# BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

# XÂY DỰNG HỆ THỐNG QUẢN LÝ NGHIÊN CỨU KHOA HỌC CHO TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT NAM ĐỊNH

Sinh viên thực hiện: Nguyễn Hữu Thắng

Giảng viên hướng dẫn: TS. Tô Đức Nhuận

Ngành: Công nghệ thông tin

Khoa: Công nghệ thông tin

# DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Viết tắt** | **Nghĩa đầy đủ** |
| NCKH | Nghiên cứu khoa học |
| ĐHSPKT | Đại học Sư phạm Kỹ thuật |
| KHCN | Khoa học Công nghệ |
| BGH | Ban Giám hiệu |
| RBAC | Role-Based Access Control – Kiểm soát truy cập theo vai trò |
| SPA | Single Page Application – Ứng dụng trang đơn |
| MPA | Multi Page Application – Ứng dụng đa trang |
| REST API | Representational State Transfer Application Programming Interface |
| JWT | JSON Web Token – Mã xác thực dạng JSON |
| ORM | Object-Relational Mapping – Ánh xạ đối tượng – quan hệ |
| CRUD | Create, Read, Update, Delete – Các thao tác dữ liệu cơ bản |
| SLA | Service Level Agreement – Cam kết thời hạn xử lý |
| DOCX | Định dạng tài liệu Microsoft Word |
| PDF | Portable Document Format – Định dạng tài liệu di động |
| SHA-256 | Secure Hash Algorithm 256-bit – Thuật toán băm bảo mật |
| ERD | Entity Relationship Diagram – Sơ đồ thực thể quan hệ |
| ACID | Atomicity, Consistency, Isolation, Durability – Bốn tính chất của giao dịch CSDL |
| CI/CD | Continuous Integration / Continuous Deployment |
| CSDL | Cơ sở dữ liệu |
| SSR | Server-Side Rendering – Kết xuất phía máy chủ |
| CSR | Client-Side Rendering – Kết xuất phía trình duyệt |
| DTO | Data Transfer Object – Đối tượng truyền tải dữ liệu |
| MVCC | Multi-Version Concurrency Control – Kiểm soát đồng thời đa phiên bản |
| IoC | Inversion of Control – Đảo ngược quyền điều khiển |

# MỞ ĐẦU

Trong bối cảnh toàn cầu hóa và cuộc cách mạng công nghiệp lần thứ tư, giáo dục đại học Việt Nam đang đứng trước yêu cầu cấp bách phải chuyển đổi số toàn diện. Nghị quyết số 52-NQ/TW ngày 27 tháng 9 năm 2019 của Bộ Chính trị về một số chủ trương, chính sách chủ động tham gia cuộc cách mạng công nghiệp lần thứ tư đã xác định chuyển đổi số là nhiệm vụ trọng tâm của mọi ngành, mọi lĩnh vực, trong đó giáo dục và đào tạo được xếp vào nhóm ưu tiên hàng đầu. Quyết định số 131/QĐ-TTg năm 2022 của Thủ tướng Chính phủ phê duyệt Đề án "Tăng cường ứng dụng công nghệ thông tin và chuyển đổi số trong giáo dục và đào tạo giai đoạn 2022–2025, định hướng đến 2030" càng khẳng định xu hướng tất yếu này. Theo đó, các trường đại học không chỉ cần số hóa hoạt động giảng dạy mà còn phải hiện đại hóa công tác quản lý hành chính, quản lý nghiên cứu khoa học và các hoạt động học thuật khác.

Hoạt động nghiên cứu khoa học là một trong ba trụ cột cốt lõi của giáo dục đại học, bên cạnh đào tạo và phục vụ cộng đồng. Chất lượng nghiên cứu khoa học không chỉ phản ánh năng lực học thuật của đội ngũ giảng viên mà còn ảnh hưởng trực tiếp đến thứ hạng, uy tín và khả năng thu hút nguồn lực của nhà trường. Tại Việt Nam, hoạt động NCKH trong trường đại học được quản lý qua một quy trình hành chính nhiều bước, từ đề xuất ý tưởng, xét duyệt ở các cấp, phân bổ kinh phí, triển khai thực hiện cho đến nghiệm thu và bàn giao sản phẩm. Mỗi bước đều liên quan đến nhiều bên tham gia với vai trò và trách nhiệm khác nhau, đòi hỏi sự phối hợp chặt chẽ và theo dõi liên tục.

Tại Trường Đại học Sư phạm Kỹ thuật Nam Định, quy trình quản lý đề tài NCKH trải qua nhiều giai đoạn – từ đề xuất, xét duyệt cấp Khoa, phân bổ Hội đồng đánh giá, phê duyệt, triển khai thực hiện, nghiệm thu đến bàn giao sản phẩm – với sự tham gia của nhiều bên liên quan: giảng viên, lãnh đạo Khoa, Phòng KHCN, thành viên Hội đồng và Ban Giám hiệu. Tuy nhiên, phần lớn các công đoạn này vẫn được thực hiện theo phương thức thủ công, dựa trên hồ sơ giấy, file Word rời rạc và trao đổi qua email. Đây là thực trạng chung của nhiều trường đại học địa phương tại Việt Nam, nơi mà nguồn lực đầu tư cho công nghệ thông tin còn hạn chế và thói quen làm việc truyền thống vẫn chi phối.

Thực trạng này dẫn đến bốn vấn đề nghiêm trọng ảnh hưởng đến hiệu quả quản lý NCKH của nhà trường.

Thứ nhất, thiếu minh bạch về trạng thái hồ sơ. Khi một đề tài được nộp lên, không ai có thể nhanh chóng biết hồ sơ đang nằm ở đâu trong quy trình, ai đang chịu trách nhiệm xử lý, và đã bao nhiêu ngày kể từ bước cuối cùng. Phòng KHCN thường phải liên hệ qua nhiều đơn vị để "truy tìm" hồ sơ bị tắc nghẽn. Trong khi đó, giảng viên – người chủ nhiệm đề tài – cũng không có kênh nào để chủ động theo dõi tiến trình xử lý hồ sơ của mình, dẫn đến tâm lý thụ động và thiếu động lực.

Thứ hai, rủi ro thất lạc và không thống nhất phiên bản. Khi hồ sơ tồn tại dưới dạng nhiều bản Word được gửi qua email, rất dễ xảy ra tình trạng các bên làm việc trên các phiên bản khác nhau. Một giảng viên có thể đã cập nhật đề cương theo yêu cầu của Hội đồng nhưng bản cũ vẫn đang được lưu hành tại Phòng KHCN. Việc không có hệ thống quản lý phiên bản cũng tiềm ẩn rủi ro pháp lý khi cần tra cứu lại hồ sơ gốc để đối chiếu hoặc giải trình.

Thứ ba, chậm tiến độ do thiếu cơ chế nhắc hạn. Không có quy định rõ ràng về thời hạn xử lý ở mỗi bước, dẫn đến nhiều đề tài bị trì hoãn kéo dài tại các khâu phê duyệt mà không ai hay biết cho đến khi kiểm tra thủ công. Một đề tài có thể nằm chờ duyệt tại cấp Khoa hàng tuần đơn giản vì người phụ trách quên hoặc bận công việc khác, trong khi không có cơ chế tự động nhắc nhở hay cảnh báo.

Thứ tư, báo cáo tổng hợp tốn nhiều công sức. Mỗi khi cần lập báo cáo về tình hình NCKH toàn trường – phục vụ họp giao ban, kiểm định chất lượng hay báo cáo cơ quan chủ quản – Phòng KHCN phải tổng hợp dữ liệu từ nhiều nguồn vào bảng tính Excel. Đây là công việc mất nhiều thời gian, dễ sai sót do nhập liệu thủ công, và phải lặp lại từ đầu mỗi kỳ báo cáo.

Xuất phát từ những vấn đề trên, đề tài "Xây dựng hệ thống quản lý nghiên cứu khoa học cho Trường Đại học Sư phạm Kỹ thuật Nam Định" được thực hiện nhằm số hóa toàn bộ quy trình quản lý đề tài NCKH, chuyển từ cách làm thủ công sang một hệ thống phần mềm có quy trình rõ ràng, trạng thái minh bạch và khả năng theo dõi tự động. Đề tài không chỉ đơn thuần là việc tin học hóa các biểu mẫu giấy, mà hướng đến việc tái cấu trúc quy trình quản lý theo hướng hiện đại, áp dụng các mẫu thiết kế phần mềm tiên tiến như máy trạng thái để đảm bảo tính chặt chẽ và nhất quán của toàn bộ quy trình.

Đề tài hướng đến bốn mục tiêu chính. Mục tiêu thứ nhất là xây dựng một hệ thống web cho phép quản lý toàn bộ vòng đời của đề tài NCKH, từ lúc đề xuất cho đến khi hoàn thành và bàn giao sản phẩm, thay thế hoàn toàn quy trình giấy tờ hiện tại. Mục tiêu thứ hai là tự động hóa quy trình xử lý thông qua một máy trạng thái với mười lăm trạng thái khác nhau, đảm bảo mỗi đề tài chỉ có thể chuyển sang bước tiếp theo khi đủ điều kiện, loại bỏ khả năng bỏ sót bước hoặc xử lý sai quy trình. Mục tiêu thứ ba là tích hợp chức năng sinh tài liệu tự động cho mười tám loại biểu mẫu theo chuẩn của trường, giúp giảm thiểu thao tác soạn thảo thủ công và đảm bảo tính chính xác, thống nhất của các văn bản hành chính. Mục tiêu thứ tư là xây dựng hệ thống phân quyền theo vai trò với tám nhóm người dùng khác nhau, đảm bảo mỗi người chỉ được truy cập và thao tác trong phạm vi quyền hạn của mình, vừa bảo vệ dữ liệu vừa đơn giản hóa giao diện sử dụng.

Về phạm vi, hệ thống tập trung vào quy trình quản lý đề tài NCKH cấp cơ sở tại Trường ĐHSPKT Nam Định. Các nhóm chức năng chính bao gồm: quản lý đề tài (tạo mới, chỉnh sửa, nộp, xem chi tiết, tìm kiếm), xử lý quy trình (chuyển trạng thái qua mười lăm bước với các tình huống ngoại lệ), quản lý Hội đồng đánh giá (tạo, phân công, gán cho đề tài), đánh giá và chấm điểm, sinh biểu mẫu tự động (mười tám loại dưới dạng DOCX và PDF), bảng điều khiển theo vai trò, quản lý thời hạn xử lý, và nhật ký kiểm toán. Hệ thống phục vụ nội bộ trường, chưa tích hợp với các hệ thống bên ngoài, chưa có module quản lý kinh phí đề tài và chưa hỗ trợ ứng dụng di động.

Đề tài được thực hiện theo phương pháp kết hợp. Giai đoạn đầu tập trung vào nghiên cứu tài liệu – tìm hiểu quy trình quản lý NCKH thực tế tại trường, phân tích các biểu mẫu hiện hành và các quy định liên quan. Giai đoạn tiếp theo là phân tích yêu cầu – xác định các nhóm chức năng cần xây dựng, mô tả chi tiết yêu cầu nghiệp vụ cho từng vai trò người dùng. Sau đó là thiết kế hệ thống – xây dựng kiến trúc tổng thể, thiết kế cơ sở dữ liệu, định nghĩa quy trình xử lý và giao diện người dùng. Giai đoạn phát triển phần mềm áp dụng kiến trúc monorepo cho hệ thống chính và microservice cho dịch vụ sinh tài liệu, sử dụng Docker để đóng gói triển khai. Cuối cùng, giai đoạn kiểm thử thực hiện kiểm thử chức năng theo kịch bản nghiệp vụ thực tế, đi qua toàn bộ quy trình từ đề xuất đến hoàn thành.

# CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

## 1.1. Giới thiệu bài toán quản lý nghiên cứu khoa học

Quản lý nghiên cứu khoa học trong trường đại học là hoạt động tổ chức, điều hành và giám sát toàn bộ vòng đời của các đề tài nghiên cứu. Vòng đời này bắt đầu từ khi giảng viên nảy sinh ý tưởng và viết đề xuất, trải qua nhiều lần xét duyệt ở các cấp khác nhau, được triển khai thực hiện, và kết thúc bằng việc nghiệm thu sản phẩm và bàn giao kết quả. Khác với việc quản lý một dự án thông thường trong doanh nghiệp, quản lý NCKH trong trường đại học Việt Nam mang những đặc thù riêng biệt xuất phát từ cơ cấu tổ chức và quy trình hành chính của hệ thống giáo dục công lập.

Đặc thù đầu tiên và quan trọng nhất của bài toán này nằm ở tính đa cấp và đa vai trò. Một đề tài phải đi qua nhiều tầng phê duyệt theo thứ tự bắt buộc: từ cấp Khoa (đơn vị trực tiếp quản lý giảng viên), qua Phòng KHCN (đơn vị chuyên trách ở cấp trường), đến Hội đồng đánh giá (gồm các chuyên gia được mời), và cuối cùng là Ban Giám hiệu (cấp phê duyệt cao nhất). Mỗi tầng có người chịu trách nhiệm và tiêu chí đánh giá riêng – cấp Khoa xem xét tính phù hợp với định hướng nghiên cứu của đơn vị, Hội đồng đánh giá về mặt khoa học, Phòng KHCN kiểm tra tính hợp lệ về thủ tục hành chính. Cấu trúc đa tầng này đảm bảo chất lượng nhưng đồng thời tạo ra nhiều điểm có thể gây tắc nghẽn trong quy trình.

Đặc thù thứ hai là sự đa dạng của các tình huống ngoại lệ. Bên cạnh luồng xử lý chính khi đề tài đi suôn sẻ qua tất cả các bước, thực tế luôn phát sinh nhiều tình huống cần xử lý đặc biệt. Đề tài có thể bị yêu cầu chỉnh sửa và nộp lại nếu Hội đồng nhận thấy đề cương chưa đạt yêu cầu nhưng có tiềm năng. Đề tài có thể bị từ chối hoàn toàn nếu không đáp ứng tiêu chí. Chủ nhiệm đề tài có thể chủ động hủy hoặc rút hồ sơ vì lý do cá nhân. Trong quá trình thực hiện, đề tài có thể bị tạm dừng vì lý do khách quan như thiếu kinh phí, thiên tai hoặc dịch bệnh. Mỗi tình huống ngoại lệ đều cần được xử lý một cách có hệ thống, ghi nhận đầy đủ và có thể phục hồi khi điều kiện cho phép.

Đặc thù thứ ba là yêu cầu về sinh tài liệu hành chính. Mỗi bước trong quy trình đều cần sinh ra các văn bản theo mẫu chuẩn do nhà trường hoặc cơ quan quản lý ban hành: phiếu đề xuất, đề cương chi tiết, biên bản họp Hội đồng, phiếu đánh giá, biên bản nghiệm thu, biên bản giao nhận sản phẩm, đơn xin gia hạn. Tổng cộng có mười tám loại biểu mẫu khác nhau. Các biểu mẫu này có cấu trúc cố định nhưng nội dung thay đổi theo từng đề tài, đòi hỏi sự chính xác cao trong việc điền thông tin. Đây là công đoạn tốn nhiều thời gian nhất khi thực hiện thủ công và cũng là nguồn gây ra nhiều lỗi nhất.

Đặc thù cuối cùng là yêu cầu về truy vết và giải trình. Vì NCKH trong trường công lập liên quan đến kinh phí nhà nước, mọi quyết định trong quy trình đều cần có khả năng tra cứu lại: ai đã phê duyệt, vào thời điểm nào, với nhận xét gì, điểm đánh giá ra sao. Yêu cầu này đặc biệt quan trọng khi có thanh tra, kiểm toán hoặc khi phát sinh tranh chấp.

## 1.2. Khảo sát thực trạng tại Trường ĐHSPKT Nam Định

Qua khảo sát thực tế và trao đổi với cán bộ Phòng KHCN, quy trình quản lý NCKH tại Trường ĐHSPKT Nam Định hiện tại hoạt động chủ yếu dựa trên giấy tờ và công cụ văn phòng thông thường. Toàn bộ quy trình có thể mô tả qua các bước sau: giảng viên soạn đề xuất bằng file Word theo mẫu rồi in ra nộp qua Khoa hoặc gửi email; lãnh đạo Khoa xem xét, ký duyệt rồi chuyển hồ sơ giấy kèm file mềm lên Phòng KHCN; Phòng KHCN tổng hợp, lập danh mục, mời Hội đồng đánh giá; sau khi Hội đồng họp và có kết quả, Phòng KHCN trình Ban Giám hiệu phê duyệt; cuối cùng, kết quả được thông báo về các Khoa và giảng viên.

Cách làm này tồn tại nhiều bất cập cụ thể. Về mặt quản lý thông tin, không có một nơi tập trung để tra cứu trạng thái của tất cả đề tài. Dữ liệu nằm rải rác trong hộp thư email của nhiều người, trong các thư mục trên máy tính cá nhân của cán bộ Phòng KHCN, và trong các cặp hồ sơ giấy tại văn phòng Khoa. Khi cán bộ chuyển công tác hoặc nghỉ phép, việc bàn giao và tiếp nối công việc gặp nhiều khó khăn.

Về mặt quy trình, không có cơ chế tự động để kiểm soát thứ tự các bước xử lý. Việc tuân thủ quy trình phụ thuộc hoàn toàn vào sự hiểu biết và ý thức của từng cá nhân. Đã từng xảy ra trường hợp một đề tài được triển khai thực hiện mà chưa hoàn tất thủ tục phê duyệt ở một cấp nào đó, gây khó khăn khi quyết toán kinh phí.

Về mặt biểu mẫu, giảng viên phải tự tải mẫu Word về máy, điền thông tin thủ công, dẫn đến nhiều lỗi phổ biến: sai tên đề tài giữa các biểu mẫu khác nhau, sai ngày tháng, thiếu chữ ký, dùng mẫu cũ đã hết hiệu lực. Phòng KHCN phải mất nhiều thời gian kiểm tra và yêu cầu sửa lại, có khi phải đi lại nhiều lần mới hoàn chỉnh được bộ hồ sơ.

Về mặt báo cáo, mỗi kỳ báo cáo tổng kết (thường là cuối học kỳ hoặc cuối năm), cán bộ Phòng KHCN phải lập bảng tính Excel tổng hợp tình hình NCKH toàn trường. Công việc này bao gồm thu thập số liệu từ các Khoa, đối chiếu với hồ sơ lưu trữ, phân loại theo nhiều tiêu chí (theo Khoa, theo trạng thái, theo năm), và tính toán các chỉ số thống kê. Toàn bộ quy trình này mất từ một đến hai tuần làm việc và kết quả vẫn có thể chứa sai sót do nhập liệu thủ công.

## 1.3. Các hệ thống tương tự và khoảng trống

*Hình 1.1. Sơ đồ Use Case tổng quát*

Trước khi thiết kế giải pháp, tác giả đã tìm hiểu và phân tích một số hệ thống quản lý nghiên cứu khoa học đang được sử dụng trong và ngoài nước nhằm rút ra bài học kinh nghiệm và xác định khoảng trống cần lấp đầy.

Trên phạm vi quốc tế, có thể kể đến hai hệ thống tiêu biểu. Thứ nhất là Pure (của Elsevier), một nền tảng quản lý nghiên cứu toàn diện được sử dụng bởi hàng trăm trường đại học và viện nghiên cứu trên thế giới. Pure cung cấp các chức năng quản lý dự án nghiên cứu, theo dõi kết quả công bố, báo cáo phân tích và tích hợp với các cơ sở dữ liệu khoa học quốc tế. Tuy nhiên, Pure được thiết kế theo quy trình quản lý nghiên cứu của phương Tây, không phù hợp với hệ thống phê duyệt đa cấp theo kiểu hành chính Việt Nam, chi phí bản quyền rất cao (hàng chục nghìn đô la Mỹ mỗi năm), và giao diện hoàn toàn bằng tiếng Anh. Thứ hai là các module quản lý nghiên cứu trong các hệ thống ERP giáo dục như Oracle PeopleSoft hoặc Ellucian Banner, cũng gặp các hạn chế tương tự về chi phí, ngôn ngữ và khả năng tùy biến quy trình.

Trong nước, một số trường đại học lớn đã tự phát triển hoặc thuê phát triển phần mềm quản lý NCKH. Tuy nhiên, qua tìm hiểu, các hệ thống này thường dừng ở mức quản lý danh sách đề tài đơn giản – cho phép nhập thông tin đề tài, tra cứu và xuất báo cáo cơ bản. Phần lớn chưa hỗ trợ workflow nhiều bước với các tình huống ngoại lệ, chưa có chức năng sinh biểu mẫu tự động từ dữ liệu, và chưa tích hợp cơ chế theo dõi thời hạn xử lý.

Ngoài lĩnh vực quản lý NCKH, có thể tham khảo các hệ thống quản lý quy trình nghiệp vụ tổng quát như Jira, Monday.com hoặc Trello. Các công cụ này rất mạnh về quản lý dự án và theo dõi tiến độ, nhưng không được thiết kế cho quy trình hành chính đặc thù của trường đại học Việt Nam: không hỗ trợ sinh biểu mẫu theo mẫu chuẩn, không có cơ chế phân quyền theo cấu trúc tổ chức (Khoa, Phòng, Ban), và thiếu tính pháp lý cần thiết cho các văn bản hành chính.

