**TRƯỜNG ĐẠI HỌC TRÀ VINH**

**TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Logo

Description automatically generated**

**ĐỒ ÁN MÔN HỌC**

**CÔNG NGHỆ PHẦN MỀM**

**(MSHP: 220055)**

**TÊN ĐỀ TÀI**

**XÂY DỰNG ỨNG DỤNG QUẢN LÝ**

**THỜI GIAN BIỂU SCHEDULE MANAGER**

**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 6 năm 2025**

**TRƯỜNG ĐẠI HỌC TRÀ VINH**

**TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Logo

Description automatically generated**

**ĐỒ ÁN MÔN HỌC**

**CÔNG NGHỆ PHẦN MỀM**

**(MSHP: 220055)**

**TÊN ĐỀ TÀI**

**XÂY DỰNG ỨNG DỤNG QUẢN LÝ**

**THỜI GIAN BIỂU SCHEDULE MANAGER**

**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 6 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](#_Toc204076160)

[1.1 Mô tả bài toán 1](#_Toc204076161)

[1.2 Mô tả ứng dụng 2](#_Toc204076162)

[1.3 Đặc tả các chức năng hệ thống 2](#_Toc204076163)

[1.3.1 Chức năng của người dùng 2](#_Toc204076164)

[1.3.2 Chức năng quản lý sự kiện 2](#_Toc204076165)

[1.3.3 Chức năng quản lý danh mục 2](#_Toc204076166)

[1.3.4 Quản lý người dùng 2](#_Toc204076167)

[1.4 Thiết kế dữ liệu 3](#_Toc204076168)

[1.4.1 Lược đồ cơ sở dữ liệu 3](#_Toc204076169)

[1.4.2 Mô hình dữ liệu 4](#_Toc204076170)

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 7](#_Toc204076171)

[2.1 Tổng quan về Agile 7](#_Toc204076172)

[2.1.1 Giới thiệu về Agile 7](#_Toc204076173)

[2.1.2 Tuyên ngôn Agile 7](#_Toc204076174)

[2.1.2.1 Giá trị của Tuyên ngôn Agile 7](#_Toc204076175)

[2.1.2.2 Nguyên tắc của Tuyên ngôn Agile 7](#_Toc204076176)

[2.2 Tổng quan về SCRUM 8](#_Toc204076177)

[2.2.1 SCRUM là gì ? 8](#_Toc204076178)

[2.2.2 Phương pháp SCRUM 8](#_Toc204076179)

[2.2.2.1 Thuật ngữ SCRUM 8](#_Toc204076180)

[2.2.2.2 Các vai trò quan trọng trong Scrum 9](#_Toc204076181)

[2.2.3 Scrum và Sprints 9](#_Toc204076182)

[2.2.4 Các thực hành chính của Scrum 9](#_Toc204076183)

[2.2.5 Trạng thái của Product Backlog Item 10](#_Toc204076184)

[2.3 Công nghệ sử dụng 10](#_Toc204076185)

[2.3.1 RESTful API 10](#_Toc204076186)

[2.3.1.1 Tổng quan về RESTful API 10](#_Toc204076187)

[2.3.1.2 RESTful hoạt động như thế nào? 10](#_Toc204076188)

[2.3.2 Next.js & React.js 11](#_Toc204076189)

[2.3.3 Node.js & framework Express.js 11](#_Toc204076190)

[2.3.3.1 Node.js 11](#_Toc204076191)

[2.3.3.2 Express.js 12](#_Toc204076192)

[2.3.4 Cơ sở dữ liệu (PostgreSQL & MySQL) 12](#_Toc204076193)

[2.3.4.1 PostgreSQL (Backend chính) 13](#_Toc204076194)

[2.3.4.2 MySQL (Frontend API) 13](#_Toc204076195)

[2.3.4.3 Database Tools 13](#_Toc204076196)

[2.3.4.4 Redis 13](#_Toc204076197)

[2.3.5 Kiến trúc MVC 14](#_Toc204076198)

[2.3.5.1 MVC là gì? 14](#_Toc204076199)

[2.3.5.2 MVC hoạt động như thế nào? 14](#_Toc204076200)

[CHƯƠNG 3 XÁC ĐỊNH NHU CẦU 15](#_Toc204076201)

[3.1 Product Backlog 15](#_Toc204076202)

[3.2 Sprint backlog 16](#_Toc204076203)

[3.3 Technical Backlog 16](#_Toc204076204)

[3.4 Support Backlog 17](#_Toc204076205)

[CHƯƠNG 4 LẬP KẾ HOẠCH SCRUM 18](#_Toc204076206)

[4.1 Sprint 1: 18](#_Toc204076207)

[4.2 Sprint 2: 20](#_Toc204076208)

[CHƯƠNG 5 THIẾT KẾ HỆ THỐNG 21](#_Toc204076209)

[5.1. Cấu trúc thư mục dự án 21](#_Toc204076210)

[5.2. Cài đặt và chạy dự án 22](#_Toc204076211)

[5.2.1. Yêu cầu hệ thống 22](#_Toc204076212)

[5.2.2. Cài đặt dependencies 22](#_Toc204076213)

[5.2.3. Cấu hình environment 22](#_Toc204076214)

[5.2.4. Chạy ứng dụng 23](#_Toc204076215)

[5.3. Database Migration 23](#_Toc204076216)

[CHƯƠNG 6 KẾT QUẢ THỰC NGHIỆM 24](#_Toc204076217)

[6.1. Kết quả đạt được 24](#_Toc204076218)

[6.1.1. Chức năng đã hoàn thành 24](#_Toc204076219)

[6.1.2. API Documentation 52](#_Toc204076220)

[6.1.3. Performance 52](#_Toc204076221)

[6.2. Tích hợp tính năng 52](#_Toc204076222)

[6.2.1. Tích hợp swagger mô tả các API 52](#_Toc204076223)

[6.2.2. Triển khai lên host 52](#_Toc204076224)

**DANH MỤC HÌNH ẢNH**

[Hình 1.1. Mô hình dữ liệu mức Quan niệm 4](#_Toc204076225)

[Hình 1.2. Mô hình dữ liệu mức Luận lý 5](#_Toc204076226)

[Hình 1.3. Mô hình dữ liệu mứcVật lý 6](#_Toc204076227)

[Hình 5.1. Cấu trúc thư mục 21](#_Toc204076228)

[Hình 6.1 Giao diện quản lý người dùng trên figma 25](#_Toc204076229)

[Hình 6.2 Giao diện quản lý người dùng 26](#_Toc204076230)

[Hình 6.3 Mã nguồn 28](#_Toc204076231)

[Hình 6.4 Giao diện quản lý sự kiện trên figma 29](#_Toc204076232)

[Hình 6.5 Giao diện quản lý sự kiện 30](#_Toc204076233)

[Hình 6.6 Mã nguồn 32](#_Toc204076234)

[Hình 6.7 Giao diện quản lý danh mục figma 33](#_Toc204076235)

[Hình 6.8 Giao diện quản lý danh mục 34](#_Toc204076236)

[Hình 6.9 Mã nguồn 36](#_Toc204076237)

[Hình 6.10 Giao diện thông báo sự kiện trên figma 36](#_Toc204076238)

[Hình 6.11 Giao diện thông báo sự kiện 37](#_Toc204076239)

[Hình 6.12 Mã nguồn 38](#_Toc204076240)

[Hình 6.13 Giao diện trang dashboards trên figma 39](#_Toc204076241)

[Hình 6.14 Giao diện trang dashboards 39](#_Toc204076242)

[Hình 6.15 Mã nguồn 41](#_Toc204076243)

[Hình 6.16 Giao diện trang calendar trên figma 42](#_Toc204076244)

[Hình 6.17 Giao diện trang calendar 42](#_Toc204076245)

[Hình 6.18 Mã nguồn 44](#_Toc204076246)

[Hình 6.19 Giao diện trang setting trên figma 47](#_Toc204076247)

[Hình 6.20 Giao diện trang Setting 50](#_Toc204076248)

[Hình 6.21 Mã nguồn 51](#_Toc204076249)

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:** Hiển 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:** Hiển 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**

