

**TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN MÔN HỌC
CÔNG NGHỆ PHẦN MỀM
(MSHP: 220055)**

**TÊN ĐỀ TÀI
XÂY DỰNG ỨNG DỤNG
QUẢN LÝ THỜI GIAN BIỂU**

Sinh viên thực hiện:

110122062	Nguyễn Thanh Duy	DA22TTD
110122054	Trần Lâm Phú Đức	DA22TTD
110122089	Phan Đình Khải	DA22TTD

Giáo viên hướng dẫn: Nguyễn Bảo Ân

Trà Vinh, tháng 7 năm 2025

**TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN MÔN HỌC
CÔNG NGHỆ PHẦN MỀM
(MSHP: 220055)**

**TÊN ĐỀ TÀI
XÂY DỰNG ỨNG DỤNG
QUẢN LÝ THỜI GIAN BIỂU**

Sinh viên thực hiện:

110122062	Nguyễn Thanh Duy	DA22TTD
110122054	Trần Lâm Phú Đức	DA22TTD
110122089	Phan Đình Khải	DA22TTD

Giáo viên hướng dẫn: Nguyễn Bảo Ân

Trà Vinh, tháng 7 năm 2025

LỜI NHẬN XÉT CỦA GIÁO VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày tháng năm

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày tháng năm

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Nhóm em xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc đối với thầy Nguyễn Bảo Ân đã tận tình truyền đạt kiến thức về môn công nghệ phần mềm. Với vốn kiến thức được tiếp thu trong quá trình học không chỉ là nền tảng cho quá trình nghiên cứu bài báo cáo mà nó còn là hành trang quý báu để nhóm em áp dụng vào thực tế một cách vững chắc và tự tin.

Chúng em cũng xin chân thành cảm ơn Thầy Nguyễn Bảo Ân là người hướng dẫn giúp đỡ và cung cấp những kiến thức quý báu giúp chúng em hoàn thành tốt bài báo cáo của nhóm mình. Do còn hạn chế về kiến thức cũng như những kinh nghiệm thực tế cho nên không tránh khỏi được những sai sót trong quá trình tìm hiểu và trình bày rất mong nhận được sự đóng góp ý kiến của thầy để bài báo cáo được hoàn chỉnh hơn.

Sau cùng, nhóm em xin kính chúc thầy Nguyễn Bảo Ân thật nhiều sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau.

Nhóm em xin chân thành cảm ơn !

Nguyễn Thanh Duy Trần Lâm Phú Đức Phan Đình Khải

MỤC LỤC

CHƯƠNG 1 GIỚI THIỆU	1
1.1 Mô tả bài toán.....	1
1.2 Mô tả ứng dụng	2
1.3 Đặc tả các chức năng hệ thống.....	2
1.3.1 Chức năng của người dùng	2
1.3.2 Chức năng quản lý sự kiện.....	2
1.3.3 Chức năng quản lý danh mục.....	2
1.3.4 Quản lý người dùng	2
1.4 Thiết kế dữ liệu.....	3
1.4.1 Lược đồ cơ sở dữ liệu	3
1.4.2 Mô hình dữ liệu.....	4
CHƯƠNG 2 CƠ SỞ LÝ THUYẾT	7
2.1 Tổng quan về Agile	7
2.1.1 Giới thiệu về Agile.....	7
2.1.2 Tuyên ngôn Agile.....	7
2.1.2.1 Giá trị của Tuyên ngôn Agile.....	7
2.1.2.2 Nguyên tắc của Tuyên ngôn Agile.....	7
2.2 Tổng quan về SCRUM.....	8
2.2.1 SCRUM là gì ?.....	8
2.2.2 Phương pháp SCRUM	8
2.2.2.1 Thuật ngữ SCRUM.....	8
2.2.2.2 Các vai trò quan trọng trong Scrum	9
2.2.3 Scrum và Sprints	9

Đề tài: Xây dựng ứng dụng quản lý thời gian biểu

2.2.4 Các thực hành chính của Scrum.....	9
2.2.5 Trạng thái của Product Backlog Item	10
2.3 Công nghệ sử dụng.....	10
2.3.1 RESTful API	10
2.3.1.1 Tổng quan về RESTful API.....	10
2.3.1.2 RESTful hoạt động như thế nào?	10
2.3.2 Next.js & React.js	11
2.3.3 Node.js & framework Express.js	11
2.3.3.1 Node.js.....	11
2.3.3.2 Express.js.....	12
2.3.4 Cơ sở dữ liệu (PostgreSQL & MySQL)	12
2.3.4.1 PostgreSQL.....	13
2.3.4.2 Database Tools.....	13
2.3.4.3 Redis	13
2.3.5 Kiến trúc MVC	13
2.3.5.1 MVC là gì?	13
2.3.5.2 MVC hoạt động như thế nào?.....	13
CHƯƠNG 3 XÁC ĐỊNH NHU CẦU	15
3.1 Product Backlog	15
3.2 Sprint backlog	16
3.3 Technical Backlog	16
3.4 Support Backlog.....	17
CHƯƠNG 4 LẬP KẾ HOẠCH SCRUM.....	18
4.1 Sprint 1:	18

Đề tài: Xây dựng ứng dụng quản lý thời gian biểu

4.2 Sprint 2:	20
CHƯƠNG 5 THIẾT KẾ HỆ THỐNG	21
5.1. Cấu trúc thư mục và kiến trúc hệ thống của dự án.....	21
5.2. Cài đặt và chạy dự án	22
5.2.1. Yêu cầu hệ thống.....	22
5.2.2. Cài đặt dependencies.....	23
5.2.3. Cấu hình environment.....	23
5.2.4. Chạy ứng dụng	24
5.3. Database Migration	24
CHƯƠNG 6 KẾT QUẢ THỰC NGHIỆM	25
6.1. Kết quả đạt được	25
6.1.1. Chức năng đã hoàn thành.....	25
6.1.2. API Documentation.....	67
6.1.3. Performance	67
6.2. Tích hợp tính năng	68
6.2.1. Tích hợp Swagger mô tả các API.....	68
6.2.2. Triển khai lên host.....	68
6.2.3. Tích hợp Github Action	70
CHƯƠNG 7 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	71
7.1. Kết luận	71
7.1.1. Điểm mạnh.....	71
7.1.2. Thách thức đã vượt qua.....	71
7.1.3. Kiến thức thu được.....	71
7.2. Hướng phát triển tương lai	72

Đề tài: Xây dựng ứng dụng quản lý thời gian biểu

7.2.1. Tính năng mở rộng.....	72
7.2.2. Cải tiến kỹ thuật	72
7.3. Đánh giá tổng thể	72
TÀI LIỆU THAM KHẢO	74

DANH MỤC HÌNH ẢNH

Hình 1.1. Mô hình dữ liệu mức Quan niệm	4
Hình 1.2. Mô hình dữ liệu mức Luận lý.....	5
Hình 1.3. Mô hình dữ liệu mức Vật lý	6
Hình 5.1. Cấu trúc thư mục	21
Hình 5.2 Kiến trúc hệ thống	22
Hình 6.1 Giao diện đăng nhập.....	25
Hình 6.2 Giao diện đăng ký.....	26
Hình 6.3 Mã nguồn.....	28
Hình 6.4 Giao diện quản lý sự kiện.....	31
Hình 6.5 Giao diện thêm sự kiện.....	31
Hình 6.6 Mã nguồn.....	33
Hình 6.7 Giao diện quản lý danh mục	37
Hình 6.8 Giao diện thêm danh mục.....	37
Hình 6.9 Mã nguồn.....	39
Hình 6.10 Giao diện thông báo sự kiện.....	42
Hình 6.11 Mã nguồn.....	44
Hình 6.12 Giao diện trang dashboards	47
Hình 6.13 Mã nguồn.....	49
Hình 6.14 Giao diện trang calendar.....	53
Hình 6.15 Mã nguồn.....	55
Hình 6.16 Giao diện trang Cài đặt.....	58
Hình 6.17 Giao diện trang Hồ sơ cá nhân	59
Hình 6.18 Giao diện trang Cài đặt Giao diện.....	60

Đề tài: Xây dựng ứng dụng quản lý thời gian biểu

Hình 6.19 Giao diện trang Cài đặt thông báo	60
Hình 6.20 Giao diện trang Cài đặt Lịch	61
Hình 6.21 Giao diện trang Phản hồi & Hỗ Trợ	62
Hình 6.21 Mã nguồn.....	64
Hình 6.22 Giao diện của Swagger.....	68
Hình 6.23 Giao diện Web được triển khai lên host có URL.....	69
Hình 6.24 Giao diện sao khi tích hợp Github Action	70

CHƯƠNG 1 GIỚI THIỆU

1.1 Mô tả bài toán

Trong thời đại công nghệ phát triển như hiện nay, việc quản lý thời gian và lịch trình cá nhân trở nên vô cùng quan trọng. Với nhịp sống ngày càng bận rộn, mọi người cần có một công cụ hiệu quả để quản lý các sự kiện, cuộc hẹn hàng ngày, nhắc nhở các công việc quan trọng và theo dõi tiến độ kế hoạch cá nhân. Để đáp ứng nhu cầu này, việc xây dựng ứng dụng quản lý thời gian biểu Schedule Manager là một giải pháp hiệu quả.

Ứng dụng quản lý thời gian biểu Schedule Manager có thể giúp người dùng quản lý thông tin sự kiện một cách khoa học và dễ dàng. Ứng dụng được xây dựng bằng Next.js, React.js, Node.js & Express.js, PostgreSQL nhằm mục đích tối ưu hóa quá trình quản lý thời gian và phục vụ nhu cầu cá nhân của người dùng.

Ứng dụng sẽ có các chức năng chính sau:

- Quản lý danh sách sự kiện cá nhân.
- Quản lý thông tin danh mục sự kiện.
- Quản lý thông tin người dùng.
- Quản lý hệ thống thông báo.
- Quản lý lịch hiển thị theo tháng, tuần, ngày.
- Quản lý tính năng nhắc nhở và lặp lại sự kiện.

Việc xây dựng ứng dụng quản lý thời gian biểu Schedule Manager có thể mang lại nhiều lợi ích cho người dùng, bao gồm:

- Tăng hiệu quả quản lý thời gian cá nhân.
- Nâng cao năng suất làm việc và học tập.
- Đơn giản hoá quá trình tổ chức và theo dõi sự kiện.
- Tiết kiệm thời gian và nguồn lực.

1.2 Mô tả ứng dụng

Ứng dụng quản lý thời gian biểu Schedule Manager là ứng dụng được sử dụng bởi đối tượng người dùng cá nhân để quản lý lịch trình và sự kiện của họ.

Đối với người dùng khi tham gia hệ thống sẽ thực hiện: Đăng ký tài khoản và đăng nhập vào hệ thống. Tạo và quản lý các sự kiện cá nhân với thông tin chi tiết như tiêu đề, mô tả, thời gian, địa điểm. Phân loại sự kiện theo danh mục và gán màu sắc riêng biệt. Thiết lập nhắc nhở và tạo sự kiện lặp lại theo chu kỳ. Xem lịch theo các chế độ tháng, tuần, ngày và theo dõi thống kê sự kiện.

1.3 Đặc tả các chức năng hệ thống

1.3.1 Chức năng của người dùng

Đăng nhập/Đăng ký: Tạo tài khoản mới và đăng nhập vào hệ thống.

Dashboard: Hiện thị tổng quan về sự kiện sắp tới và thống kê.

Quản lý hồ sơ: Cập nhật thông tin cá nhân và thay đổi mật khẩu.

1.3.2 Chức năng quản lý sự kiện

Tạo sự kiện: Thêm sự kiện mới với thông tin chi tiết.

Chỉnh sửa sự kiện: Cập nhật thông tin sự kiện đã tồn tại.

Xóa sự kiện: Loại bỏ sự kiện không cần thiết.

Xem chi tiết: Hiện thị thông tin đầy đủ của sự kiện.

Tìm kiếm và lọc: Tìm kiếm sự kiện theo tiêu đề, danh mục, thời gian.

1.3.3 Chức năng quản lý danh mục

Tạo danh mục: Thêm danh mục mới để phân loại sự kiện.

Tùy chỉnh màu sắc: Gán màu sắc riêng cho từng danh mục.

Quản lý danh mục: Sửa đổi và xóa danh mục.

1.3.4 Quản lý người dùng

Đăng Nhập: Đăng nhập bằng email và mật khẩu với xác thực JWT.

Đăng Ký: Tạo tài khoản mới với thông tin cơ bản.

Quản lý phiên: Duy trì phiên đăng nhập và tự động đăng xuất khi hết hạn.

1.4 Thiết kế dữ liệu

1.4.1 Lược đồ cơ sở dữ liệu

PostgreSQL (Backend chính):

users (id UUID, name, email, password, created_at, updated_at)

categories (id UUID, user_id UUID, name, color, description, is_default, created_at, updated_at)

events (id UUID, user_id UUID, category_id UUID, title, description, start_date, end_date, all_day, location, reminder JSON, repeat JSON, created_at, updated_at)

notifications (id UUID, user_id UUID, event_id UUID, title, message, type, is_read, scheduled_at, created_at)

MySQL (Frontend API):

events (id, title, description, start_date, end_date, all_day, category_id, user_id, location, reminder_enabled, reminder_minutes, created_at, updated_at)

categories (id, name, color, user_id, created_at, updated_at)

STT Tên thực thể Database Diễn giải

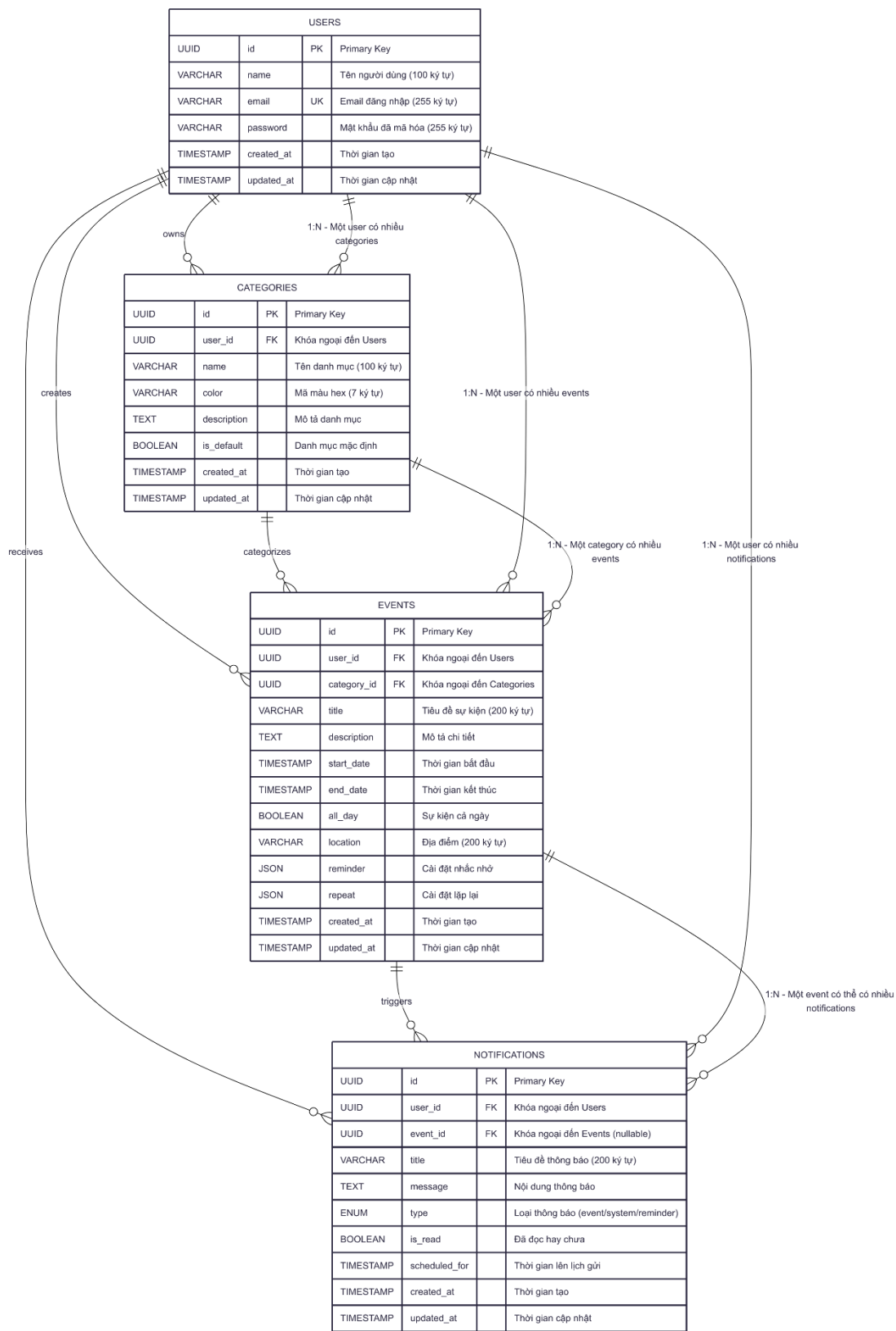
1 users PostgreSQL Lưu trữ thông tin người dùng