Bảng dưới đây tóm tắt so sánh:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tiêu chí** | **Hệ thống quốc tế (Pure, ERP)** | **Phần mềm nội địa hiện có** | **Công cụ quản lý dự án chung** | **Đề tài này** |
| Quy trình phê duyệt đa cấp | Có, nhưng theo mô hình nước ngoài | Hạn chế hoặc không có | Không chuyên biệt | Có, mười lăm trạng thái theo thực tế VN |
| Sinh biểu mẫu theo mẫu chuẩn | Không phù hợp mẫu VN | Không có | Không có | Có, mười tám loại biểu mẫu |
| Theo dõi SLA | Có | Không có | Có cơ bản | Có, tính theo ngày làm việc thực tế |
| Nhật ký kiểm toán | Có | Hạn chế | Có cơ bản | Có, chi tiết đến từng thao tác |
| Chi phí | Rất cao | Trung bình | Trung bình | Thấp (mã nguồn mở) |
| Ngôn ngữ tiếng Việt | Không | Có | Hạn chế | Có |

Khoảng trống rõ ràng nhất là thiếu một hệ thống vừa hỗ trợ quy trình phê duyệt đa cấp theo đúng thực tế Việt Nam, vừa có khả năng sinh tài liệu hành chính theo mẫu chuẩn của trường, vừa cung cấp công cụ giám sát tiến độ và báo cáo tổng hợp, với chi phí phù hợp cho một trường đại học địa phương.

## 1.4. Giải pháp đề xuất

Sau khi phân tích bài toán, khảo sát thực trạng và nghiên cứu các giải pháp hiện có, đề tài đề xuất xây dựng một hệ thống web gồm ba thành phần chính.

Thành phần thứ nhất là ứng dụng backend, đóng vai trò trung tâm xử lý nghiệp vụ, quản lý dữ liệu, điều phối quy trình workflow và cung cấp giao diện lập trình cho các thành phần khác. Backend được thiết kế theo kiến trúc module, mỗi module đảm nhận một nhóm chức năng nghiệp vụ riêng biệt, đảm bảo tính độc lập và dễ bảo trì. Tất cả các quy tắc nghiệp vụ – từ việc xác định ai được phép thực hiện thao tác gì, đến việc kiểm tra tính hợp lệ của mỗi lần chuyển trạng thái – đều được tập trung tại tầng này.

Thành phần thứ hai là ứng dụng frontend, là giao diện web mà người dùng tương tác trực tiếp. Frontend được thiết kế dạng ứng dụng trang đơn để mang lại trải nghiệm mượt mà, không phải tải lại trang khi chuyển đổi giữa các chức năng. Lý do chọn kiến trúc SPA là vì hệ thống quản lý NCKH yêu cầu người dùng thao tác liên tục trên nhiều biểu mẫu và danh sách – nếu mỗi thao tác đều tải lại toàn bộ trang, trải nghiệm sẽ bị gián đoạn và chậm chạp, đặc biệt khi điền biểu mẫu đề xuất dài nhiều phần.

Thành phần thứ ba là dịch vụ sinh tài liệu, một dịch vụ độc lập chuyên xử lý việc tạo văn bản DOCX và PDF từ các mẫu Word sẵn có. Dịch vụ này được tách riêng thành một microservice vì hai lý do. Về mặt kỹ thuật, ngôn ngữ Python sở hữu hệ sinh thái thư viện xử lý tài liệu vượt trội so với các ngôn ngữ khác, cho phép kiểm soát chi tiết đến từng đoạn văn, bảng biểu và định dạng trong file Word. Về mặt kiến trúc, việc tách riêng cho phép dịch vụ này được phát triển, triển khai và mở rộng độc lập với hệ thống chính.

Ba thành phần này giao tiếp với nhau thông qua giao thức HTTP theo kiến trúc REST, và chia sẻ chung một cơ sở dữ liệu quan hệ. Việc lựa chọn nền tảng web thay vì ứng dụng desktop hoặc di động xuất phát từ thực tế rằng tất cả giảng viên và cán bộ trong trường đều đã có máy tính kết nối mạng nội bộ, không cần cài đặt phần mềm bổ sung, và có thể truy cập hệ thống từ bất kỳ trình duyệt nào. Giải pháp web cũng thuận lợi cho việc triển khai tập trung, cập nhật tức thì và quản trị từ xa.

## 1.5. Các bên liên quan

Hệ thống phục vụ tám nhóm người dùng với vai trò và quyền hạn khác nhau, phản ánh đúng cơ cấu tổ chức và phân công trách nhiệm trong quy trình quản lý NCKH tại trường.

Giảng viên, hay còn gọi là Chủ nhiệm đề tài, là người khởi tạo và chịu trách nhiệm chính về đề tài nghiên cứu. Trong quy trình, giảng viên đóng vai trò khởi đầu: soạn thảo đề xuất, điền đầy đủ thông tin theo biểu mẫu, nộp hồ sơ lên Khoa, và theo dõi quá trình xử lý. Khi bị yêu cầu chỉnh sửa, giảng viên cập nhật nội dung theo góp ý rồi nộp lại. Trong giai đoạn thực hiện, giảng viên triển khai nghiên cứu theo đề cương đã được phê duyệt. Khi hoàn thành, giảng viên nộp sản phẩm nghiệm thu. Giảng viên cũng có quyền xuất các biểu mẫu liên quan đến đề tài của mình và chủ động hủy hoặc rút hồ sơ trong các trường hợp được phép.

Quản lý Khoa, thường là Trưởng Khoa hoặc Phó Khoa phụ trách NCKH, đóng vai trò cổng kiểm soát đầu tiên. Họ xem xét các đề tài do giảng viên trong khoa nộp lên, đánh giá tính phù hợp với định hướng nghiên cứu của đơn vị, và quyết định phê duyệt, yêu cầu chỉnh sửa hoặc từ chối. Quản lý Khoa cũng tham gia vào quá trình nghiệm thu cấp Khoa. Trên giao diện hệ thống, Quản lý Khoa chỉ nhìn thấy các đề tài thuộc khoa mình quản lý, đảm bảo tính phân cấp trong tổ chức.

Thư ký Khoa hỗ trợ Quản lý Khoa trong việc kiểm tra tính đầy đủ của hồ sơ – số lượng biểu mẫu, chữ ký, đầy đủ thông tin bắt buộc – và chuẩn bị các văn bản cần thiết trước khi trình lãnh đạo Khoa xem xét. Vai trò này giúp giảm tải cho Quản lý Khoa, đồng thời đảm bảo hồ sơ đã hoàn chỉnh về mặt thủ tục trước khi được đánh giá nội dung.

Phòng Khoa học Công nghệ đóng vai trò điều phối ở cấp trường. Sau khi đề tài qua được cổng kiểm soát cấp Khoa, Phòng KHCN tiếp nhận hồ sơ, tổng hợp các đề tài từ tất cả các Khoa, lập Hội đồng đánh giá bằng cách mời các chuyên gia trong và ngoài trường, phân bổ Hội đồng cho từng đề tài, và theo dõi tiến độ xử lý chung. Phòng KHCN cũng là đơn vị chịu trách nhiệm lập báo cáo tổng hợp tình hình NCKH toàn trường.

Thư ký Hội đồng phụ trách ghi biên bản các buổi họp hội đồng, tổng hợp kết quả đánh giá từ các phiếu chấm điểm của thành viên, tính điểm trung bình, và chuẩn bị hồ sơ trình duyệt ở bước tiếp theo. Vai trò này đòi hỏi sự chính xác cao trong việc ghi chép và tổng hợp số liệu.

Thành viên Hội đồng, bao gồm chủ tịch, phản biện và ủy viên, tham gia đánh giá và chấm điểm đề tài thông qua phiếu đánh giá trên hệ thống. Mỗi thành viên đánh giá độc lập, không nhìn thấy phiếu của người khác cho đến khi tất cả đã hoàn tất. Cơ chế này đảm bảo tính khách quan trong đánh giá.

Ban Giám hiệu xem báo cáo tổng hợp toàn trường và thực hiện phê duyệt cuối cùng ở những bước cần thiết. Trên bảng điều khiển, Ban Giám hiệu nhìn thấy bức tranh tổng thể: số lượng đề tài theo từng trạng thái, phân bố theo Khoa, tỷ lệ hoàn thành đúng hạn, và các đề tài có vấn đề cần quan tâm.

Quản trị hệ thống quản lý tài khoản người dùng (tạo mới, khóa, đặt lại mật khẩu, nhập hàng loạt từ Excel), cấu hình lịch làm việc (đánh dấu ngày lễ, ngày nghỉ, ngày làm bù), quản lý mẫu biểu mẫu và xem nhật ký kiểm toán. Đây là vai trò kỹ thuật, không tham gia vào quy trình nghiệp vụ nhưng đảm bảo hệ thống hoạt động trơn tru.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ SỬ DỤNG

## 2.1. Kiến trúc ứng dụng web hiện đại

Trước khi trình bày lựa chọn công nghệ cụ thể, cần hiểu rõ các mô hình kiến trúc phổ biến trong phát triển ứng dụng web hiện nay, bởi mỗi lựa chọn kiến trúc đều ảnh hưởng sâu sắc đến cách hệ thống được thiết kế, phát triển và vận hành.

Mô hình Client–Server là nền tảng của mọi ứng dụng web. Trong mô hình này, trình duyệt đóng vai trò client – gửi yêu cầu đến máy chủ, và máy chủ (server) tiếp nhận yêu cầu, xử lý logic nghiệp vụ, truy vấn cơ sở dữ liệu, và trả về kết quả. Mô hình này tạo ra sự tách biệt rõ ràng giữa hai phía: phía client chịu trách nhiệm hiển thị giao diện và thu thập thao tác người dùng, phía server chịu trách nhiệm xử lý dữ liệu và áp dụng quy tắc nghiệp vụ. Sự tách biệt này mang lại nhiều lợi ích: cho phép phát triển song song hai phía bởi các nhóm khác nhau, cho phép thay đổi giao diện mà không ảnh hưởng đến logic phía sau, và cho phép nhiều loại client khác nhau (trình duyệt web, ứng dụng di động, hệ thống bên thứ ba) cùng sử dụng chung một backend.

Trong mô hình Client–Server truyền thống, mỗi lần người dùng thao tác (bấm nút, chuyển trang, gửi biểu mẫu), trình duyệt gửi yêu cầu đến server và nhận về toàn bộ trang HTML mới. Toàn bộ trang – bao gồm cả các phần không thay đổi như menu, header, footer – đều được tải lại, gây lãng phí băng thông và tạo cảm giác giật, chậm cho người dùng. Mô hình hiện đại khắc phục hạn chế này bằng cách tách biệt rõ ràng hơn: server chỉ cung cấp dữ liệu thô qua giao diện lập trình, còn client tự xây dựng và cập nhật giao diện dựa trên dữ liệu nhận được.

Ứng dụng trang đơn (SPA – Single Page Application) là hiện thực hóa của mô hình hiện đại nói trên. Khi người dùng truy cập lần đầu, trình duyệt tải về toàn bộ mã nguồn giao diện (HTML, CSS, JavaScript) một lần duy nhất. Sau đó, mọi thao tác của người dùng đều được xử lý ngay trên trình duyệt; chỉ khi cần dữ liệu mới, trình duyệt mới gửi yêu cầu đến server và nhận về dữ liệu dạng JSON – không phải trang HTML hoàn chỉnh. Kết quả là giao diện phản hồi gần như tức thì, không có hiện tượng "nhấp nháy" khi chuyển trang, mang lại trải nghiệm giống ứng dụng desktop. Đối lập với SPA là MPA (Multi Page Application), mô hình truyền thống trong đó mỗi trang là một file HTML riêng trên server. MPA đơn giản hơn trong một số trường hợp, nhưng không phù hợp cho các ứng dụng quản lý phức tạp yêu cầu thao tác liên tục và biểu mẫu nhiều bước.

REST API (Representational State Transfer) là phong cách thiết kế giao diện lập trình phổ biến nhất hiện nay cho giao tiếp giữa client và server. Trong REST, mỗi tài nguyên nghiệp vụ (đề tài, người dùng, hội đồng, biểu mẫu) được biểu diễn bằng một địa chỉ URL duy nhất, và các thao tác trên tài nguyên tương ứng với các phương thức HTTP chuẩn: GET để đọc dữ liệu, POST để tạo mới, PATCH hoặc PUT để cập nhật, DELETE để xóa. Thiết kế REST mang lại sự nhất quán và dễ hiểu – bất kỳ lập trình viên nào đã quen với HTTP đều có thể nhanh chóng hiểu và sử dụng API. REST cũng hỗ trợ tốt cho việc phát triển độc lập giữa frontend và backend: hai nhóm chỉ cần thống nhất về cấu trúc URL và định dạng dữ liệu JSON, sau đó có thể làm việc song song.

Kiến trúc microservice là cách tổ chức hệ thống phần mềm thành nhiều dịch vụ nhỏ, độc lập, mỗi dịch vụ đảm nhận một chức năng riêng biệt và giao tiếp với nhau qua mạng (thường là HTTP hoặc hàng đợi tin nhắn). Đây là sự đối lập với kiến trúc nguyên khối (monolith), trong đó toàn bộ chức năng được đóng gói trong một ứng dụng duy nhất. Microservice mang lại nhiều lợi ích: mỗi dịch vụ có thể được viết bằng ngôn ngữ lập trình phù hợp nhất cho chức năng đó, triển khai và cập nhật độc lập mà không ảnh hưởng đến toàn hệ thống, và mở rộng (scale) riêng lẻ khi nhu cầu tăng. Tuy nhiên, microservice cũng mang lại sự phức tạp trong vận hành – cần quản lý nhiều dịch vụ, đảm bảo giao tiếp giữa chúng ổn định, và xử lý các lỗi mạng. Do đó, việc lựa chọn tách dịch vụ nào ra cần cân nhắc kỹ lưỡng giữa lợi ích và chi phí phức tạp.

Hệ thống trong đề tài này áp dụng cách tiếp cận kết hợp: phần lõi (backend + frontend) được tổ chức trong một monorepo – tức là cùng nằm trong một kho mã nguồn để chia sẻ kiểu dữ liệu, hằng số và cấu hình; dịch vụ sinh tài liệu được tách thành microservice riêng vì lý do sẽ được giải thích chi tiết ở mục 2.5. Cách tiếp cận này tận dụng được sự đơn giản của monolith cho phần lõi (nơi các module có sự phụ thuộc chặt chẽ), đồng thời hưởng lợi từ microservice cho phần có yêu cầu công nghệ khác biệt.

## 2.2. Lựa chọn công nghệ Backend

Trong chương trình đào tạo tại trường, sinh viên được học chủ yếu hai nền tảng backend là PHP và Java. Đề tài này lựa chọn NestJS – một framework chạy trên nền Node.js – sau khi cân nhắc kỹ lưỡng các phương án. Việc lựa chọn dựa trên phân tích các tiêu chí quan trọng nhất đối với bài toán cụ thể: khả năng tổ chức mã nguồn cho hệ thống phức tạp, hiệu năng xử lý đồng thời nhiều thao tác đọc/ghi cơ sở dữ liệu, và khả năng chia sẻ mã nguồn giữa backend và frontend.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí** | **PHP (Laravel)** | **Java (Spring Boot)** | **Node.js (NestJS)** |
| Ngôn ngữ | PHP – ngôn ngữ kịch bản, kiểu dữ liệu động | Java – ngôn ngữ biên dịch, kiểu tĩnh mạnh | TypeScript – ngôn ngữ kiểu tĩnh trên nền JavaScript |
| Mô hình xử lý | Đồng bộ, mỗi yêu cầu một tiến trình/luồng | Đa luồng, mỗi yêu cầu một luồng từ thread pool | Bất đồng bộ đơn luồng (event loop), xử lý hàng nghìn kết nối đồng thời với tài nguyên thấp |
| Hiệu năng I/O | Trung bình – phải dùng thêm thư viện cho async | Tốt – hỗ trợ reactive nhưng cấu hình phức tạp | Rất tốt – async/await là cơ chế mặc định, phù hợp hệ thống nhiều thao tác đọc/ghi CSDL |
| Kiến trúc | MVC đơn giản, ít ràng buộc cấu trúc | Rất chặt chẽ, nhiều annotation và cấu hình | Module-based, mỗi chức năng là một module độc lập với controller–service–DTO |
| An toàn kiểu dữ liệu | Yếu – PHP là dynamic typing | Mạnh – Java static typing | Mạnh – TypeScript cung cấp kiểm tra kiểu tại thời điểm biên dịch, ORM tự sinh kiểu dữ liệu từ CSDL |
| Hệ sinh thái | Rất phong phú (Composer) | Rất phong phú (Maven/Gradle) | Phong phú (npm), chia sẻ thư viện với frontend React |
| Đường cong học tập | Thấp – dễ bắt đầu | Cao – nhiều khái niệm (Bean, IoC, AOP) | Trung bình – quen thuộc nếu biết TypeScript |
| Chia sẻ mã nguồn | Không thể chia sẻ với frontend | Không thể chia sẻ với frontend | Cùng ngôn ngữ TypeScript, chia sẻ kiểu dữ liệu, hằng số, validation giữa backend và frontend |

Về mô hình xử lý, sự khác biệt giữa ba nền tảng là đáng kể. PHP truyền thống xử lý đồng bộ – mỗi yêu cầu HTTP được giao cho một tiến trình riêng; khi tiến trình đang chờ cơ sở dữ liệu trả về kết quả, nó không làm gì khác được, gây lãng phí tài nguyên. Java sử dụng đa luồng – mỗi yêu cầu được giao cho một luồng từ bể luồng; cách này hiệu quả hơn nhưng số lượng luồng bị giới hạn bởi bộ nhớ. Node.js sử dụng mô hình bất đồng bộ đơn luồng – khi chờ cơ sở dữ liệu, event loop chuyển sang xử lý yêu cầu khác, quay lại khi có kết quả. Mô hình này đặc biệt phù hợp với hệ thống quản lý vốn thực hiện rất nhiều thao tác đọc/ghi cơ sở dữ liệu (I/O-bound) chứ không phải tính toán nặng (CPU-bound).

Về khả năng tổ chức mã nguồn, NestJS vượt trội nhờ kiến trúc module có tính kỷ luật cao. Mỗi nhóm chức năng nghiệp vụ được đóng gói thành một module riêng biệt, có cấu trúc nhất quán gồm controller (tiếp nhận yêu cầu), service (xử lý logic), và DTO (định nghĩa cấu trúc dữ liệu). Với hệ thống có hơn hai mươi bảy module chức năng như trong đề tài, kiến trúc này giúp mã nguồn gọn gàng, dễ tìm kiếm và bảo trì. Laravel tuy có kiến trúc MVC nhưng ít ràng buộc hơn, dễ dẫn đến mã nguồn thiếu tổ chức khi dự án lớn lên. Spring Boot có kiến trúc chặt chẽ nhưng đường cong học tập dốc với nhiều khái niệm trừu tượng.

Về chia sẻ mã nguồn, lợi thế quyết định của NestJS so với PHP và Java nằm ở việc cùng sử dụng TypeScript với frontend React. Trong hệ thống quản lý NCKH, có rất nhiều cấu trúc dữ liệu được sử dụng ở cả hai phía: định nghĩa các trạng thái đề tài, danh sách quyền hạn, cấu trúc biểu mẫu, quy tắc validation. Với monorepo TypeScript, các định nghĩa này chỉ cần viết một lần và được chia sẻ giữa backend và frontend, loại bỏ hoàn toàn rủi ro không đồng bộ giữa hai phía. Nếu dùng PHP hoặc Java cho backend, mỗi khi thay đổi cấu trúc dữ liệu, lập trình viên phải nhớ cập nhật ở cả hai phía – một nguồn lỗi phổ biến trong phát triển phần mềm.

## 2.3. Lựa chọn công nghệ Frontend

Tương tự, sinh viên thường được tiếp cận các công nghệ frontend truyền thống như jQuery kết hợp với template engine phía server. Đề tài này lựa chọn React để xây dựng giao diện người dùng dạng SPA, dựa trên phân tích yêu cầu cụ thể của hệ thống quản lý NCKH.

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **jQuery + Template Engine (Blade/JSP)** | **React** |
| Mô hình hiển thị | Multi-Page Application – mỗi thao tác tải lại toàn bộ trang | Single-Page Application – chỉ cập nhật phần giao diện thay đổi |
| Trải nghiệm người dùng | Chậm, giật khi chuyển trang | Mượt mà, phản hồi tức thì, giống ứng dụng desktop |
| Quản lý trạng thái | Khó khăn khi ứng dụng phức tạp – phải tự quản lý biến toàn cục | Có hệ sinh thái quản lý trạng thái phong phú, phù hợp khi nhiều thành phần cần chia sẻ dữ liệu |
| Tái sử dụng giao diện | Hạn chế – chủ yếu sao chép mã HTML | Component-based – mỗi phần giao diện là một thành phần độc lập có thể tái sử dụng ở nhiều nơi |
| Xử lý biểu mẫu phức tạp | Phải viết nhiều mã JavaScript thủ công | Có cơ chế quản lý form hiệu quả với validation tích hợp |
| Tải dữ liệu từ server | Tải lại toàn trang hoặc AJAX thủ công | Tích hợp thư viện quản lý cache và đồng bộ dữ liệu tự động |
| Chia tải mã nguồn | Không hỗ trợ – tải toàn bộ JavaScript một lần | Lazy loading – chỉ tải mã của trang đang xem, tải thêm khi cần |

Sự khác biệt giữa MPA và SPA đặc biệt rõ rệt trong trải nghiệm người dùng. Với MPA, mỗi khi giảng viên chuyển từ trang danh sách đề tài sang trang chi tiết đề tài, toàn bộ trang – bao gồm menu bên trái, header, footer – đều được tải lại từ server, gây ra hiện tượng "trắng trang" kéo dài từ vài trăm mili giây đến vài giây tùy tốc độ mạng. Với SPA, việc chuyển trang chỉ cập nhật phần nội dung chính, menu và header giữ nguyên, và dữ liệu mới được tải không đồng bộ (async) nên giao diện luôn phản hồi ngay lập tức. Trải nghiệm này đặc biệt quan trọng khi người dùng cần thao tác nhiều lần trong ngày – mỗi giây tiết kiệm được cộng dồn thành sự hài lòng đáng kể.