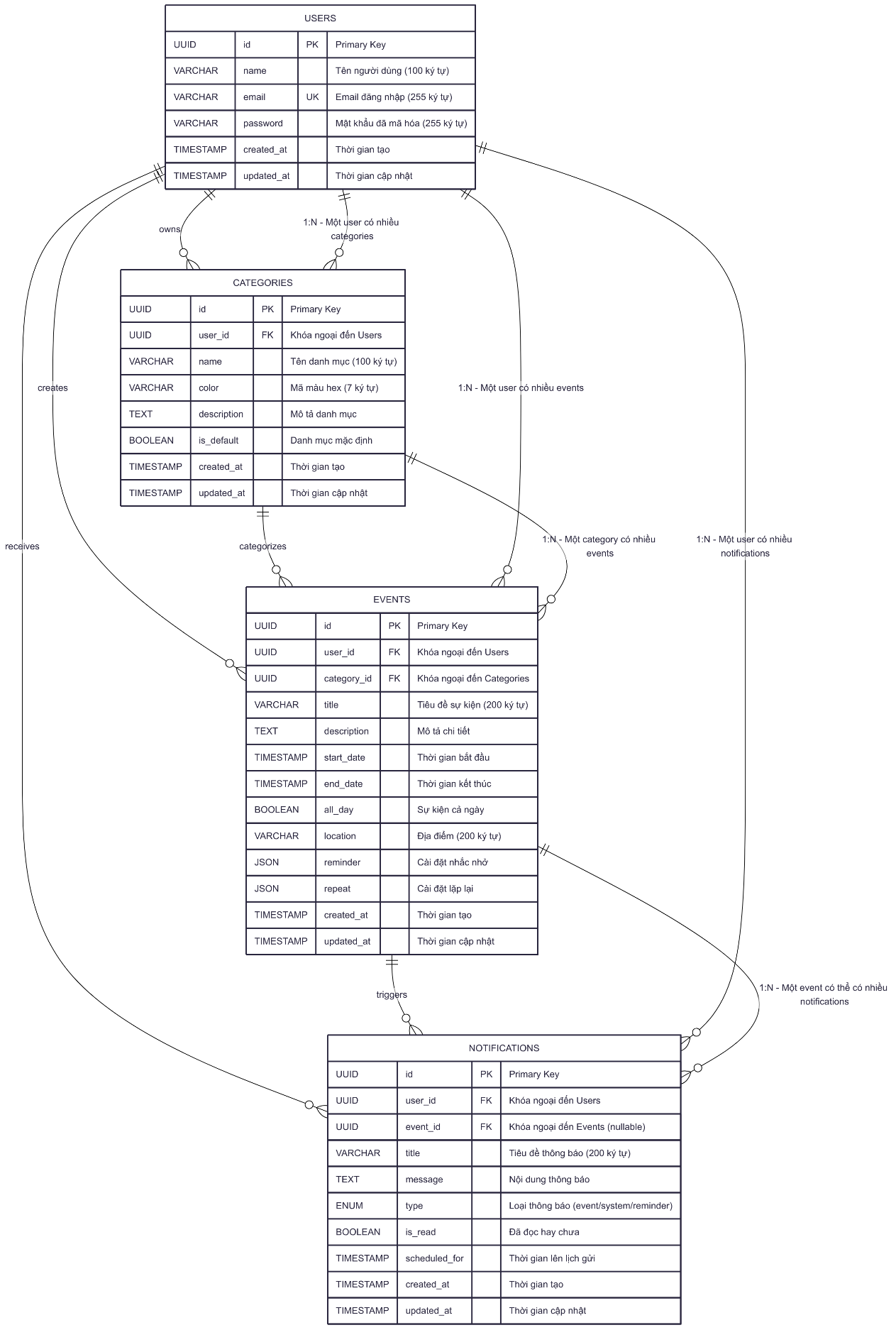
**1 users PostgreSQL** Lưu trữ thông tin người dùng

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

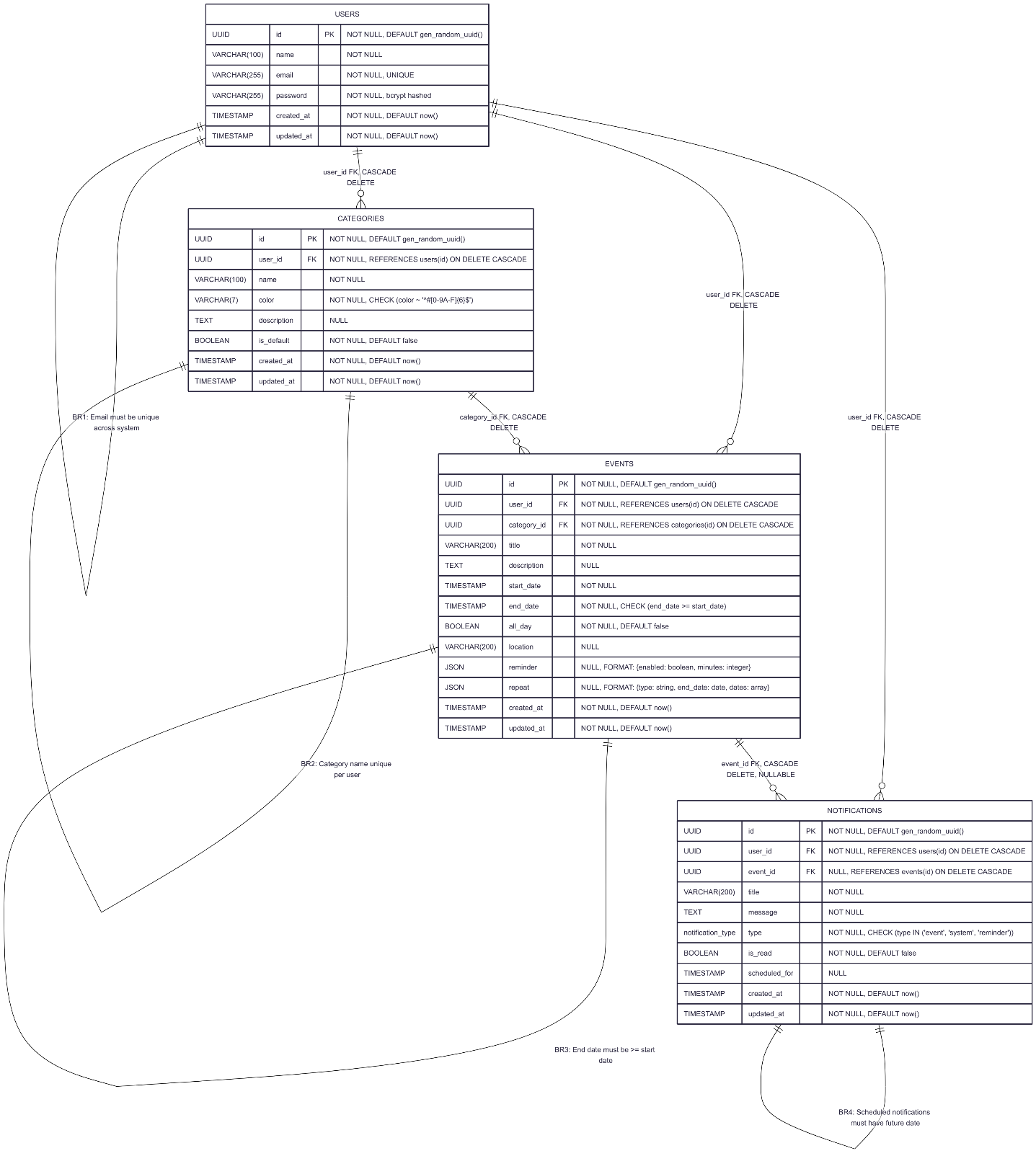
**3 event PostgreSQL + MySQL** Lưu trữ thông tin sự kiện với reminder/repeat

**4 notifications PostgreSQL** Lưu trữ thông tin thông báo hệ thống

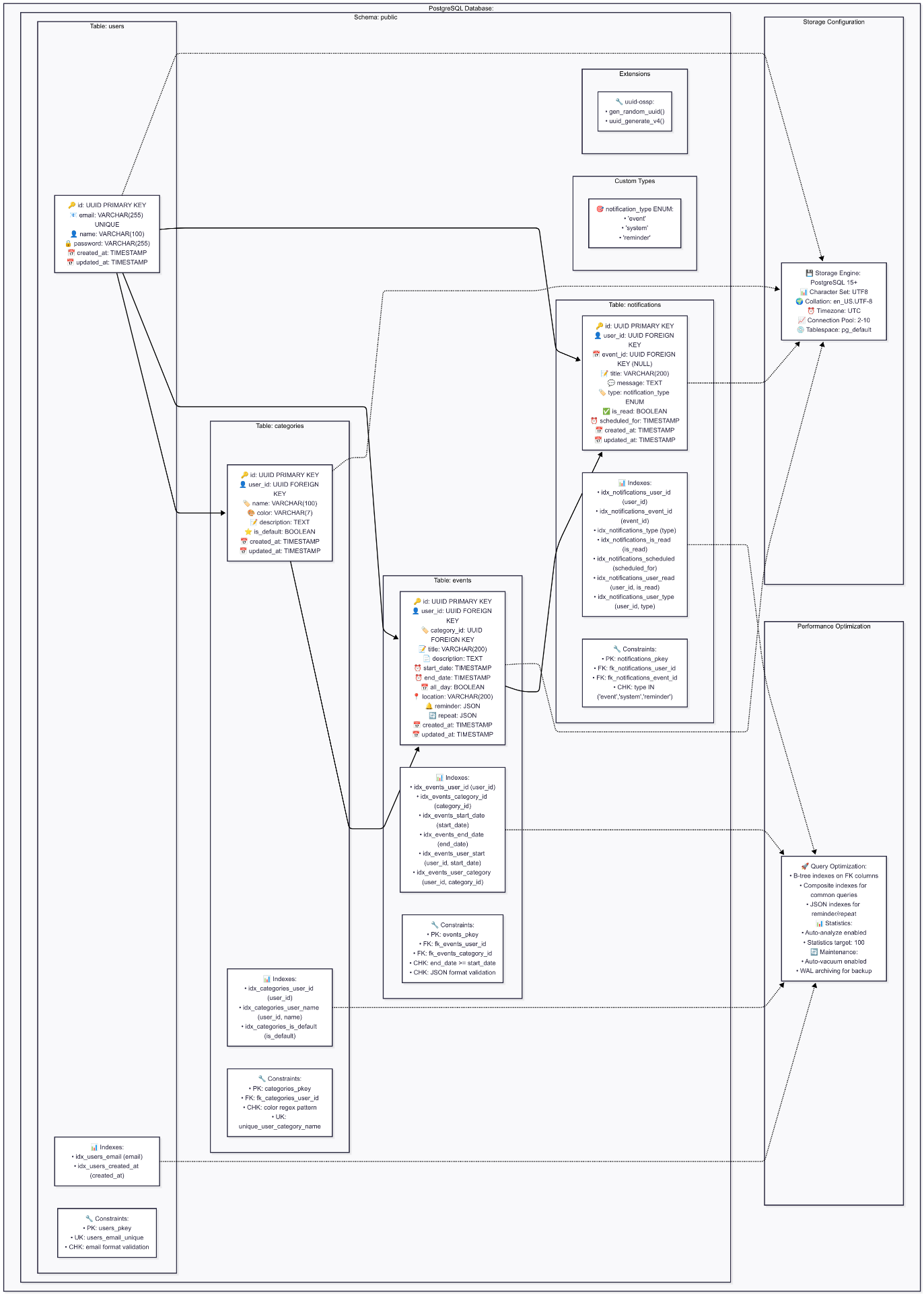
1.4.2 Mô hình dữ liệu



Hình 1.1. Mô hình dữ liệu mức Quan niệm



Hình 1.2. Mô hình dữ liệu mức Luận lý



Hình 1.3. Mô hình dữ liệu mứcVật 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):** Xoá 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 (Backend chính)