2 categories PostgreSQL + MySQL Lưu trữ thông tin danh mục sự kiện

3 event PostgreSQL + MySQL Lưu trữ thông tin sự kiện với reminder/repeat

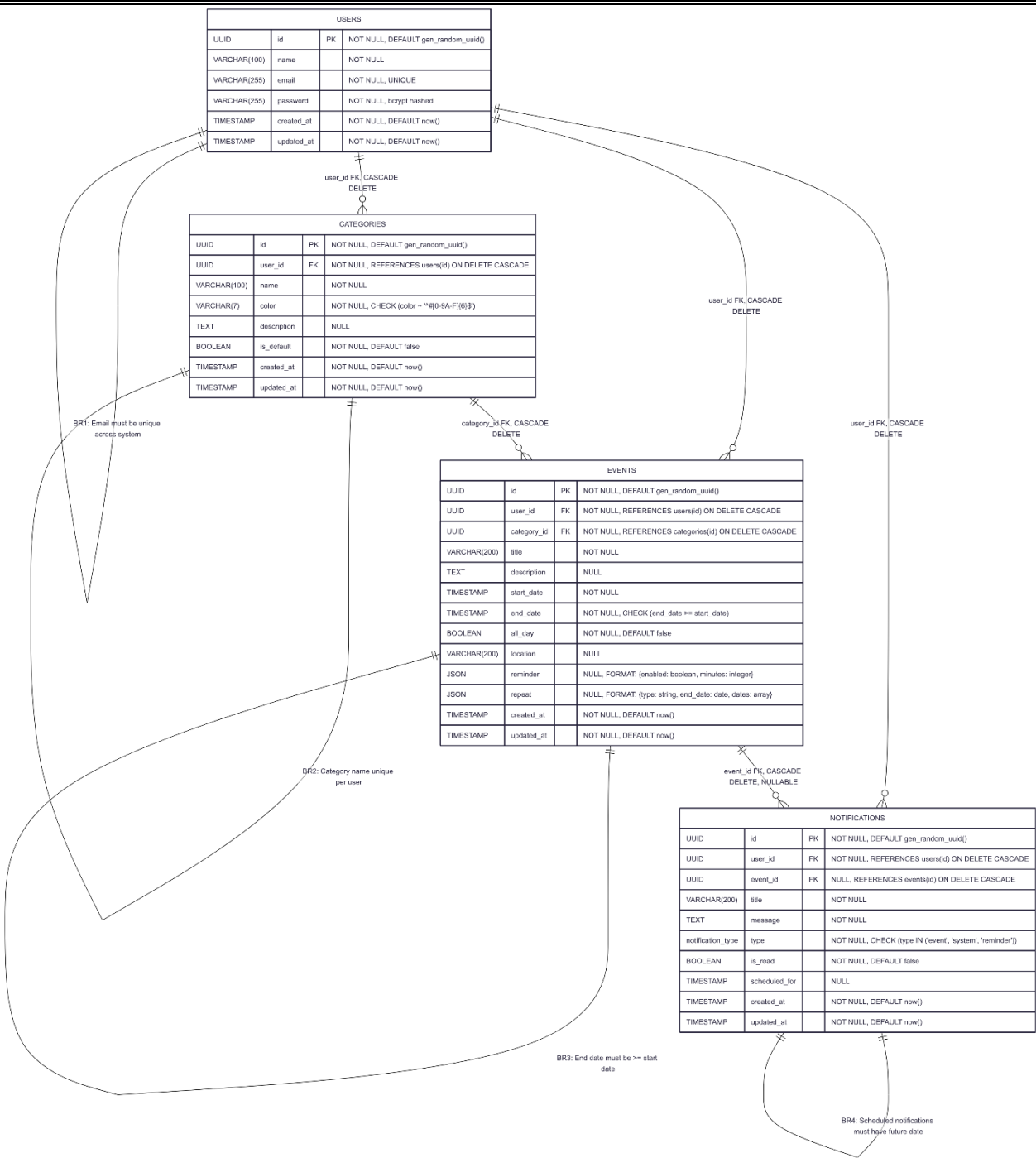
4 notifications PostgreSQL Lưu trữ thông tin thông báo hệ thống

1.4.2 Mô hình dữ liệu



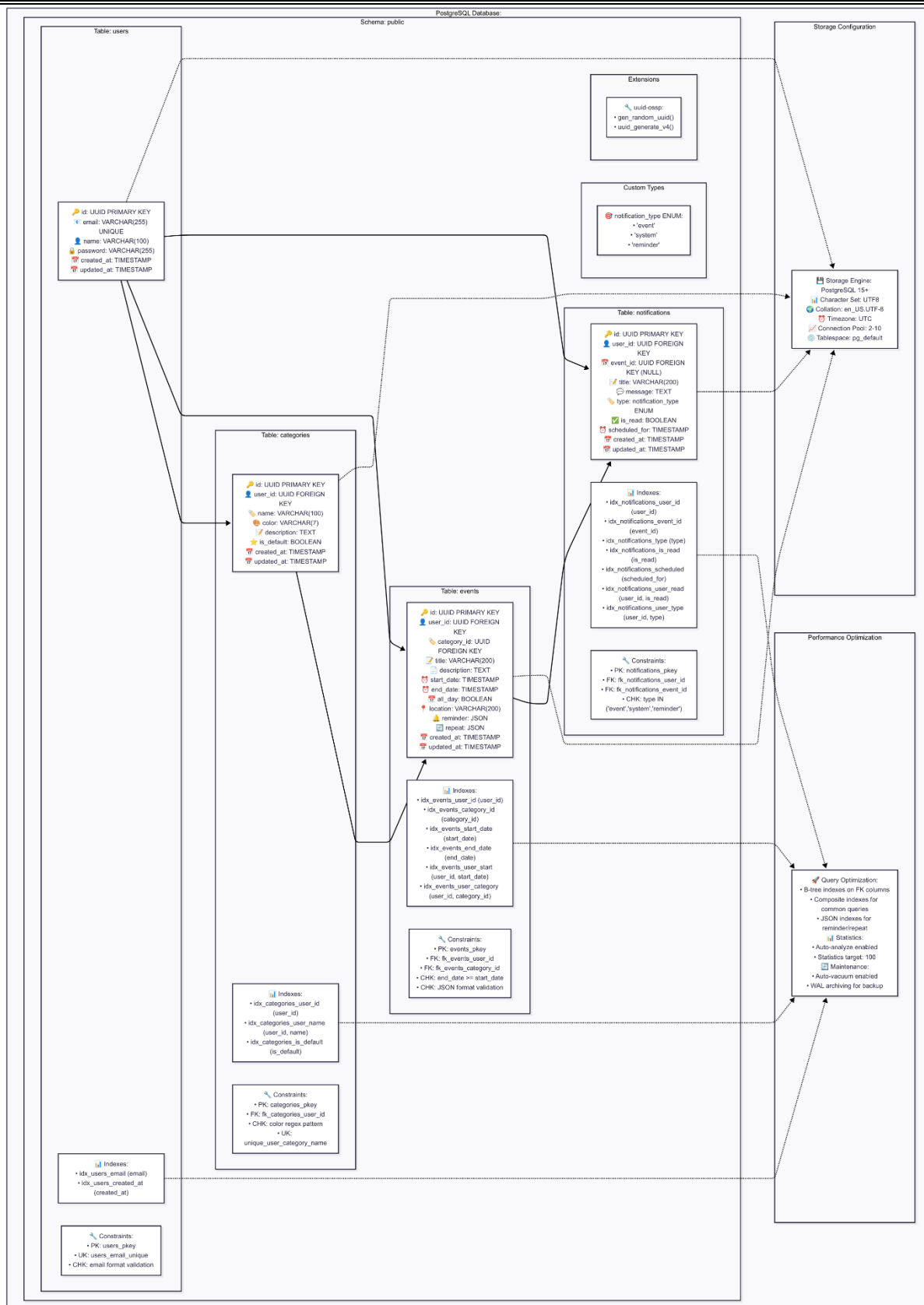
Hình 1.1. Mô hình dữ liệu mức Quan niệm

Đề tài: Xây dựng ứng dụng quản lý thời gian biểu



Hình 1.2. Mô hình dữ liệu mức Luận lý

Đề tài: Xây dựng ứng dụng quản lý thời gian biểu



Hình 1.3. Mô hình dữ liệu mức Vật lý

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về Agile

2.1.1 Giới thiệu về Agile

Agile là một phương pháp luận phát triển phần mềm linh hoạt, tập trung vào việc cung cấp các chức năng phần mềm một cách nhanh chóng và hiệu quả. Agile cho phép các nhóm phát triển phản hồi nhanh chóng đối với các thay đổi và phản hồi từ khách hàng, giảm thiểu chi phí phát triển thông qua các vòng lặp phát triển ngắn và tăng trưởng liên tục.

2.1.2 Tuyên ngôn Agile

Tuyên ngôn Agile, công bố năm 2001, đề ra 4 giá trị cốt lõi và 12 nguyên tắc hướng dẫn cho phương pháp phát triển phần mềm linh hoạt.

2.1.2.1 Giá trị của Tuyên ngôn Agile

Cá nhân và tương tác hơn là quy trình và công cụ: Tôn trọng các cá nhân và mối quan hệ tương tác giữa các thành viên.

Phần mềm hoạt động tốt hơn là tài liệu đầy đủ: Ưu tiên việc tạo ra phần mềm hoạt động hơn là viết tài liệu chi tiết.

Hợp tác với khách hàng hơn là đàm phán hợp đồng: Tăng cường hợp tác liên tục với khách hàng để đảm bảo sản phẩm phát triển đúng hướng.

Ứng phó với các thay đổi hơn là làm theo kế hoạch: Linh hoạt và sẵn sàng thay đổi để đáp ứng các yêu cầu mới.

2.1.2.2 Nguyên tắc của Tuyên ngôn Agile

Làm hài lòng khách hàng thông qua việc cung cấp phần mềm liên tục và sớm.

Chào đón sự thay đổi trong suốt quá trình phát triển.

Cung cấp phần mềm hoạt động định kỳ.

Tạo điều kiện cho các cuộc hợp tác hàng ngày giữa đội phát triển và khách hàng.

Xây dựng các dự án xoay quanh các cá nhân có động lực.

Thúc đẩy các cuộc trò chuyện trực tiếp.

Phần mềm hoạt động là thước đo chính của tiến độ.

Các quy trình Agile thúc đẩy sự phát triển bền vững.

Liên tục quan tâm đến kỹ thuật xuất sắc và thiết kế tốt.

Tính đơn giản.

Các nhóm tự tổ chức.

Phản ánh và điều chỉnh thường xuyên.

2.2 Tổng quan về SCRUM

2.2.1 SCRUM là gì ?

Scrum là một phương pháp linh hoạt tập trung vào lập kế hoạch và quản lý linh hoạt. Không giống như XP, nó không xác định các thực hành kỹ thuật sẽ được sử dụng. Nhóm phát triển có thể sử dụng bất kỳ phương pháp kỹ thuật nào mà họ cho là phù hợp với sản phẩm đang được phát triển.

Trong Scrum, công việc cần hoàn thành được duy trì trong product backlog một danh sách các hạng mục công việc cần hoàn thành. Mỗi phần tăng trưởng của phần mềm thực hiện một số hạng mục công việc từ product backlog.

2.2.2 Phương pháp SCRUM

SCRUM là một phương pháp quản lý dự án thuộc Agile, cung cấp một khuôn khổ linh hoạt cho việc tổ chức và lập kế hoạch dự án. Không bắt buộc bất kỳ thực hành kỹ thuật cụ thể nào, SCRUM giúp quản lý các công việc cần hoàn thành, thời gian và chi phí phát triển phần mềm, cũng như thời điểm sản phẩm có thể chuyển giao/bán ra thị trường.

2.2.2.1 Thuật ngữ SCRUM

Daily Scrum: Cuộc họp nhóm hàng ngày để xem xét tiến độ và công việc phải hoàn thành.

Sprint: Khoảng thời gian ngắn (thường 2-4 tuần) khi phát triển một phần gia tăng của sản phẩm.

ScrumMaster: Người hướng dẫn nhóm sử dụng hiệu quả phương pháp Scrum.

Product: Sản phẩm phần mềm đang được phát triển bởi nhóm Scrum.

ProductOwner: Người chịu trách nhiệm xác định các tính năng và thuộc tính của sản phẩm, xem xét công việc đã hoàn thành và giúp kiểm tra sản phẩm.

Product backlog: Danh sách việc cần làm bao gồm các lỗi, tính năng và cải tiến sản phẩm chưa hoàn thành.

Development team: Nhóm nhỏ tự tổ chức từ 5-8 người chịu trách nhiệm phát triển sản phẩm.

Potentially shippable product increment: Đầu ra của một Sprint có chất lượng đủ cao để triển khai cho khách hàng sử dụng.

Velocity: Ước tính khối lượng công việc mà một nhóm có thể thực hiện trong một Sprint

2.2.2.2 Các vai trò quan trọng trong Scrum

Product Owner: Đảm bảo nhóm phát triển tập trung vào sản phẩm, thường là người quản lý sản phẩm trong các công ty sử dụng Scrum.

Scrum Master: Hướng dẫn nhóm sử dụng hiệu quả phương pháp Scrum, không phải là người quản lý dự án thông thường mà là huấn luyện viên cho nhóm. Trong nhiều công ty, Scrum Master có thể kiêm nhiệm một số trách nhiệm quản lý dự án.

2.2.3 Scrum và Sprints

Trong Scrum, phần mềm được phát triển qua các Sprint - khoảng thời gian cố định (thường là 2-4 tuần) trong đó các tính năng phần mềm được phát triển và chuyển giao. Mỗi Sprint tạo ra một 'phần gia tăng sản phẩm có thể chuyển giao được'. Nhóm có các cuộc họp hàng ngày (Scrums) để xem xét tiến độ và cập nhật danh sách các hạng mục công việc chưa hoàn thành.

2.2.4 Các thực hành chính của Scrum

Product backlog: Danh sách việc cần làm bao gồm các mục sẽ được triển khai, được xem xét và cập nhật trước mỗi Sprint.

Timeboxed sprints: Khoảng thời gian cố định (2-4 tuần) trong đó các mục từ product backlog được triển khai.

Self-organizing teams: Các nhóm tự tổ chức đưa ra quyết định của riêng họ và làm việc bằng cách thảo luận các vấn đề và đưa ra quyết định bằng sự đồng thuận.

2.2.5 Trạng thái của Product Backlog Item

Sẵn sàng để xem xét: Ý tưởng cấp cao và mô tả tính năng sẽ được xem xét để đưa vào sản phẩm.

Sẵn sàng để tinh chỉnh: Hạng mục quan trọng cần được triển khai với định nghĩa rõ ràng về yêu cầu, cần thêm công việc để hiểu và tinh chỉnh.

Sẵn sàng triển khai: PBI có đủ thông tin chi tiết để nhóm ước tính nỗ lực liên quan và triển khai, đã xác định các phụ thuộc.

2.3 Công nghệ sử dụng

2.3.1 RESTful API

2.3.1.1 Tổng quan về RESTful API

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động...), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

2.3.1.2 RESTful hoạt động như thế nào?

REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

GET (SELECT): Trả về một Resource hoặc một danh sách Resource.

POST (CREATE): Tạo mới một Resource.

PUT (UPDATE): Cập nhật thông tin cho Resource.

DELETE (DELETE): Xóa một Resource.

Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa.

2.3.2 Next.js & React.js

Next.js là một framework React hiện đại được sử dụng để xây dựng ứng dụng web full-stack. Next.js cung cấp nhiều tính năng mạnh mẽ như Server-Side Rendering (SSR), Static Site Generation (SSG), và API routes tích hợp.

React.js là một framework JavaScript mã nguồn mở được sử dụng để xây dựng giao diện người dùng web. React.js sử dụng mô hình component-based, trong đó giao diện web được xây dựng từ các component nhỏ. Điều này giúp giao diện web trở nên linh hoạt và dễ bảo trì.

2.3.3 Node.js & framework Express.js

2.3.3.1 Node.js

Giới thiệu Node.js

Node.js là một nền tảng JavaScript chạy trên máy chủ. Node.js được sử dụng để phát triển các ứng dụng web, mobile, IoT,...Node.js sử dụng mô hình eventdriven, trong đó các sự kiện được xử lý một cách không đồng bộ. Mô hình này giúp Node.js có thể xử lý nhiều yêu cầu cùng lúc một cách hiệu quả.

Cấu trúc của Node.js

Kernel: Là lớp cơ bản nhất của Node.js, cung cấp các chức năng cơ bản như xử lý sự kiện, quản lý bộ nhớ,...

Modules: Là các thư viện chức năng được sử dụng để xây dựng ứng dụng Node.js.

API: Là các giao diện lập trình ứng dụng cung cấp cho các nhà phát triển khả năng truy cập vào các chức năng của Node.js.