Kiến trúc component-based của React cho phép xây dựng giao diện theo phương pháp "lắp ghép từ các khối nhỏ". Mỗi phần giao diện – một nút bấm, một huy hiệu trạng thái, một ô nhập liệu, một bảng dữ liệu có phân trang – được đóng gói thành một component độc lập, có thể kiểm thử riêng biệt và tái sử dụng ở nhiều nơi. Ví dụ, component huy hiệu trạng thái đề tài được sử dụng ở cả trang danh sách, trang chi tiết và bảng điều khiển; component bảng dữ liệu có phân trang được dùng cho danh sách đề tài, danh sách người dùng, danh sách hội đồng và nhật ký kiểm toán. Cách tiếp cận này giúp giảm đáng kể lượng mã nguồn cần viết và đảm bảo tính nhất quán về giao diện xuyên suốt ứng dụng.

Đối với hệ thống quản lý NCKH, React còn phù hợp vì có nhiều giao diện phức tạp: biểu mẫu đề xuất đề tài với nhiều phần (thông tin chung, nội dung nghiên cứu, dự toán kinh phí, tiến độ, danh sách nhà nghiên cứu), mỗi phần có thể chứa các bảng dữ liệu con cho phép thêm/xóa hàng động; phiếu đánh giá với bốn tiêu chí chấm điểm, mỗi tiêu chí có thanh trượt và ô nhận xét; bảng điều khiển với biểu đồ thống kê cập nhật theo thời gian thực. Những giao diện phức tạp này rất khó xây dựng hiệu quả bằng jQuery và template engine truyền thống, nhưng phù hợp tự nhiên với mô hình component của React.

## 2.4. Lựa chọn cơ sở dữ liệu

MySQL và PostgreSQL đều là hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở phổ biến, mỗi hệ thống có thế mạnh riêng. Đề tài chọn PostgreSQL sau khi đánh giá kỹ lưỡng các yêu cầu cụ thể của hệ thống quản lý NCKH.

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **MySQL** | **PostgreSQL** |
| Hỗ trợ JSON | Có từ phiên bản 5.7, nhưng hiệu năng truy vấn JSON hạn chế | Hỗ trợ JSONB (JSON nhị phân) – lưu trữ hiệu quả, có thể đánh chỉ mục và truy vấn nhanh bên trong JSON |
| Kiểu liệt kê (Enum) | Hỗ trợ cơ bản, khó thay đổi sau khi tạo | Hỗ trợ enum gốc mạnh mẽ, dễ quản lý và thay đổi |
| Tìm kiếm toàn văn | Có nhưng hạn chế | Tích hợp sẵn Full-Text Search mạnh mẽ |
| Giao dịch (Transaction) | Hỗ trợ ACID với InnoDB | Hỗ trợ ACID đầy đủ, cơ chế MVCC tiên tiến |
| Ràng buộc toàn vẹn | Cơ bản | Phong phú – hỗ trợ CHECK, EXCLUSION, partial index |
| Mở rộng | Hạn chế | Rất linh hoạt – cho phép tạo kiểu dữ liệu tùy chỉnh, hàm, trigger |

Lý do thứ nhất chọn PostgreSQL liên quan đến khả năng hỗ trợ JSON. Trong hệ thống quản lý NCKH, nhiều loại dữ liệu có cấu trúc linh hoạt không thể định nghĩa trước bằng các cột cố định. Ví dụ, dữ liệu biểu mẫu đề tài thay đổi tùy theo loại đề tài – đề tài nghiên cứu cơ bản có cấu trúc khác đề tài ứng dụng, và cấu trúc này có thể được nhà trường điều chỉnh mà không cần can thiệp mã nguồn. Dữ liệu phiếu đánh giá cũng tương tự – mỗi loại hội đồng có thể có bộ tiêu chí đánh giá khác nhau. JSONB của PostgreSQL cho phép lưu trữ những dữ liệu linh hoạt này dưới dạng JSON có hiệu năng cao, hỗ trợ đánh chỉ mục và truy vấn trực tiếp bên trong cấu trúc JSON – điều mà MySQL chỉ làm được ở mức hạn chế.

Lý do thứ hai liên quan đến kiểu liệt kê. Hệ thống sử dụng rất nhiều giá trị liệt kê: mười lăm trạng thái đề tài, tám vai trò người dùng, hơn năm mươi quyền hạn, bốn loại hội đồng, mười tám loại biểu mẫu, nhiều loại sự kiện kiểm toán. PostgreSQL cho phép định nghĩa các kiểu enum ngay ở tầng cơ sở dữ liệu, đảm bảo rằng một đề tài không thể có trạng thái ngoài danh sách cho phép – lớp bảo vệ quan trọng để tránh dữ liệu bất hợp lệ.

Lý do thứ ba liên quan đến tính toàn vẹn giao dịch. Mỗi lần chuyển trạng thái đề tài trong hệ thống cần thực hiện đồng thời nhiều thao tác: cập nhật trạng thái đề tài, ghi nhật ký workflow, ghi nhật ký kiểm toán, tính toán SLA mới, và cập nhật thông tin người/đơn vị nắm giữ hồ sơ. Tất cả phải thành công hoặc tất cả phải hủy bỏ – không cho phép tình trạng trạng thái đã cập nhật nhưng nhật ký chưa ghi, hoặc ngược lại. PostgreSQL với cơ chế MVCC (Multi-Version Concurrency Control) cung cấp khả năng quản lý giao dịch mạnh mẽ, đảm bảo tính ACID (Atomicity, Consistency, Isolation, Durability) ngay cả khi nhiều người dùng thao tác đồng thời.

## 2.5. Form Engine – Dịch vụ sinh tài liệu

Một câu hỏi tự nhiên đặt ra là: tại sao không viết chức năng sinh tài liệu ngay trong ứng dụng chính (NestJS) mà phải tách thành một dịch vụ riêng viết bằng Python? Câu trả lời nằm ở sự khác biệt về hệ sinh thái thư viện giữa hai ngôn ngữ.

Python sở hữu hệ sinh thái thư viện xử lý tài liệu vượt trội trong thế giới lập trình. Thư viện python-docx cho phép đọc, chỉnh sửa và tạo mới file DOCX với khả năng kiểm soát chi tiết đến từng đoạn văn, từng ô trong bảng biểu, header, footer, font chữ, kích thước, màu sắc và căn lề. Thư viện reportlab hỗ trợ tạo file PDF phức tạp với bố cục đa dạng. Sự trưởng thành của các thư viện này xuất phát từ truyền thống sử dụng Python trong lĩnh vực khoa học dữ liệu và tự động hóa văn phòng – nơi việc sinh báo cáo, biểu mẫu và tài liệu là nhu cầu hàng ngày. Trong khi đó, các thư viện tương đương trên Node.js hoặc không đầy đủ chức năng (thiếu khả năng xử lý bảng biểu phức tạp, header/footer), hoặc chất lượng kém hơn đáng kể, hoặc đã bị bỏ bảo trì.

Dịch vụ sinh tài liệu được xây dựng trên FastAPI – một framework Python hiện đại nổi bật với hai đặc điểm: hiệu năng cao nhờ hỗ trợ xử lý bất đồng bộ (ngang với Node.js trong nhiều benchmark), và khả năng tự động sinh tài liệu API từ mã nguồn. Cơ chế hoạt động của dịch vụ như sau: khi người dùng yêu cầu xuất biểu mẫu, ứng dụng chính gửi dữ liệu đề tài (dưới dạng JSON) đến Form Engine qua API; Form Engine xác định loại biểu mẫu cần tạo, mở file mẫu Word tương ứng, duyệt qua từng đoạn văn và bảng biểu trong mẫu để tìm và thay thế các trường dữ liệu (tên đề tài, tên chủ nhiệm, ngày tháng, kết quả đánh giá, danh sách thành viên) vào đúng vị trí, lưu thành file DOCX mới, chuyển đổi sang PDF, tính toán mã băm SHA-256 cho mỗi file để xác minh tính toàn vẹn, và trả về đường dẫn tải file.

Việc tách riêng dịch vụ này còn mang lại lợi ích quan trọng về mặt kiến trúc hệ thống. Thứ nhất, Form Engine có thể được mở rộng độc lập khi nhu cầu sinh tài liệu tăng cao – ví dụ vào cuối năm khi nhiều đề tài đồng loạt nghiệm thu – mà không ảnh hưởng đến hiệu năng của hệ thống chính. Thứ hai, nếu trong tương lai cần nâng cấp hoặc thay thế công nghệ sinh tài liệu, chỉ cần thay đổi dịch vụ này mà không cần động đến phần lõi. Thứ ba, lỗi trong quá trình sinh tài liệu (file mẫu lỗi, thiếu font chữ) không ảnh hưởng đến hoạt động chung của toàn hệ thống.

## 2.6. Các công nghệ hỗ trợ

Bên cạnh các công nghệ chính đã phân tích ở trên, hệ thống còn sử dụng một số công nghệ hỗ trợ quan trọng, mỗi công nghệ giải quyết một vấn đề kỹ thuật cụ thể.

Docker là công nghệ đóng gói ứng dụng dựa trên khái niệm "container" – một môi trường cô lập chứa đầy đủ ứng dụng cùng tất cả các thành phần phụ thuộc (thư viện, công cụ, cấu hình). Container khác với máy ảo (virtual machine) ở chỗ không cần một hệ điều hành riêng biệt cho mỗi container, mà chia sẻ nhân hệ điều hành với máy chủ, dẫn đến khởi động nhanh hơn và tiêu tốn ít tài nguyên hơn đáng kể. Lợi ích chính của Docker trong đề tài này là đảm bảo tính nhất quán môi trường – ứng dụng chạy giống hệt nhau trên máy tính phát triển, máy chủ kiểm thử và máy chủ vận hành, loại bỏ hoàn toàn vấn đề "trên máy tôi chạy được mà trên server không chạy". Docker Compose cho phép định nghĩa và khởi chạy toàn bộ hệ thống (backend, frontend, cơ sở dữ liệu, Form Engine) bằng một lệnh duy nhất, giúp quá trình triển khai trở nên đơn giản và có thể lặp lại.

Trong đề tài, Docker được sử dụng cụ thể cho hai mục đích. Thứ nhất, chạy cơ sở dữ liệu PostgreSQL trong container, giúp không cần cài đặt PostgreSQL trực tiếp lên hệ điều hành, dễ dàng khởi tạo lại cơ sở dữ liệu sạch khi cần kiểm thử, và đảm bảo đúng phiên bản PostgreSQL được sử dụng. Thứ hai, đóng gói dịch vụ Form Engine (Python) thành container riêng, cô lập hoàn toàn các thư viện Python khỏi hệ điều hành chủ.

Prisma ORM là công cụ ánh xạ đối tượng – quan hệ thế hệ mới, đóng vai trò cầu nối giữa mã nguồn TypeScript và cơ sở dữ liệu PostgreSQL. Thuật ngữ ORM (Object-Relational Mapping) chỉ kỹ thuật chuyển đổi tự động giữa đối tượng trong mã nguồn và bản ghi trong bảng cơ sở dữ liệu, cho phép lập trình viên thao tác với dữ liệu bằng các lệnh trực quan thay vì viết câu truy vấn SQL trực tiếp. Ưu điểm chính là giảm lỗi cú pháp SQL, tăng tốc phát triển, và đảm bảo an toàn trước tấn công SQL injection (vì toàn bộ đầu vào đều được xử lý tự động).

Điểm nổi bật của Prisma so với các ORM truyền thống là khả năng tự động sinh ra kiểu dữ liệu TypeScript tương ứng với từng bảng trong cơ sở dữ liệu. Khi lập trình viên định nghĩa cấu trúc bảng Đề tài có các trường như tiêu đề (kiểu chuỗi), trạng thái (kiểu enum), ngày tạo (kiểu ngày), Prisma tự động tạo ra một kiểu dữ liệu TypeScript tương ứng với đầy đủ các trường và kiểu dữ liệu đúng. Nhờ vậy, nếu lập trình viên vô tình truy cập một trường không tồn tại hoặc gán sai kiểu dữ liệu, lỗi được phát hiện ngay khi viết mã (tại thời điểm biên dịch) chứ không phải khi chạy chương trình – tiết kiệm đáng kể thời gian gỡ lỗi. Prisma cũng hỗ trợ quản lý phiên bản cấu trúc cơ sở dữ liệu thông qua cơ chế migration – mỗi thay đổi cấu trúc (thêm bảng, thêm cột, đổi kiểu dữ liệu) được ghi thành một bản ghi có thể áp dụng hoặc hoàn tác, cho phép theo dõi lịch sử thay đổi và đồng bộ cấu trúc giữa các môi trường.

JWT (JSON Web Token) là cơ chế xác thực người dùng phổ biến trong ứng dụng web hiện đại, đặc biệt phù hợp với kiến trúc SPA và REST API. Trong mô hình xác thực truyền thống (session-based), server lưu thông tin phiên đăng nhập trong bộ nhớ hoặc cơ sở dữ liệu, và gửi cho client một mã phiên (session ID) qua cookie; mỗi lần client gửi yêu cầu, server tra cứu mã phiên để xác định người dùng. Cách này gây áp lực lên server khi số lượng người dùng đồng thời lớn, và khó mở rộng khi có nhiều server.

JWT giải quyết vấn đề này bằng cách mã hóa thông tin người dùng (mã người dùng, vai trò, quyền hạn, thời hạn) thành một chuỗi ký tự (token) ký bằng khóa bí mật của server. Token được gửi cho client sau khi đăng nhập thành công; mỗi lần client gửi yêu cầu, token được đính kèm; server chỉ cần xác minh chữ ký của token là xác định được danh tính người dùng mà không cần tra cứu cơ sở dữ liệu – giảm tải đáng kể cho server.

Hệ thống trong đề tài sử dụng cặp token để cân bằng giữa bảo mật và tiện dụng. Access token có thời hạn ngắn (mười lăm phút) – đủ dài cho một phiên làm việc liên tục, nhưng đủ ngắn để giảm thiểu rủi ro nếu token bị đánh cắp. Refresh token có thời hạn dài (bảy ngày) – được lưu trong cookie HttpOnly (không thể truy cập bằng JavaScript, chống tấn công XSS), chỉ dùng để xin cấp access token mới khi token cũ hết hạn. Cơ chế này giúp người dùng không phải đăng nhập lại quá thường xuyên, đồng thời duy trì mức bảo mật cao.

# CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 3.1. Phân tích yêu cầu chức năng

Dựa trên khảo sát thực tế và trao đổi với cán bộ Phòng KHCN, các yêu cầu chức năng của hệ thống được phân thành tám nhóm chính, mỗi nhóm đáp ứng một nhu cầu nghiệp vụ cụ thể:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Nhóm chức năng** | **Mô tả** | **Vai trò liên quan** |
| 1 | Quản lý đề tài | Tạo mới, chỉnh sửa, xóa, tìm kiếm và xem chi tiết đề tài NCKH. Hỗ trợ lọc theo trạng thái, khoa, chủ nhiệm và thời hạn SLA. | Tất cả |
| 2 | Xử lý quy trình | Chuyển trạng thái đề tài qua mười lăm bước, bao gồm nộp, duyệt, trả lại, từ chối, hủy, rút, tạm dừng và tiếp tục. Mỗi lần chuyển trạng thái đều được ghi nhật ký. | Tất cả |
| 3 | Quản lý Hội đồng | Tạo và quản lý các Hội đồng đánh giá. Phân công chủ tịch, thư ký và ủy viên. Gán Hội đồng cho đề tài. | Phòng KHCN |
| 4 | Đánh giá đề tài | Thành viên Hội đồng chấm điểm theo bốn tiêu chí, mỗi tiêu chí tối đa 25 điểm. Phiếu đánh giá có hai trạng thái: nháp và hoàn tất. | Thành viên Hội đồng |
| 5 | Sinh biểu mẫu tự động | Tạo mười tám loại biểu mẫu hành chính dưới dạng DOCX và PDF. | Tất cả |
| 6 | Bảng điều khiển | Hiển thị tổng quan phù hợp từng vai trò. | Tất cả |
| 7 | Quản lý SLA | Cấu hình lịch làm việc. Tự động tính hạn chót theo ngày làm việc thực tế. Cảnh báo hồ sơ sắp và đã quá hạn. | Phòng KHCN, Admin |
| 8 | Quản trị hệ thống | Quản lý tài khoản, nhập dữ liệu, quản lý mẫu biểu, xem nhật ký kiểm toán, thao tác hàng loạt. | Admin |

Để làm rõ hơn yêu cầu chức năng, tác giả mô tả một số ca sử dụng tiêu biểu bằng lời.

Ca sử dụng "Giảng viên tạo và nộp đề tài mới" diễn ra như sau. Giảng viên đăng nhập vào hệ thống, chọn chức năng tạo đề tài mới. Hệ thống hiển thị biểu mẫu động nhiều bước theo cấu trúc đã được cấu hình sẵn. Giảng viên điền thông tin lần lượt qua từng bước: thông tin chung (tên đề tài, lĩnh vực, thời gian thực hiện), nội dung và phương pháp nghiên cứu, kết quả dự kiến, dự toán kinh phí, tiến độ thực hiện, và danh sách nhà nghiên cứu tham gia. Tại mỗi bước, hệ thống kiểm tra dữ liệu nhập vào theo các quy tắc validation đã định nghĩa. Giảng viên có thể lưu nháp bất kỳ lúc nào và quay lại chỉnh sửa sau. Khi hoàn thiện, giảng viên bấm nút "Nộp hồ sơ", hệ thống kiểm tra tính đầy đủ của toàn bộ biểu mẫu, nếu đạt yêu cầu thì chuyển đề tài sang trạng thái "Đang xét duyệt tại Khoa", đồng thời ghi nhật ký workflow và nhật ký kiểm toán.

Ca sử dụng "Phòng KHCN phân bổ Hội đồng đánh giá" diễn ra sau khi đề tài đã được Khoa phê duyệt. Cán bộ Phòng KHCN đăng nhập, nhìn thấy danh sách đề tài đang ở trạng thái "Đang phân bổ Hội đồng". Cán bộ có thể tạo Hội đồng mới hoặc sử dụng Hội đồng đã có, phân công các thành viên với vai trò cụ thể (chủ tịch, thư ký, ủy viên, phản biện), sau đó gán Hội đồng cho đề tài. Khi gán xong, đề tài chuyển sang trạng thái "Đang xét duyệt bởi Hội đồng" và các thành viên Hội đồng sẽ nhìn thấy đề tài trong danh sách đánh giá của mình.

Ca sử dụng "Sinh biểu mẫu tự động" có thể được kích hoạt ở nhiều thời điểm khác nhau trong quy trình. Người dùng chọn đề tài cần xuất biểu mẫu, chọn loại biểu mẫu từ danh sách mười tám loại. Hệ thống gửi yêu cầu đến dịch vụ Form Engine kèm dữ liệu đề tài, nhận về file DOCX và PDF, hiển thị đường dẫn tải về. Đồng thời, hệ thống lưu bản kê dữ liệu (manifest) ghi nhận chính xác dữ liệu nào đã được sử dụng để sinh tài liệu – phục vụ truy vết và xác minh sau này.

## 3.2. Phân tích yêu cầu phi chức năng

Ngoài các chức năng nghiệp vụ, hệ thống cần đáp ứng các yêu cầu về chất lượng, ảnh hưởng đến toàn bộ quyết định thiết kế và cài đặt.

Về hiệu năng, hệ thống cần phản hồi nhanh cho các thao tác thường gặp – tải danh sách đề tài, mở trang chi tiết, chuyển trạng thái – trong vòng vài trăm mili giây. Cơ sở dữ liệu được tối ưu với hơn hai mươi lăm chỉ mục trên các trường truy vấn thường xuyên như trạng thái đề tài, mã chủ nhiệm, mã khoa và thời hạn SLA. Việc lựa chọn chỉ mục dựa trên phân tích các câu truy vấn phổ biến nhất: lấy danh sách đề tài theo trạng thái và khoa (phục vụ bảng điều khiển), tìm đề tài theo chủ nhiệm (phục vụ trang cá nhân giảng viên), và lọc đề tài theo tình trạng SLA (phục vụ cảnh báo quá hạn). Ở tầng frontend, kỹ thuật lazy loading giúp giảm thời gian tải ban đầu bằng cách chỉ tải mã nguồn của trang đang xem.

Về bảo mật, hệ thống áp dụng nhiều tầng bảo vệ. Tầng xác thực sử dụng JWT với cơ chế token kép đã mô tả ở Chương 2. Tầng phân quyền triển khai mô hình RBAC với hơn năm mươi quyền hạn chi tiết, được kiểm tra ở cả tầng giao diện (ẩn/hiện các phần tử) và tầng server (từ chối yêu cầu không hợp lệ) – việc kiểm tra ở cả hai tầng là nguyên tắc "phòng thủ theo chiều sâu", vì kiểm tra ở tầng giao diện có thể bị vô hiệu hóa bởi người dùng có kỹ năng kỹ thuật. Cookie được cấu hình HttpOnly để chống tấn công đánh cắp token qua JavaScript (XSS – Cross-Site Scripting). Sử dụng ORM để ngăn chặn tấn công SQL injection – mọi đầu vào từ người dùng đều được xử lý tự động bởi ORM thay vì ghép trực tiếp vào câu SQL. Mật khẩu được mã hóa một chiều bằng thuật toán bcrypt trước khi lưu vào cơ sở dữ liệu.

Về độ tin cậy, mọi thao tác chuyển trạng thái đều được thực hiện trong giao dịch cơ sở dữ liệu để đảm bảo tính nguyên tử – hoặc tất cả thay đổi đều thành công, hoặc không có thay đổi nào được lưu. Hệ thống còn áp dụng cơ chế bảo đảm tính bất biến (idempotency) để ngăn việc xử lý trùng lặp khi người dùng vô tình bấm nút hai lần hoặc khi trình duyệt gửi lại yêu cầu do lỗi mạng. Cụ thể, trước khi thực hiện chuyển trạng thái, hệ thống kiểm tra trạng thái hiện tại của đề tài; nếu trạng thái đã khác so với trạng thái kỳ vọng (vì yêu cầu trước đó đã được xử lý), hệ thống từ chối yêu cầu trùng lặp thay vì thực hiện lại.