PostgreSQL là hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở được sử dụng làm cơ sở dữ liệu chính cho backend API. 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 MySQL (Frontend API)

MySQL được sử dụng cho các API routes trong frontend Next.js để xử lý một số tác vụ cụ thể:

- Kết nối trực tiếp từ frontend đến database

- Xử lý các truy vấn đơn giản và nhanh chóng

- Tương thích tốt với mysql2 driver

- Connection pooling hiệu quả

Cấu trúc dữ liệu chính được lưu trữ trong PostgreSQL, trong khi MySQL hỗ trợ các tác vụ phụ và cache dữ liệu.

2.3.4.3 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.

2.3.4.4 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 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 lí, khi người dùng thao tác bất kì với hệ thống controller sẽ gửi thông tin đến model để xử lí 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 fronend 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/ đăng 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

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 |
| 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 |
| 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 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

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

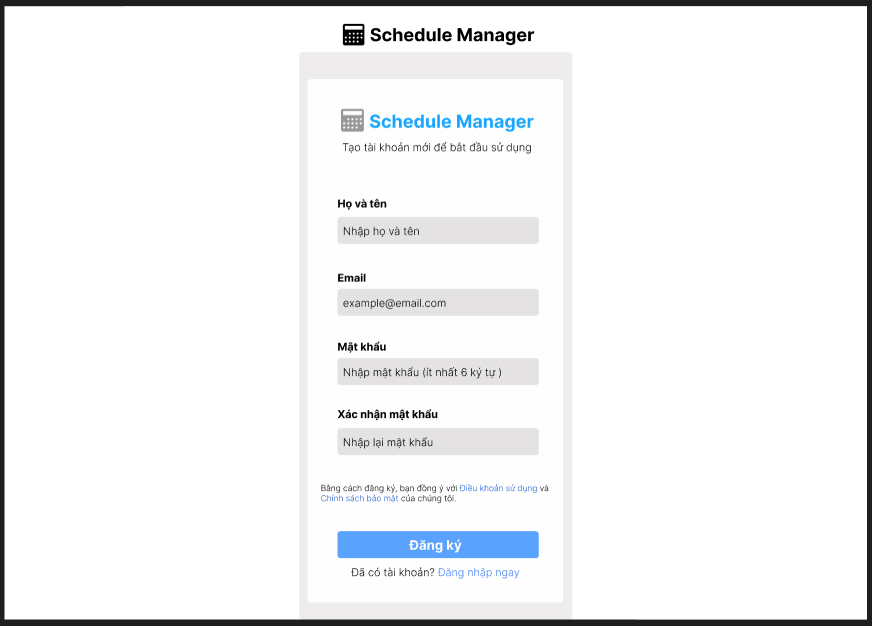
**Quản lý người dùng:**

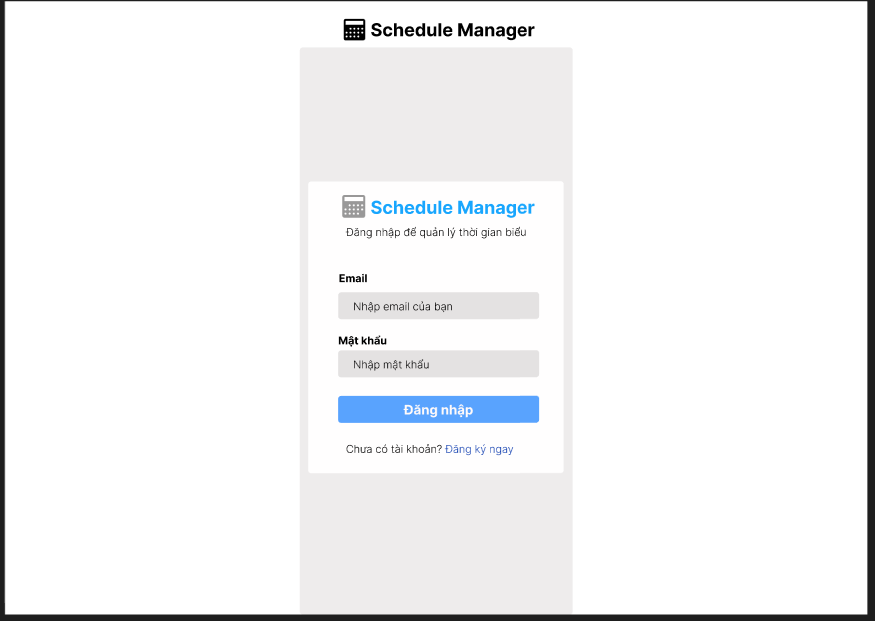
- Đăng ký, đăng nhập, đăng xuất

- Quản lý profile và đổi mật khẩu

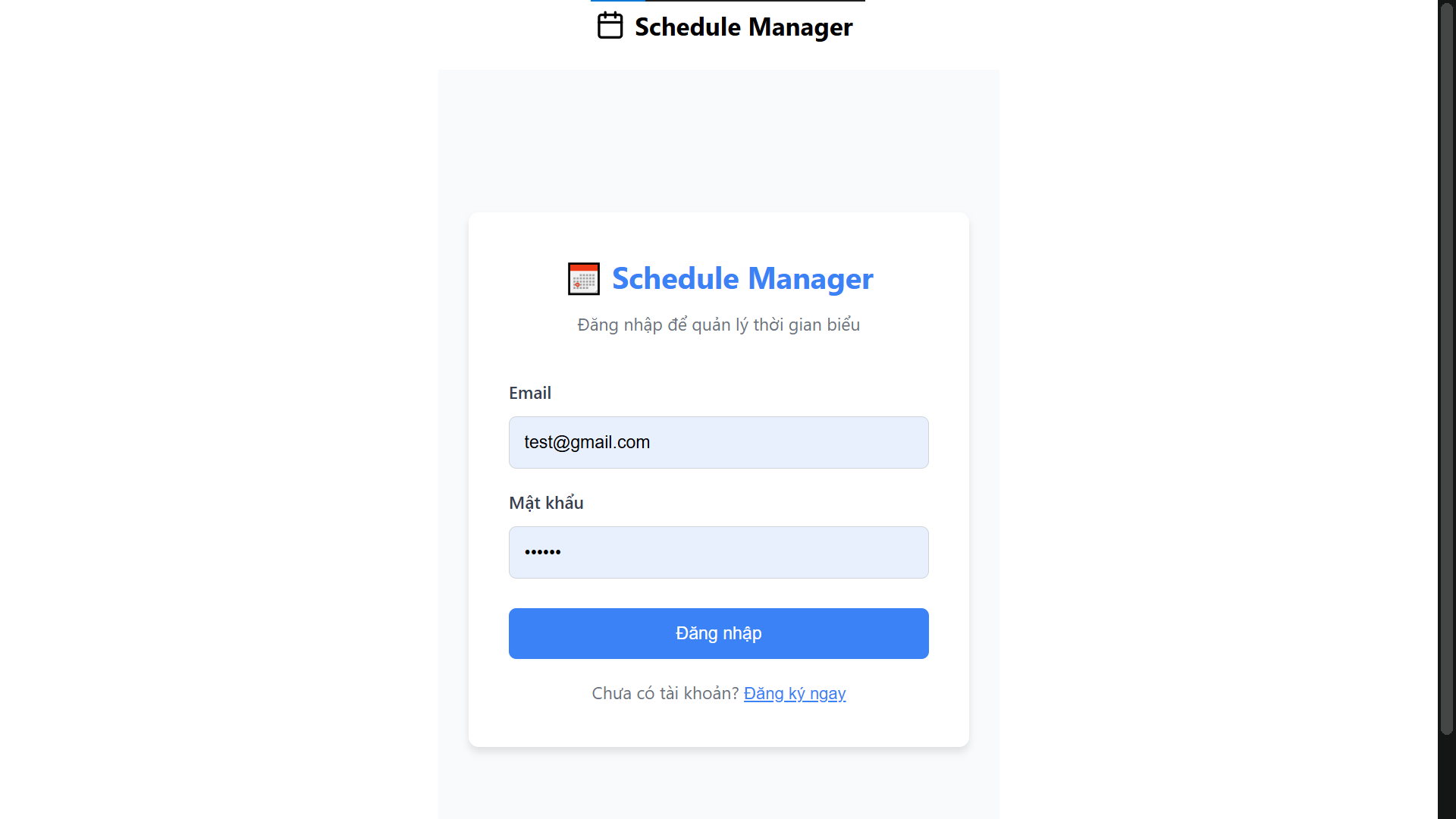
- Xác thực JWT token (backend) và localStorage (frontend)

