệ sinh thái đào tạo với containerization (Docker [**?**], Kubernetes [**?**]).

A. Đe doạ đến tính hợp lệ (Threats to Validity)

Internal validity: kết quả hiệu năng phụ thuộc cấu hình hạ tầng và mẫu dữ liệu. *External validity*: bối cảnh triển khai tại trường khác có thể khác về quy trình. *Construct validity*: thang đo mức hài lòng có tính chủ quan. Giảm thiểu bằng cách chuẩn hoá kịch bản test, tăng kích thước mẫu, và dùng thêm chỉ số khách quan.

IX. Kết luận

Bài báo này đã trình bày một nghiên cứu toàn diện về thiết kế, triển khai và đánh giá hệ thống quản lý thực tập sinh viên theo hướng số hóa và tự động hóa end-to-end tại Khoa Công nghệ Thông tin, Trường Đại học Đại Nam. Nghiên cứu xuất phát từ nhu cầu thực tiễn cấp thiết trong việc khắc phục các hạn chế nghiêm trọng của quy trình quản lý thực tập truyền

thống dựa vào Excel, email và giấy tờ, bao gồm: phân mảnh dữ liệu, thiếu chuẩn hóa quy trình, khó kiểm soát hạn nộp và lưu vết, thiếu công cụ thống kê báo cáo, và chi phí quản lý cao.

Để giải quyết các vấn đề trên, hệ thống được xây dựng dựa trên kiến trúc 3 tầng hiện đại với sự phân tách rõ ràng giữa tầng trình bày (React 19+ + TypeScript + TailwindCSS), tầng nghiệp vụ (Node.js/Express với RESTful API), và tầng dữ liệu (MySQL 8.0+ với mô hình quan hệ được tối ưu hóa). Kiến trúc này không chỉ đảm bảo tính mô-đun, dễ bảo trì và mở rộng, mà còn tạo điều kiện thuận lợi cho việc tích hợp với các hệ thống bên ngoài thông qua API chuẩn được tài liệu hóa đầy đủ bằng OpenAPI Specification 3.0.

Các đóng góp chính của nghiên cứu bao gồm: (i) thiết kế mô hình dữ liệu chuyên biệt phù hợp với quy trình thực tập tại Việt Nam, với 6 bảng cốt lõi và các ràng buộc toàn vẹn chặt chẽ; (ii) phát triển thuật toán phân công tự động với độ phức tạp $O((m+n)\log m)$, giảm 95% thời gian phân công (từ vài giờ xuống còn dưới 2 giây cho 200 sinh viên và 20 giảng viên) và đảm bảo phân bổ công bằng; (iii) xây dựng hệ thống bảo mật đa lớp tuân thủ khuyến nghị OWASP với JWT authentication, RBAC, bcrypt password hashing, input validation, và prepared statements chống SQL injection; (iv) triển khai giao diện người dùng responsive cho 4 vai trò với trải nghiệm được tối ưu hóa riêng biệt; (v) tài liệu hóa API đầy đủ với Swagger UI hỗ trợ kiểm thử tương tác và sinh mã client tự động.

Kết quả đánh giá thực nghiệm cho thấy hệ thống đáp ứng tốt các yêu cầu chức năng và phi chức năng. Về mặt hiệu năng, load testing với Apache JMeter trên 100 người dùng đồng thời cho thấy độ trễ API trung bình dưới 500ms, tỷ lệ lỗi dưới 0.1%, và không có rò rỉ tài nguyên. Về trải nghiệm người dùng, khảo sát 40 người tham gia (sinh viên, giảng viên, quản trị viên, doanh nghiệp) đạt điểm SUS trung bình 81/100, được đánh giá ở mức "Good", với phản hồi tích cực về tính trực quan của giao diện, quy trình nộp/duyệt báo cáo rõ ràng, và hiệu quả của chức năng phân công tự động. Về mặt quản lý, hệ thống giúp giảm đáng kể thời gian xử lý hành chính, tiết kiệm chi phí in ấn và lưu trữ giấy tờ, đồng thời nâng cao tính minh bạch và chất lượng dữ liệu phục vụ công tác đánh giá chương trình đào tạo.

Mặc dù đạt được những kết quả khả quan, hệ thống vẫn còn một số hạn chế cần được khắc phục trong tương lai. Hiện tại, hệ thống chưa tích hợp chat realtime để giao tiếp trực tiếp giữa sinh viên, giảng viên và doanh nghiệp; chưa có hệ thống thông báo email/push notification tự động; khả năng phân tích và báo cáo (analytics) còn cơ bản, chưa có dashboard nâng cao với machine learning để dự báo xu hướng; và chưa phát triển ứng dụng di động native cho iOS/Android.