Về khả năng mở rộng, kiến trúc monorepo kết hợp microservice cho phép mở rộng từng thành phần độc lập. Nếu tải sinh tài liệu tăng cao, có thể chạy thêm nhiều instance của Form Engine mà không cần thay đổi backend. Docker hóa giúp việc nhân bản dịch vụ trở nên đơn giản – chỉ cần khởi chạy thêm container mới từ cùng một image.

Về khả năng bảo trì, mã nguồn TypeScript với kiểu dữ liệu tĩnh giúp dễ đọc hiểu và bảo trì. Khi một lập trình viên mới tiếp nhận dự án, kiểu dữ liệu đóng vai trò như tài liệu sống – cho biết mỗi biến chứa gì, mỗi hàm nhận đầu vào gì và trả về gì. Cơ sở dữ liệu được mô tả tường minh trong file schema của Prisma, đóng vai trò như tài liệu thiết kế cơ sở dữ liệu luôn được cập nhật theo mã nguồn thực tế.

Về khả năng kiểm toán, mọi thao tác quan trọng (tạo, sửa, xóa, chuyển trạng thái, đăng nhập, đăng xuất, xuất tài liệu) đều được ghi vào bảng nhật ký kiểm toán với đầy đủ thông tin bối cảnh: thời điểm, người thực hiện, hành động, đối tượng bị tác động, dữ liệu bổ sung, địa chỉ IP, thông tin trình duyệt và mã yêu cầu. Nhật ký kiểm toán được thiết kế theo nguyên tắc chỉ ghi thêm (append-only) – không cho phép sửa hoặc xóa, đảm bảo tính toàn vẹn.

## 3.3. Kiến trúc tổng thể

Hệ thống được tổ chức theo kiến trúc ba tầng kết hợp microservice. Tầng giao diện (frontend), tầng nghiệp vụ (backend) và tầng dữ liệu (cơ sở dữ liệu) được tổ chức trong một monorepo, trong khi dịch vụ sinh tài liệu (Form Engine) là một microservice độc lập.

Tầng giao diện là ứng dụng React chạy trên trình duyệt người dùng, cung cấp giao diện dạng SPA. Giao diện được chia thành các thành phần tái sử dụng: bố cục trang (layout) với thanh điều hướng bên trái và phần nội dung chính, thanh điều hướng hiển thị menu chức năng phù hợp với vai trò người dùng đang đăng nhập, các thành phần nhập liệu (biểu mẫu động, ô tìm kiếm, bộ lọc), các thành phần hiển thị dữ liệu (bảng có phân trang, huy hiệu trạng thái, dòng thời gian workflow, biểu đồ thống kê). Hệ thống sử dụng cơ chế tải lười (lazy loading) – mã nguồn của mỗi trang chỉ được tải khi người dùng truy cập trang đó lần đầu, giảm đáng kể thời gian tải ban đầu. Zustand quản lý trạng thái phía client (thông tin người dùng đang đăng nhập, cài đặt giao diện), trong khi TanStack Query quản lý việc tải và đồng bộ dữ liệu từ server (với cơ chế cache tự động, refetch khi dữ liệu cũ, và retry khi lỗi mạng).

*Hình 3.1. Sơ đồ kiến trúc tổng thể hệ thống*

Tầng nghiệp vụ là ứng dụng NestJS được tổ chức thành hơn hai mươi bảy module, có thể nhóm thành bốn nhóm chính. Nhóm xác thực và phân quyền xử lý đăng nhập, đăng xuất, gia hạn token, kiểm tra quyền. Nhóm quản lý nghiệp vụ xử lý các thao tác trên đề tài, hội đồng, đánh giá, biểu mẫu. Nhóm hạ tầng cung cấp các dịch vụ nền tảng như kiểm toán, SLA, nhập/xuất dữ liệu. Nhóm điều phối kết nối với các dịch vụ bên ngoài như Form Engine. Mỗi module có cấu trúc nhất quán gồm: controller (tiếp nhận yêu cầu HTTP, kiểm tra đầu vào, trả về kết quả), service (xử lý logic nghiệp vụ, gọi cơ sở dữ liệu, tương tác với module khác), và DTO (định nghĩa cấu trúc dữ liệu đầu vào/đầu ra với các quy tắc validation).

Luồng dữ liệu điển hình khi người dùng thực hiện một thao tác diễn ra như sau: trình duyệt gửi yêu cầu HTTP kèm access token đến backend; bộ lọc xác thực kiểm tra token hợp lệ và trích xuất thông tin người dùng; bộ lọc phân quyền kiểm tra người dùng có quyền thực hiện thao tác yêu cầu hay không; controller tiếp nhận yêu cầu và chuyển đến service tương ứng; service thực hiện logic nghiệp vụ (kiểm tra điều kiện, cập nhật dữ liệu, ghi nhật ký) trong một giao dịch cơ sở dữ liệu; kết quả được trả về qua controller đến trình duyệt.

Tầng dữ liệu là cơ sở dữ liệu PostgreSQL lưu trữ mười tám bảng dữ liệu. Prisma ORM đóng vai trò trung gian, chuyển đổi giữa đối tượng TypeScript và bản ghi cơ sở dữ liệu, quản lý connection pool (bể kết nối) để tái sử dụng kết nối thay vì tạo kết nối mới cho mỗi yêu cầu.

Dịch vụ sinh tài liệu (Form Engine) là microservice Python FastAPI hoạt động độc lập, chỉ được gọi khi cần sinh biểu mẫu. Giao tiếp giữa backend NestJS và Form Engine diễn ra qua HTTP REST API nội bộ.

## 3.4. Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu gồm mười tám bảng, có thể chia thành năm nhóm chính theo chức năng. Việc phân nhóm giúp hiểu rõ vai trò và mối quan hệ giữa các bảng trong toàn bộ hệ thống.

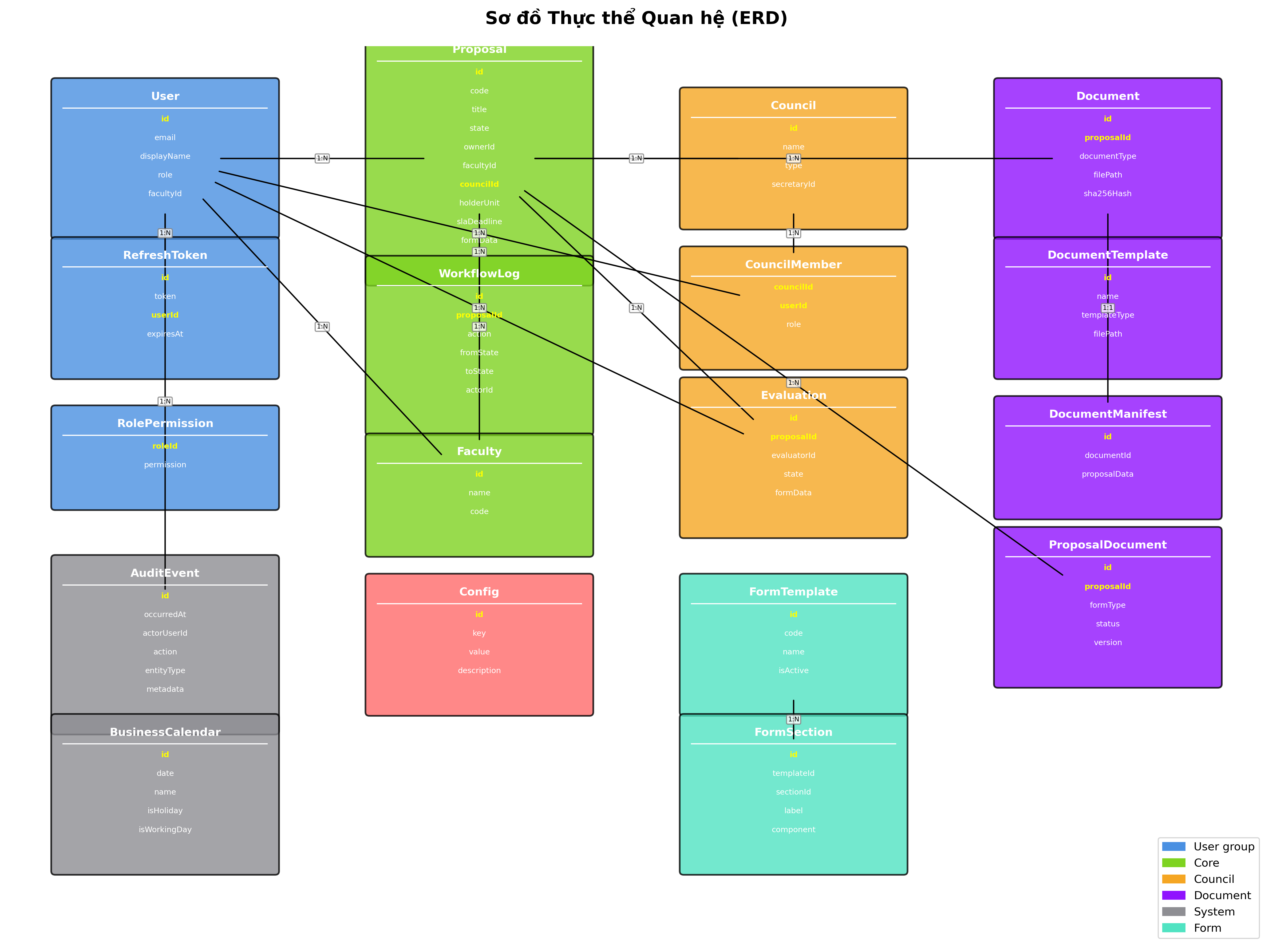
Nhóm quản lý người dùng gồm ba bảng, chịu trách nhiệm lưu trữ thông tin tài khoản và phân quyền. Bảng Người dùng lưu thông tin cốt lõi của mỗi tài khoản: email (duy nhất, dùng để đăng nhập), mật khẩu đã mã hóa bằng bcrypt, tên hiển thị, vai trò (một trong tám giá trị enum), khoa trực thuộc (liên kết khóa ngoại đến bảng Khoa), và trạng thái tài khoản (hoạt động hoặc bị khóa). Bảng Mã làm mới lưu các refresh token đang có hiệu lực, cho phép một người dùng đăng nhập trên nhiều thiết bị đồng thời – mỗi thiết bị có một refresh token riêng. Khi người dùng đăng xuất hoặc đổi mật khẩu, các refresh token tương ứng bị thu hồi. Bảng Khoa lưu danh sách các đơn vị trong trường, phục vụ phân cấp quản lý – mỗi người dùng thuộc một khoa, mỗi đề tài thuộc một khoa.

Nhóm quản lý đề tài là trung tâm của hệ thống, gồm hai bảng có mối quan hệ chặt chẽ. Bảng Đề tài là bảng lớn nhất và quan trọng nhất, lưu toàn bộ thông tin về một đề tài nghiên cứu. Các trường chính bao gồm: mã đề tài (sinh tự động, duy nhất), tiêu đề, trạng thái hiện tại (một trong mười lăm giá trị enum), chủ nhiệm (khóa ngoại đến bảng Người dùng), khoa (khóa ngoại đến bảng Khoa), hội đồng được gán (khóa ngoại đến bảng Hội đồng, có thể rỗng khi chưa gán), đơn vị đang giữ hồ sơ (để xác định ai chịu trách nhiệm xử lý ở bước hiện tại), người đang giữ hồ sơ, thời hạn SLA, dữ liệu biểu mẫu động (lưu dạng JSONB – cho phép mỗi loại đề tài có cấu trúc biểu mẫu riêng mà không cần thay đổi cấu trúc bảng), cùng các trường ghi nhận tình huống ngoại lệ (thời điểm hủy, rút, từ chối, tạm dừng, trạng thái trước khi tạm dừng – để khôi phục đúng bước khi tiếp tục, và trạng thái đích khi nộp lại – để đề tài quay về đúng cấp đã yêu cầu sửa). Bảng Nhật ký workflow ghi lại mọi lần chuyển trạng thái: đề tài nào, từ trạng thái nào sang trạng thái nào, ai thực hiện, thời điểm nào, ghi chú kèm theo. Mỗi đề tài có thể có hàng chục bản ghi nhật ký workflow, tạo thành bức tranh toàn cảnh về lịch sử xử lý.

Nhóm hội đồng và đánh giá gồm ba bảng, phục vụ chức năng thành lập hội đồng và chấm điểm đề tài. Bảng Hội đồng lưu thông tin chung của hội đồng: tên, loại (xét duyệt đề cương, nghiệm thu cấp Khoa, nghiệm thu cấp Trường, hoặc gia hạn), và thư ký. Bảng Thành viên hội đồng ánh xạ người dùng với hội đồng kèm vai trò (chủ tịch, thư ký, ủy viên, phản biện) – một hội đồng có nhiều thành viên, một người dùng có thể tham gia nhiều hội đồng. Bảng Đánh giá lưu phiếu chấm điểm của từng thành viên cho từng đề tài, với ràng buộc duy nhất đảm bảo mỗi người chỉ đánh giá một đề tài một lần. Dữ liệu điểm và nhận xét được lưu dạng JSONB để linh hoạt theo từng loại đánh giá.

*Hình 3.2. Sơ đồ thực thể quan hệ (ERD)*

Nhóm tài liệu gồm năm bảng, phục vụ chức năng sinh biểu mẫu tự động. Bảng Mẫu tài liệu lưu các file mẫu Word gốc kèm mã băm SHA-256 để xác minh tính toàn vẹn – đảm bảo file mẫu không bị sửa đổi ngoài ý muốn. Bảng Tài liệu đã sinh lưu thông tin các file đã được tạo ra (loại tài liệu, đường dẫn file, mã băm, người tạo, thời điểm tạo). Bảng Bản kê tài liệu (manifest) lưu ảnh chụp dữ liệu đề tài tại thời điểm sinh tài liệu – giúp xác minh rằng nội dung tài liệu phản ánh đúng dữ liệu tại thời điểm đó, phục vụ mục đích truy vết. Bảng Mẫu biểu mẫu và Mục biểu mẫu mô tả cấu trúc của các biểu mẫu nhập liệu động, cho phép tùy biến biểu mẫu (thêm/bớt phần, thay đổi thứ tự, cập nhật nhãn) mà không cần sửa mã nguồn.



Các bảng được liên kết với nhau thông qua khóa ngoại và ràng buộc toàn vẹn. Ví dụ, mỗi đề tài thuộc về một chủ nhiệm và một khoa; khi xóa một người dùng (nếu được phép), hệ thống kiểm tra xem người đó có đang là chủ nhiệm đề tài nào không. Mỗi phiếu đánh giá gắn với một đề tài và một người đánh giá, với ràng buộc duy nhất ngăn việc một người đánh giá cùng một đề tài hai lần. Mỗi nhật ký workflow gắn với một đề tài, tạo thành quan hệ một-nhiều cho phép tra cứu toàn bộ lịch sử xử lý.

## 3.5. Thiết kế quy trình xử lý (Workflow State Machine)

Trái tim của hệ thống là máy trạng thái (state machine) quản lý vòng đời đề tài. Khái niệm máy trạng thái xuất phát từ lý thuyết tính toán, được ứng dụng rộng rãi trong phát triển phần mềm cho các hệ thống có quy trình phức tạp. Ý tưởng cốt lõi là: hệ thống tại bất kỳ thời điểm nào đều nằm ở đúng một trạng thái; từ mỗi trạng thái, chỉ có một tập hợp hữu hạn các hành động hợp lệ dẫn đến các trạng thái khác; mọi hành động không nằm trong tập hợp này đều bị từ chối. Cách tiếp cận này buộc nhà thiết kế phải suy nghĩ và định nghĩa rõ ràng mọi tình huống có thể xảy ra, từ đó giảm thiểu lỗi logic và đảm bảo quy trình luôn được tuân thủ.

Luồng xử lý chính mô tả quy trình lý tưởng khi đề tài đi qua tất cả các bước mà không gặp trở ngại. Đề tài bắt đầu ở trạng thái Nháp khi giảng viên tạo mới. Khi giảng viên hoàn thiện biểu mẫu và bấm nộp, đề tài chuyển sang trạng thái Đang xét duyệt tại Khoa – lúc này hồ sơ nằm trong tay Quản lý Khoa. Khi Quản lý Khoa phê duyệt, đề tài chuyển sang Đang phân bổ Hội đồng – hồ sơ chuyển về Phòng KHCN. Phòng KHCN gán Hội đồng đánh giá, đề tài chuyển sang Đang xét duyệt bởi Hội đồng. Sau khi Hội đồng đánh giá và phê duyệt, đề tài sang trạng thái Đã duyệt. Giảng viên bắt đầu thực hiện, đề tài sang Đang thực hiện. Hoàn thành nghiên cứu, giảng viên nộp nghiệm thu cấp Khoa, rồi nghiệm thu cấp Trường. Cuối cùng, sau khi bàn giao sản phẩm, đề tài đạt trạng thái Đã hoàn thành – đây là trạng thái kết thúc thành công.

Bên cạnh luồng chính, hệ thống xử lý năm loại tình huống ngoại lệ, mỗi loại giải quyết một nhu cầu thực tế.

Tình huống yêu cầu chỉnh sửa và nộp lại là tình huống phổ biến nhất. Tại các bước xét duyệt (cấp Khoa, phân bổ Hội đồng, Hội đồng đánh giá), người có thẩm quyền nhận thấy đề cương có tiềm năng nhưng cần bổ sung hoặc chỉnh sửa. Thay vì từ chối hoàn toàn, họ yêu cầu chủ nhiệm sửa rồi nộp lại. Đề tài chuyển sang trạng thái "Đã yêu cầu sửa" kèm ghi chú chi tiết về nội dung cần chỉnh sửa. Điểm đặc biệt trong thiết kế là hệ thống ghi nhận bước đích để nộp lại – nếu Khoa yêu cầu sửa, khi nộp lại đề tài quay về bước xét duyệt Khoa; nếu Hội đồng yêu cầu sửa, khi nộp lại đề tài quay về bước Hội đồng đánh giá, không phải đi lại từ đầu. Cơ chế này phản ánh đúng thực tế: người yêu cầu sửa là người cần xem lại kết quả chỉnh sửa.

Tình huống từ chối xảy ra khi đề tài không đáp ứng yêu cầu và không có tiềm năng cải thiện. Đề tài bị từ chối tại bước xét duyệt Khoa hoặc Hội đồng, chuyển sang trạng thái kết thúc "Đã từ chối" kèm lý do cụ thể. Đây là trạng thái kết thúc – đề tài không thể tiếp tục quy trình.

Tình huống hủy cho phép chủ nhiệm chủ động hủy đề tài khi đang ở trạng thái Nháp – tức là chưa nộp lên bất kỳ cấp nào. Đề tài chuyển sang trạng thái "Đã hủy". Giới hạn chỉ cho phép hủy ở trạng thái Nháp là hợp lý vì khi đã nộp lên Khoa, nhiều bên đã bắt đầu xử lý, việc hủy đơn phương có thể gây lãng phí nguồn lực.

*Hình 3.3. Sơ đồ máy trạng thái workflow*

Tình huống rút hồ sơ cho phép chủ nhiệm rút đề tài khi đang ở trạng thái "Đã yêu cầu sửa" – tức là đề tài đã bị trả lại và chủ nhiệm quyết định không muốn chỉnh sửa và nộp lại nữa. Đề tài chuyển sang trạng thái "Đã rút".

Tình huống tạm dừng và tiếp tục phục vụ các trường hợp đề tài cần dừng tạm thời trong quá trình thực hiện vì lý do khách quan (chủ nhiệm đi học, thiếu kinh phí, dịch bệnh). Hệ thống ghi nhận trạng thái trước khi tạm dừng (ví dụ "Đang thực hiện" hoặc "Đang nghiệm thu cấp Khoa") để khi tiếp tục, đề tài quay về đúng bước đã dừng chứ không phải bắt đầu lại từ đầu.

Mỗi trạng thái đều xác định rõ đơn vị và người đang "nắm giữ" hồ sơ. Khi đề tài ở trạng thái "Đang xét duyệt tại Khoa", hồ sơ thuộc quyền xử lý của Quản lý Khoa tương ứng; khi ở trạng thái "Đang phân bổ Hội đồng", hồ sơ chuyển về Phòng KHCN. Cơ chế gán người/đơn vị nắm giữ hồ sơ là chìa khóa để giải quyết vấn đề "không biết hồ sơ kẹt ở đâu" đã nêu ở phần mở đầu – bất kỳ lúc nào cũng có thể truy vấn để biết ai đang chịu trách nhiệm xử lý đề tài nào.

Để đảm bảo tính toàn vẹn, mỗi lần chuyển trạng thái, hệ thống thực hiện các bước sau trong cùng một giao dịch cơ sở dữ liệu. Bước một, kiểm tra tính hợp lệ: trạng thái hiện tại có cho phép chuyển sang trạng thái đích không (theo bảng chuyển đổi đã định nghĩa), người thực hiện có đủ quyền không. Bước hai, kiểm tra tính bất biến: ngăn xử lý trùng lặp nếu yêu cầu bị gửi hai lần. Bước ba, cập nhật dữ liệu: thay đổi trạng thái đề tài, gán đơn vị/người nắm giữ mới phù hợp với trạng thái mới, tính toán thời hạn SLA mới. Bước bốn, ghi nhật ký workflow với đầy đủ thông tin: trạng thái cũ, trạng thái mới, người thực hiện, thời điểm, ghi chú. Bước năm, ghi nhật ký kiểm toán. Nếu bất kỳ bước nào thất bại, toàn bộ giao dịch bị hủy bỏ, đảm bảo dữ liệu luôn nhất quán.