Các tính năng của Node.js

Ứng dụng web: Node.js được sử dụng để xây dựng các ứng dụng web động, hiệu quả và linh hoạt.

Ứng dụng mobile: Node.js được sử dụng để xây dựng các ứng dụng mobile với giao diện web.

Ứng dụng IoT: Node.js được sử dụng để xây dựng các ứng dụng IoT kết nối với các thiết bị vật lý.

2.3.3.2 Express.js

Express là một framework Node.js giúp xây dựng ứng dụng web một cách nhanh chóng và dễ dàng. Express cung cấp nhiều tính năng hữu ích cho việc phát triển ứng dụng web, bao gồm routing, middleware, session management,...

Routing

Routing là tính năng giúp định tuyến các yêu cầu đến các hàm xử lý tương ứng. Express sử dụng routing để định tuyến các yêu cầu đến các hàm xử lý dựa trên URL của yêu cầu.

Middleware

Middleware là các hàm được thực thi trước và sau các hàm xử lý. Middleware thường được sử dụng để thực hiện các tác vụ chung, chẳng hạn như xác thực người dùng, kiểm tra bảo mật,...

Session management

Session management là tính năng giúp lưu trữ thông tin trạng thái của người dùng trong một phiên. Express sử dụng session management để lưu trữ thông tin trạng thái của người dùng, chẳng hạn như tên người dùng, ID người dùng,...

Các tính năng của Express.js

Nhanh chóng: Express giúp xây dựng ứng dụng web một cách nhanh chóng.

Dễ sử dụng: Express có cú pháp đơn giản và dễ học.

Linh hoạt: Express có thể được tùy chỉnh để đáp ứng nhu cầu của các ứng dụng khác nhau.

2.3.4 Cơ sở dữ liệu (PostgreSQL & MySQL)

Dự án Schedule Manager sử dụng kiến trúc cơ sở dữ liệu hybrid với cả PostgreSQL và MySQL để tối ưu hóa hiệu suất và tính linh hoạt.

2.3.4.1 PostgreSQL

PostgreSQL là hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở được sử dụng làm cơ sở dữ liệu. PostgreSQL được chọn vì:

- ACID compliance đảm bảo tính toàn vẹn dữ liệu
- Hỗ trợ JSON để lưu trữ dữ liệu phức tạp (reminder, repeat)
- Hiệu suất cao với các truy vấn phức tạp
- Khả năng mở rộng tốt
- Hỗ trợ UUID làm primary key

2.3.4.2 Database Tools

Knex.js: SQL query builder được sử dụng với PostgreSQL, cung cấp migration system và connection pooling.

mysql2: MySQL driver cho Node.js với hỗ trợ Promise và connection pooling cho frontend API.

2.3.4.3 Redis

Redis được sử dụng cho caching và session management:

- Caching dữ liệu để tăng hiệu suất
- Lưu trữ session người dùng
- Hỗ trợ real-time features
- In-memory data structure store

2.3.5 Kiến trúc MVC

2.3.5.1 MVC là gì?

Mô hình MVC – Model-View-Controller là phương pháp chia nhỏ các thành phần dữ liệu (data), trình bày (output) và dữ liệu nhập từ người dùng (input) thành những thành phần riêng biệt.

2.3.5.2 MVC hoạt động như thế nào?

Thông thường, chúng ta biết rằng mô hình MVC gồm 3 thành phần: Model, View và Controller.

View

Về cơ bản, View đại diện cho cách dữ liệu được trình bày trong ứng dụng (UI). Các view được tạo ra dựa trên dữ liệu thu thập từ model. Bằng cách yêu cầu thông tin từ model, sau đó sẽ trả kết quả tới người dùng.

Controller

Controller là thành phần xử lý tương tác của người dùng. Dữ liệu đầu vào của người dùng được controller phân tích và xử lý, khi người dùng thao tác bất kì với hệ thống controller sẽ gửi thông tin đến model để xử lý và sau đó trả về kết quả view.

Model

Model là nơi lưu trữ dữ liệu và logic. Ví dụ, khi Controller truy xuất thông tin khách hàng từ cơ sở dữ liệu, dữ liệu được chuyển đổi giữa các thành phần controller hoặc giữa các yếu tố logic nghiệp vụ. Nó thao tác dữ liệu và gửi lại cơ sở dữ liệu, hoặc được sử dụng để hiển thị thông tin tương tự.

CHƯƠNG 3 XÁC ĐỊNH NHU CẦU

3.1 Product Backlog

PBI1: Tạo cấu trúc thư mục backend frontend, Setup Docker compose, dockerfile cho frontend backend.

PBI2: Tạo database, thiết kế bảng users trong MySQL, cài đặt Node.js, React, MySQL,...

PBI3: Thiết kế giao diện, viết giao diện và tích hợp tính năng cho trang đăng nhập/dăng ký.

PBI4: Thiết kế giao diện, viết giao diện và tích hợp tính năng cho trang Dashboard (bảng điều khiển và trang chủ).

PBI5: Thiết kế giao diện, viết giao diện và tích hợp tính năng cho trang Calendar (lịch).

PBI6: Thiết kế giao diện, viết giao diện và tích hợp tính năng cho trang Event (sự kiện).

PBI7: Thiết kế giao diện, viết giao diện và tích hợp tính năng cho trang Category (danh mục).

PBI8: Chức năng người dùng: Xem lịch theo chế độ tháng, tuần, ngày.

PBI9: Chức năng người dùng: Thiết lập nhắc nhở cho sự kiện.

PBI10: Chức năng người dùng: Tạo sự kiện lặp lại theo chu kỳ (hàng ngày, tuần, tháng).

PBI11: Thiết kế giao diện, viết giao diện và tích hợp tính năng cho trang Notifications (Thông báo).

PBI12: Thiết kế giao diện, viết giao diện cho trang Setting (tùy chỉnh).

PBI13: Tích hợp Swagger để mô tả các API.

PBI14: Triển khai lên Host.

PBI15: Viết báo cáo và hướng dẫn sử dụng.

3.2 Sprint backlog

Sprint 1 - Tạo giao diện cơ bản và xây dựng cơ sở hạ tầng back-end: Hoàn thành giao diện cơ bản cho Đăng nhập/Đăng ký, Dashboard, Calender, Category, Notifications, Setting. Xây dựng cơ sở hạ tầng back-end để xử lý đăng nhập/đăng ký, Dashboard, Calender, Category, Notifications.

Sprint 2 – Tích hợp Swagger mô tả các API, triển khai web lên Host, mô tả chức năng từng phần trong kiến trúc, thêm GitHub Actions, Viết báo cáo + hướng dẫn sử dụng.

3.3 Technical Backlog

Front-end

Thiết kế và phát triển giao diện cho mỗi trang (Đăng nhập/Đăng ký, Dashboard, Quản lý sự kiện, Lịch, Danh mục, Thông báo, Cài đặt).

Tối ưu hóa trải nghiệm người dùng trên các thiết bị di động và desktop.

Kiểm thử giao diện để đảm bảo tính nhất quán và hiệu quả.

Back-end

Thiết kế và xây dựng các API cho mỗi chức năng (xác thực người dùng, quản lý sự kiện, quản lý danh mục, thông báo, thống kê).

Tạo và quản lý cơ sở dữ liệu (PostgreSQL) để lưu trữ thông tin về người dùng, sự kiện, danh mục và thông báo.

Xử lý logic kinh doanh (business logic) như xác thực người dùng, quản lý sự kiện và phân quyền truy cập.

Đảm bảo bảo mật dữ liệu và xử lý lỗi một cách an toàn.

Tích hợp và Testing

Tích hợp giao diện front-end với back-end thông qua các API đã xây dựng.

Thực hiện kiểm thử chức năng (functional testing) để đảm bảo mọi tính năng hoạt động đúng như mong đợi.

Thực hiện kiểm thử tương tác (interaction testing) để đảm bảo giao diện và back-end hoạt động một cách nhất quán.

Kiểm tra tính bảo mật của ứng dụng để phòng tránh các lỗ hổng bảo mật tiềm ẩn.

Triển khai và Monitor

Đề tài: Xây dựng ứng dụng quản lý thời gian biểu

Triển khai ứng dụng lên môi trường sản phẩm (production environment) để người dùng có thể truy cập.

Thiết lập các công cụ theo dõi (monitoring tools) để theo dõi hiệu suất ứng dụng và phản ứng kịp thời với các vấn đề.

Phân tích dữ liệu và thông tin từ các công cụ theo dõi để cải thiện hiệu suất và trải nghiệm người dùng.

3.4 Support Backlog

Cập nhật tài liệu hướng dẫn sử dụng cho người dùng.

Xây dựng chức năng Thống kê và Báo cáo để người dùng có thể theo dõi hoạt động của mình.

Cải thiện khả năng tương thích của ứng dụng trên các trình duyệt web phổ biến như Google Chrome, Mozilla Firefox và Safari.

CHƯƠNG 4 LẬP KẾ HOẠCH SCRUM

4.1 Sprint 1:

Story 1: Lập trình viên, muốn hệ thống có cấu trúc rõ ràng và setup Docker

Story 2: Là lập trình viên, tôi muốn có database

Story 3: Là một người dùng, tôi muốn đăng ký và đăng nhập

Story 4: Là một người dùng, tôi muốn xem lịch để theo dõi các sự kiện

Story 5: Là một người dùng, tôi muốn có bảng điều khiển tổng quan để theo dõi thông tin nhanh

Story 6: Là một người dùng, tôi muốn có một trang để xem các thông báo

Story 7: Là một người dùng, tôi muốn có một trang để quản lý các sự kiện

Story 8: Là một người dùng, tôi muốn một nơi để tùy chỉnh giao diện, xem thông tin người dùng, gửi phản hồi

Story 9: Là một người dùng, tôi muốn phân loại sự kiện theo danh mục

ID	Issue	Person	Story	Start	End
SCRUM-8	Tạo cấu trúc thư mục backend/frontend	Đức	1	15/07/2025	18/07/2025
SCRUM-31	Setup Docker cho frontend/backend	Đức	1	15/07/2025	18/07/2025
SCRUM-9	Tạo database	Duy	2	15/07/2025	18/07/2025
SCRUM-14	Thiết kế bảng users trong MySQL	Duy	2	15/07/2025	18/07/2025
SCRUM-11	Cài đặt NodeJS, React, MySQL	Duy	2	15/07/2025	18/07/2025
SCRUM-10	Thiết kế giao diện đăng nhập	Khải	3	15/07/2025	18/07/2025
SCRUM-12	Viết giao diện đăng nhập	Khải	3	15/07/2025	18/07/2025
SCRUM-13	Tích hợp tính năng và hoàn thiện trang đăng nhập đăng ký	Khải	3	15/07/2025	18/07/2025

Đề tài: Xây dựng ứng dụng quản lý thời gian biểu

SCRUM-23	Thiết kế dashboard (trang chủ)	Khải	5	15/07/2025	18/07/2025
SCRUM-25	Viết giao diện dashboard (trang chủ)	Khải	5	15/07/2025	18/07/2025
SCRUM-24	Tích hợp tính năng cho trang dashboard (trang chủ)	Khải	5	15/07/2025	18/07/2025
SCRUM-15	Thiết kế giao diện calendar	Duy	4	15/07/2025	18/07/2025
SCRUM-16	Viết giao diện calendar	Duy	4	15/07/2025	18/07/2025
SCRUM-17	Tích hợp tính năng và hoàn thiện cho trang calendar	Duy	4	15/07/2025	18/07/2025
SCRUM-21	Thiết kế giao diện event	Đức	7	15/07/2025	18/07/2025
SCRUM-22	Viết giao diện event	Đức	7	15/07/2025	18/07/2025
SCRUM-24	Tích hợp tính năng và hoàn thiện cho trang event	Đức	7	15/07/2025	18/07/2025
SCRUM-16	Thiết kế giao diện cho trang category	Đức	9	15/07/2025	18/07/2025
SCRUM-17	Viết giao diện cho trang category	Đức	9	15/07/2025	18/07/2025
SCRUM-18	Tích hợp tính năng và hoàn thiện cho trang category	Đức	9	15/07/2025	18/07/2025
SCRUM-35	Thiết kế giao diện cho trang Notifications	Duy	6	15/07/2025	18/07/2025
SCRUM-36	Viết giao diện cho trang Notifications	Duy	6	15/07/2025	18/07/2025
SCRUM-37	Tích hợp tính năng và hoàn thiện cho trang Notification	Duy	6	15/07/2025	18/07/2025
SCRUM-39	Thiết kế giao diện setting	Khải	8	15/07/2025	18/07/2025

Đề tài: Xây dựng ứng dụng quản lý thời gian biểu

SCRUM-40	Viết giao diện setting	Khải	8	15/07/2025	18/07/2025
----------	------------------------	------	---	------------	------------

4.2 Sprint 2:

Story 10: Là một lập trình viên, tôi muốn trang web được tích hợp đầy đủ các tính năng được yêu cầu

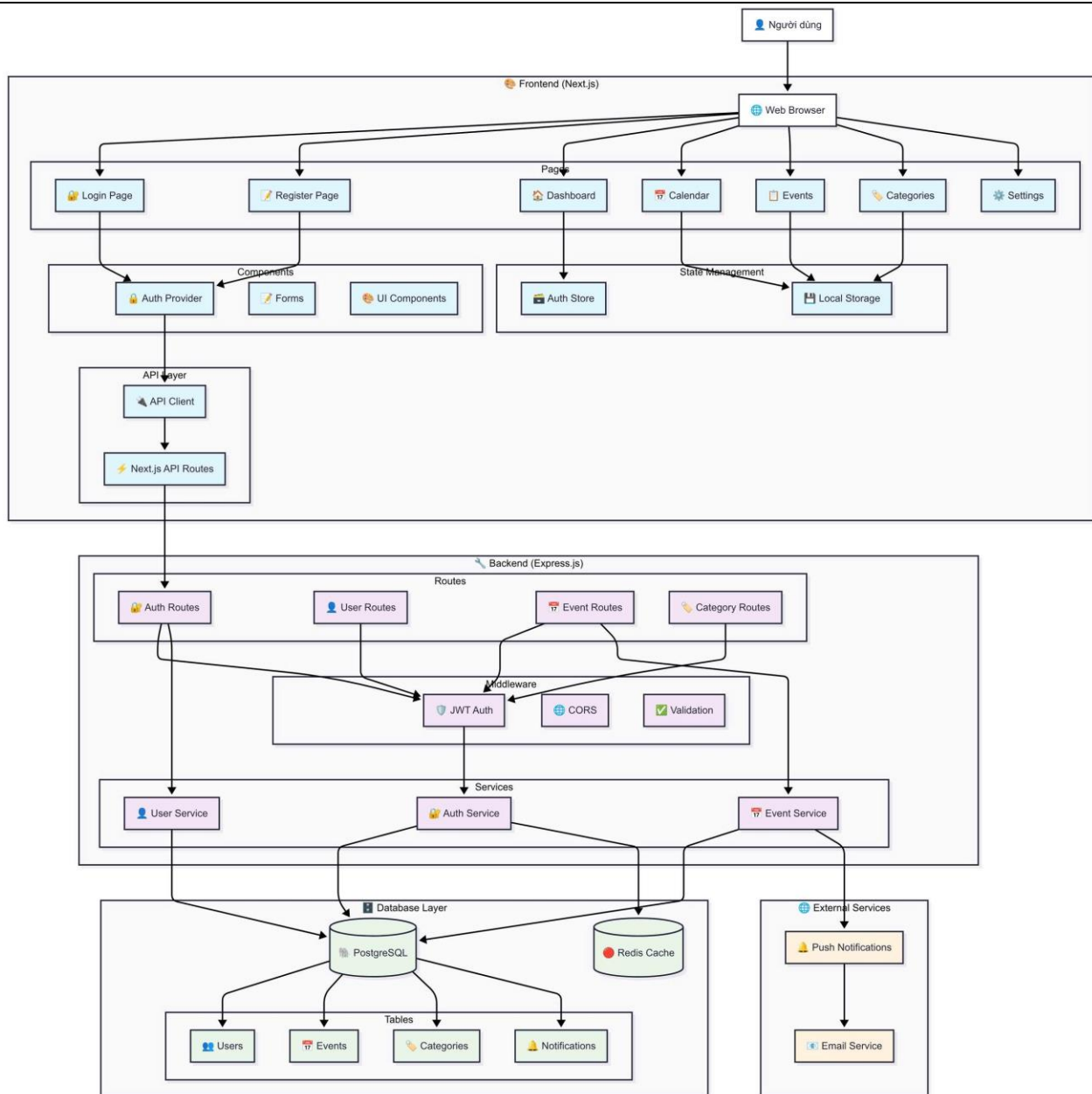
ID	Issue	Person	Story	Start	End
SCRUM-28	Tích hợp swagger mô tả các API	Duy	10	18/07/2025	23/07/2025
SCRUM-42	Triển khai lên host	Khải	10	18/07/2025	23/07/2025
SCRUM-29	Mô tả chức năng từng phần trong kiến trúc	Cả nhóm	10	18/07/2025	23/07/2025
SCRUM-30	Thêm GitHub Actions	Cả nhóm	10	18/07/2025	23/07/2025
SCRUM-32	Viết báo cáo + hướng dẫn sử dụng	Cả nhóm	10	18/07/2025	23/07/2025