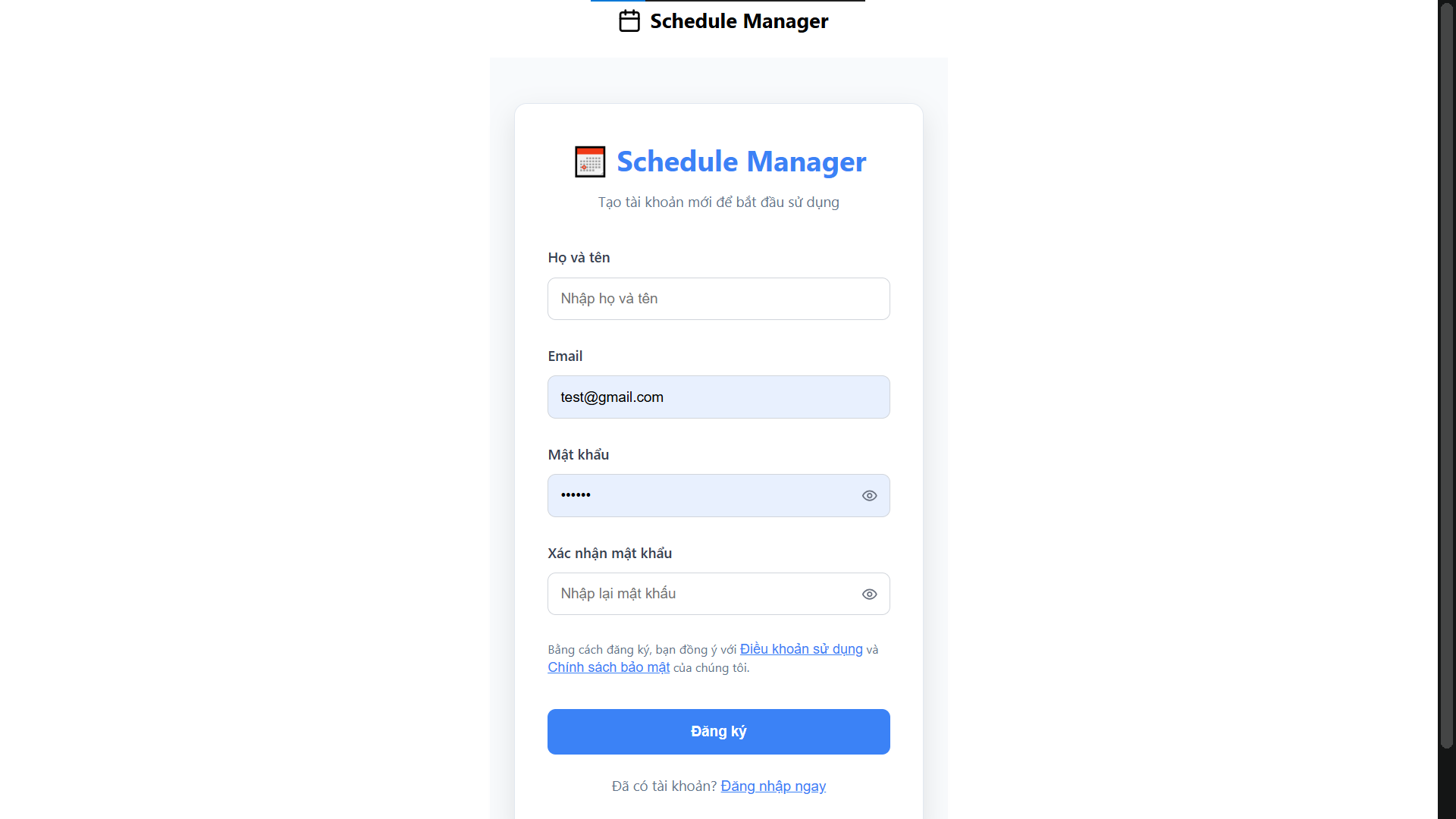
- Hỗ trợ refresh token





Hình 6.1 Giao diện quản lý người dùng trên figma





Hình 6.2 Giao diện quản lý người dùng

**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

**Quản lý sự kiện:**

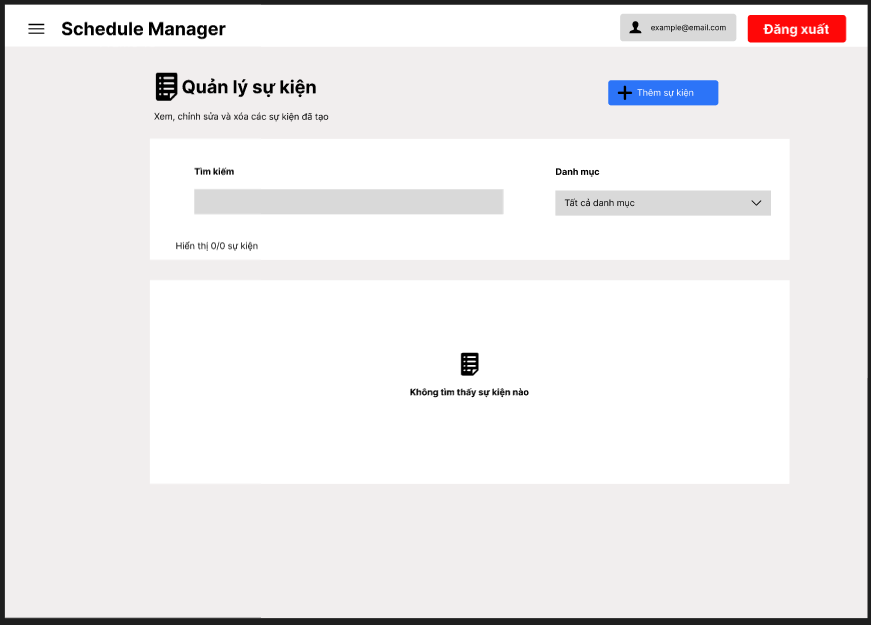
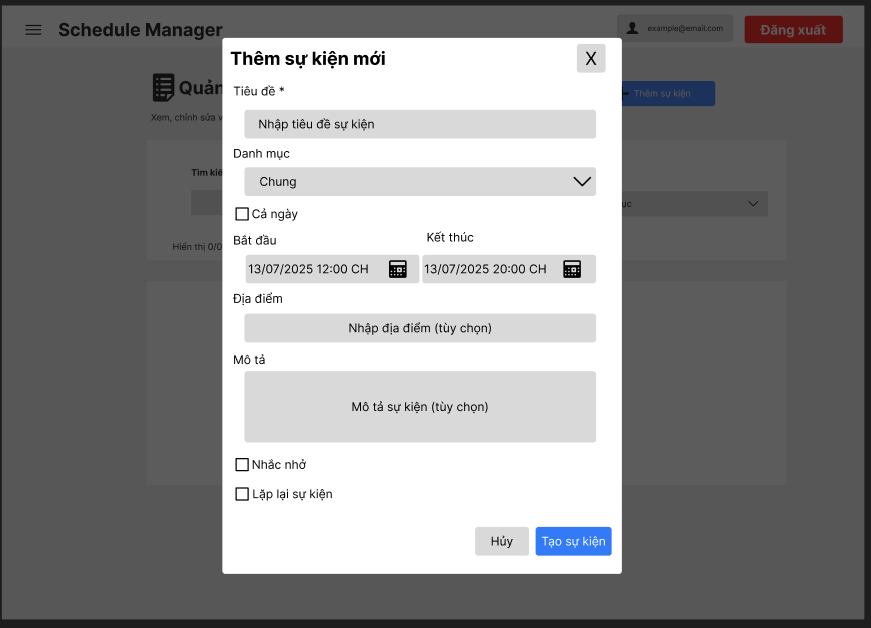
- CRUD operations đầy đủ