## 3.6. Thiết kế giao diện người dùng

Giao diện hệ thống tuân theo bố cục phổ biến của ứng dụng quản trị doanh nghiệp hiện đại: thanh điều hướng dọc bên trái, thanh tiêu đề ngang ở trên cùng, và vùng nội dung chính ở giữa. Bố cục này đã trở thành tiêu chuẩn không chính thức cho các hệ thống quản lý vì mang lại nhiều lợi ích: người dùng dễ làm quen do đã quen thuộc từ các hệ thống khác, tận dụng hiệu quả diện tích màn hình, và phù hợp với cả màn hình rộng lẫn màn hình dọc.

Thanh điều hướng bên trái hiển thị menu phân cấp hai mức, chia theo nhóm chức năng. Nội dung menu thay đổi tự động dựa trên vai trò người dùng đang đăng nhập: giảng viên chỉ thấy các mục liên quan đến quản lý đề tài cá nhân, báo cáo tiến độ và xuất biểu mẫu; Quản lý Khoa thấy thêm các mục xét duyệt đề tài thuộc khoa và báo cáo theo khoa; Phòng KHCN thấy toàn bộ chức năng quản lý hội đồng, phân bổ, báo cáo toàn trường và cấu hình SLA; quản trị hệ thống thấy các mục quản lý tài khoản, nhật ký kiểm toán và cấu hình hệ thống. Thiết kế này đảm bảo giao diện gọn gàng, không làm người dùng bối rối với các chức năng không liên quan đến vai trò của họ.

Thanh tiêu đề chứa tên hệ thống, thông tin người dùng hiện tại (tên, vai trò, ảnh đại diện), nút thông báo và nút đăng xuất. Phần thông báo hiển thị các sự kiện quan trọng như đề tài được phê duyệt, yêu cầu chỉnh sửa, hoặc cảnh báo sắp quá hạn xử lý. Mỗi thông báo có liên kết trực tiếp đến đối tượng liên quan, giúp người dùng nhanh chóng tiếp cận công việc cần xử lý.

Vùng nội dung chính thay đổi tùy theo trang đang xem. Đối với trang danh sách (danh sách đề tài, danh sách hội đồng, nhật ký kiểm toán), vùng này hiển thị bảng dữ liệu với phân trang, các cột có thể sắp xếp, bộ lọc nâng cao và ô tìm kiếm. Đối với trang chi tiết đề tài, vùng này hiển thị thông tin đề tài chia thành các tab: thông tin chung, nội dung nghiên cứu, lịch sử xử lý, phiếu đánh giá, và tài liệu đính kèm. Đối với trang biểu mẫu nhập liệu, vùng này hiển thị biểu mẫu động nhiều bước với thanh tiến trình phía trên, các nút điều hướng giữa các bước, và tóm tắt lỗi validation nếu có.

Về responsive design, giao diện được tối ưu cho ba nhóm kích thước màn hình. Trên màn hình lớn (từ 1280 pixel trở lên), hiển thị đầy đủ thanh điều hướng, nội dung chính và có thể có thanh phụ bên phải cho bộ lọc. Trên màn hình trung bình (từ 768 đến 1279 pixel), thanh điều hướng có thể thu gọn chỉ hiển thị icon, tự động mở rộng khi di chuột vào. Trên màn hình nhỏ (dưới 768 pixel), thanh điều hướng ẩn mặc định, chỉ hiện khi bấm nút menu, cho phép sử dụng trên máy tính bảng hoặc điện thoại khi cần tra cứu thông tin khẩn cấp (mặc dù hệ thống chủ yếu phục vụ sử dụng trên máy tính để bàn).

Giao diện theo vai trò được triển khai ở cả phía client và phía server. Ở phía client, khi người dùng đăng nhập, thông tin vai trò và quyền hạn được lưu trong state quản lý bởi Zustand. Mỗi component kiểm tra quyền trước khi hiển thị: nút "Phê duyệt" chỉ hiện với Quản lý Khoa khi xem đề tài thuộc khoa mình; nút "Phân bổ Hội đồng" chỉ hiện với Phòng KHCN; nút "Chấm điểm" chỉ hiện với thành viên Hội đồng được gán cho đề tài đó. Ở phía server, mỗi endpoint API đều có guard kiểm tra quyền, đảm bảo dù người dùng có cố tình gửi yêu cầu trái phép (bằng cách chỉnh sửa mã nguồn client hoặc dùng công cụ như Postman), server vẫn từ chối.

Biểu mẫu nhập liệu được thiết kế với nhiều lớp hỗ trợ người dùng. Thứ nhất, validation thời gian thực: khi người dùng rời khỏi một ô nhập liệu, hệ thống kiểm tra ngay và hiển thị thông báo lỗi nếu dữ liệu không hợp lệ (ví dụ: email sai định dạng, số điện thoại không đủ chữ số, ngày kết thúc trước ngày bắt đầu). Thứ hai, các trường bắt buộc được đánh dấu bằng dấu sao đỏ, giúp người dùng dễ nhận biết. Thứ ba, placeholder và tooltip gợi ý cách điền cho những trường phức tạp. Thứ tư, nút "Lưu nháp" luôn khả dụng, cho phép người dùng lưu tiến độ và quay lại sau mà không cần hoàn thiện toàn bộ biểu mẫu. Thứ năm, khi có lỗi validation ở nhiều ô, trang tự động cuộn đến ô lỗi đầu tiên và hiển thị danh sách tất cả lỗi ở đầu biểu mẫu.

Màu sắc và biểu tượng được sử dụng nhất quán xuyên suốt hệ thống để tạo ngôn ngữ hình ảnh dễ nhận biết. Trạng thái đề tài sử dụng huy hiệu màu: xanh lá cho các trạng thái thành công (Đã duyệt, Đã hoàn thành), vàng cho các trạng thái đang xử lý (Đang xét duyệt, Đang thực hiện), đỏ cho các trạng thái cảnh báo hoặc kết thúc không thành công (Quá hạn, Đã từ chối, Đã hủy), xám cho các trạng thái tạm thời (Nháp, Đã tạm dừng). Các hành động sử dụng biểu tượng chuẩn: mắt để xem chi tiết, bút chì để chỉnh sửa, kiểm để phê duyệt, đồng hồ cát để xem lịch sử, tải xuống để xuất tài liệu. Bảng điều khiển sử dụng biểu đồ cột và tròn để trực quan hóa số liệu thống kê, với màu sắc nhất quán với ý nghĩa trạng thái.

# CHƯƠNG 4: CÀI ĐẶT VÀ THỬ NGHIỆM

## 4.1. Môi trường phát triển và triển khai

Hệ thống được phát triển trên nền tảng monorepo sử dụng Nx, một công cụ quản lý monorepo tiên tiến cho các dự án JavaScript và TypeScript. Lựa chọn monorepo thay vì tách riêng từng thành phần ra các repository khác nhau xuất phát từ ba lợi ích chính. Thứ nhất, khả năng chia sẻ mã nguồn: các định nghĩa kiểu dữ liệu, hằng số, hàm tiện ích và quy tắc validation được viết một lần và sử dụng ở cả backend lẫn frontend, loại bỏ hoàn toàn rủi ro không đồng bộ. Thứ hai, quản lý phụ thuộc đơn giản: toàn bộ dự án có một file quản lý thư viện duy nhất, đảm bảo tất cả các thành phần dùng cùng phiên bản của các thư viện chung, tránh xung đột phiên bản. Thứ ba, khả năng refactor an toàn: khi thay đổi cấu trúc dữ liệu hoặc chữ ký hàm, các công cụ phát triển có thể phát hiện ngay tất cả nơi bị ảnh hưởng xuyên suốt cả backend và frontend.

Môi trường phát triển bao gồm các công cụ chính. Node.js phiên bản mười tám hoặc cao hơn làm nền tảng chạy ứng dụng. PostgreSQL phiên bản mười bốn chạy trong Docker container, được khởi động bằng một lệnh duy nhất qua Docker Compose, đảm bảo mọi lập trình viên trong nhóm đều làm việc với cùng phiên bản cơ sở dữ liệu và cấu hình giống hệt nhau. Prisma CLI được sử dụng để quản lý migration cơ sở dữ liệu – mỗi khi thay đổi cấu trúc bảng, lập trình viên chạy lệnh tạo migration mới, Prisma tự động sinh ra câu lệnh SQL tương ứng và lưu thành file, cho phép áp dụng cùng thay đổi lên cơ sở dữ liệu trên các môi trường khác nhau (development, staging, production). Git được dùng để quản lý phiên bản mã nguồn, với quy ước commit rõ ràng và branching strategy phù hợp với nhóm nhỏ. Visual Studio Code là trình soạn thảo chính, được cấu hình với các extension hỗ trợ TypeScript, Prisma và React để cung cấp gợi ý mã nguồn thông minh và phát hiện lỗi sớm.

Quy trình phát triển điển hình diễn ra như sau. Lập trình viên bắt đầu bằng việc định nghĩa hoặc chỉnh sửa schema Prisma để phản ánh yêu cầu mới. Chạy lệnh migration để tạo và áp dụng thay đổi lên cơ sở dữ liệu local, đồng thời Prisma tự động sinh lại các kiểu dữ liệu TypeScript tương ứng. Cài đặt logic nghiệp vụ ở backend: viết service xử lý nghiệp vụ, controller tiếp nhận yêu cầu HTTP, và DTO định nghĩa cấu trúc đầu vào/đầu ra với các quy tắc validation. Cài đặt giao diện ở frontend: viết component React hiển thị dữ liệu và thu thập đầu vào, sử dụng TanStack Query để gọi API, và Zustand để quản lý trạng thái phía client nếu cần. Kiểm thử thủ công bằng cách chạy backend và frontend trên máy local, sử dụng trình duyệt để kiểm tra chức năng hoạt động đúng, và công cụ như Postman để kiểm tra API trực tiếp. Commit mã nguồn vào Git với message mô tả rõ ràng thay đổi.

Về kết nối giữa các thành phần, backend NestJS và frontend React giao tiếp qua HTTP REST API. Backend chạy trên cổng 3000, cung cấp các endpoint theo chuẩn REST. Frontend chạy trên cổng 4200 trong quá trình phát triển, được cấu hình proxy để chuyển tiếp các yêu cầu API đến cổng 3000, tránh vấn đề CORS (Cross-Origin Resource Sharing). Khi triển khai production, frontend được build thành các file tĩnh (HTML, CSS, JavaScript) và được phục vụ bởi một web server như Nginx; Nginx cũng đóng vai trò reverse proxy, chuyển tiếp các yêu cầu đến backend.

Backend kết nối với PostgreSQL thông qua Prisma Client, sử dụng connection pool với cấu hình mặc định mười kết nối đồng thời – đủ để xử lý tải của một trường đại học cỡ trung với vài chục người dùng đồng thời. Connection string được cấu hình qua biến môi trường, cho phép dễ dàng thay đổi khi chuyển môi trường mà không cần sửa mã nguồn.

Backend giao tiếp với Form Engine qua HTTP. Khi cần sinh tài liệu, backend gửi yêu cầu POST đến endpoint của Form Engine kèm dữ liệu đề tài dưới dạng JSON. Form Engine xử lý yêu cầu, tạo file DOCX và PDF, lưu vào thư mục được chia sẻ giữa hai dịch vụ (qua Docker volume), và trả về đường dẫn file cho backend. Backend sau đó tạo bản ghi trong bảng Tài liệu đã sinh và trả URL tải về cho frontend.

Docker Compose được sử dụng để điều phối toàn bộ hệ thống. File cấu hình mô tả bốn dịch vụ: PostgreSQL với volume lưu trữ dữ liệu, backend NestJS với các biến môi trường cần thiết, frontend React (chỉ trong môi trường development), và Form Engine Python. Một lệnh duy nhất khởi động toàn bộ hệ thống, bao gồm cả việc tạo mạng nội bộ cho các container giao tiếp với nhau. Cách tiếp cận này giúp một lập trình viên mới tham gia dự án có thể chạy toàn bộ hệ thống trên máy cá nhân chỉ trong vài phút, mà không cần cài đặt thủ công từng thành phần.

## 4.2. Kết quả cài đặt các chức năng chính

Hệ thống hoàn chỉnh bao gồm tám nhóm chức năng chính, mỗi nhóm được cài đặt với sự chú trọng đặc biệt vào tính đúng đắn, hiệu năng và trải nghiệm người dùng.

Chức năng quản lý đề tài cung cấp giao diện toàn diện cho việc tạo mới, chỉnh sửa, tìm kiếm và xem chi tiết đề tài. Trang danh sách đề tài hiển thị bảng dữ liệu với phân trang, mỗi trang hai mươi bản ghi, giúp tải nhanh ngay cả khi có hàng nghìn đề tài trong hệ thống. Các cột chính bao gồm mã đề tài, tiêu đề, chủ nhiệm, khoa, trạng thái (hiển thị dưới dạng huy hiệu màu), và hạn chót SLA (hiển thị màu đỏ nếu quá hạn, vàng nếu còn dưới ba ngày). Bộ lọc nâng cao cho phép lọc đồng thời theo nhiều tiêu chí: trạng thái (cho phép chọn nhiều trạng thái cùng lúc), khoa, chủ nhiệm, năm tạo, và tình trạng SLA (đúng hạn, sắp hết hạn, quá hạn). Ô tìm kiếm toàn văn cho phép gõ từ khóa và hệ thống tìm kiếm trong tiêu đề, tóm tắt và nội dung đề tài. Trang chi tiết đề tài hiển thị toàn bộ thông tin được tổ chức thành các tab: thông tin cơ bản, nội dung nghiên cứu, kế hoạch thực hiện, dự toán kinh phí, lịch sử xử lý (dòng thời gian trực quan hiển thị các mốc quan trọng), phiếu đánh giá (nếu đã được đánh giá), và tài liệu đính kèm. Biểu mẫu tạo và chỉnh sửa đề tài được thiết kế dạng wizard nhiều bước với thanh tiến trình, cho phép người dùng điền thông tin từng phần, lưu nháp bất kỳ lúc nào, và chỉ cho phép nộp khi tất cả các trường bắt buộc đã được điền đầy đủ và hợp lệ.

Chức năng xử lý quy trình là trái tim của hệ thống, triển khai máy trạng thái với mười lăm trạng thái và hơn hai mươi chuyển đổi hợp lệ. Mỗi hành động chuyển trạng thái (nộp, duyệt, yêu cầu sửa, từ chối, hủy, rút, tạm dừng, tiếp tục) được cài đặt như một endpoint riêng với logic nghiệp vụ cụ thể. Trước khi thực hiện chuyển đổi, hệ thống kiểm tra ba điều kiện: trạng thái hiện tại có cho phép chuyển sang trạng thái đích không, người thực hiện có quyền thực hiện hành động này không, và các điều kiện tiên quyết có được đáp ứng không (ví dụ: chỉ cho phép chuyển sang trạng thái Đang xét duyệt bởi Hội đồng nếu đã có Hội đồng được gán). Khi chuyển trạng thái, hệ thống tự động thực hiện các tác vụ phụ: cập nhật đơn vị và người nắm giữ hồ sơ, tính toán thời hạn SLA mới dựa trên lịch làm việc, ghi nhật ký workflow, ghi nhật ký kiểm toán, và gửi thông báo đến người liên quan (tính năng thông báo được thiết kế nhưng chưa kích hoạt trong phiên bản hiện tại). Toàn bộ quá trình diễn ra trong một giao dịch cơ sở dữ liệu, đảm bảo tính nguyên tử: nếu bất kỳ bước nào thất bại, không có thay đổi nào được lưu vào cơ sở dữ liệu. Giao diện hiển thị rõ ràng các hành động khả dụng ở mỗi trạng thái: với giảng viên khi đề tài ở trạng thái Nháp, hiện nút Nộp và Hủy; với Quản lý Khoa khi đề tài ở trạng thái Đang xét duyệt tại Khoa, hiện nút Phê duyệt, Yêu cầu sửa và Từ chối, kèm ô văn bản cho ghi chú.

Chức năng quản lý Hội đồng cho phép Phòng KHCN tạo và quản lý các Hội đồng đánh giá. Trang danh sách Hội đồng hiển thị tất cả các Hội đồng đã tạo với thông tin tên, loại, số lượng thành viên, và số lượng đề tài được gán. Biểu mẫu tạo Hội đồng mới cho phép nhập tên, chọn loại (xét duyệt đề cương, nghiệm thu cấp Khoa, nghiệm thu cấp Trường, gia hạn), và phân công thành viên. Giao diện phân công thành viên được thiết kế trực quan: chia thành hai cột, cột trái là danh sách người dùng khả dụng (có thể lọc theo khoa, tìm kiếm theo tên), cột phải là danh sách thành viên đã chọn kèm vai trò (chủ tịch, thư ký, ủy viên, phản biện). Người dùng kéo thả hoặc bấm nút để thêm/bớt thành viên, chọn vai trò từ dropdown. Hệ thống kiểm tra ràng buộc: một Hội đồng phải có ít nhất một chủ tịch, một thư ký và một phản biện; một người có thể đồng thời là thành viên của nhiều Hội đồng nhưng mỗi người chỉ được gán một vai trò trong mỗi Hội đồng. Khi gán Hội đồng cho đề tài, hệ thống kiểm tra xung đột: không cho phép gán Hội đồng mà chủ tịch hoặc phản biện là chính chủ nhiệm đề tài đó (để tránh xung đột lợi ích).

Chức năng đánh giá đề tài cung cấp giao diện cho thành viên Hội đồng chấm điểm. Mỗi thành viên nhìn thấy danh sách các đề tài mình cần đánh giá, với chỉ báo rõ ràng đề tài nào đã chấm, đề tài nào chưa. Khi mở phiếu đánh giá, giao diện hiển thị thông tin đề tài ở phía trên (tiêu đề, chủ nhiệm, tóm tắt nội dung) và biểu mẫu đánh giá ở phía dưới. Biểu mẫu đánh giá gồm bốn tiêu chí, mỗi tiêu chí có thanh trượt cho phép chấm từ 0 đến 25 điểm, kèm ô văn bản cho nhận xét chi tiết. Tổng điểm được tính tự động và hiển thị nổi bật. Phiếu đánh giá có hai trạng thái: Nháp (cho phép lưu tạm và quay lại chỉnh sửa) và Hoàn tất (không thể chỉnh sửa nữa). Thành viên Hội đồng không nhìn thấy phiếu đánh giá của người khác cho đến khi tất cả thành viên đã hoàn tất phiếu của mình, đảm bảo tính độc lập trong đánh giá. Sau khi tất cả phiếu đánh giá hoàn tất, thư ký Hội đồng tổng hợp kết quả: hệ thống tự động tính điểm trung bình từ các phiếu, hiển thị phân bố điểm của từng thành viên, và cho phép thư ký nhập biên bản họp Hội đồng. Đề tài được coi là đạt yêu cầu nếu điểm trung bình từ sáu mươi điểm trở lên (trên thang điểm một trăm).

Chức năng sinh biểu mẫu tự động tích hợp chặt chẽ với dịch vụ Form Engine. Từ trang chi tiết đề tài, người dùng bấm nút Xuất tài liệu, hệ thống hiển thị danh sách mười tám loại biểu mẫu khả dụng. Khi chọn loại biểu mẫu, frontend gửi yêu cầu đến backend kèm mã đề tài và mã biểu mẫu. Backend truy vấn toàn bộ dữ liệu liên quan đến đề tài (thông tin chủ nhiệm, thông tin khoa, dữ liệu biểu mẫu, kết quả đánh giá, thành viên Hội đồng, lịch sử workflow), đóng gói thành một đối tượng JSON, và gửi đến Form Engine qua HTTP POST. Form Engine nhận dữ liệu, xác định file mẫu Word tương ứng, mở mẫu bằng thư viện python-docx, duyệt qua từng đoạn văn và bảng biểu để tìm và thay thế các trường dữ liệu (đánh dấu bằng cú pháp dấu ngoặc nhọn kép, ví dụ tiêu đề đề tài, tên chủ nhiệm, ngày bắt đầu), tạo file DOCX mới, chuyển đổi sang PDF bằng thư viện unoconv hoặc libreoffice headless, tính mã băm SHA-256 cho cả hai file để xác minh tính toàn vẹn, lưu file vào thư mục chia sẻ, và trả về đường dẫn tải về cho backend. Backend tạo bản ghi trong bảng Tài liệu đã sinh, lưu manifest (bản kê) chứa toàn bộ dữ liệu đã sử dụng để sinh tài liệu, ghi nhật ký kiểm toán, và trả URL tải về cho frontend. Frontend hiển thị liên kết tải về, khi người dùng bấm vào, file được tải trực tiếp từ server. Toàn bộ quá trình từ lúc bấm nút đến lúc hiển thị liên kết tải về mất khoảng hai đến năm giây tùy độ phức tạp của biểu mẫu.

Chức năng bảng điều khiển hiển thị tổng quan phù hợp với từng vai trò. Giảng viên nhìn thấy số lượng đề tài của mình theo từng trạng thái, danh sách đề tài đang xử lý hoặc cần chú ý (có yêu cầu chỉnh sửa, sắp hết hạn), và biểu đồ tiến độ theo thời gian. Quản lý Khoa nhìn thấy tổng quan các đề tài thuộc khoa: số lượng theo trạng thái, tỷ lệ hoàn thành đúng hạn, danh sách đề tài đang chờ xét duyệt, và so sánh năng suất nghiên cứu giữa các năm. Phòng KHCN nhìn thấy bức tranh toàn cảnh toàn trường: tổng số đề tài đang hoạt động, phân bố theo khoa (biểu đồ tròn), phân bố theo trạng thái (biểu đồ cột), xu hướng theo thời gian, danh sách đề tài quá hạn và sắp quá hạn, và các chỉ số quan trọng như thời gian xử lý trung bình ở mỗi bước. Ban Giám hiệu nhìn thấy báo cáo tổng hợp cấp cao với các chỉ số chính và biểu đồ xu hướng nhiều năm. Tất cả biểu đồ được cài đặt bằng thư viện Recharts, hỗ trợ tương tác (di chuột để xem chi tiết từng điểm dữ liệu) và responsive (tự động điều chỉnh kích thước theo màn hình).