CHƯƠNG 5 THIẾT KẾ HỆ THỐNG

5.1. Cấu trúc thư mục và kiến trúc hệ thống của dự án

```
schedule-manager/
├── backend/           # Backend API
│   ├── src/
│   │   ├── config/    # Database & app config
│   │   ├── middleware/ # Authentication middleware
│   │   ├── routes/     # API routes
│   │   └── server.js   # Main server file
│   ├── migrations/    # Database migrations
│   ├── Dockerfile     # Docker config
│   └── package.json    # Dependencies
├── frontend/         # Frontend app
│   ├── src/
│   │   ├── app/       # Next.js app directory
│   │   ├── components/ # React components
│   │   ├── hooks/     # Custom hooks
│   │   ├── lib/       # Utilities & API client
│   │   ├── store/     # State management
│   │   └── types/     # TypeScript types
│   ├── Dockerfile     # Docker config
│   └── package.json    # Dependencies
├── shared/           # Shared types
└── docker-compose.yml # Docker orchestration
```

Hình 5.1. Cấu trúc thư mục



Hình 5.2 Kiến trúc hệ thống

5.2. Cài đặt và chạy dự án

5.2.1. Yêu cầu hệ thống

- Node.js 18+
- PostgreSQL 15+ (Backend chính)
- MySQL 8.0+ (Frontend API)
- Redis 7+ (Caching)
- Docker & Docker Compose (khuyến nghị)

5.2.2. Cài đặt dependencies

```
# Backend  
cd backend  
npm install
```

```
# Frontend  
cd frontend  
npm install
```

5.2.3. Cấu hình environment

```
# Backend .env (PostgreSQL)  
DB_HOST=localhost  
DB_PORT=5432  
DB_USER=postgres  
DB_PASSWORD=password  
DB_NAME=schedule_manager  
JWT_SECRET=your-secret-key  
PORT=5001
```

```
# Frontend .env.local (MySQL)  
DB_HOST=localhost  
DB_PORT=3306  
DB_USER=root  
DB_PASSWORD=password  
DB_NAME=schedule_manager  
NEXT_PUBLIC_API_URL=http://localhost:5001/api
```

5.2.4. Chạy ứng dụng

Chạy với Docker

docker-compose up -d

Hoặc chạy manual

Terminal 1: Backend

cd backend

npm run dev

Terminal 2: Frontend

cd frontend

npm run dev

5.3. Database Migration

cd backend

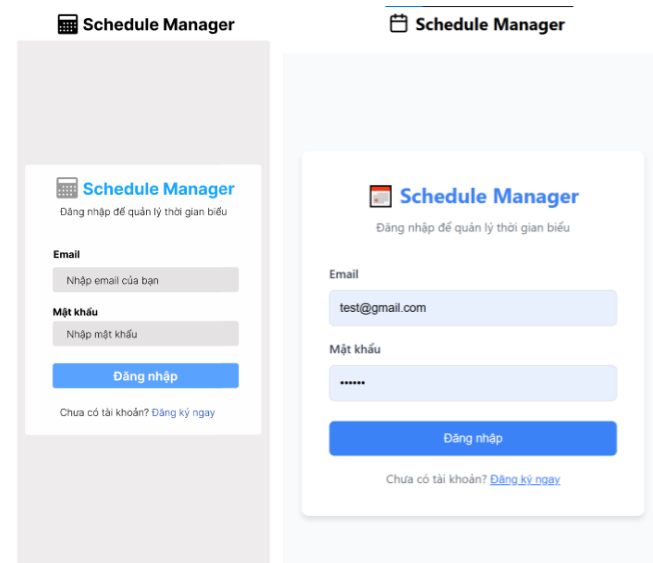
npx knex migrate:latest

CHƯƠNG 6 KẾT QUẢ THỰC NGHIỆM

6.1. Kết quả đạt được

6.1.1. Chức năng đã hoàn thành

Quản lý người dùng:

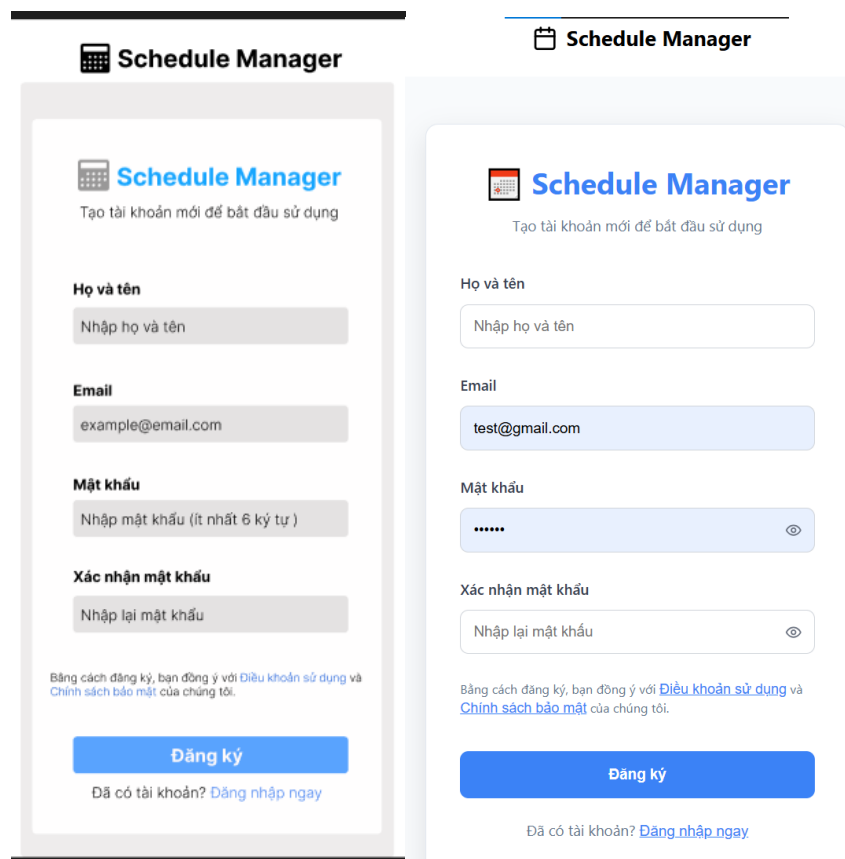


Hình 6.1 Giao diện đăng nhập

Chức năng giao diện:

Giao diện đăng nhập được thiết kế với phong cách hiện đại và thân thiện, sử dụng centered layout với form đăng nhập nằm chính giữa màn hình trên nền xám nhạt. Phần header hiển thị logo emoji 📅 cùng tên ứng dụng "Schedule Manager" bằng màu xanh dương nổi bật, kèm dòng mô tả "Đăng nhập để quản lý thời gian biểu" giúp người dùng hiểu rõ mục đích.

Form bao gồm hai trường input chính là email và mật khẩu với thiết kế border bo tròn và padding thoải mái. Nút đăng nhập có màu xanh dương đậm, chuyển sang xám khi đang xử lý và hiển thị text "Đang đăng nhập..." để thông báo trạng thái. Bên dưới có link chuyển đến trang đăng ký với màu xanh và gạch chân. Khi có lỗi, hệ thống hiển thị thông báo với nền đỏ nhạt và viền đỏ phía trên form. Toàn bộ giao diện responsive với maxWidth 400px, hoạt động tốt trên cả desktop và mobile.



Hình 6.2 Giao diện đăng ký

Chức năng giao diện:

Giao diện đăng ký mở rộng từ thiết kế trang đăng nhập nhưng với form phức tạp hơn để thu thập thông tin người dùng mới. Layout tương tự để đảm bảo tính nhất quán, nhưng form được mở rộng bao gồm bốn trường: họ tên, email, mật khẩu và xác nhận mật khẩu.

Form có validation real-time được powered bởi React Hook Form và Zod schema, giúp người dùng nhận feedback ngay lập tức khi nhập liệu không hợp lệ. Các trường mật khẩu tích hợp tính năng toggle hiện/ẩn để người dùng dễ kiểm tra mật khẩu đã nhập. Validation messages hiển thị dưới mỗi trường input với màu đỏ, chỉ xuất hiện khi có lỗi. Nút đăng ký có behavior tương tự nút đăng nhập với loading và disabled state. Phía dưới form có link chuyển về trang đăng nhập cho người dùng đã có tài khoản.

Backend

```
src/backend/
├── src/
│   ├── config/
│   │   └── database.js           // Cấu hình kết nối PostgreSQL
│   ├── middleware/
│   │   └── auth.js              // JWT authentication middleware
│   ├── routes/
│   │   ├── auth.js              // Routes đăng ký, đăng nhập, refresh token
│   │   └── users.js             // Routes quản lý profile, đổi mật khẩu
│   └── server.js                // Main server file
├── migrations/
│   └── 001_create_users_table.js // Database schema cho users
├── package.json                 // Dependencies: express, bcryptjs,
jsonwebtoken, joi, knex, pg
└── knexfile.js                 // Knex configuration
```

Frontend

```
src/frontend/src/
├── app/
│   ├── (auth)/
│   │   ├── login/
│   │   │   └── page.tsx         // Trang đăng nhập
│   │   ├── register/
│   │   │   └── page.tsx         // Trang đăng ký
│   │   └── profile/
│   │       └── page.tsx         // Trang quản lý profile
│   └── api/
```

```
| | └─ auth/
| |   └─ login/
| |     └─ route.ts           // API route đăng nhập
| |       └─ register/
| |         └─ route.ts       // API route đăng ký
| └─ layout.tsx              // Root layout
|   └─ page.tsx              // Home page
└─ components/
    └─ auth/
        └─ AuthProvider.tsx   // Context provider cho auth
        └─ AuthLayout.tsx     // Layout cho auth pages
        └─ LoginForm.tsx      // Form đăng nhập
        └─ RegisterForm.tsx   // Form đăng ký
        └─ ProfileForm.tsx     // Form cập nhật profile
        └─ ChangePasswordForm.tsx // Form đổi mật khẩu
        └─ PasswordToggle.tsx // Component show/hide password
    └─ ui/                    // UI components (Button, Input, Card...)
└─ store/
    └─ authStore.ts           // Zustand store cho localStorage auth
    └─ authStoreAPI.ts        // Zustand store cho API-based auth
└─ lib/
    └─ api.ts                 // API client class
    └─ validations.ts         // Zod validation schemas
    └─ utils.ts               // Utility functions
    └─ constants.ts           // App constants và API endpoints
└─ types/
    └─ index.ts               // TypeScript type definitions
```

Hình 6.3 Mã nguồn

Mô tả mã nguồn:

Backend Architecture

Mã nguồn backend được tổ chức theo kiến trúc modular với Express.js làm core framework. Cấu trúc thư mục src/ chứa các module chính: config/database.js quản lý kết nối PostgreSQL với connection pooling và error handling, middleware/auth.js implement JWT authentication middleware để verify tokens và protect routes.

Thư mục routes/ chứa auth.js xử lý authentication endpoints (register, login, refresh token) với bcryptjs để hash passwords và jsonwebtoken để generate/verify JWT tokens. users.js handle user management operations như profile updates và password changes với proper authorization checks.

server.js là entry point chính, setup Express application với CORS, body parsing, route mounting, và error handling middleware. Database migrations trong migrations/001_create_users_table.js define user schema với proper constraints và indexes. package.json specify dependencies: express cho web framework, bcryptjs cho password hashing, jsonwebtoken cho JWT handling, joi cho input validation, knex cho query builder, và pg cho PostgreSQL driver. knexfile.js chứa database configuration cho different environments (development, production).

Frontend Architecture

Frontend được xây dựng với Next.js App Router architecture, sử dụng TypeScript cho type safety. Cấu trúc app/ follow Next.js 13+ conventions với route groups: (auth)/ chứa authentication pages như login/page.tsx và register/page.tsx, profile/page.tsx cho user profile management.

api/auth/ chứa API routes với login/route.ts và register/route.ts handle authentication requests, có thể proxy đến backend hoặc handle directly. layout.tsx là root layout component, page.tsx là home page.

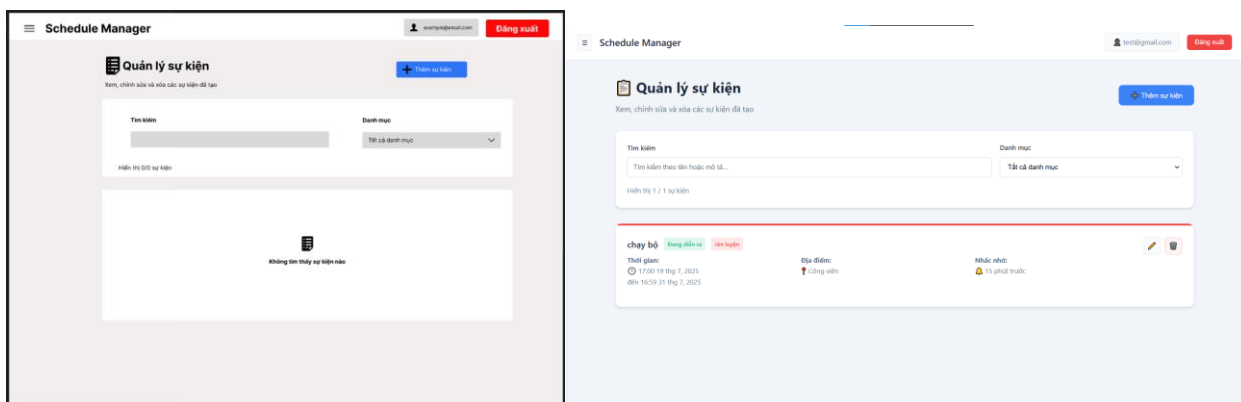
components/auth/ chứa authentication-related components: AuthProvider.tsx provide authentication context với React Context API, AuthLayout.tsx layout wrapper cho auth pages, LoginForm.tsx và RegisterForm.tsx là form components với validation, ProfileForm.tsx và ChangePasswordForm.tsx cho user management, PasswordToggle.tsx là reusable component cho show/hide password functionality.

store/ implement state management với Zustand: authStore.ts cho localStorage-based authentication state, authStoreAPI.ts cho API-based authentication với server synchronization.

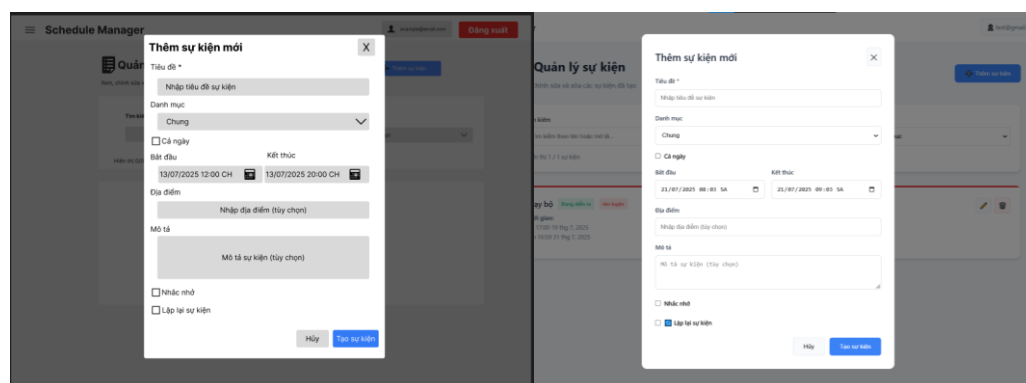
lib/ chứa utility modules: api.ts là centralized API client class với request/response interceptors, validations.ts define Zod schemas cho form validation, utils.ts chứa helper functions, constants.ts define app constants và API endpoints.

types/index.ts chứa TypeScript type definitions cho User, AuthState, API responses, ensuring type safety across application. Architecture này support scalability với clear separation of concerns và reusable components.

Quản lý sự kiện:



Hình 6.4 Giao diện quản lý sự kiện



Hình 6.5 Giao diện thêm sự kiện

Chức năng giao diện:

Giao diện quản lý sự kiện được thiết kế như comprehensive event management dashboard, cho phép xem, tìm kiếm, lọc và quản lý tất cả sự kiện trong một view. Filter bar đầu trang cung cấp search input với placeholder hướng dẫn và dropdown filter theo danh mục, giúp nhanh chóng tìm sự kiện cần thiết. Results counter hiển thị số lượng sự kiện được filter so với tổng số.

Danh sách sự kiện hiển thị dưới dạng cards với thiết kế rich information. Mỗi event card có color bar phía trên tương ứng category color, title với status badge và category badge, description nếu có, và grid layout

hiển thị thông tin chi tiết như thời gian, địa điểm, reminder settings, repeat patterns. Status badges sử dụng màu sắc khác nhau phân biệt "Sắp tới", "Đang diễn ra", và "Đã qua". Action buttons đặt góc phải mỗi card với edit và delete functionality. Empty state thiết kế thoughtfully với different messages tùy thuộc filter được apply hay không, cung cấp appropriate call-to-action cho từng tình huống.