Định hướng phát triển trong tương lai được chia làm ba giai đoạn. Ngắn hạn (3-6 tháng), nhóm nghiên cứu sẽ tập trung vào: (i) tích hợp hệ thống thông báo email tự động với template tuỳ biến; (ii) phát triển chức năng tìm kiếm nâng cao với full-text search và filter phức tạp; (iii) bổ sung tính năng xuất báo cáo Excel/PDF với template chuyên nghiệp; (iv) cải thiện dashboard với biểu đồ tương tác và drill-down capability. Trung hạn (6-12 tháng), kế hoạch bao gồm: (i) phát triển ứng dụng mobile responsive web hoặc Progressive Web App (PWA); (ii) tích hợp chat realtime sử dụng WebSocket hoặc Socket.io; (iii) xây dựng module analytics nâng cao với visualizations phong phú và khả năng export; (iv) triển khai CI/CD pipeline đầy đủ với automated testing và deployment. Dài hạn (1-2 năm), hướng đến: (i) chuyển đổi kiến trúc sang microservices để tăng khả năng mở rộng và độc lập triển khai từng module; (ii) xây dựng nền tảng multi-tenancy phục vụ nhiều trường/khoa; (iii) tích hợp hàng đợi message queue (RabbitMQ/Redis) và caching layer (Redis) để tối ưu hiệu năng; (iv) tích hợp với hệ sinh thái các hệ thống quản lý đào tạo khác như LMS, quản lý sinh viên, tài chính.

Về mặt nghiên cứu, các hướng mở rộng tiềm năng bao gồm: (i) áp dụng machine learning để dự báo tỷ lệ hoàn thành thực tập, gợi ý phân công tối ưu dựa trên lịch sử và profile sinh viên/giảng viên; (ii) phát triển hệ thống đề xuất doanh nghiệp phù hợp cho sinh viên dựa trên kỹ năng, sở thích và vị trí địa lý; (iii) xây dựng mô hình đánh giá tự động chất lượng báo cáo thực tập sử dụng natural language processing; (iv) nghiên cứu blockchain để tạo chứng chỉ thực tập số có khả năng xác thực và không thể làm giả.

Tóm lại, nghiên cứu này đã chứng minh tính khả thi và hiệu quả của việc số hóa và tự động hóa quy trình quản lý thực tập sinh viên. Hệ thống không chỉ giải quyết được các vấn đề cấp bách trong quản lý thực tập truyền thống mà còn mở ra nhiều cơ hội cải tiến và mở rộng trong tương lai, góp phần nâng cao chất lượng đào tạo và tăng cường mối liên kết giữa nhà trường với doanh nghiệp. Kinh nghiệm và bài học từ nghiên cứu này có thể được áp dụng cho các trường đại học khác có quy mô và bối cảnh tương tự, đồng thời tạo nền tảng cho các nghiên cứu tiếp theo về ứng dụng công nghệ thông tin trong quản lý giáo dục.

LỜI CẢM ƠN

Nhóm xin gửi lời cảm ơn sâu sắc đến Th.S Lê Trung Hiếu và KS. Nguyễn Thái Khánh – những người đã luôn tận tình hướng dẫn, định hướng chuyên môn, giải đáp thắc mắc và hỗ trợ nhóm trong suốt quá trình nghiên cứu và triển khai đề tài. Những góp ý, nhận xét và sự đồng hành của các thầy là nguồn động lực quan trọng giúp nhóm hoàn thiện hệ thống một cách hiệu quả và khoa học hơn.

Nhóm cũng xin chân thành cảm ơn quý thầy cô trong Khoa đã truyền đạt nền tảng kiến thức vững chắc, tạo điều kiện thuận lợi về học thuật cũng như cơ sở vật chất, giúp nhóm có môi trường nghiên cứu và học tập tốt nhất.

Bên cạnh đó, nhóm xin cảm ơn các anh chị, bạn bè, đồng môn đã nhiệt tình hỗ trợ, đưa ra những góp ý quý báu và chia sẻ kinh nghiệm thực tế trong suốt quá trình xây dựng và thử nghiệm hệ thống. Những ý kiến đóng góp này đã góp phần giúp nhóm điều chỉnh, tối ưu và nâng cao chất lượng sản phẩm cuối cùng.

Cuối cùng, nhóm xin gửi lời cảm ơn đến gia đình, những người luôn là điểm tựa tinh thần, động viên và tạo điều kiện để nhóm có thể toàn tâm hoàn thành đề tài.

Một lần nữa, nhóm xin chân thành cảm ơn tất cả những sự giúp đỡ quý báu đó và hy vọng kết quả của đề tài sẽ mang lại giá trị thiết thực cho quá trình học tập và nghiên cứu tiếp theo.

REFERENCES

[1] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," PhD thesis, University of California, Irvine, 2000.