Chức năng quản lý SLA tự động hóa việc tính toán và giám sát thời hạn xử lý. Quản trị viên cấu hình lịch làm việc thông qua giao diện dạng lịch, đánh dấu các ngày nghỉ lễ, ngày nghỉ bù, và ngày làm bù. Hệ thống lưu thông tin này vào bảng Lịch làm việc. Khi một đề tài chuyển sang trạng thái mới, hệ thống tra cứu quy định thời hạn xử lý cho trạng thái đó (ví dụ: Đang xét duyệt tại Khoa có hạn năm ngày làm việc, Đang xét duyệt bởi Hội đồng có hạn mười ngày làm việc), tính toán ngày hết hạn bằng cách cộng dồn từng ngày làm việc (bỏ qua thứ bảy, chủ nhật và các ngày nghỉ lễ), và lưu vào trường thời hạn SLA của đề tài. Một tiến trình nền chạy định kỳ (mỗi giờ hoặc mỗi ngày) để quét tất cả đề tài đang hoạt động, so sánh thời hạn SLA với thời điểm hiện tại, và đánh dấu tình trạng SLA: Đúng hạn (còn nhiều hơn ba ngày), Sắp hết hạn (còn từ một đến ba ngày), và Quá hạn (đã qua thời hạn). Danh sách đề tài quá hạn và sắp hết hạn được hiển thị nổi bật trên bảng điều khiển của Phòng KHCN và người phụ trách tương ứng, kèm số ngày trễ hoặc số ngày còn lại.

*Hình 4.1. Giao diện đăng nhập*

Chức năng quản trị hệ thống cung cấp các công cụ cho quản trị viên. Quản lý tài khoản cho phép tạo tài khoản mới (nhập thủ công hoặc import hàng loạt từ file Excel), chỉnh sửa thông tin, đặt lại mật khẩu, khóa/mở khóa tài khoản, và chuyển đổi vai trò. Giao diện import hàng loạt hỗ trợ tải xuống file Excel mẫu, điền thông tin theo mẫu, tải lên, hệ thống kiểm tra tính hợp lệ từng dòng (email đúng định dạng, email không trùng, vai trò hợp lệ, khoa tồn tại), hiển thị báo cáo lỗi nếu có, và cho phép xác nhận import khi tất cả dữ liệu hợp lệ. Quản lý mẫu biểu mẫu cho phép tải lên file mẫu Word mới, cập nhật mẫu hiện có (khi có thay đổi về định dạng hoặc nội dung cố định), và xem lịch sử thay đổi mẫu. Xem nhật ký kiểm toán cung cấp giao diện tìm kiếm và lọc mạnh mẽ: lọc theo người thực hiện, loại hành động, đối tượng bị tác động, khoảng thời gian, và địa chỉ IP; xuất kết quả tìm kiếm ra file CSV phục vụ phân tích hoặc lưu trữ lâu dài. Nhật ký kiểm toán được thiết kế không cho phép xóa hoặc sửa, chỉ cho phép đọc và xuất, đảm bảo tính toàn vẹn.

## 4.3. Thử nghiệm và đánh giá

Hệ thống được thử nghiệm qua hai loại kịch bản chính: kịch bản end-to-end mô phỏng quy trình thành công từ đầu đến cuối, và kịch bản ngoại lệ kiểm tra xử lý các tình huống đặc biệt.

Kịch bản end-to-end thứ nhất mô phỏng một đề tài đi qua toàn bộ quy trình lý tưởng. Bước một, giảng viên Nguyễn Văn A thuộc Khoa Công nghệ Thông tin đăng nhập vào hệ thống, tạo đề tài mới với tiêu đề "Ứng dụng trí tuệ nhân tạo trong chẩn đoán hình ảnh y tế", điền đầy đủ các phần thông tin qua biểu mẫu động (mục tiêu, phương pháp, kết quả dự kiến, tiến độ, dự toán kinh phí), lưu nháp, sau đó nộp hồ sơ. Hệ thống kiểm tra tính đầy đủ, chuyển đề tài sang trạng thái Đang xét duyệt tại Khoa, ghi nhật ký workflow và kiểm toán, tính thời hạn SLA (năm ngày làm việc kể từ ngày nộp). Bước hai, Trưởng Khoa Công nghệ Thông tin đăng nhập, thấy đề tài trong danh sách chờ duyệt, xem chi tiết, đánh giá nội dung, bấm nút Phê duyệt kèm nhận xét "Đề tài phù hợp với định hướng nghiên cứu của khoa, đồng ý đề xuất". Hệ thống chuyển đề tài sang trạng thái Đang phân bổ Hội đồng, gán đơn vị nắm giữ là Phòng KHCN, tính thời hạn SLA mới (ba ngày làm việc). Bước ba, cán bộ Phòng KHCN đăng nhập, thấy đề tài trong danh sách cần phân bổ Hội đồng, chọn Hội đồng xét duyệt đề cương số 01/2024 (đã được tạo trước đó với năm thành viên), gán Hội đồng cho đề tài. Hệ thống chuyển đề tài sang trạng thái Đang xét duyệt bởi Hội đồng, gửi thông báo đến các thành viên Hội đồng, tính thời hạn SLA mới (mười ngày làm việc). Bước bốn, năm thành viên Hội đồng lần lượt đăng nhập, mở phiếu đánh giá, chấm điểm theo bốn tiêu chí (tính cấp thiết, tính khoa học, tính khả thi, dự kiến tác động), nhập nhận xét, và hoàn tất phiếu. Hệ thống tính điểm trung bình (giả sử đạt bảy mươi lăm điểm trên một trăm). Bước năm, thư ký Hội đồng tổng hợp kết quả, nhập biên bản họp, bấm nút Phê duyệt. Hệ thống chuyển đề tài sang trạng thái Đã duyệt, gán lại cho chủ nhiệm, cho phép chủ nhiệm bắt đầu thực hiện. Bước sáu, giảng viên Nguyễn Văn A bấm nút Bắt đầu thực hiện, đề tài chuyển sang trạng thái Đang thực hiện. Sau sáu tháng (mô phỏng bằng cách thay đổi thời gian hệ thống), giảng viên hoàn thành nghiên cứu, bấm nút Nộp nghiệm thu cấp Khoa, đề tài chuyển sang trạng thái tương ứng. Bước bảy, quy trình nghiệm thu diễn ra tương tự xét duyệt: Khoa tổ chức Hội đồng nghiệm thu cấp Khoa, đánh giá, thông qua, đề tài chuyển sang Nộp nghiệm thu cấp Trường. Bước tám, Phòng KHCN tổ chức Hội đồng nghiệm thu cấp Trường, đánh giá, thông qua. Bước chín, giảng viên bàn giao sản phẩm (báo cáo tổng kết, mã nguồn phần mềm nếu có, bài báo khoa học), bấm nút Bàn giao sản phẩm, đề tài chuyển sang trạng thái Đã hoàn thành. Trong suốt quá trình, hệ thống ghi nhận mười ba lần chuyển trạng thái, tương ứng mười ba bản ghi nhật ký workflow, và hơn ba mươi bản ghi nhật ký kiểm toán (bao gồm cả các thao tác đăng nhập, xem chi tiết, xuất tài liệu). Giảng viên có thể xuất bất kỳ biểu mẫu nào trong mười tám loại tại các thời điểm khác nhau, mỗi lần xuất mất khoảng ba giây và tạo ra cả file DOCX lẫn PDF với nội dung đúng và định dạng chuẩn.

Kịch bản ngoại lệ thứ nhất kiểm tra tình huống yêu cầu chỉnh sửa và nộp lại. Giảng viên nộp đề tài, Khoa xét duyệt nhận thấy phần phương pháp nghiên cứu chưa rõ ràng, bấm nút Yêu cầu sửa kèm ghi chú "Cần bổ sung chi tiết về phương pháp thu thập dữ liệu và tiêu chí đánh giá kết quả". Hệ thống chuyển đề tài sang trạng thái Đã yêu cầu sửa, ghi nhận trạng thái đích để nộp lại là Đang xét duyệt tại Khoa. Giảng viên nhận thông báo, mở đề tài, thấy ghi chú yêu cầu sửa, chỉnh sửa phần phương pháp nghiên cứu, bấm nút Nộp lại. Hệ thống chuyển đề tài quay về trạng thái Đang xét duyệt tại Khoa (không phải Đang xét duyệt bởi Hội đồng hay bước nào khác), Quản lý Khoa xem lại phần đã sửa, hài lòng, phê duyệt. Đề tài tiếp tục quy trình bình thường. Kịch bản xác nhận rằng cơ chế ghi nhận trạng thái đích hoạt động đúng.

Kịch bản ngoại lệ thứ hai kiểm tra tình huống từ chối. Giảng viên nộp đề tài, Hội đồng đánh giá và nhận thấy đề tài không khả thi (dự toán kinh phí quá cao so với nguồn lực của trường, phương pháp chưa phù hợp), bấm nút Từ chối kèm nhận xét chi tiết. Hệ thống chuyển đề tài sang trạng thái Đã từ chối, không cho phép bất kỳ hành động nào khác ngoài xem lại thông tin và xuất biểu mẫu để lưu hồ sơ. Kịch bản xác nhận rằng trạng thái kết thúc không cho phép chuyển tiếp.

Kịch bản ngoại lệ thứ ba kiểm tra tình huống tạm dừng và tiếp tục. Đề tài đang ở trạng thái Đang thực hiện, chủ nhiệm gặp khó khăn cá nhân (ốm dài ngày, hoặc được cử đi đào tạo), nộp đơn xin tạm dừng. Quản lý Khoa phê duyệt đơn tạm dừng, hệ thống ghi nhận trạng thái trước khi tạm dừng là Đang thực hiện, chuyển đề tài sang trạng thái Đã tạm dừng. Sau ba tháng, khi điều kiện cho phép, chủ nhiệm nộp đơn xin tiếp tục. Quản lý Khoa phê duyệt, hệ thống tra cứu trạng thái trước khi tạm dừng, khôi phục đề tài về trạng thái Đang thực hiện. Kịch bản xác nhận cơ chế lưu và khôi phục trạng thái hoạt động đúng.

Kịch bản ngoại lệ thứ tư kiểm tra xử lý lỗi và ràng buộc. Thử tạo tài khoản với email đã tồn tại, hệ thống từ chối và hiển thị thông báo lỗi rõ ràng. Thử phân bổ Hội đồng mà chủ tịch là chính chủ nhiệm đề tài, hệ thống từ chối và giải thích lý do xung đột lợi ích. Thử chuyển đề tài từ trạng thái Nháp trực tiếp sang Đã duyệt (bỏ qua các bước trung gian) bằng cách gửi yêu cầu API thủ công qua Postman, hệ thống từ chối vì vi phạm quy tắc chuyển đổi trạng thái. Thử đăng nhập với tài khoản bị khóa, hệ thống từ chối. Thử truy cập chức năng không có quyền (ví dụ giảng viên cố truy cập trang quản lý tài khoản), hệ thống chuyển hướng về trang chủ kèm thông báo "Bạn không có quyền truy cập trang này". Tất cả các trường hợp đều được xử lý an toàn, không gây ra lỗi hệ thống (crash) hay để lộ thông tin nhạy cảm.

Kết quả thử nghiệm được tổng hợp trong bảng sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Nhóm chức năng** | **Kịch bản kiểm thử** | **Kết quả** | **Ghi chú** |
| 1 | Quản lý đề tài | Tạo, sửa, xóa nháp, tìm kiếm, lọc | Đạt | Tất cả thao tác hoạt động đúng, lọc phức tạp trả về kết quả chính xác trong dưới 500ms |
| 2 | Xử lý quy trình – luồng chính | Quy trình từ nộp đến hoàn thành | Đạt | Mười ba lần chuyển trạng thái, tất cả nhật ký được ghi đầy đủ |
| 3 | Xử lý quy trình – yêu cầu sửa | Trả lại, sửa, nộp lại đúng bước | Đạt | Đề tài quay về đúng bước yêu cầu sửa |
| 4 | Xử lý quy trình – từ chối | Từ chối tại Khoa và Hội đồng | Đạt | Đề tài chuyển sang trạng thái kết thúc, không cho phép tiếp tục |
| 5 | Xử lý quy trình – tạm dừng | Tạm dừng và tiếp tục | Đạt | Khôi phục đúng trạng thái trước khi tạm dừng |
| 6 | Quản lý Hội đồng | Tạo, phân công, gán cho đề tài | Đạt | Kiểm tra xung đột lợi ích hoạt động đúng |
| 7 | Đánh giá | Năm thành viên chấm điểm độc lập | Đạt | Phiếu của người khác không nhìn thấy cho đến khi hoàn tất, điểm trung bình tính đúng |
| 8 | Sinh biểu mẫu | Mười tám loại biểu mẫu | Đạt | Tất cả biểu mẫu sinh đúng nội dung và định dạng, thời gian sinh từ 2-5 giây |
| 9 | Bảng điều khiển | Hiển thị theo từng vai trò | Đạt | Mỗi vai trò nhìn thấy đúng dữ liệu phạm vi quyền hạn |
| 10 | SLA | Tính thời hạn, cảnh báo quá hạn | Đạt | Bỏ qua đúng ngày nghỉ, tính đúng ngày làm việc, cảnh báo hiển thị kịp thời |
| 11 | Nhật ký kiểm toán | Ghi lại mọi thao tác quan trọng | Đạt | Hơn ba mươi bản ghi cho một quy trình hoàn chỉnh, đầy đủ thông tin bối cảnh |
| 12 | Quản lý tài khoản | Tạo, sửa, khóa, import hàng loạt | Đạt | Import từ Excel với 50 tài khoản hoàn tất trong 10 giây, kiểm tra lỗi chính xác |
| 13 | Phân quyền | Kiểm tra quyền ở cả client và server | Đạt | Không thể bypass bằng cách sửa mã client hoặc gọi API trực tiếp |
| 14 | Xử lý lỗi | Email trùng, xung đột, trạng thái không hợp lệ | Đạt | Tất cả lỗi được xử lý an toàn với thông báo rõ ràng |
| 15 | Hiệu năng | Tải trang với 1000 đề tài, tìm kiếm | Đạt | Tải danh sách dưới 500ms, tìm kiếm toàn văn dưới 300ms nhờ chỉ mục PostgreSQL |

Qua thử nghiệm, hệ thống hoạt động ổn định, đáp ứng đầy đủ các yêu cầu chức năng và phi chức năng đã đặt ra. Tất cả mười lăm nhóm kịch bản kiểm thử đều cho kết quả đạt yêu cầu. Một số điểm đáng chú ý: thời gian phản hồi của các thao tác phổ biến đều dưới nửa giây, mang lại trải nghiệm người dùng mượt mà; cơ chế xử lý lỗi toàn diện, không có trường hợp nào gây crash hệ thống hoặc để lộ thông tin nhạy cảm; nhật ký kiểm toán ghi lại đầy đủ, chi tiết, đáp ứng yêu cầu truy vết; chức năng sinh biểu mẫu tự động hoạt động ổn định, tạo ra tài liệu đúng định dạng và nội dung.

# KẾT LUẬN

Đề tài "Xây dựng hệ thống quản lý nghiên cứu khoa học cho Trường Đại học Sư phạm Kỹ thuật Nam Định" đã hoàn thành mục tiêu đề ra: số hóa toàn bộ quy trình quản lý đề tài NCKH từ khi đề xuất đến khi hoàn thành, thay thế phương thức thủ công bằng hệ thống phần mềm có quy trình rõ ràng, trạng thái minh bạch và khả năng theo dõi tự động. Hệ thống triển khai thành công bốn mục tiêu chính đã xác định trong phần mở đầu.

Thứ nhất, hệ thống web hoàn chỉnh cho phép quản lý toàn bộ vòng đời đề tài NCKH đã được xây dựng với kiến trúc ba tầng kết hợp microservice, sử dụng công nghệ hiện đại và phù hợp với yêu cầu thực tế. Backend NestJS với hơn hai mươi bảy module được tổ chức theo kiến trúc rõ ràng, dễ bảo trì và mở rộng. Frontend React dạng SPA mang lại trải nghiệm người dùng mượt mà, phản hồi nhanh. Cơ sở dữ liệu PostgreSQL với mười tám bảng được thiết kế chuẩn hóa, đảm bảo tính toàn vẹn dữ liệu thông qua các ràng buộc khóa ngoại và kiểm tra.

Thứ hai, máy trạng thái với mười lăm trạng thái và hơn hai mươi chuyển đổi hợp lệ đã tự động hóa quy trình xử lý, đảm bảo mỗi đề tài chỉ có thể chuyển sang bước tiếp theo khi đủ điều kiện, loại bỏ khả năng bỏ sót bước hoặc xử lý sai quy trình. Cơ chế này không chỉ áp dụng cho luồng xử lý chính mà còn bao quát năm loại tình huống ngoại lệ: yêu cầu chỉnh sửa, từ chối, hủy, rút và tạm dừng/tiếp tục. Mỗi lần chuyển trạng thái đều được thực hiện trong một giao dịch cơ sở dữ liệu, kèm theo các tác vụ phụ như ghi nhật ký, tính SLA và cập nhật người/đơn vị nắm giữ hồ sơ, đảm bảo tính nguyên tử và nhất quán.

Thứ ba, chức năng sinh tài liệu tự động cho mười tám loại biểu mẫu đã được tích hợp thông qua dịch vụ Form Engine riêng biệt viết bằng Python, tận dụng hệ sinh thái thư viện xử lý tài liệu phong phú. Người dùng chỉ cần bấm nút chọn loại biểu mẫu, hệ thống tự động điền dữ liệu vào mẫu Word, tạo cả file DOCX lẫn PDF với định dạng chuẩn, tính mã băm để xác minh tính toàn vẹn, và lưu manifest để truy vết. Chức năng này giảm thiểu đáng kể thời gian soạn thảo thủ công (từ hàng giờ xuống còn vài giây), loại bỏ lỗi do nhập liệu thủ công, và đảm bảo tính thống nhất của các văn bản hành chính.

Thứ tư, hệ thống phân quyền theo vai trò với tám nhóm người dùng và hơn năm mươi quyền hạn chi tiết đã được triển khai ở cả tầng giao diện và tầng server. Mỗi người dùng chỉ được truy cập và thao tác trong phạm vi quyền hạn của mình, vừa bảo vệ dữ liệu vừa đơn giản hóa giao diện sử dụng. Giao diện thay đổi động theo vai trò, hiển thị đúng chức năng và dữ liệu phù hợp với từng người. Cơ chế kiểm tra kép (client và server) đảm bảo an toàn ngay cả khi người dùng cố tình bypass kiểm tra phía client.

Bên cạnh những kết quả đạt được, hệ thống vẫn còn một số hạn chế cần khắc phục trong tương lai. Hạn chế thứ nhất là chưa tích hợp với các hệ thống hiện có của nhà trường như hệ thống quản lý nhân sự, hệ thống tài chính kế toán, và hệ thống thư viện. Việc tích hợp sẽ giúp đồng bộ dữ liệu giảng viên, tự động hóa quy trình thanh toán kinh phí đề tài, và kết nối với kho tài liệu số. Hạn chế thứ hai là chưa có module quản lý chi tiết kinh phí đề tài – theo dõi dự toán, thực chi, quyết toán và báo cáo tài chính. Module này cần được bổ sung để hệ thống trở thành giải pháp toàn diện, thay thế hoàn toàn bảng tính Excel hiện đang dùng để quản lý kinh phí. Hạn chế thứ ba là chưa hỗ trợ ứng dụng di động, hạn chế khả năng truy cập khi người dùng đang di chuyển. Tuy nhiên, do quy trình quản lý NCKH chủ yếu diễn ra trong giờ hành chính tại văn phòng, hạn chế này không ảnh hưởng nhiều đến sử dụng thực tế. Hạn chế thứ tư là chưa có cơ chế thông báo tự động qua email hoặc tin nhắn SMS khi có sự kiện quan trọng (đề tài được phê duyệt, yêu cầu chỉnh sửa, sắp quá hạn). Tính năng thông báo đã được thiết kế nhưng chưa kích hoạt vì cần cấu hình máy chủ email và đàm phán với nhà cung cấp dịch vụ SMS.

Hướng phát triển trong ngắn hạn (từ ba đến sáu tháng tới) bao gồm: triển khai thử nghiệm hệ thống tại một khoa thí điểm, thu thập phản hồi từ người dùng thực tế, điều chỉnh giao diện và quy trình cho phù hợp với thói quen làm việc; kích hoạt chức năng thông báo tự động qua email, giúp người dùng nhận được cảnh báo kịp thời; và bổ sung báo cáo thống kê nâng cao với nhiều chiều phân tích hơn (theo lĩnh vực nghiên cứu, theo nguồn kinh phí, theo mức độ hoàn thành). Hướng phát triển trong trung hạn (từ sáu tháng đến một năm) bao gồm: xây dựng module quản lý kinh phí đề tài với các chức năng quản lý dự toán chi tiết, theo dõi thực chi theo từng khoản mục, quyết toán và đối chiếu với hệ thống tài chính; tích hợp với hệ thống quản lý nhân sự để đồng bộ thông tin giảng viên, học vị, chức danh và lịch sử công tác; tích hợp với hệ thống thư viện để lưu trữ và quản lý sản phẩm nghiên cứu (báo cáo, bài báo, mã nguồn phần mềm) dưới dạng tài liệu số có thể tra cứu và tải về. Hướng phát triển trong dài hạn (từ một đến hai năm) bao gồm: phát triển ứng dụng di động (iOS và Android) cho phép tra cứu thông tin đề tài, xem lịch sử xử lý, và nhận thông báo đẩy khi có sự kiện quan trọng; mở rộng hệ thống để quản lý cả đề tài nghiên cứu sinh và đề tài cấp Bộ, cấp Nhà nước với quy trình phức tạp hơn; áp dụng trí tuệ nhân tạo để hỗ trợ phân loại đề tài tự động theo lĩnh vực, gợi ý thành viên Hội đồng phù hợp dựa trên chuyên môn, và phát hiện bất thường trong quy trình (đề tài nằm im quá lâu ở một bước, điểm đánh giá chênh lệch lớn giữa các thành viên Hội đồng).