Backend

```
src/backend/
├── src/
│   ├── routes/
│   │   ├── events.js           // CRUD operations cho events
│   │   └── server.js          // Main server với event routes
│   └── migrations/
│       └── 003_create_events_table.js // Database schema cho events
└── package.json                // Dependencies: joi cho validation
```

Frontend

```
src/frontend/src/
├── app/
│   ├── (dashboard)/
│   │   ├── events/
│   │   │   ├── page.tsx       // Trang quản lý events chính
│   │   │   └── [id]/
│   │   │       └── page.tsx    // Trang chi tiết event
│   │   └── api/
│   │       └── events/
│   │           └── route.ts    // API routes cho events
```

```
|      └── [id]/
|      └── route.ts           // API routes cho event cụ thể
|── components/
|   └── events/
|       ├── EventFormModal.tsx    // Form tạo/sửa event
|       ├── EventDetailsModal.tsx // Modal hiển thị chi tiết event
|       ├── EventList.tsx         // Danh sách events
|       ├── EventCard.tsx        // Card hiển thị event
|       ├── EventSearch.tsx      // Tìm kiếm events
|       └── EventFilters.tsx      // Bộ lọc events
|   └── calendar/
|       ├── CalendarView.tsx      // Component calendar chính
|       ├── MonthView.tsx        // View theo tháng
|       ├── WeekView.tsx         // View theo tuần
|       ├── DayView.tsx          // View theo ngày
|       └── CalendarHeader.tsx    // Header với navigation
|── store/
|   ├── eventStore.ts            // Zustand store cho events
|   └── calendarStore.ts         // Zustand store cho calendar
|── lib/
|   ├── eventStorage.ts          // LocalStorage utilities cho events
|   └── validations.ts           // Zod schemas cho event validation
|── types/
|   └── index.ts                 // TypeScript definitions cho Event
```

Hình 6.6 Mã nguồn

Mô tả mã nguồn:

Backend Architecture

Mã nguồn backend cho event management được thiết kế theo RESTful API pattern với Express.js. File `src/routes/events.js` implement đầy đủ CRUD operations cho events bao gồm GET (list/detail), POST (create), PUT (update), DELETE (remove) với proper HTTP status codes và error handling. Route handlers sử dụng `async/await` pattern để handle database operations và include input validation với Joi schemas.

`migrations/003_create_events_table.js` define database schema cho events table với các fields: id (primary key), title, description, startDate, endDate, isAllDay (boolean), categoryId (foreign key), location, reminder settings, repeat patterns, và timestamps. Schema include proper indexes cho performance và foreign key constraints để maintain data integrity.

`server.js` được update để mount event routes với proper middleware stack bao gồm authentication middleware cho protected routes, CORS configuration, và error handling middleware. `Package.json` include Joi dependency cho server-side validation, ensuring data consistency trước khi persist vào database.

Frontend Architecture

Frontend event management được xây dựng với Next.js App Router và component-based architecture. `app/(dashboard)/events/page.tsx` là main events management page implement comprehensive event listing với search, filtering, và pagination capabilities.

`app/(dashboard)/events/[id]/page.tsx` handle individual event detail pages với dynamic routing.

API routes trong `app/api/events/route.ts` provide RESTful endpoints cho event operations, có thể proxy requests đến backend hoặc handle directly với database. `app/api/events/[id]/route.ts` handle specific event operations với dynamic route parameters.

Component architecture trong `components/events/` được tổ chức theo single responsibility principle: `EventFormModal.tsx` handle event creation/editing với React Hook Form và Zod validation, `EventDetailsModal.tsx` display comprehensive event information với action buttons, `EventList.tsx` render paginated event lists với virtualization cho performance, `EventCard.tsx` là reusable component cho event display với consistent styling, `EventSearch.tsx` implement real-time search với debouncing, `EventFilters.tsx` provide advanced filtering options với category, date range, status filters.

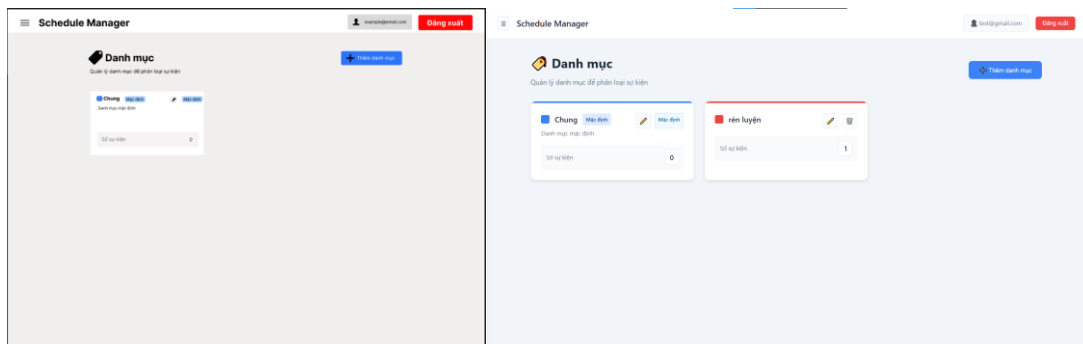
Calendar components trong `components/calendar/` implement multiple view modes: `CalendarView.tsx` là main container component với view switching logic, `MonthView.tsx` render traditional month grid với event indicators, `WeekView.tsx` display week timeline với time slots, `DayView.tsx` show detailed day schedule, `CalendarHeader.tsx` provide navigation controls và view mode switching.

State management sử dụng Zustand với `store/eventStore.ts` managing event data, CRUD operations, loading states, và error handling. `store/calendarStore.ts` handle calendar-specific state như current view, selected date, view mode preferences.

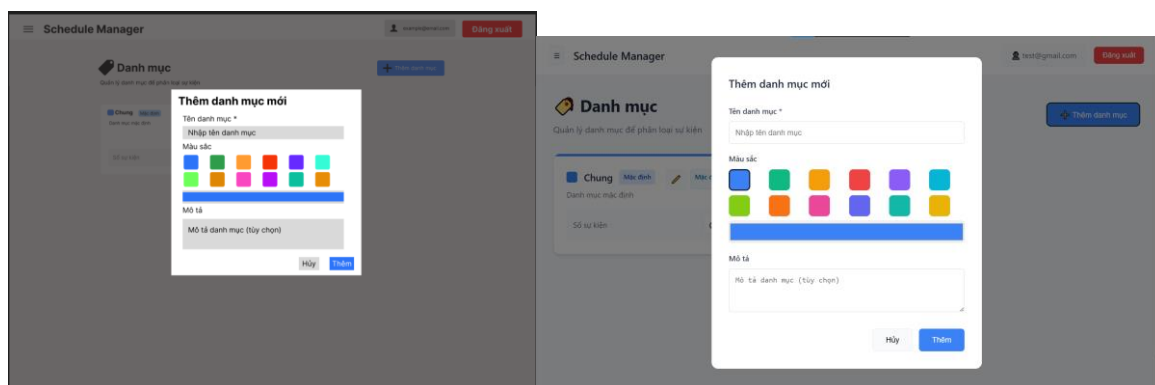
`lib/eventStorage.ts` provide `localStorage` utilities cho offline event management với serialization/deserialization, data migration, và sync capabilities. `lib/validations.ts` define comprehensive Zod schemas cho event validation bao gồm date validation, repeat pattern validation, reminder validation.

TypeScript definitions trong `types/index.ts` include Event interface với all properties, `EventFormData` cho form handling, `CalendarView` enums, `FilterOptions` interfaces, ensuring type safety across event management features.

Quản lý danh mục:



Hình 6.7 Giao diện quản lý danh mục



Hình 6.8 Giao diện thêm danh mục

Chức năng giao diện:

Giao diện quản lý danh mục giúp người dùng tổ chức và phân loại sự kiện hiệu quả. Trang sử dụng grid layout responsive hiển thị danh mục dưới dạng cards đẹp mắt, mỗi card có color bar phía trên tương ứng màu danh mục. Header có nút "Thêm danh mục" nổi bật với icon và text rõ ràng.

Mỗi category card hiển thị đầy đủ thông tin: tên danh mục với color indicator, mô tả nếu có, badge "Mặc định" cho default categories, và số lượng sự kiện đang sử dụng. Action buttons thiết kế thông minh với edit button luôn available, nhưng delete button chỉ enabled khi danh mục không phải default và không có sự kiện sử dụng. Modal form thêm/sửa danh mục có thiết kế

user-friendly với color picker hỗ trợ predefined colors và custom color selection. Grid predefined colors cho phép chọn màu phổ biến nhanh chóng, color input cho phép chọn màu tùy chỉnh. Form validation đảm bảo tên danh mục không trống và cung cấp feedback real-time.

Backend

```
src/backend/
├── src/
│   ├── routes/
│   │   ├── categories.js           // CRUD operations cho categories
│   │   └── server.js              // Main server với category routes
│   └── migrations/
│       └── 002_create_categories_table.js // Database schema cho categories
└── package.json                  // Dependencies: joi cho validation
```

Frontend

```
src/frontend/src/
├── app/
│   ├── (dashboard)/
│   │   ├── categories/
│   │   │   └── page.tsx           // Trang quản lý categories
│   │   └── api/
│   │       ├── categories/
│   │       │   ├── route.ts       // API routes cho categories
│   │       │   └── [id]/
│   │       │       └── route.ts    // API routes cho category cụ thể
│   └── components/
│       └── categories/
```

```
| | |— CategoryForm.tsx      // Form tạo/sửa category
| | |— CategoryList.tsx     // Danh sách categories
| | |— CategoryCard.tsx     // Card hiển thị category
| | |— CategoryPicker.tsx   // Selector cho forms
| | |— ColorPicker.tsx      // Component chọn màu
| | |— ui/                  // UI components
| | |— store/
| | |— eventStore.ts        // Category actions trong EventStore
| | |— lib/
| | |— categoryStorage.ts    // LocalStorage utilities cho categories
| | |— validations.ts       // Zod schemas cho category validation
| | |— types/
| | |— index.ts             // TypeScript definitions cho Category
```

Hình 6.9 Mã nguồn

Mô tả mã nguồn:

Backend Architecture

Mã nguồn backend cho category management implement RESTful API pattern với Express.js trong src/routes/categories.js. Route handlers cung cấp đầy đủ CRUD operations: GET endpoints cho listing categories và retrieving specific category, POST cho creating new categories với validation, PUT cho updating existing categories, và DELETE cho removing categories với business logic checking (prevent deletion nếu category đang được sử dụng bởi events).

migrations/002_create_categories_table.js define database schema cho categories table với các fields: id (primary key), name (unique constraint), description (optional), color (hex color code), isDefault (boolean

flag), createdAt, updatedAt timestamps. Schema include unique constraint trên name field để prevent duplicate category names và index trên isDefault field cho performance queries.

Route handlers implement business logic như checking category usage trước khi deletion, ensuring ít nhất một default category tồn tại, và validating color format. Server-side validation sử dụng Joi schemas để validate input data bao gồm name length, color format (hex), và required fields. Error handling provide meaningful error messages cho client consumption.

Frontend Architecture

Frontend category management được xây dựng với component-driven architecture trong Next.js. `app/(dashboard)/categories/page.tsx` là main categories management page implement grid layout cho category display, modal management cho create/edit operations, và integration với category store cho data management.

API routes trong `app/api/categories/route.ts` handle category collection operations (GET all, POST create) với proper error handling và response formatting. `app/api/categories/[id]/route.ts` handle individual category operations (GET, PUT, DELETE) với dynamic routing và parameter validation.

Component architecture trong `components/categories/` được thiết kế theo reusability principle: `CategoryForm.tsx` là comprehensive form component handle cả create và edit modes với React Hook Form integration, Zod validation, và `ColorPicker` integration. `CategoryList.tsx` render categories trong responsive grid layout với loading states và empty states.

CategoryCard.tsx display individual category information với color indicator, usage statistics, action buttons, và conditional rendering cho default categories.

CategoryPicker.tsx là reusable selector component được sử dụng trong event forms và filters, provide dropdown interface với color indicators và search functionality. ColorPicker.tsx implement advanced color selection với predefined color palette, custom color input, và color validation.

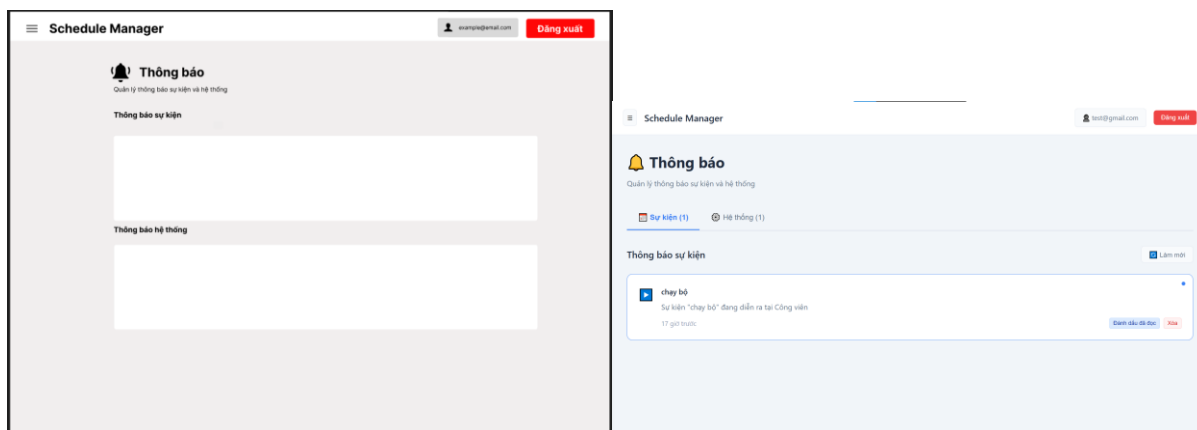
State management integrate với existing store/eventStore.ts thông qua category actions, maintaining consistency giữa events và categories. Store handle category CRUD operations, loading states, error handling, và relationship management với events.

lib/categoryStorage.ts provide localStorage utilities cho offline category management với data persistence, default category initialization, và sync capabilities với server state. Storage utilities handle serialization/deserialization và data migration cho schema changes.

Validation schemas trong lib/validations.ts define Zod schemas cho category validation bao gồm name validation (length, uniqueness), color validation (hex format), description validation, ensuring data integrity across client-side operations.

TypeScript definitions trong types/index.ts include Category interface với all properties, CategoryFormData cho form handling, ColorOption interfaces cho color picker, ensuring type safety và IntelliSense support across category management features.

Thông báo sự kiện:



Hình 6.10 Giao diện thông báo sự kiện

Chức năng giao diện:

Giao diện quản lý sự kiện được thiết kế như comprehensive event management dashboard, cho phép xem, tìm kiếm, lọc và quản lý tất cả sự kiện trong một view. Filter bar đầu trang cung cấp search input với placeholder hướng dẫn và dropdown filter theo danh mục, giúp nhanh chóng tìm sự kiện cần thiết. Results counter hiển thị số lượng sự kiện được filter so với tổng số.

Danh sách sự kiện hiển thị dưới dạng cards với thiết kế rich information. Mỗi event card có color bar phía trên tương ứng category color, title với status badge và category badge, description nếu có, và grid layout hiển thị thông tin chi tiết như thời gian, địa điểm, reminder settings, repeat patterns. Status badges sử dụng màu sắc khác nhau phân biệt "Sắp tới", "Đang diễn ra", và "Đã qua". Action buttons đặt góc phải mỗi card với edit và delete functionality. Empty state thiết kế thoughtfully với different messages tùy thuộc filter được apply hay không, cung cấp appropriate call-to-action cho từng tình huống.

Backend

src/backend/

```
|— src/
|   |— routes/
|   |   └─ notifications.js      // CRUD operations cho notifications
|   └─ server.js                // Main server với notification routes
|— migrations/
|   └─ 004_create_notifications_table.js // Database schema cho notifications
└─ package.json                // Dependencies: node-cron cho scheduled tasks
```

Frontend

src/frontend/src/

```
|— app/
|   |— (dashboard)/
|   |   └─ notifications/
|   |       └─ page.tsx        // Trang quản lý notifications
|   └─ api/
|       └─ notifications/
|           |— route.ts        // API routes cho notifications
|           └─ [id]/
|               └─ route.ts     // API routes cho notification cụ thể
|— components/
|   |— notifications/
|   |   |— NotificationList.tsx // Danh sách notifications
|   |   |— NotificationItem.tsx // Item hiển thị notification
|   |   └─ NotificationSettings.tsx // Cài đặt notifications
```

```
| | └─ ReminderForm.tsx          // Form cài đặt reminder cho event
| |   └─ NotificationBell.tsx    // Icon bell với badge count
|   └─ events/
|       └─ EventFormModal.tsx    // Form có reminder settings
└─ store/
    └─ notificationStore.ts      // Zustand store cho notifications
└─ hooks/
    └─ useNotifications.ts       // Custom hook cho notifications
└─ lib/
    └─ notificationStorage.ts    // LocalStorage utilities
    └─ notificationService.ts    // Service worker cho browser
notifications
└─ types/
    └─ index.ts                  // TypeScript definitions
```

Hình 6.11 Mã nguồn

Mô tả mã nguồn:

Backend Architecture

Mã nguồn backend cho notification system implement comprehensive notification management với Express.js trong src/routes/notifications.js. Route handlers cung cấp CRUD operations cho notifications bao gồm GET endpoints cho listing user notifications với pagination và filtering, POST cho creating manual notifications, PUT cho marking notifications as read/unread, và DELETE cho removing notifications. Routes cũng include specialized endpoints cho bulk operations như mark all as read và delete all read notifications.

migrations/004_create_notifications_table.js define database schema cho notifications table với các fields: id (primary key), userId (foreign key), eventId (foreign key, optional), type (enum: reminder, event_update, system), title, message, isRead (boolean), scheduledAt (timestamp cho scheduled notifications), createdAt, updatedAt. Schema include indexes trên userId, isRead, scheduledAt fields cho performance queries và foreign key constraints để maintain data integrity.