- Tìm kiếm và lọc sự kiện

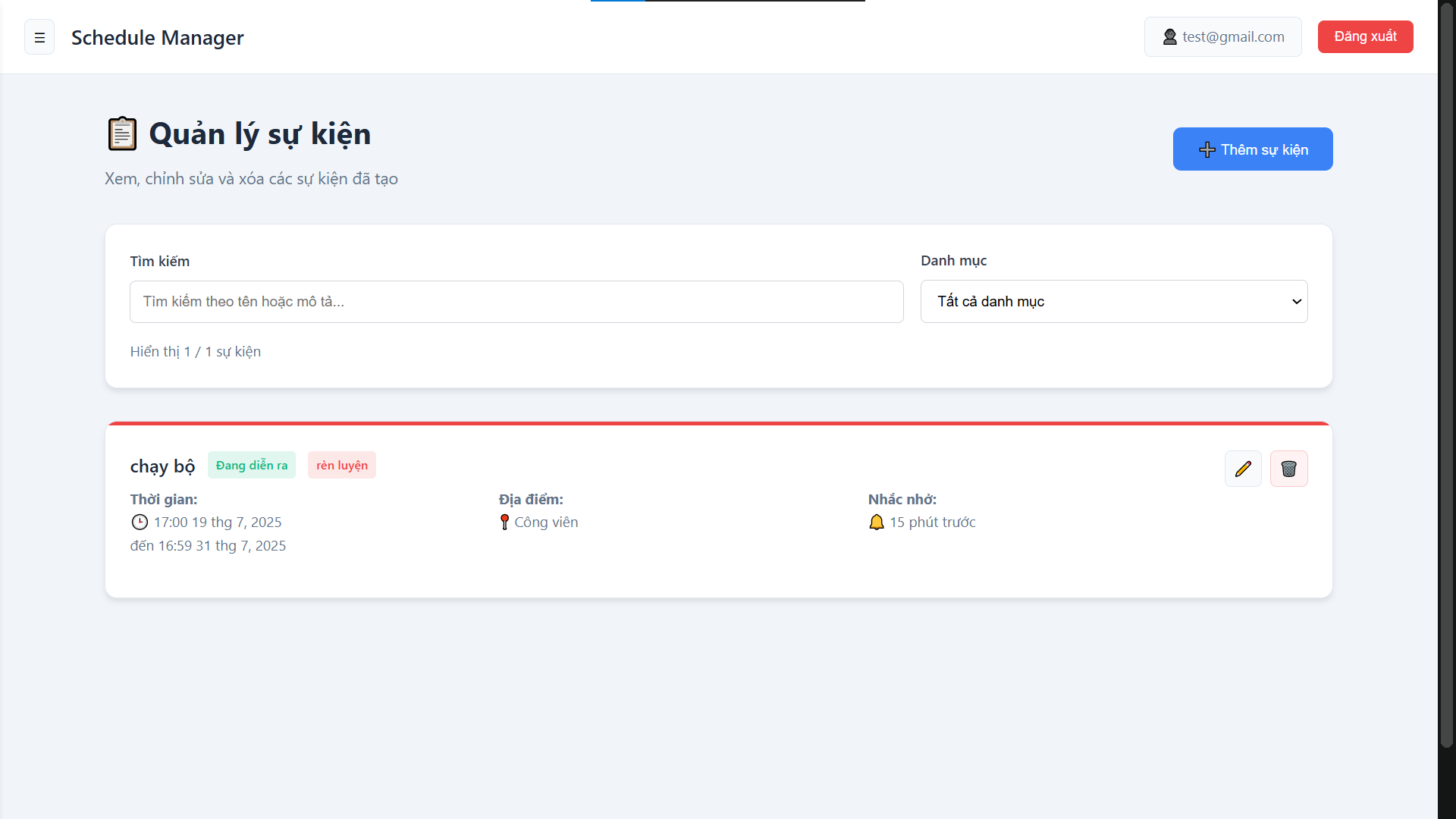
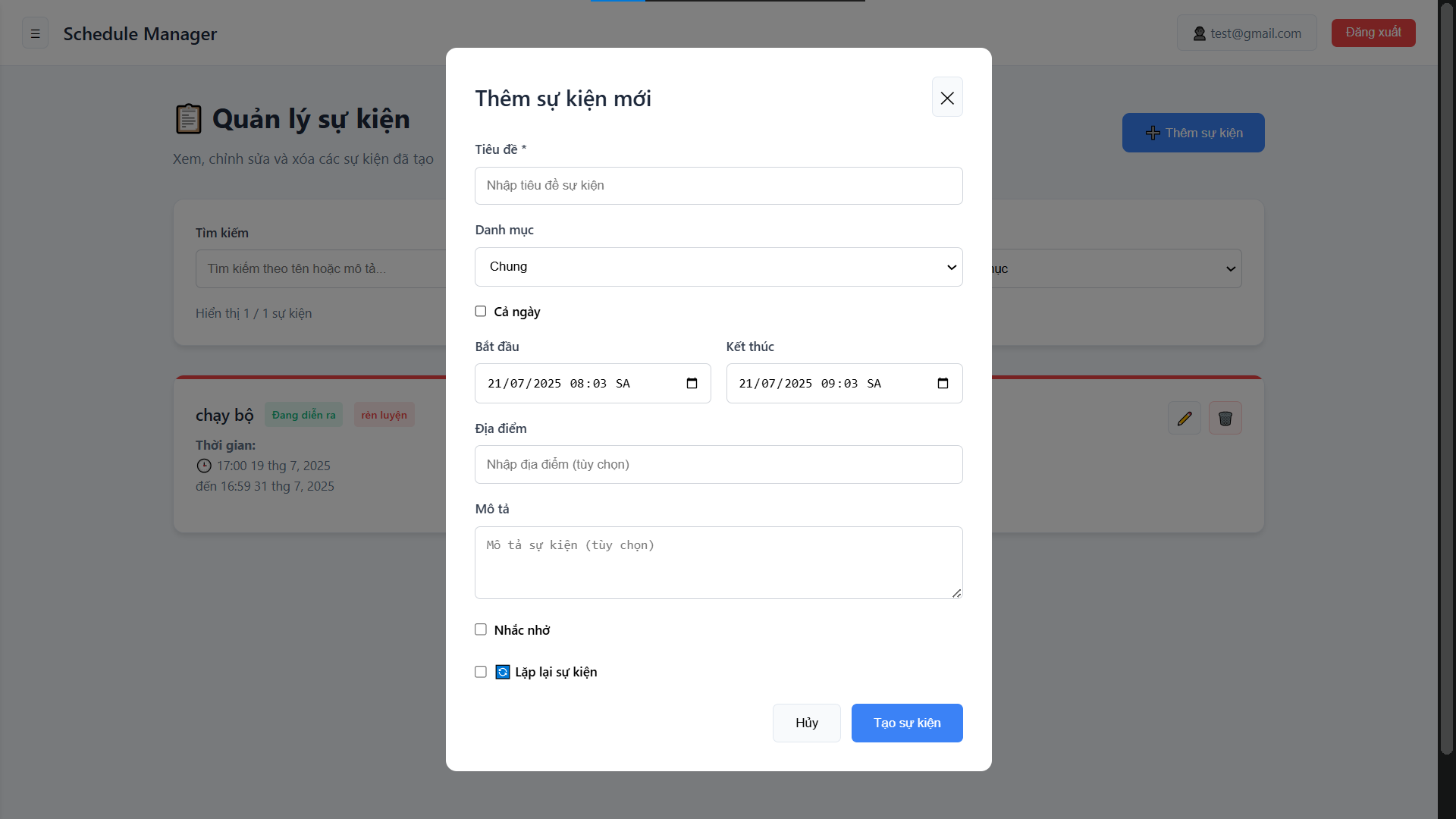
- Hỗ trợ sự kiện cả ngày

- Thiết lập địa điểm

- Thêm sự kiện trên lịch



Hình 6.4 Giao diện quản lý sự kiện trên figma

Hình 6.5 Giao diện quản lý sự kiện

**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

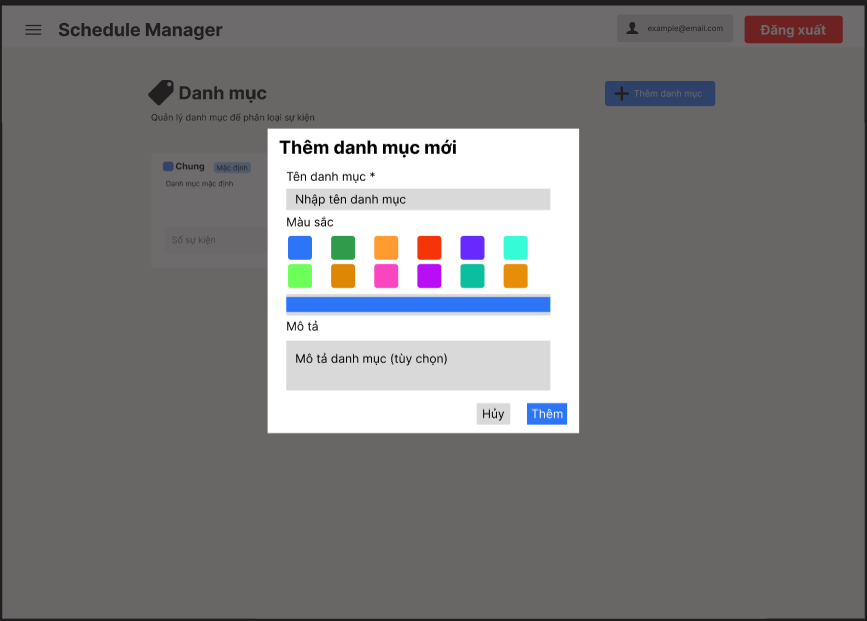
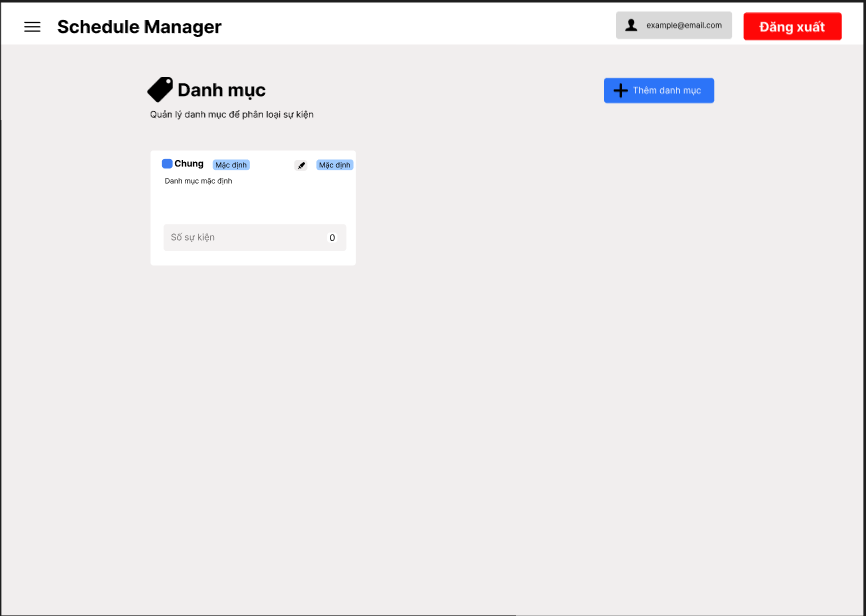
Hình 6.6 Mã nguồn