Qua quá trình thực hiện đề tài, tác giả đã rút ra nhiều bài học kinh nghiệm quý giá. Bài học thứ nhất về phương pháp làm việc: việc đầu tư thời gian khảo sát kỹ quy trình thực tế và trao đổi với người dùng cuối (cán bộ Phòng KHCN, Quản lý Khoa, giảng viên) trước khi thiết kế là vô cùng quan trọng. Nhiều yêu cầu chi tiết chỉ phát hiện được qua trao đổi trực tiếp, không thể suy luận từ văn bản quy định. Bài học thứ hai về công nghệ: lựa chọn công nghệ phù hợp với bài toán cụ thể quan trọng hơn chạy theo công nghệ mới nhất. Việc chọn PostgreSQL thay vì MySQL, NestJS thay vì PHP, và tách riêng dịch vụ sinh tài liệu bằng Python đều xuất phát từ phân tích kỹ lưỡng yêu cầu thực tế. Bài học thứ ba về thiết kế: đầu tư thời gian vào thiết kế máy trạng thái rõ ràng ngay từ đầu giúp tránh được rất nhiều lỗi logic trong quá trình cài đặt. Việc vẽ sơ đồ chuyển đổi trạng thái, liệt kê tất cả các tình huống ngoại lệ, và thống nhất với người dùng trước khi viết mã giúp tiết kiệm đáng kể thời gian sửa lỗi sau này. Bài học thứ tư về bảo mật: luôn kiểm tra quyền ở cả tầng client và tầng server, không tin tưởng hoàn toàn vào kiểm tra phía client vì người dùng có thể bypass. Bài học thứ năm về kiểm thử: việc viết kịch bản kiểm thử chi tiết và thực hiện kiểm thử theo kịch bản giúp phát hiện nhiều lỗi mà kiểm thử ngẫu hứng không phát hiện được. Đặc biệt, kiểm thử các tình huống ngoại lệ (lỗi, xung đột, trạng thái không hợp lệ) quan trọng không kém kiểm thử luồng chính.

Với những kết quả đạt được, hệ thống sẵn sàng triển khai thử nghiệm tại Trường Đại học Sư phạm Kỹ thuật Nam Định. Tác giả kỳ vọng hệ thống sẽ góp phần nâng cao hiệu quả quản lý NCKH, tăng tính minh bạch trong quy trình xử lý, giảm thiểu thời gian hành chính cho cả cán bộ quản lý lẫn giảng viên, và tạo nền tảng dữ liệu tin cậy phục vụ công tác báo cáo, đánh giá và định hướng phát triển hoạt động nghiên cứu của nhà trường trong tương lai.

# TÀI LIỆU THAM KHẢO

Tài liệu chính thức và quy định

Bộ Chính trị (2019). Nghị quyết số 52-NQ/TW ngày 27 tháng 9 năm 2019 về một số chủ trương, chính sách chủ động tham gia cuộc cách mạng công nghiệp lần thứ tư.

Thủ tướng Chính phủ (2022). Quyết định số 131/QĐ-TTg phê duyệt Đề án "Tăng cường ứng dụng công nghệ thông tin và chuyển đổi số trong giáo dục và đào tạo giai đoạn 2022–2025, định hướng đến 2030".

Tài liệu kỹ thuật về công nghệ backend

NestJS Documentation (2024). Official documentation for NestJS framework. Truy cập từ https://docs.nestjs.com

Node.js Documentation (2024). Official Node.js documentation and API reference. Truy cập từ https://nodejs.org/docs

Prisma Documentation (2024). Prisma ORM documentation and guides. Truy cập từ https://www.prisma.io/docs

PostgreSQL Documentation (2024). PostgreSQL 14 official documentation. Truy cập từ https://www.postgresql.org/docs/14

Tài liệu kỹ thuật về công nghệ frontend

React Documentation (2024). Official React documentation and tutorial. Truy cập từ https://react.dev

Zustand Documentation (2024). Zustand state management library documentation. Truy cập từ https://docs.pmnd.rs/zustand

TanStack Query Documentation (2024). Powerful asynchronous state management for React. Truy cập từ https://tanstack.com/query

Tài liệu kỹ thuật về Python và xử lý tài liệu

FastAPI Documentation (2024). FastAPI modern Python web framework documentation. Truy cập từ https://fastapi.tiangolo.com

python-docx Documentation (2024). Python library for creating and updating Microsoft Word files. Truy cập từ https://python-docx.readthedocs.io

Tài liệu về Docker và DevOps

Docker Documentation (2024). Docker containerization platform documentation. Truy cập từ https://docs.docker.com

Docker Compose Documentation (2024). Docker Compose multi-container orchestration. Truy cập từ https://docs.docker.com/compose

Tài liệu về Nx monorepo

Nx Documentation (2024). Nx build system and monorepo tool documentation. Truy cập từ https://nx.dev/getting-started/intro

Sách chuyên khảo về kiến trúc và thiết kế phần mềm

Martin, Robert C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Casciaro, Mario and Luciano Mammino (2020). Node.js Design Patterns, Third Edition. Packt Publishing.

Nghiên cứu liên quan đến quản lý nghiên cứu khoa học

Elsevier Pure (2024). Research Information Management System. Truy cập từ https://www.elsevier.com/solutions/pure

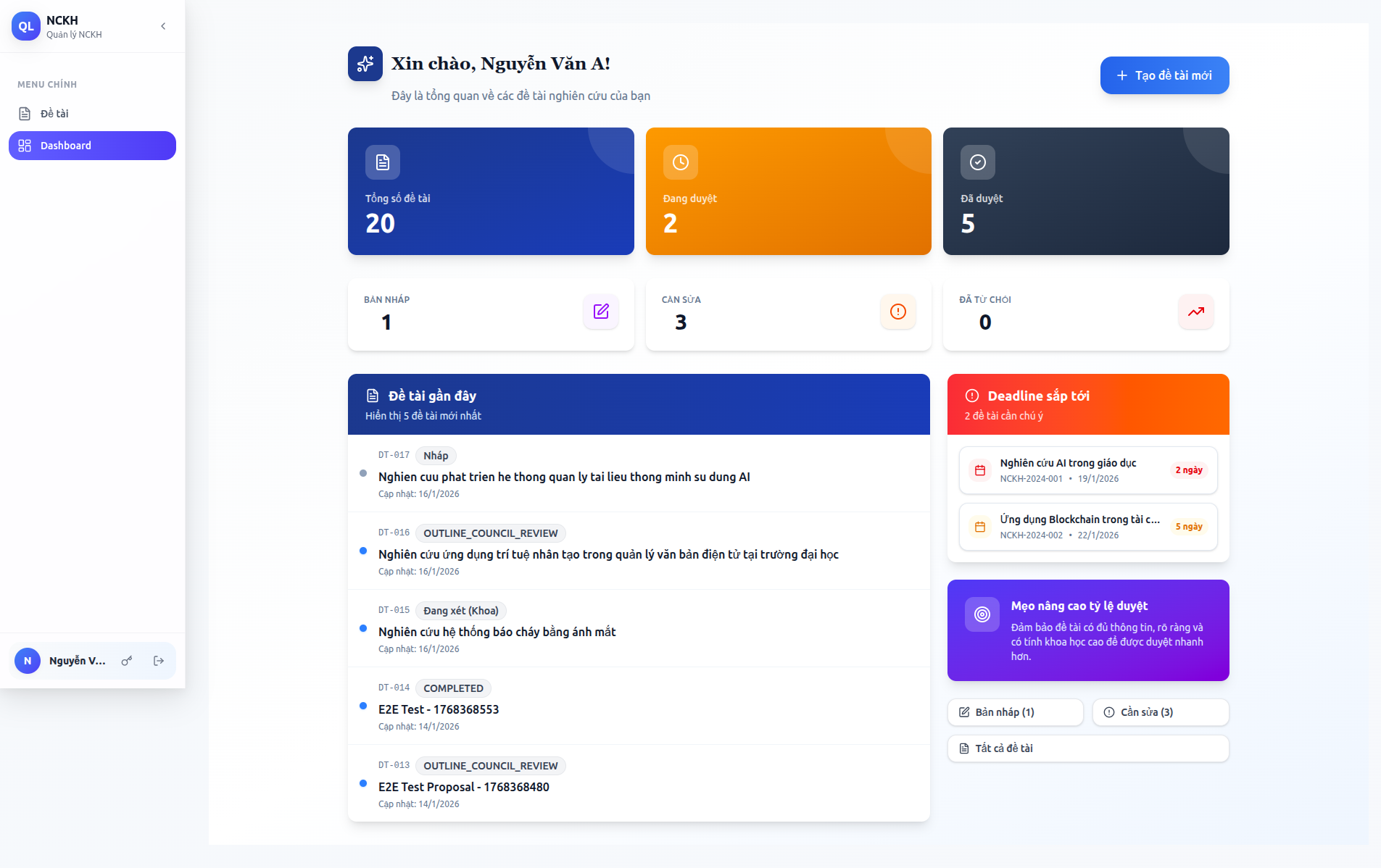
Tài liệu về bảo mật web

OWASP (2024). Open Web Application Security Project – Top 10 Web Application Security Risks. Truy cập từ https://owasp.org/www-project-top-ten

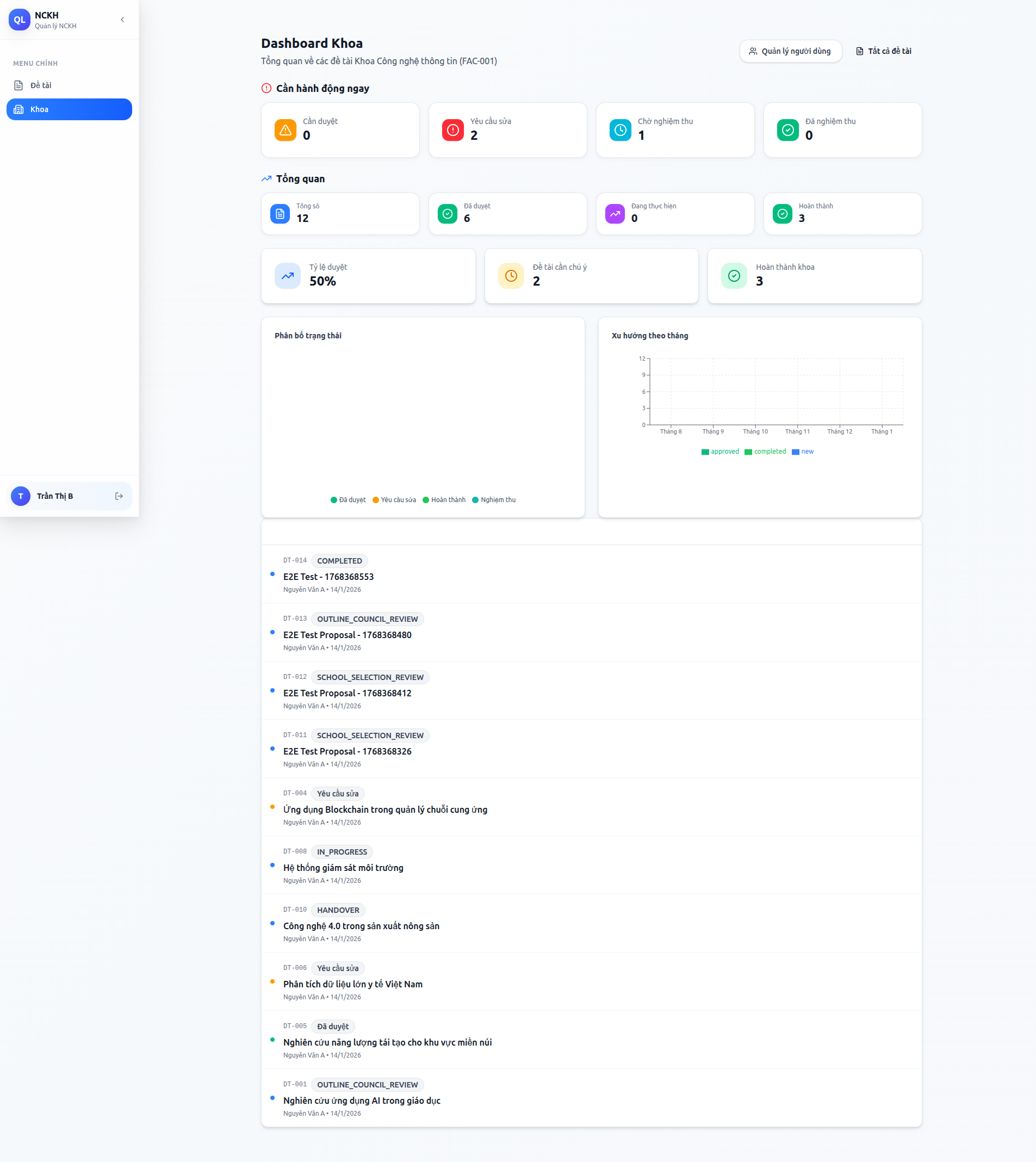
JWT.io (2024). JSON Web Tokens introduction and debugger. Truy cập từ https://jwt.io

bcrypt Documentation (2024). bcrypt password hashing library. Truy cập từ https://github.com/kelektiv/node.bcrypt.js

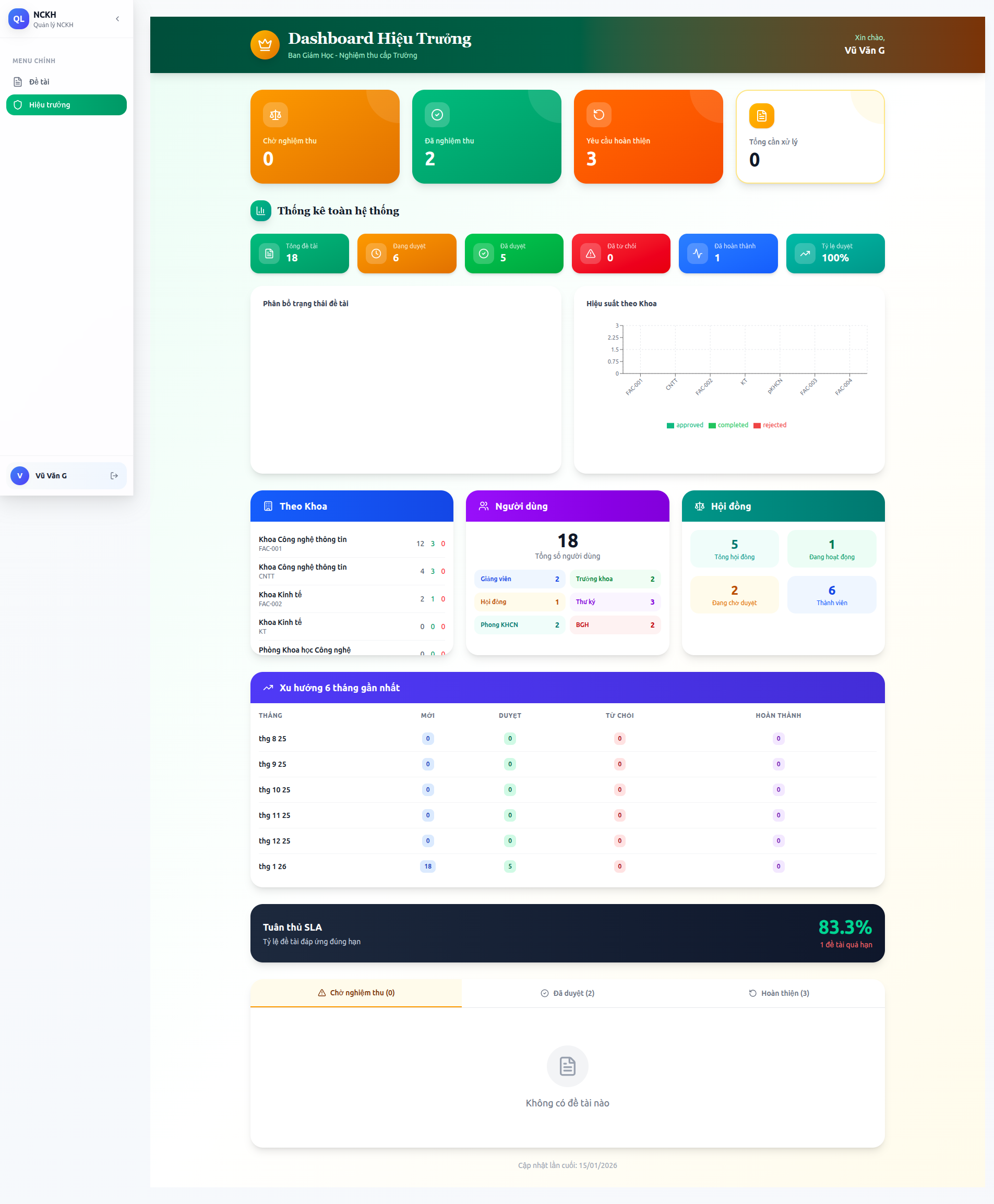
*Hết*



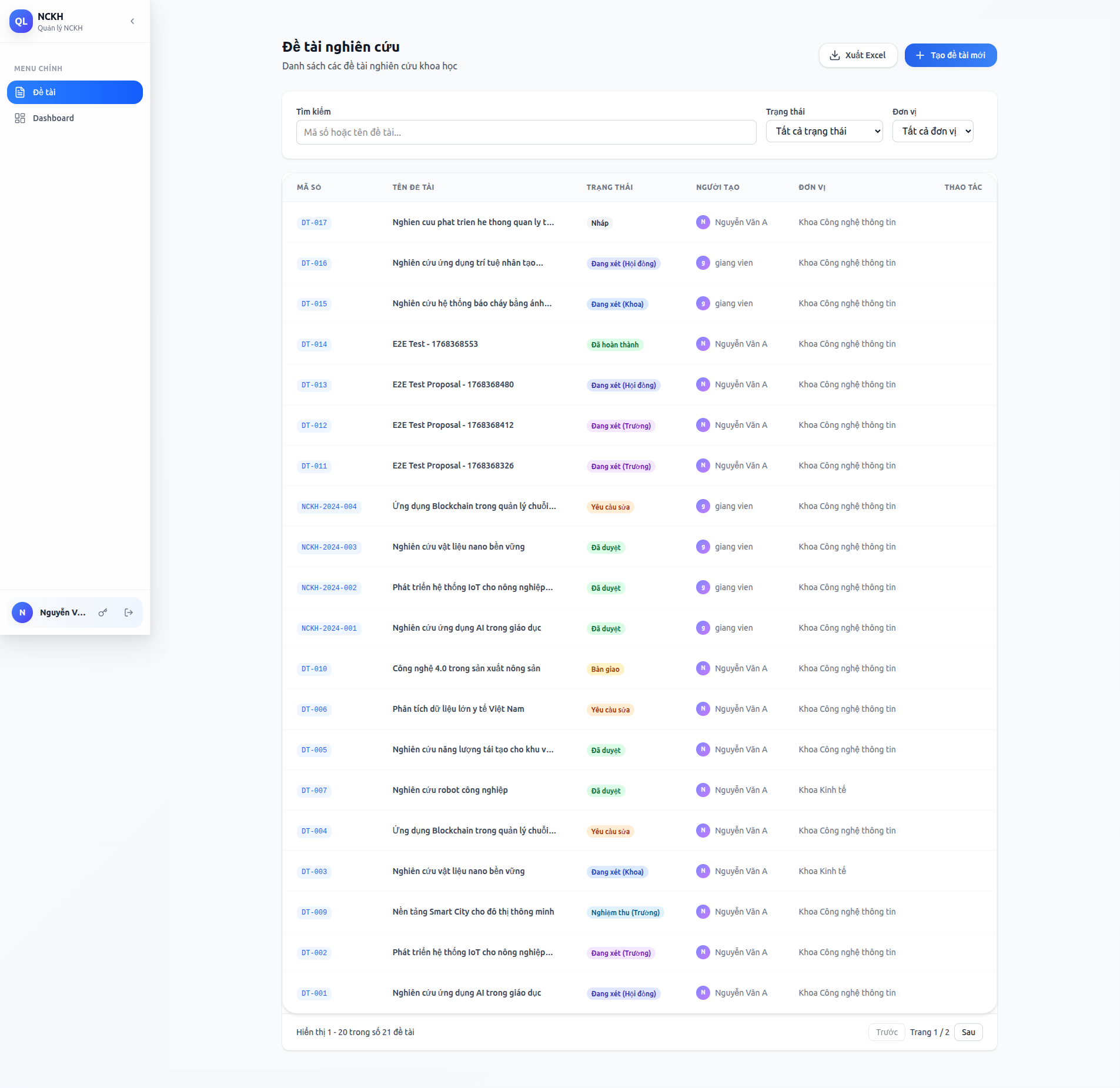
*Hình 4.2. Bảng điều khiển vai trò Giảng viên*