Backend integrate với node-cron dependency để handle scheduled notification tasks. Cron jobs được setup để check scheduled notifications, generate event reminders based trên event reminder settings, và cleanup old read notifications. Notification generation logic handle different notification types với appropriate message formatting và scheduling logic.

Frontend Architecture

Frontend notification system được xây dựng với real-time capabilities và comprehensive user interface. app/(dashboard)/notifications/page.tsx implement main notifications management page với notification list, filtering options, bulk actions, và real-time updates. Page handle notification state management và user interactions như marking as read, deleting, và filtering.

API routes trong app/api/notifications/route.ts handle notification collection operations với proper authentication và authorization. app/api/notifications/[id]/route.ts handle individual notification operations với parameter validation và error handling.

Component architecture trong components/notifications/ được thiết kế cho reusability và performance: NotificationList.tsx render notifications với

virtualization cho large lists, infinite scrolling, và real-time updates. NotificationItem.tsx display individual notifications với read/unread states, action buttons, và timestamp formatting. NotificationSettings.tsx provide comprehensive notification preferences management với toggle switches cho different notification types.

ReminderForm.tsx integrate với event creation/editing workflow, allowing users set reminder preferences với multiple reminder times, notification methods (browser, email), và custom messages. NotificationBell.tsx là header component display notification count badge với real-time updates và dropdown preview của recent notifications.

EventFormModal.tsx được enhanced với reminder settings integration, allowing users configure notifications khi creating hoặc editing events. Form include reminder time options, notification preferences, và repeat reminder settings.

State management sử dụng dedicated store/notificationStore.ts với Zustand, handling notification data, real-time updates, read/unread states, filtering, và user preferences. Store implement optimistic updates cho better user experience và sync với server state.

hooks/useNotifications.ts provide custom hook cho notification functionality bao gồm real-time subscription, notification permission handling, browser notification API integration, và cleanup logic. Hook handle notification lifecycle và provide convenient interface cho components.

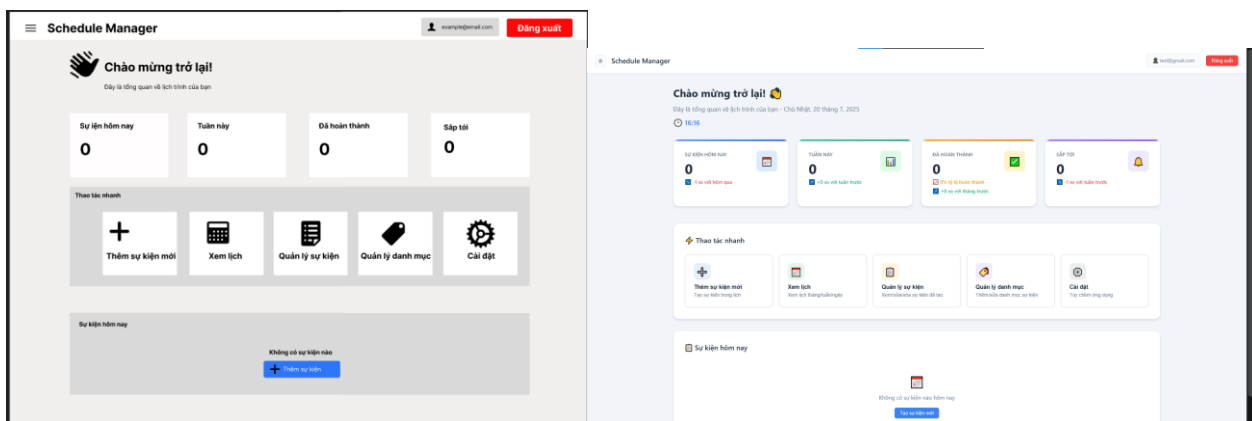
lib/notificationStorage.ts provide localStorage utilities cho offline notification management với data persistence, notification queue cho offline

Đề tài: Xây dựng ứng dụng quản lý thời gian biểu

scenarios, và sync capabilities. lib/notificationService.ts implement service worker integration cho browser notifications, background sync, và push notification handling.

TypeScript definitions trong types/index.ts include Notification interface với all properties, NotificationSettings cho user preferences, ReminderSettings cho event reminders, NotificationFilter cho filtering options, ensuring type safety across notification system features.

Giao diện trang Dashboards:



Hình 6.12 Giao diện trang dashboards

Chức năng giao diện:

Dashboard là trung tâm điều khiển của ứng dụng, cung cấp cái nhìn tổng quan toàn diện về lịch trình và hoạt động người dùng. Giao diện sử dụng modern dashboard layout với background xám nhạt và các card trắng tạo độ tương phản. Welcome section hiển thị lời chào thân thiện với emoji 🤝, ngày tháng hiện tại format tiếng Việt, và đồng hồ thời gian thực cập nhật mỗi phút.

Phần thống kê chính là grid responsive với bốn thẻ thống kê quan trọng, mỗi thẻ có gradient color bar phía trên, icon lớn bên phải, và số liệu chính với font size lớn. Đặc biệt, mỗi thẻ hiển thị trend comparison với periods trước sử dụng arrow icons và màu sắc biểu thị xu hướng. Quick

actions được thiết kế như grid các nút thao tác nhanh với icon, tiêu đề và mô tả, có hover effects mượt mà với animation transform. Cuối cùng là preview sự kiện hôm nay hiển thị danh sách với thông tin chi tiết và color coding theo danh mục.

Backend

src/backend/

```
|— src/
|   |— routes/
|   |   |— dashboard.js           // Dashboard API endpoints
|   |   |— stats.js              // Statistics endpoints
|   |   |— analytics.js          // Analytics endpoints
|   |— services/
|   |   |— statsService.js        // Business logic cho statistics
|   |   |— analyticsService.js    // Business logic cho analytics
|   |— package.json              // Dependencies: moment, lodash cho data
|   processing
```

Frontend

src/frontend/src/

```
|— app/
|   |— (dashboard)/
|   |   |— dashboard/
|   |   |   |— page.tsx          // Trang dashboard chính
|   |   |   |— layout.tsx        // Layout cho dashboard pages
|   |   |   |— loading.tsx       // Loading component
|   |— components/
|   |   |— dashboard/
|   |   |   |— DashboardOverview.tsx // Tổng quan dashboard
```

```
| | └─ StatsCards.tsx           // Cards hiển thị thống kê
| | └─ EventsChart.tsx         // Biểu đồ events theo thời gian
| | └─ CategoryChart.tsx       // Biểu đồ phân bố theo category
| | └─ UpcomingEvents.tsx      // Danh sách events sắp tới
| | └─ RecentActivity.tsx      // Hoạt động gần đây
| | └─ QuickActions.tsx        // Các action nhanh
| | └─ WeatherWidget.tsx       // Widget thời tiết (optional)
| └─ charts/
|   └─ LineChart.tsx           // Line chart component
|   └─ PieChart.tsx            // Pie chart component
|   └─ BarChart.tsx            // Bar chart component
|   └─ DonutChart.tsx          // Donut chart component
| └─ layout/
|   └─ Sidebar.tsx             // Sidebar navigation
|   └─ Header.tsx              // Header với user menu
|   └─ Breadcrumb.tsx          // Breadcrumb navigation
└─ store/
  └─ dashboardStore.ts         // Zustand store cho dashboard data
└─ lib/
  └─ chartUtils.ts             // Utilities cho charts
  └─ dashboardApi.ts           // API calls cho dashboard
└─ types/
  └─ dashboard.ts              // TypeScript definitions cho dashboard
```

Hình 6.13 Mã nguồn

Mô tả mã nguồn:

Backend Architecture

Mã nguồn backend cho dashboard system implement comprehensive analytics và statistics API với Express.js. `src/routes/dashboard.js` provide main dashboard endpoints bao gồm overview data aggregation, real-time metrics, và dashboard configuration endpoints. Route handlers aggregate data từ multiple sources để provide unified dashboard view.

`src/routes/stats.js` implement specialized statistics endpoints cho event statistics, user activity metrics, category distribution, time-based analytics, và comparative statistics. Endpoints support flexible date ranges, grouping options (daily, weekly, monthly), và filtering capabilities. `src/routes/analytics.js` provide advanced analytics endpoints cho trend analysis, performance metrics, usage patterns, và predictive analytics.

Business logic được tách riêng trong `src/services/statsService.js` và `src/services/analyticsService.js`. `statsService.js` implement complex statistical calculations bao gồm event completion rates, category usage statistics, time distribution analysis, và comparative metrics với previous periods. Service sử dụng `moment.js` cho date manipulation và `lodash` cho data processing và aggregation.

`analyticsService.js` handle advanced analytics logic bao gồm trend calculation, pattern recognition, performance analysis, và data visualization preparation. Service implement caching mechanisms cho expensive calculations và provide data transformation utilities cho chart consumption.

Package.json include moment dependency cho comprehensive date handling và lodash cho efficient data manipulation, aggregation, và utility functions. Services implement proper error handling, input validation, và performance optimization cho large datasets.

Frontend Architecture

Frontend dashboard system được xây dựng với comprehensive data visualization và real-time updates. app/(dashboard)/dashboard/page.tsx là main dashboard page orchestrate multiple dashboard components, handle data fetching, real-time updates, và responsive layout management. Page implement lazy loading cho performance và error boundaries cho robust error handling.

app/(dashboard)/layout.tsx provide consistent layout cho tất cả dashboard pages với sidebar navigation, header, breadcrumb, và responsive design. app/(dashboard)/loading.tsx provide skeleton loading states cho better user experience during data fetching.

Component architecture trong components/dashboard/ được thiết kế theo modular principle: DashboardOverview.tsx provide high-level dashboard summary với key metrics và navigation shortcuts. StatsCards.tsx render statistical information trong card format với trend indicators, comparison data, và interactive elements.

EventsChart.tsx implement time-series visualization cho event data với multiple chart types, date range selection, và drill-down capabilities. CategoryChart.tsx display category distribution với pie/donut charts,

interactive legends, và filtering options. UpcomingEvents.tsx show prioritized upcoming events với smart sorting và quick actions.

RecentActivity.tsx display user activity timeline với real-time updates, activity filtering, và detailed activity information. QuickActions.tsx provide shortcut buttons cho common dashboard actions với customizable action sets. WeatherWidget.tsx integrate external weather API cho location-based weather information.

Chart components trong components/charts/ provide reusable visualization components: LineChart.tsx, PieChart.tsx, BarChart.tsx, DonutChart.tsx implement responsive charts với interactive features, animation, tooltips, và export capabilities. Charts support theming, customization, và accessibility features.

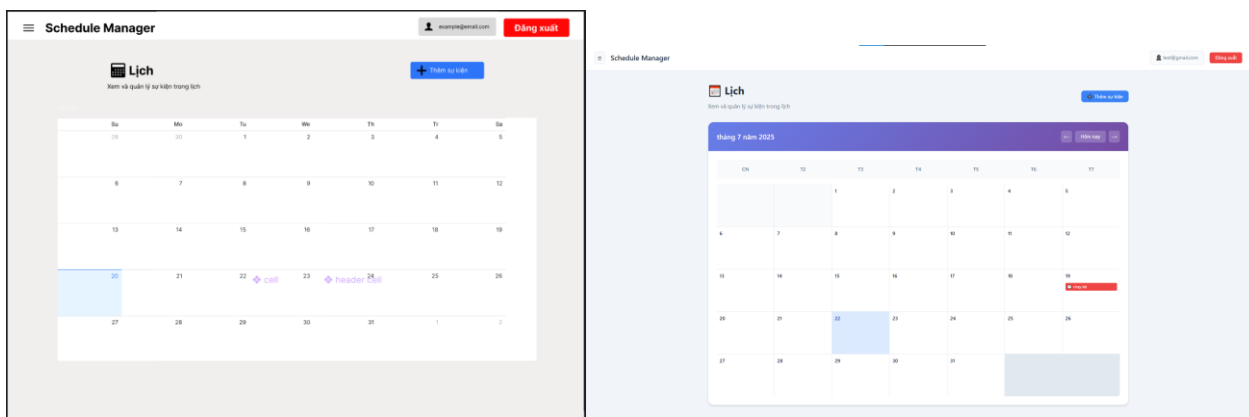
Layout components trong components/layout/ provide consistent navigation experience: Sidebar.tsx implement collapsible navigation với active state management, user preferences, và responsive behavior. Header.tsx include user menu, notifications, search functionality, và global actions. Breadcrumb.tsx provide contextual navigation với dynamic breadcrumb generation.

State management sử dụng store/dashboardStore.ts với Zustand, handling dashboard data, real-time updates, user preferences, chart configurations, và caching logic. Store implement optimistic updates, error handling, và data synchronization với server state.

lib/chartUtils.ts provide utilities cho chart data processing, color schemes, formatting functions, và chart configuration helpers. lib/dashboardApi.ts centralize tất cả dashboard-related API calls với proper error handling, caching, và request optimization.

TypeScript definitions trong types/dashboard.ts include comprehensive interfaces cho DashboardData, StatsData, ChartData, AnalyticsData, DashboardConfig, ensuring type safety và IntelliSense support across dashboard features.

Giao diện trang Calendar:



Hình 6.14 Giao diện trang calendar

Chức năng giao diện:

Giao diện lịch là phần phức tạp nhất, hiển thị sự kiện theo dạng calendar view truyền thống với nhiều tính năng hiện đại. Header có gradient background đẹp mắt với navigation controls di chuyển qua các tháng, nút "Hôm nay" quay về tháng hiện tại, và nút "Thêm sự kiện" tạo sự kiện mới.

Calendar grid thiết kế theo layout 7x6 chuẩn với header hiển thị các ngày trong tuần. Mỗi ô ngày chứa multiple events dưới dạng colored bars nhỏ

với text truncation. Ngày hôm nay được highlight với background xanh nhạt và font weight đậm. Hover vào các ô ngày có subtle background color change tạo feedback. Click vào ngày xuất hiện modal hiển thị tất cả sự kiện trong ngày với thông tin chi tiết, có thiết kế clean với header chứa ngày được chọn, danh sách sự kiện với full information, và action buttons edit/delete. Nếu ngày không có sự kiện, modal hiển thị empty state với call-to-action tạo sự kiện mới.

Frontend

src/frontend/src/

```
├── app/
│   ├── (dashboard)/
│       ├── calendar/
│           └── page.tsx           // Trang calendar chính
├── components/
│   ├── calendar/
│       ├── CalendarView.tsx     // Main calendar component
│       ├── CalendarHeader.tsx  // Header với navigation và view
│   ├── switcher
│       ├── MonthView.tsx       // Calendar view theo tháng
│       ├── CalendarGrid.tsx    // Grid layout cho calendar
│       ├── CalendarCell.tsx    // Cell cho mỗi ngày
│       ├── EventBlock.tsx      // Block hiển thị event trên calendar
│       ├── TimeSlot.tsx        // Time slot cho week/day view
│       ├── AllDayEvents.tsx    // Section cho all-day events
│       ├── CalendarSidebar.tsx // Sidebar với mini calendar và filters
│       ├── MiniCalendar.tsx    // Mini calendar navigator
│       └── CalendarFilters.tsx // Filters cho categories
```

		└─ CalendarLegend.tsx	// Legend cho category colors
		└─ events/	
		└─ EventFormModal.tsx	// Form tạo event từ calendar
		└─ EventDetailsModal.tsx	// Modal chi tiết event
		└─ QuickEventForm.tsx	// Quick add event form
		└─ ui/	
		└─ DatePicker.tsx	// Date picker component
		└─ TimePicker.tsx	// Time picker component
		└─ store/	
		└─ calendarStore.ts	// Zustand store cho calendar state
		└─ eventStore.ts	// Store cho events data
		└─ lib/	
		└─ calendarUtils.ts	// Utilities cho calendar calculations
		└─ dateUtils.ts	// Date manipulation utilities
		└─ constants.ts	// Calendar constants
		└─ types/	
		└─ calendar.ts	// TypeScript definitions cho calendar

Hình 6.15 Mã nguồn

Mô tả mã nguồn:

Frontend Architecture

Frontend calendar system được xây dựng với comprehensive calendar functionality và multiple view modes. `app/((dashboard))/calendar/page.tsx` là main calendar page orchestrate toàn bộ calendar interface, handle view mode switching, date navigation, event management, và responsive layout. Page implement keyboard shortcuts cho navigation, drag-and-drop functionality cho event management, và real-time synchronization với event data.