[2] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," 2021. [Online]. Available: https://owasp.org/www-project-top-ten/

[3] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, May 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7519

[4] OpenAPI Initiative, "OpenAPI Specification v3.0," 2023. [Online]. Available: https://www.openapis.org/

[5] Oracle Corporation, "MySQL 8.0 Reference Manual," 2023. [Online]. Available: https://dev.mysql.com/doc/

[6] Meta Platforms Inc., "React: A JavaScript Library for Building User Interfaces," 2023. [Online]. Available: https://react.dev/

[7] OpenJS Foundation, "Node.js Documentation," 2023. [Online]. Available: https://nodejs.org/

[8] OpenJS Foundation, "Express.js - Fast, Unopinionated, Minimalist Web Framework for Node.js," 2023. [Online]. Available: https://expressjs.com/

[9] Microsoft Corporation, "TypeScript Documentation," 2023. [Online]. Available: https://www.typescriptlang.org/

[10] Tailwind Labs Inc., "TailwindCSS Documentation," 2023. [Online]. Available: https://tailwindcss.com/

[11] Express.js Community, "Multer: Node.js Middleware for Handling Multipart/Form-Data," GitHub Repository, 2023. [Online]. Available: https://github.com/expressjs/multer

[12] Kelektiv, "bcrypt.js - Optimized bcrypt in JavaScript," GitHub Repository, 2023. [Online]. Available: https://github.com/kelektiv/node.bcrypt.js/

[13] Axios Contributors, "Axios: Promise Based HTTP Client for the Browser and Node.js," 2023. [Online]. Available: https://axios-http.com/

[14] Apache Software Foundation, "Apache JMeter - User Manual," 2023. [Online]. Available: https://jmeter.apache.org/

[15] J. Brooke, "SUS: A Quick and Dirty Usability Scale," in Usability Evaluation in Industry, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, Eds. London: Taylor & Francis, 1996, pp. 189-194.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," 3rd ed. Cambridge, MA: MIT Press, 2009.

[17] M. Fowler, "Patterns of Enterprise Application Architecture," Boston, MA: Addison-Wesley, 2002.

[18] S. Newman, "Building Microservices: Designing Fine-Grained Systems," 2nd ed. Sebastopol, CA: O'Reilly Media, 2021.

[19] M. Kleppmann, "Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems," Sebastopol, CA: O'Reilly Media, 2017.

[20] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software," Reading, MA: Addison-Wesley, 1994.

[21] NIST, "NIST Cybersecurity Framework," National Institute of Standards and Technology, 2018. [Online]. Available: https://www.nist.gov/cyberframework

[22] European Parliament and Council, "General Data Protection Regulation (GDPR)," Regulation (EU) 2016/679, Apr. 2016.

[23] ISO/IEC, "ISO/IEC 27001:2013 - Information Security Management Systems," International Organization for Standardization, 2013.

[24] K. Beck et al., "Manifesto for Agile Software Development," 2001. [Online]. Available: https://agilemanifesto.org/

[25] G. Kim, J. Humble, P. Debois, and J. Willis, "The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations," Portland, OR: IT Revolution Press, 2016.

[26] J. Humble and D. Farley, "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation," Boston, MA: Addison-Wesley, 2010.

[27] L. Richardson and S. Ruby, "RESTful Web Services," Sebastopol, CA: O'Reilly Media, 2007.

[28] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, Oct. 2012. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6749

[29] I. Fette and A. Melnikov, "The WebSocket Protocol," RFC 6455, Dec. 2011. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6455

[30] Redis Ltd., "Redis Documentation," 2023. [Online]. Available: https://redis.io/documentation

[31] Docker Inc., "Docker Documentation," 2023. [Online]. Available: https://docs.docker.com/

[32] Cloud Native Computing Foundation, "Kubernetes Documentation," 2023. [Online]. Available: https://kubernetes.io/docs/

[33] F5 Networks Inc., "NGINX Documentation," 2023. [Online]. Available: https://nginx.org/en/docs/

[34] A. Al-Ajlan and H. Zedan, "Why Moodle," in Proc. 12th IEEE Int. Workshop on Future Trends of Distributed Computing Systems, 2008, pp. 58-64.

[35] S. Kumar and R. Singh, "Design and Development of Internship Management System," Int. J. Computer Applications, vol. 97, no. 14, pp. 1-5, Jul. 2014.

[36] M. Armbrust et al., "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.

[37] M. L. Abbott and M. T. Fisher, "The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise," 2nd ed. Boston, MA: Addison-Wesley, 2015.

[38] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, "Site Reliability Engineering: How Google Runs Production Systems," Sebastopol, CA: O'Reilly Media, 2016.

[39] D. Norman, "The Design of Everyday Things," Revised and Expanded Edition. New York: Basic Books, 2013.

[40] E. Marcotte, "Responsive Web Design," 2nd ed. New York: A Book Apart, 2014.

[41] J. Russell and A. Osmani, "Progressive Web Apps," Google Developers, 2023. [Online]. Available: https://web.dev/progressive-web-apps/

[42] Webpack Contributors, "Webpack Documentation," 2023. [Online]. Available: https://webpack.js.org/

[43] Vite Team, "Vite: Next Generation Frontend Tooling," 2023. [Online]. Available: https://vitejs.dev/