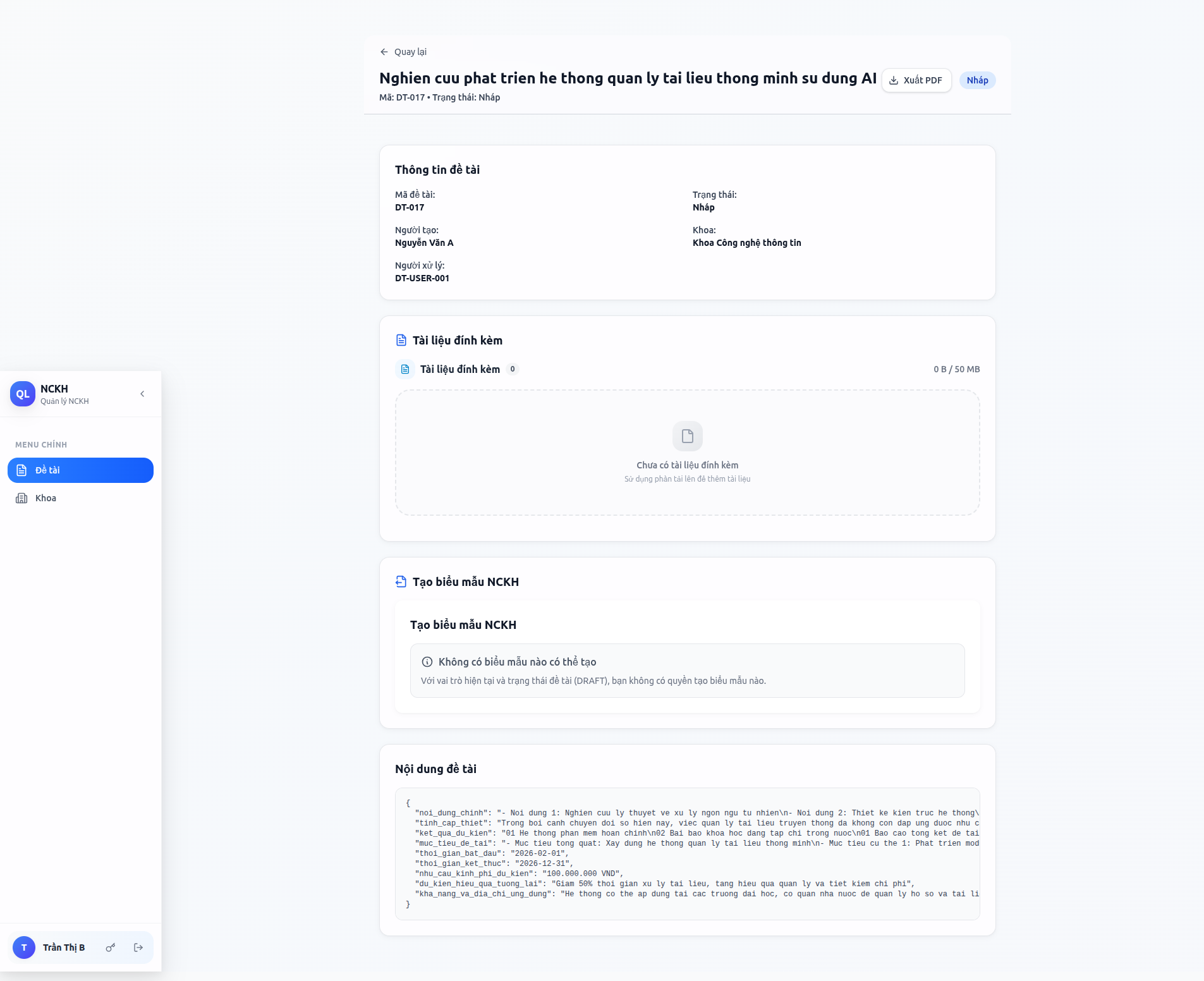
*Hình 4.3. Bảng điều khiển vai trò Quản lý Khoa*



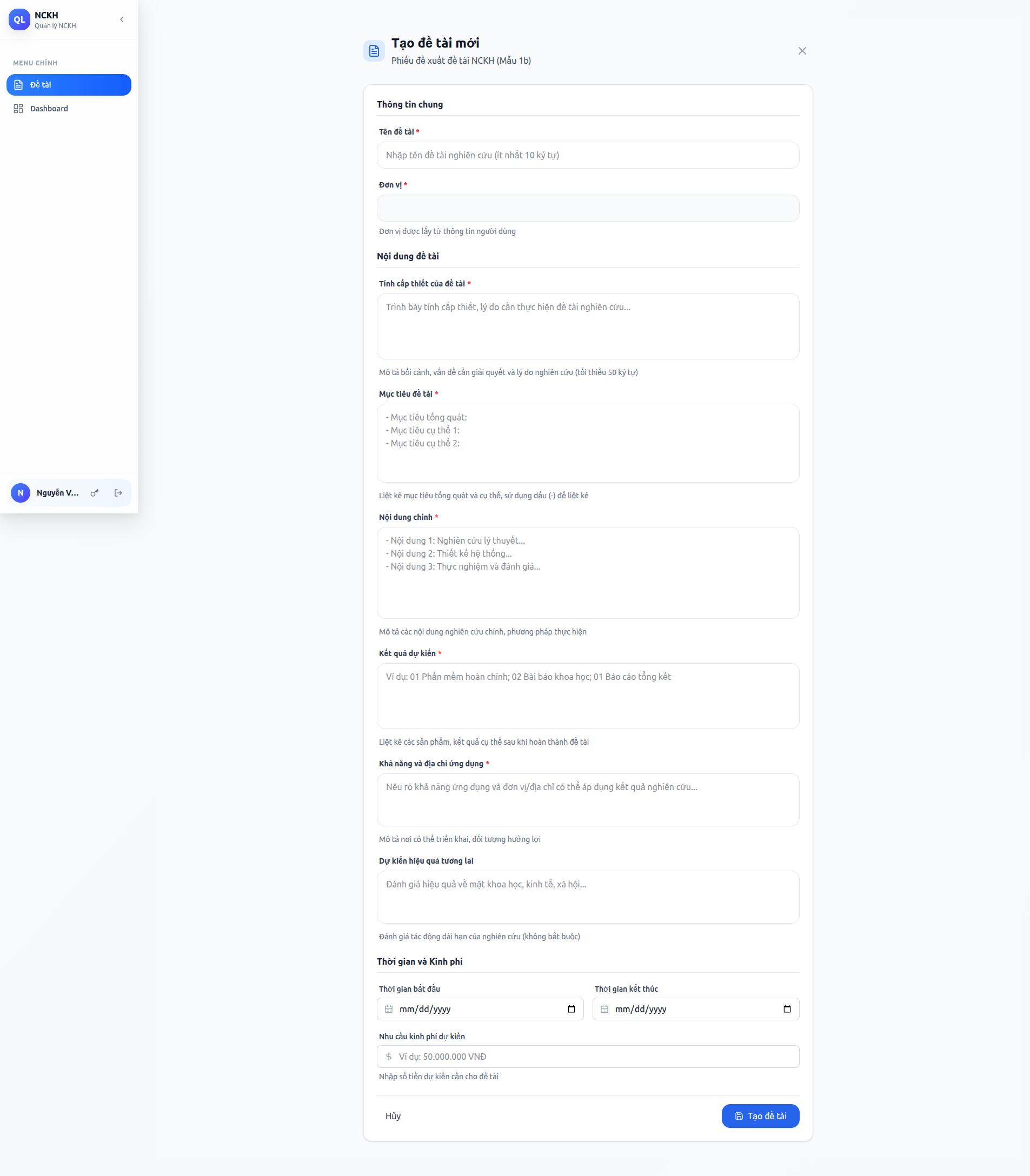
*Hình 4.4. Bảng điều khiển vai trò Ban Giám hiệu*



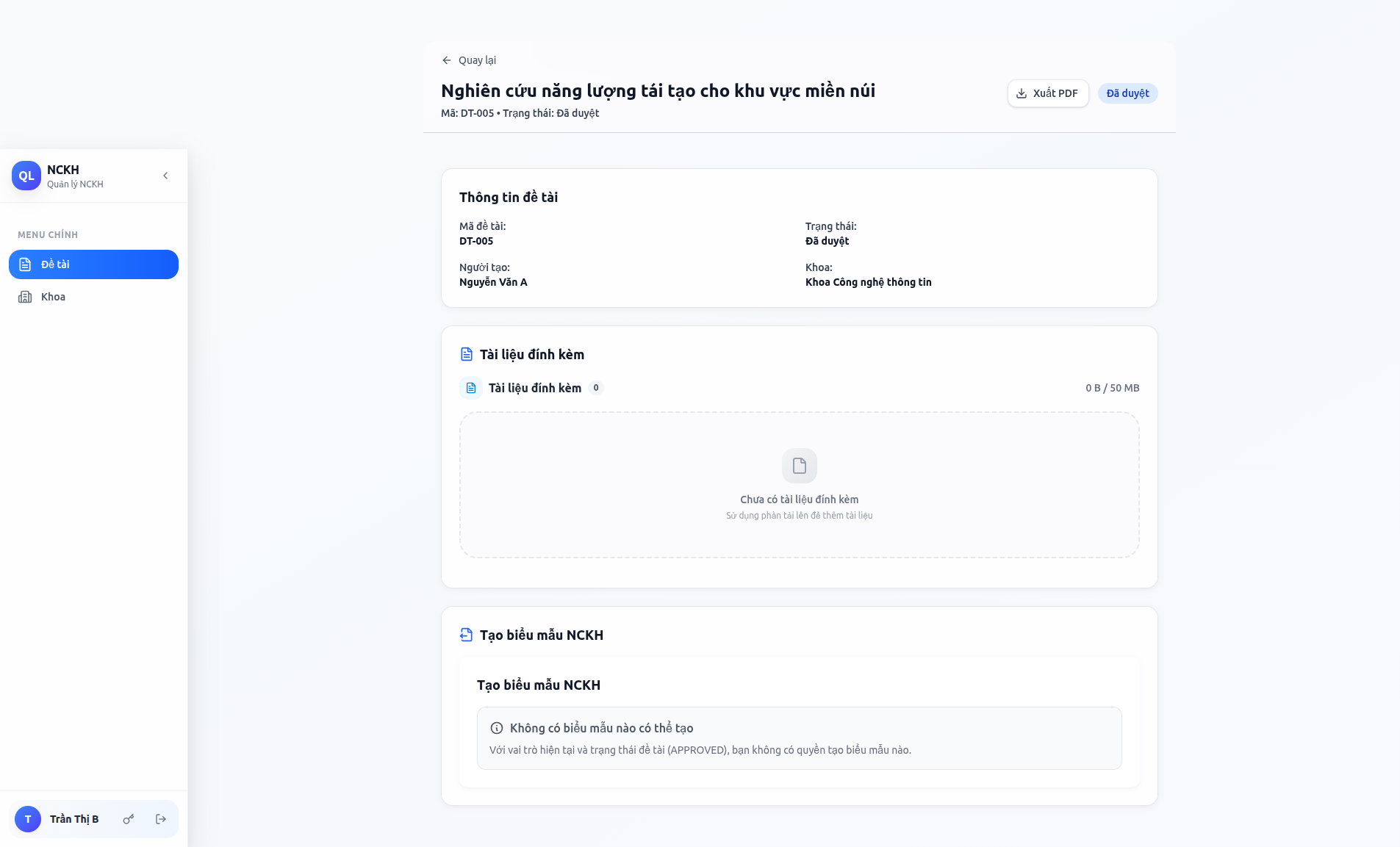
*Hình 4.5. Danh sách đề tài NCKH*



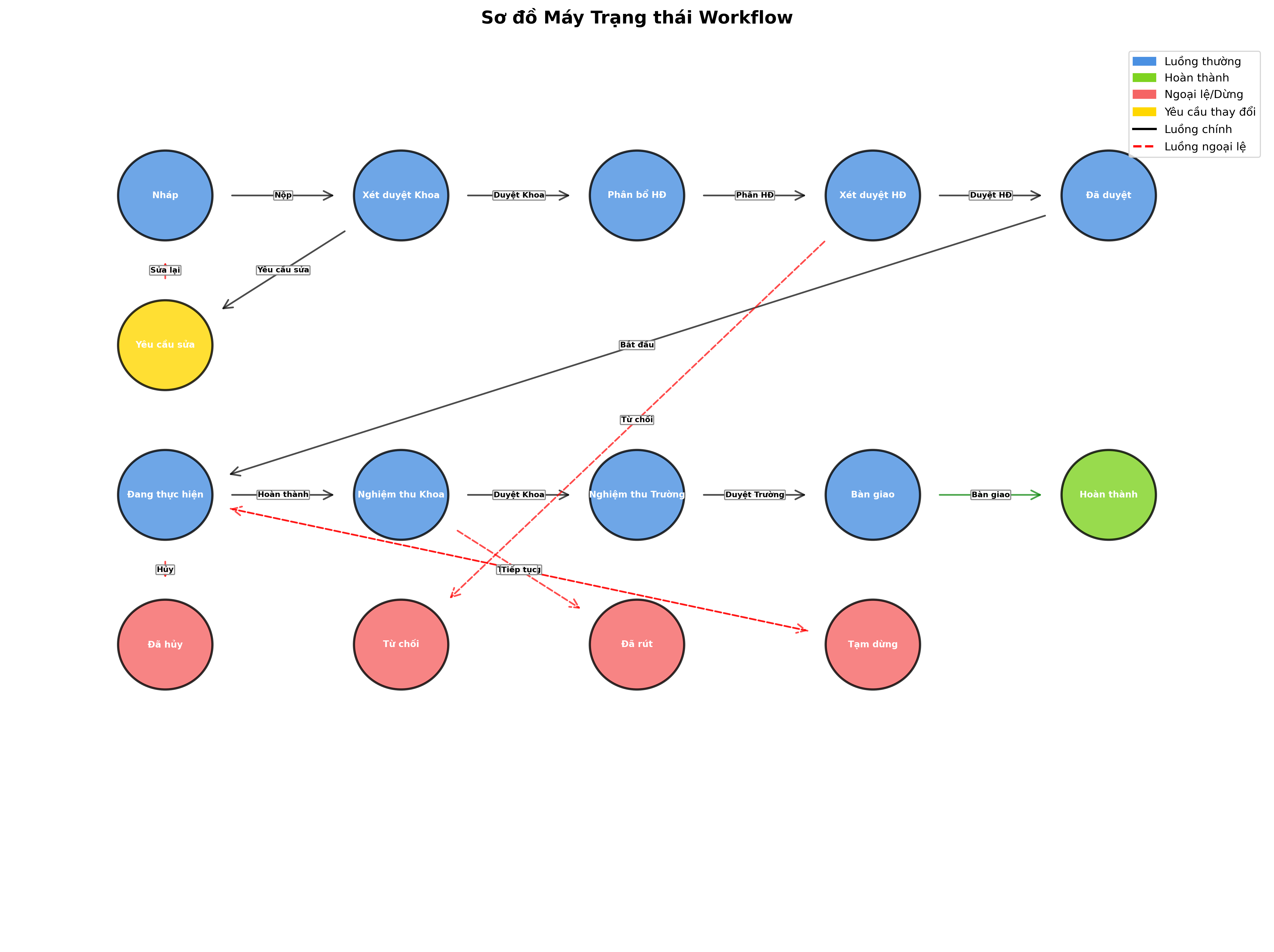
*Hình 4.6. Chi tiết đề tài ở trạng thái Nháp*



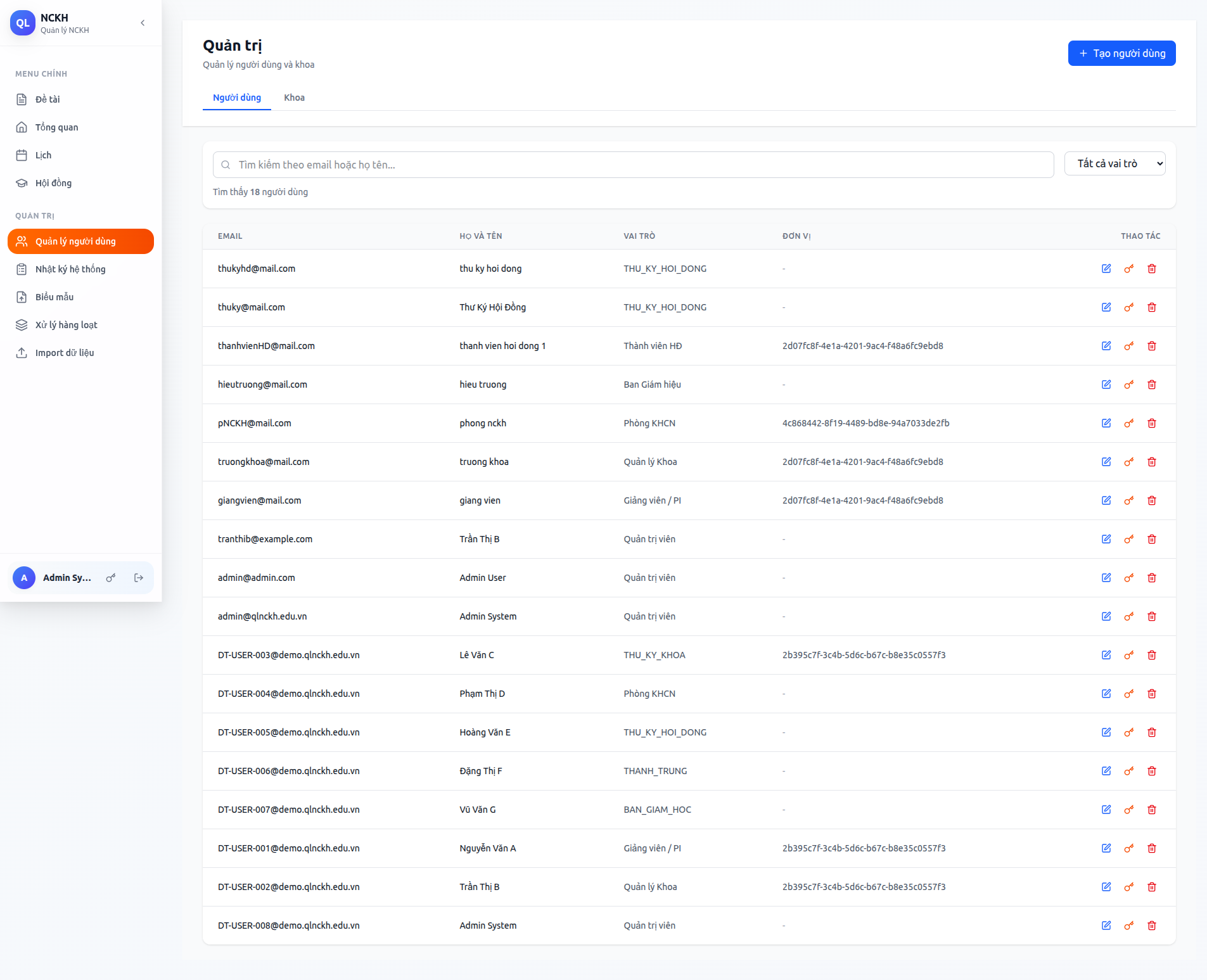
*Hình 4.7. Biểu mẫu tạo đề tài mới*



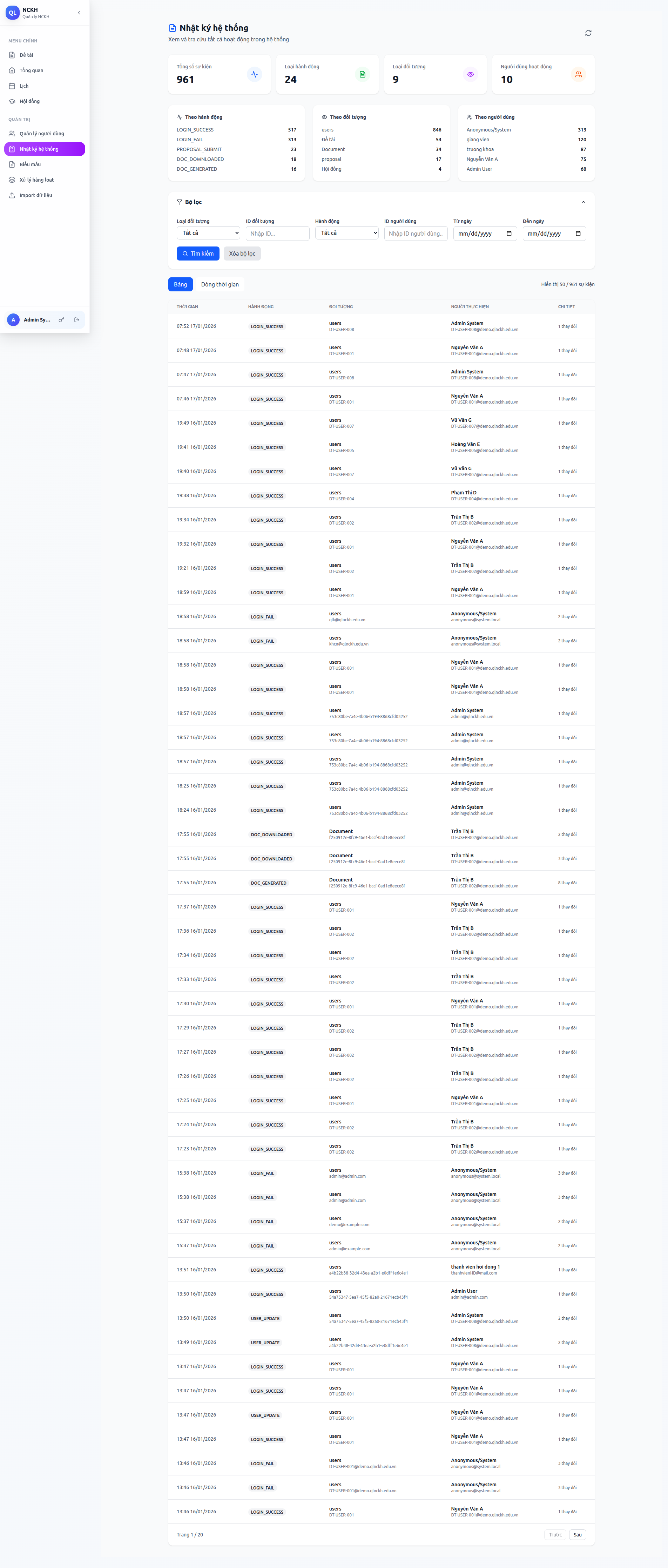
*Hình 4.8. Đề tài đã được phê duyệt*



*Hình 4.9. Phân bổ Hội đồng cho đề tài*



*Hình 4.11. Quản lý tài khoản người dùng*



*Hình 4.12. Nhật ký kiểm toán hệ thống*