Component architecture trong components/calendar/ được thiết kế theo hierarchical structure: CalendarView.tsx là main container component handle view mode logic, date state management, event filtering, và coordinate giữa các sub-components. Component support multiple view modes (month, week, day) với smooth transitions và state persistence.

CalendarHeader.tsx provide comprehensive navigation interface với date navigation controls, view mode switcher, today button, event creation shortcuts, và calendar settings access. Header implement responsive design với collapsible elements cho mobile devices và keyboard navigation support.

MonthView.tsx implement traditional month calendar layout với CalendarGrid.tsx providing flexible grid system cho different calendar layouts. CalendarCell.tsx render individual day cells với event indicators, date highlighting, selection states, và interactive hover effects. Cell component handle click events, drag-and-drop operations, và accessibility features.

EventBlock.tsx display events trên calendar với smart positioning, color coding theo categories, text truncation cho long titles, và click handlers cho event details. Component support different event types (all-day, timed, multi-day) với appropriate visual representations.

TimeSlot.tsx implement time-based calendar views cho week và day modes với hourly slots, event positioning calculations, và time grid rendering. AllDayEvents.tsx handle all-day event display với separate section và overflow handling cho multiple events.

CalendarSidebar.tsx provide additional calendar functionality với MiniCalendar.tsx cho quick date navigation, CalendarFilters.tsx cho category-based filtering, và CalendarLegend.tsx cho category color reference. Sidebar implement collapsible design và user preference persistence.

Event management components trong components/events/ integrate seamlessly với calendar: EventFormModal.tsx handle event creation từ calendar clicks với pre-filled date/time information, EventDetailsModal.tsx display comprehensive event information với edit/delete actions, QuickEventForm.tsx provide rapid event creation với minimal input requirements.

UI components trong components/ui/ provide specialized input controls: DatePicker.tsx implement advanced date selection với calendar popup, keyboard input, date validation, và localization support. TimePicker.tsx provide time selection với dropdown interface, 12/24 hour format support, và time validation.

State management sử dụng store/calendarStore.ts với Zustand cho calendar-specific state bao gồm current view mode, selected date, view preferences, filter settings, và UI state. store/eventStore.ts handle event data với calendar integration, real-time updates, và optimistic updates cho better user experience.

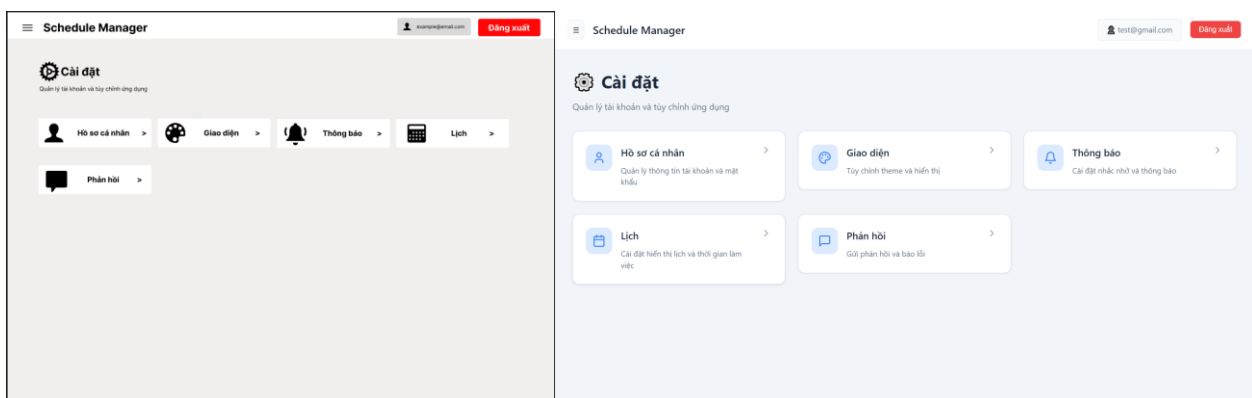
lib/calendarUtils.ts provide comprehensive calendar calculation utilities bao gồm month grid generation, week calculations, date range utilities, event positioning algorithms, và calendar layout helpers.

lib/dateUtils.ts implement date manipulation functions với timezone handling, localization, formatting utilities, và date arithmetic operations.

lib/constants.ts define calendar-related constants bao gồm view modes, date formats, time slots, default settings, và configuration options.


TypeScript definitions trong types/calendar.ts include comprehensive interfaces cho CalendarView enums, CalendarState, EventPosition, TimeSlot, CalendarConfig, DateRange, ensuring type safety và IntelliSense support across calendar functionality.

Giao diện trang Setting:

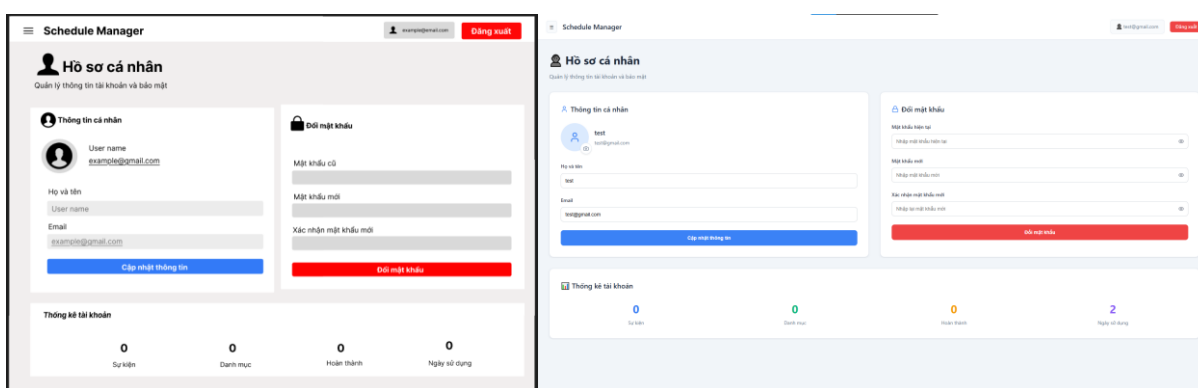


Hình 6.16 Giao diện trang Cài đặt

Chức năng giao diện:

Giao diện cài đặt được thiết kế như một hub trung tâm cho tất cả các tùy chỉnh và cấu hình của ứng dụng. Trang có layout clean và organized với header hiển thị icon  và tiêu đề "Cài đặt" cùng mô tả ngắn gọn "Quản lý tài khoản và tùy chỉnh ứng dụng".

Phần chính của trang được bố trí theo grid layout responsive với 6 thẻ cài đặt chính, mỗi thẻ có thiết kế card đẹp mắt với icon màu sắc, tiêu đề rõ ràng và mô tả ngắn gọn về chức năng. Các thẻ bao gồm "Hồ sơ cá nhân" để quản lý thông tin tài khoản, "Giao diện" cho tùy chỉnh theme và hiển thị, "Thông báo" để cài đặt notifications, "Lịch" cho calendar preferences, và "Phản hồi" để gửi feedback. Mỗi thẻ có hover effects với subtle animation và color transitions, tạo trải nghiệm tương tác mượt mà.

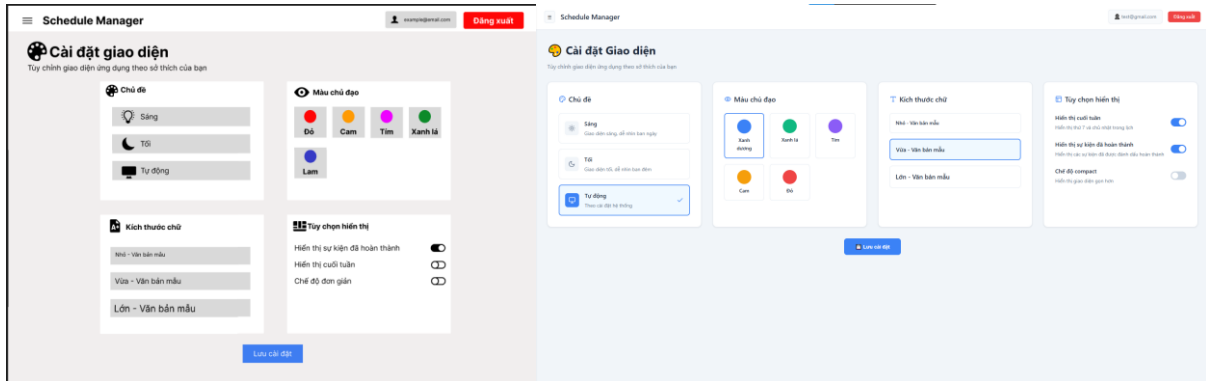


Hình 6.17 Giao diện trang Hồ sơ cá nhân

Chức năng giao diện:

Giao diện hồ sơ cá nhân được thiết kế để quản lý thông tin tài khoản và bảo mật một cách toàn diện. Trang được chia thành ba phần chính: thông tin cá nhân với avatar và form cập nhật, phần đổi mật khẩu với security focus, và thống kê tài khoản hiển thị usage metrics.

Phần thông tin cá nhân có avatar placeholder với nút camera để upload ảnh đại diện, hiển thị tên và email hiện tại, và form cập nhật với validation real-time. Phần đổi mật khẩu sử dụng PasswordToggle component cho tất cả password fields, đảm bảo security và user experience tốt. Phần thống kê hiển thị số lượng sự kiện, danh mục, events hoàn thành, và số ngày sử dụng ứng dụng.

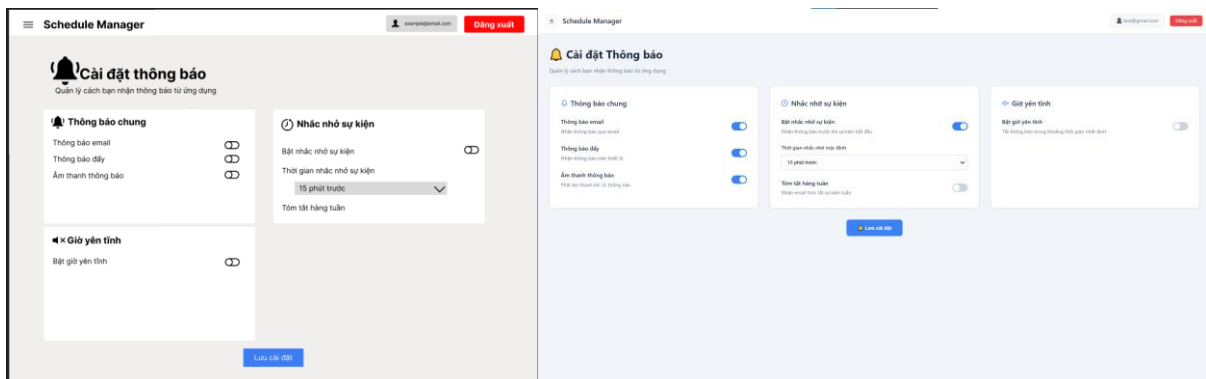


Hình 6.18 Giao diện trang Cài đặt Giao diện

Chức năng giao diện:

Giao diện cài đặt giao diện cho phép người dùng tùy chỉnh toàn diện về visual appearance của ứng dụng. Trang được chia thành bốn phần chính: chọn theme (sáng/tối/tự động), màu chủ đạo với 5 options màu sắc, kích thước font với 3 levels, và các tùy chọn hiển thị với toggle switches.

Phần chọn theme có 3 options với icons và descriptions rõ ràng, visual feedback khi selected. Color scheme section hiển thị color circles với tên màu, cho phép preview trực quan. Font size section có sample text để người dùng thấy được effect. Display options sử dụng custom toggle switches với smooth animations cho weekend display, completed events visibility, và compact mode.

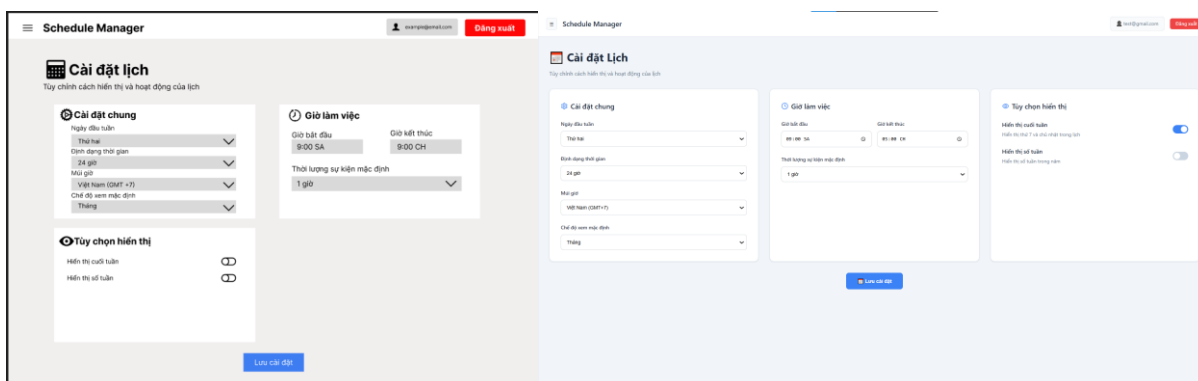


Hình 6.19 Giao diện trang Cài đặt thông báo

Chức năng giao diện:

Giao diện cài đặt thông báo cho phép người dùng kiểm soát toàn diện cách nhận thông báo từ ứng dụng. Trang được chia thành ba phần chính: thông báo chung với email/push/sound toggles, nhắc nhở sự kiện với default reminder time và weekly digest option, và giờ yên tĩnh với time range configuration.

Mỗi section có icon và title rõ ràng, sử dụng custom toggle switches với smooth animations. Event reminders section có conditional rendering cho reminder time dropdown khi enabled. Quiet hours section có time inputs cho start/end times khi activated. Tất cả settings được organize logical và có descriptions giúp người dùng hiểu rõ chức năng.

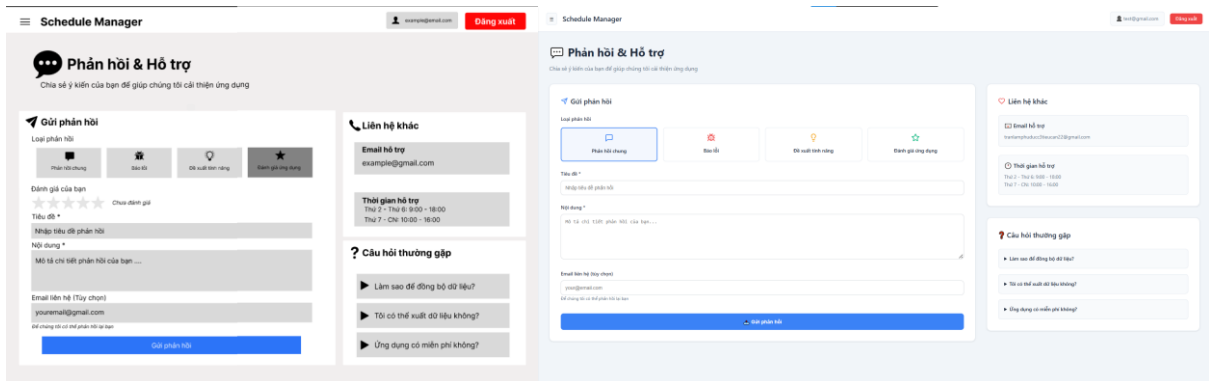


Hình 6.20 Giao diện trang Cài đặt Lịch

Chức năng giao diện:

Giao diện cài đặt lịch cho phép người dùng tùy chỉnh chi tiết cách hiển thị và hoạt động của calendar component. Trang được chia thành ba phần chính: cài đặt chung với week start, time format, timezone và default view; giờ làm việc với start/end times và default event duration; và tùy chọn hiển thị với weekend và week numbers toggles.

Phần cài đặt chung có các dropdown cho week start (Sunday/Monday), time format (12h/24h), timezone selection với multiple options, và default calendar view. Working hours section có time inputs cho start/end times và dropdown cho default event duration. Display options sử dụng toggle switches cho weekend display và week numbers visibility.