**Quản lý danh mục:**

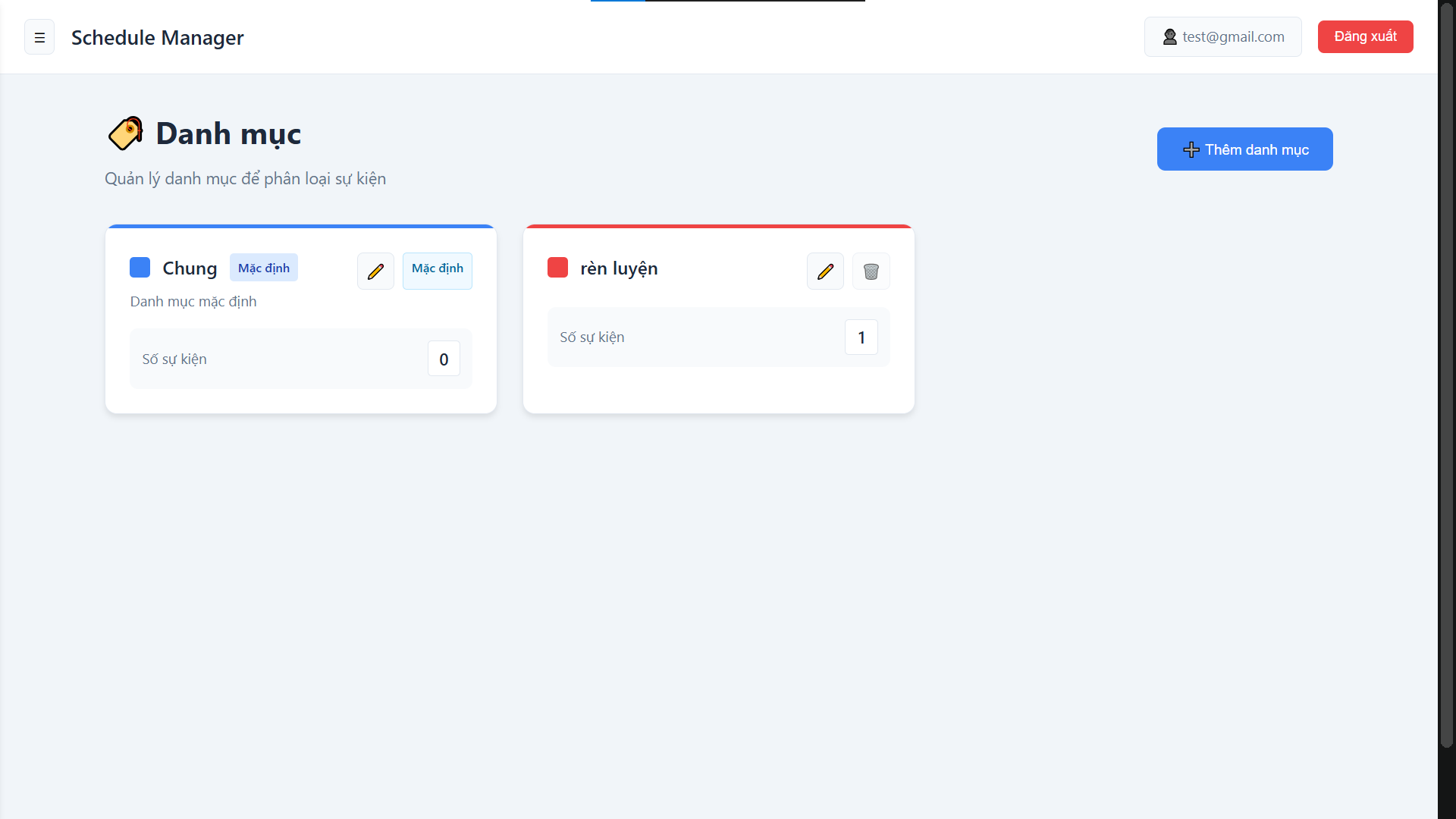
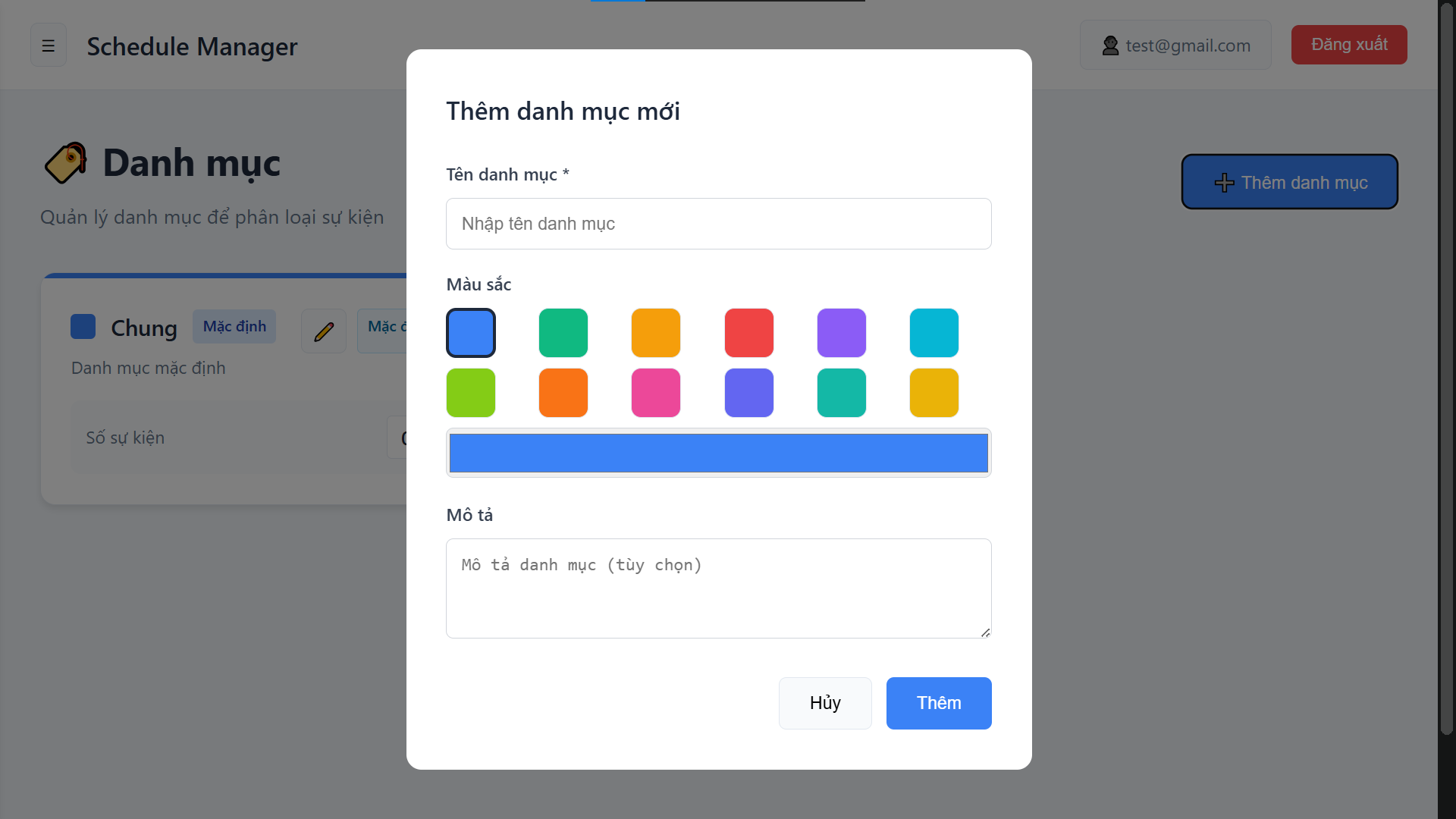
- Tạo và quản lý danh mục

- Tùy chỉnh màu sắc

- Khi tạo một danh mục thì danh mục đó sẽ được hiển thị lên khi tạo sự kiện



Hình 6.7 Giao diện quản lý danh mục figma

Hình 6.8 Giao diện quản lý danh mục

**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

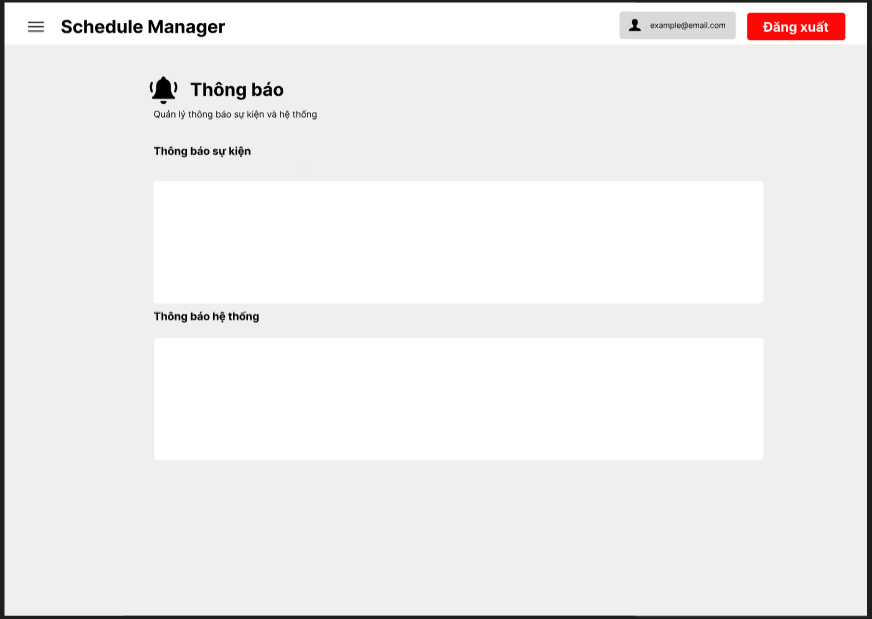
└── types/

    └── index.ts                           // TypeScript definitions cho Category

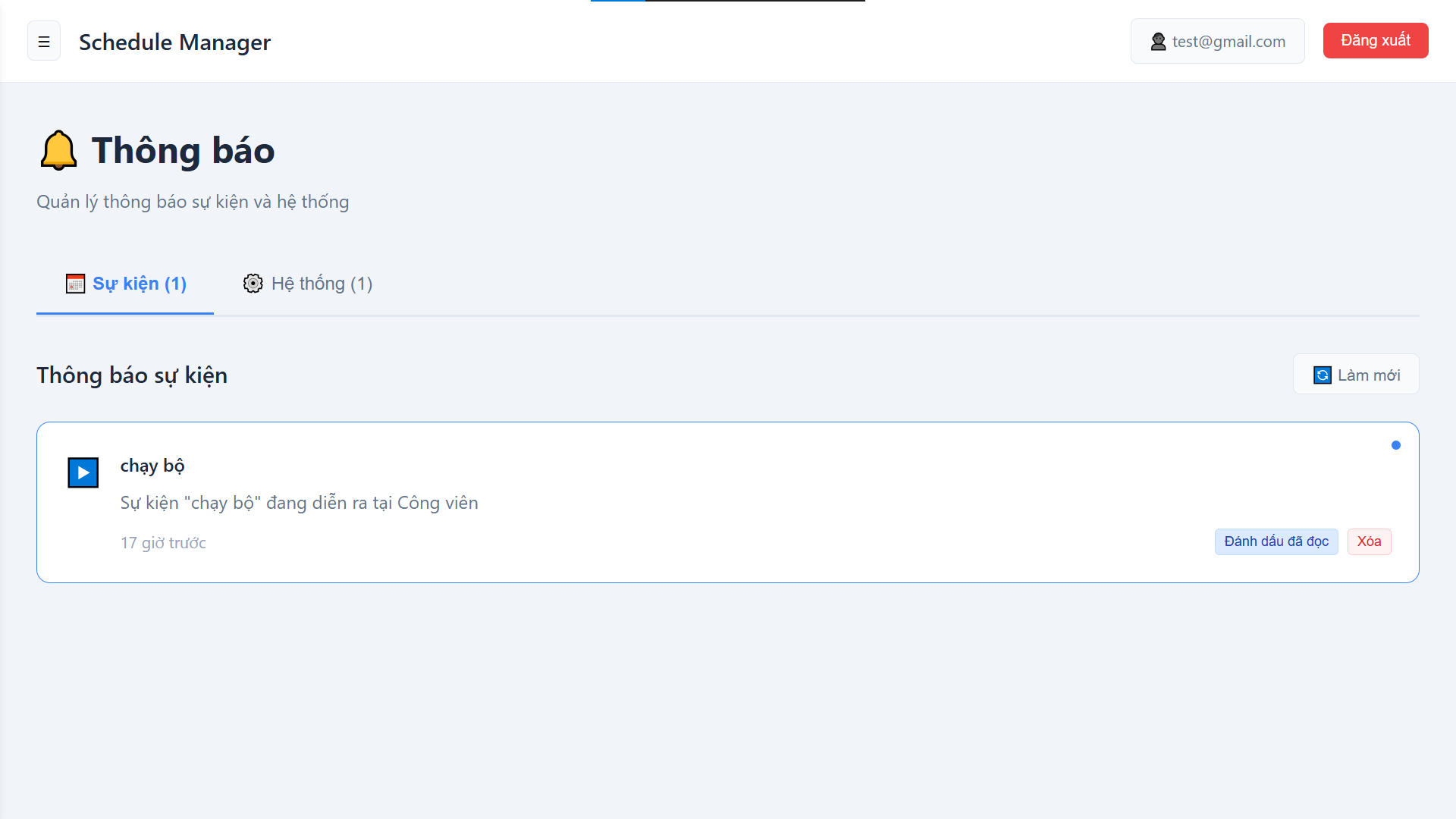
Hình 6.9 Mã nguồn

**Thông báo sự kiện:**

- Nhắc nhở sự kiện (với tùy chọn thời gian từ 0-1440 phút)