Hình 6.21 Giao diện trang Phản hồi & Hỗ Trợ

Chức năng giao diện:

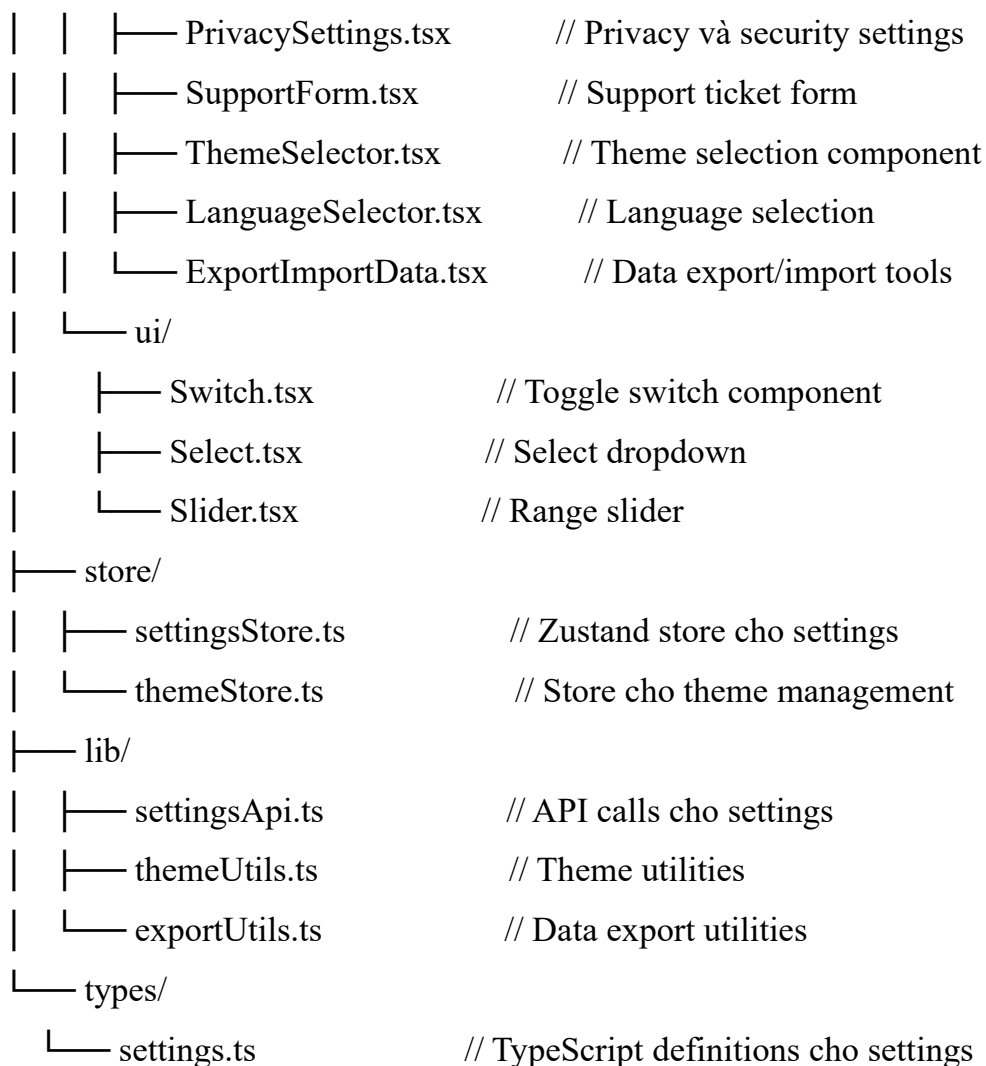
Giao diện phản hồi được thiết kế như một comprehensive support center với feedback form và support information. Trang có layout 2-column với main feedback form bên trái và sidebar chứa contact info và FAQ bên phải. Feedback form bao gồm feedback type selection với 4 categories (general, bug report, feature request, rating), conditional star rating cho rating type, subject và message fields, và optional email field.

Sidebar chứa contact information với support email và business hours, cùng với expandable FAQ section sử dụng HTML details/summary elements. Feedback types được hiển thị dưới dạng selectable cards với icons và colors khác nhau. Star rating component có interactive stars với hover và click effects.

Frontend

src/frontend/src/

```
|— app/
|   |— (dashboard)/
|       |— settings/
|           |— page.tsx          // Trang settings chính
|           |— profile/
|               |— page.tsx      // Cài đặt profile
|           |— appearance/
|               |— page.tsx      // Cài đặt giao diện
|           |— notifications/
|               |— page.tsx      // Cài đặt thông báo
|           |— calendar/
|               |— page.tsx      // Cài đặt lịch
|           |— privacy/
|               |— page.tsx      // Cài đặt riêng tư
|           |— support/
|               |— page.tsx      // Phản hồi & hỗ trợ
|— components/
|   |— settings/
|       |— SettingsLayout.tsx    // Layout cho settings pages
|       |— SettingsSidebar.tsx  // Sidebar navigation
|       |— SettingsHeader.tsx   // Header cho settings
|       |— SettingsCard.tsx     // Card wrapper cho settings sections
|       |— ProfileSettings.tsx  // Profile settings form
|       |— AppearanceSettings.tsx // Theme và appearance settings
|       |— NotificationSettings.tsx // Notification preferences
|       |— CalendarSettings.tsx // Calendar preferences
```



Hình 6.21 Mã nguồn

Mô tả mã nguồn:

Frontend Architecture

Frontend settings system được xây dựng với comprehensive configuration management và modular architecture. Cấu trúc pages trong `app/(dashboard)/settings/` implement nested routing với `page.tsx` là main settings hub, và các sub-pages cho specific setting categories: `profile/page.tsx` cho user profile management, `appearance/page.tsx` cho theme và UI customization, `notifications/page.tsx` cho notification preferences,

calendar/page.tsx cho calendar configuration, privacy/page.tsx cho security settings, và support/page.tsx cho help và feedback.

Component architecture trong components/settings/ được thiết kế theo consistent layout pattern: SettingsLayout.tsx provide unified layout cho tất cả settings pages với responsive sidebar, breadcrumb navigation, và consistent spacing. SettingsSidebar.tsx implement navigation menu với active state management, collapsible sections, và user preference persistence.

SettingsHeader.tsx provide page-specific headers với titles, descriptions, action buttons, và contextual help. SettingsCard.tsx là reusable wrapper component cho settings sections với consistent styling, collapsible functionality, và form organization.

Specialized settings components handle specific configuration areas: ProfileSettings.tsx implement comprehensive user profile management với avatar upload, personal information editing, account settings, và security options. Component include form validation, image processing, và data persistence.

AppearanceSettings.tsx provide extensive theme customization với ThemeSelector.tsx cho theme switching (light, dark, auto), color scheme selection, font size adjustment, và layout preferences. Component implement real-time preview, CSS custom properties management, và theme persistence.

NotificationSettings.tsx handle comprehensive notification preferences với granular controls cho different notification types, delivery

methods, quiet hours configuration, và notification history management. Component integrate với browser notification API và permission handling.

CalendarSettings.tsx provide calendar-specific configuration bao gồm default view preferences, working hours setup, timezone selection, week start configuration, và calendar display options. Settings integrate seamlessly với calendar components.

PrivacySettings.tsx handle security và privacy configuration với password management, two-factor authentication setup, data privacy controls, account deletion options, và security audit logs.

SupportForm.tsx implement comprehensive support ticket system với category selection, priority levels, file attachments, và ticket tracking. Form include rich text editing, validation, và submission handling.

Utility components trong components/ui/ provide specialized input controls: Switch.tsx implement animated toggle switches với accessibility support, Select.tsx provide enhanced dropdown functionality với search, multi-select, và custom styling, Slider.tsx implement range sliders với value display và step controls.

Additional components include LanguageSelector.tsx cho internationalization support và ExportImportData.tsx cho data portability với JSON export/import, backup creation, và data migration tools.

State management sử dụng store/settingsStore.ts với Zustand cho comprehensive settings state management bao gồm user preferences, form

states, validation states, và persistence logic. store/themeStore.ts handle theme-specific state với CSS custom properties management, theme switching logic, và system preference detection.

lib/settingsApi.ts centralize tất cả settings-related API calls với proper error handling, optimistic updates, và data synchronization. lib/themeUtils.ts provide theme manipulation utilities bao gồm CSS custom properties management, theme calculation, và color scheme utilities.

lib/exportUtils.ts implement data export/import functionality với JSON serialization, data validation, backup creation, và migration utilities.

TypeScript definitions trong types/settings.ts include comprehensive interfaces cho UserSettings, ThemeConfig, NotificationPreferences, CalendarSettings, PrivacySettings, SupportTicket, ensuring type safety và IntelliSense support across settings functionality.

6.1.2. API Documentation

- Swagger UI tại: `http://localhost:5001/api-docs`
- 29 endpoints được document đầy đủ
- Interactive testing interface

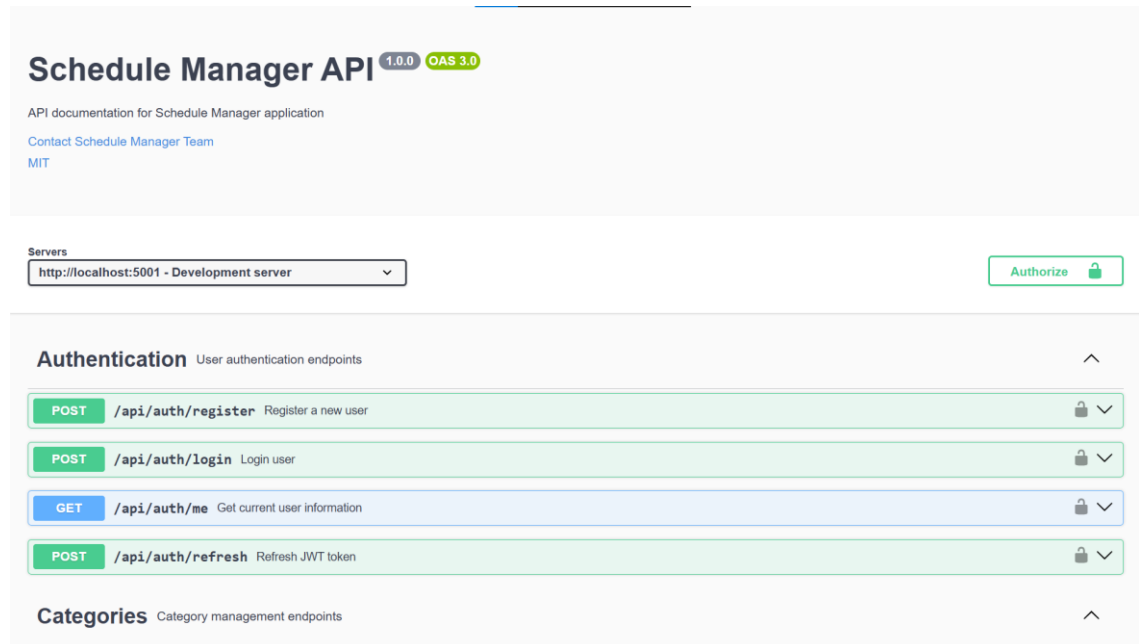
6.1.3. Performance

- Thời gian tải trang: < 2 giây
- API response time: < 500ms
- Database query optimization

6.2. Tích hợp tính năng

6.2.1. Tích hợp Swagger mô tả các API

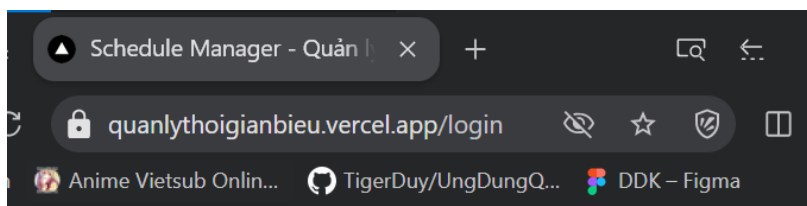
- Sử dụng Swagger để test chức năng của Login
- Sử dụng Swagger để test chức năng của Register
- Link Swagger: <http://localhost:5001/api-docs/>



Hình 6.22 Giao diện của Swagger

6.2.2. Triển khai lên host

- Sử dụng Vercel để triển khai web lên host
- Sử dụng Supabase để triển khai database
- Frontend và backend được tách ra để triển khai riêng trên 2 project trên Vercel.
- Link web khi được triển khai: (<https://schedule-manager-frontend-three.vercel.app/login>)

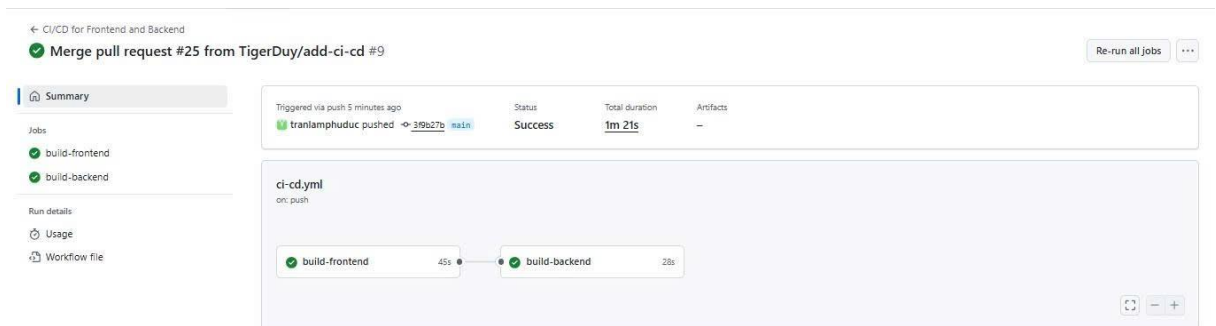


Schedule Manager

Hình 6.23 Giao diện Web được triển khai lên host có URL

6.2.3. Tích hợp Github Action

- Tự động build frontend
- Tự động cài đặt, test backend
- Hai job này chạy nối tiếp nhau: chỉ khi frontend build thành công thì backend mới được xử lý.



Hình 6.24 Giao diện sao khi tích hợp Github Action

CHƯƠNG 7 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

7.1. Kết luận

Dự án **Schedule Manager** đã được hoàn thành thành công với đầy đủ các chức năng cơ bản và nâng cao. Ứng dụng cung cấp một giải pháp toàn diện cho việc quản lý thời gian biểu cá nhân với:

7.1.1. Điểm mạnh

- **Kiến trúc hiện đại:** Sử dụng công nghệ mới nhất (Next.js 15, React 19)
- **User Experience tốt:** Giao diện trực quan, responsive
- **Performance cao:** Tối ưu hóa loading và API response
- **Bảo mật:** Implement các best practices về security
- **Scalability:** Kiến trúc có thể mở rộng dễ dàng

7.1.2. Thách thức đã vượt qua

- Tích hợp multiple technologies stack
- Thiết kế database schema phức tạp
- Implement real-time notifications
- Responsive design cho mobile devices
- Triển khai lên host

7.1.3. Kiến thức thu được

- Full-stack development với modern technologies
- Database design và optimization
- API design và documentation
- Security best practices
- DevOps với Docker

7.2. Hướng phát triển tương lai

7.2.1. Tính năng mở rộng

Phase 2 Features:

- Hoàn thiện chức năng cho trang setting
- **Real-time collaboration:** Chia sẻ lịch với người khác
- **Mobile app:** React Native hoặc Flutter
- **AI integration:** Smart scheduling suggestions
- **Calendar sync:** Tích hợp Google Calendar, Outlook
- **Advanced analytics:** Detailed reports và insights

Phase 3 Features:

- **Team management:** Quản lý lịch nhóm
- **Video conferencing:** Tích hợp Zoom/Meet
- **Workflow automation:** Zapier integration
- **Multi-language:** Internationalization
- **Offline support:** PWA capabilities

7.2.2. Cải tiến kỹ thuật

- **Microservices architecture:** Tách services riêng biệt
- **GraphQL API:** Thay thế REST API
- **Redis caching:** Improve performance
- **Elasticsearch:** Advanced search capabilities
- **Kubernetes:** Container orchestration

7.3. Đánh giá tổng thể

Dự án Schedule Manager đã đạt được mục tiêu đề ra và có thể được sử dụng trong thực tế. Với kiến trúc vững chắc và code quality tốt, ứng dụng có tiềm năng phát triển thành một sản phẩm thương mại.

TÀI LIỆU THAM KHẢO

- [1] Nextjs, "Next.js Docs," Nextjs, [Online]. Available: <https://nextjs.org/docs>. [Accessed 01 07 2025].
- [2] React, "React The library for web and native user interfaces," React, [Online]. Available: <https://react.dev>. [Accessed 01 07 2025].
- [3] Knex.js, "Knex.js SQL query builder," Knex.js, [Online]. Available: <https://knexjs.org>. [Accessed 01 07 2025].
- [4] Swagger, "API Development forEveryone," Swagger , [Online]. Available: <https://swagger.io>. [Accessed 01 07 2025].
- [5] L. Gupta, "REST API Tutorial," restfulapi.net, 01 04 2025. [Online]. Available: <https://restfulapi.net>. [Accessed 02 07 2025].
- [6] S. Brown, "Explore the world of cyber security," owasp.org, 28 06 2025. [Online]. Available: <https://owasp.org>. [Accessed 02 07 2025].
- [7] freecodecamp.org, "Build Your Skills for Free.," freecodecamp.org, [Online]. Available: freeCodeCamp.org. [Accessed 03 07 2025].
- [8] nodebestpractices, "nodebestpractices," github.com, 16 04 2025. [Online]. Available: <https://github.com/goldbergonyi/nodebestpractices>. [Accessed 05 07 2025].