Hình 6.10 Giao diện thông báo sự kiện trên figma



Hình 6.11 Giao diện thông báo sự kiện

**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

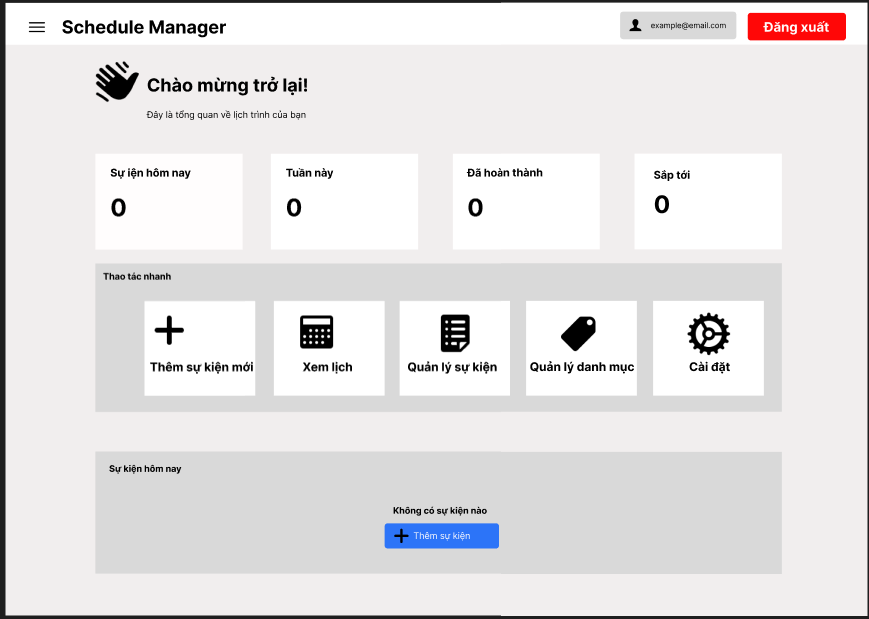
└── types/

    └── index.ts                           // TypeScript definitions

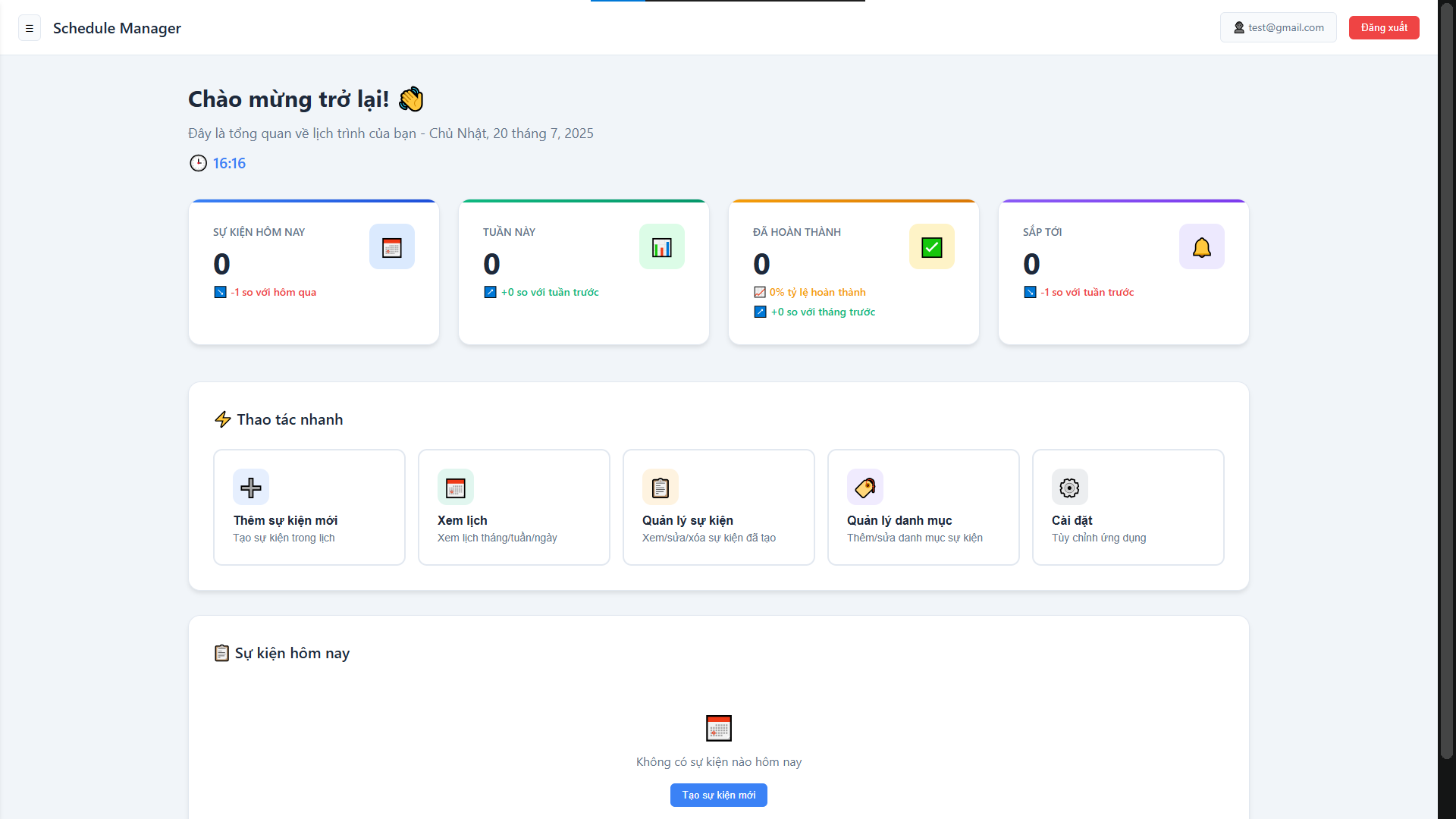
Hình 6.12 Mã nguồn

**Giao diện trang Dashboards:**

- Hiển thị thông số chi tiết các sự kiện



Hình 6.13 Giao diện trang dashboards trên figma



Hình 6.14 Giao diện trang dashboards

**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 chodata 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

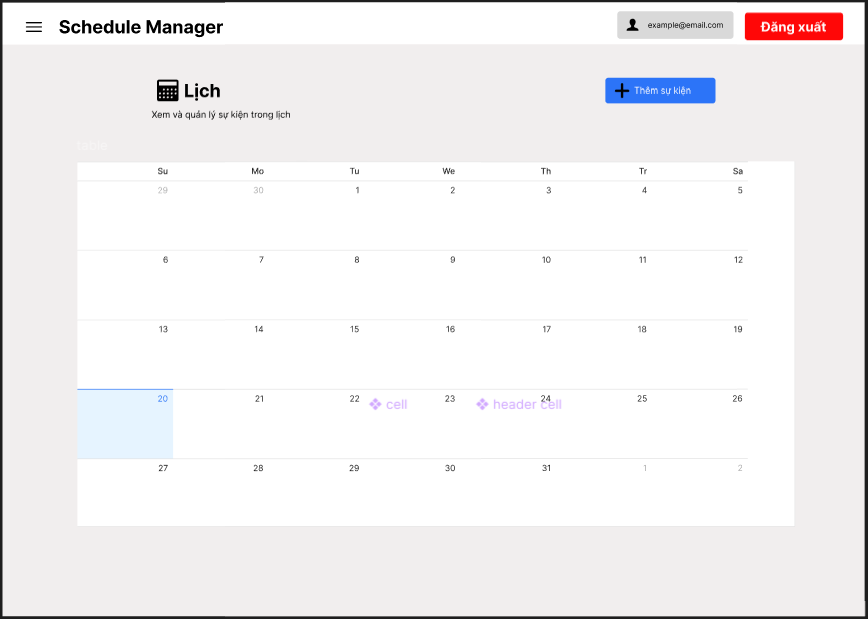
└── types/

    └── dashboard.ts                       // TypeScript definitions cho dashboard

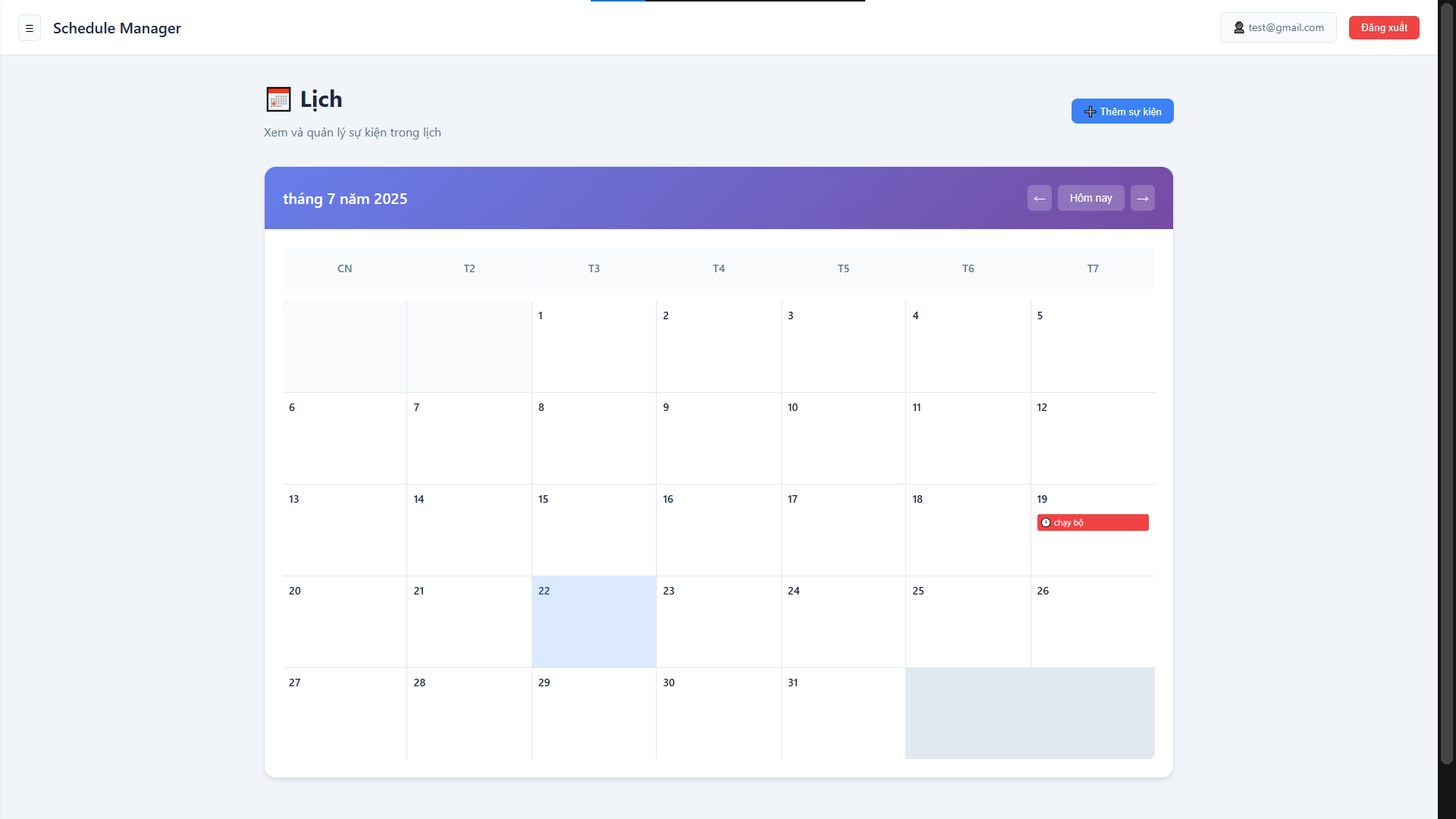
Hình 6.15 Mã nguồn

**Giao diện trang Calendar:**

- Xem sự kiện theo dạng lịch



Hình 6.16 Giao diện trang calendar trên figma



Hình 6.17 Giao diện trang calendar

**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

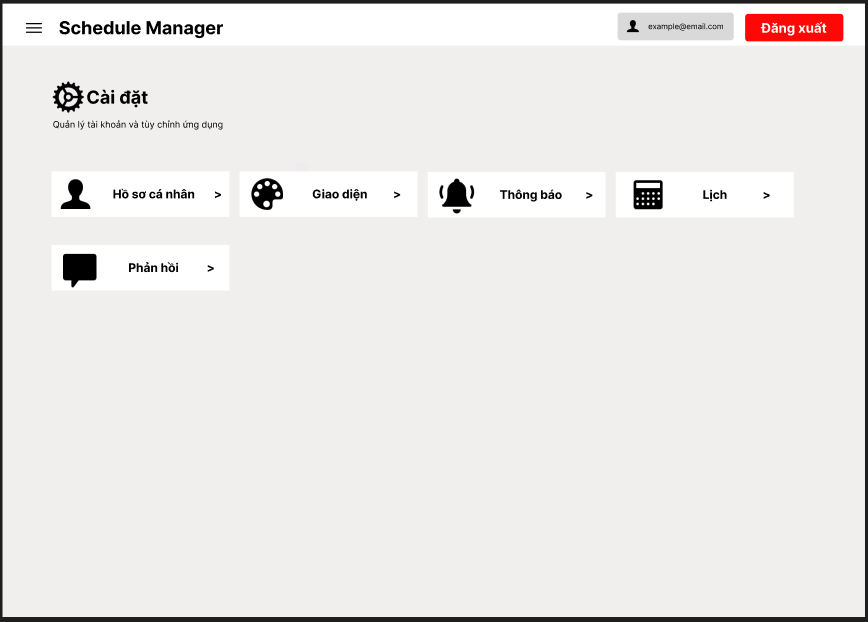
Hình 6.18 Mã nguồn

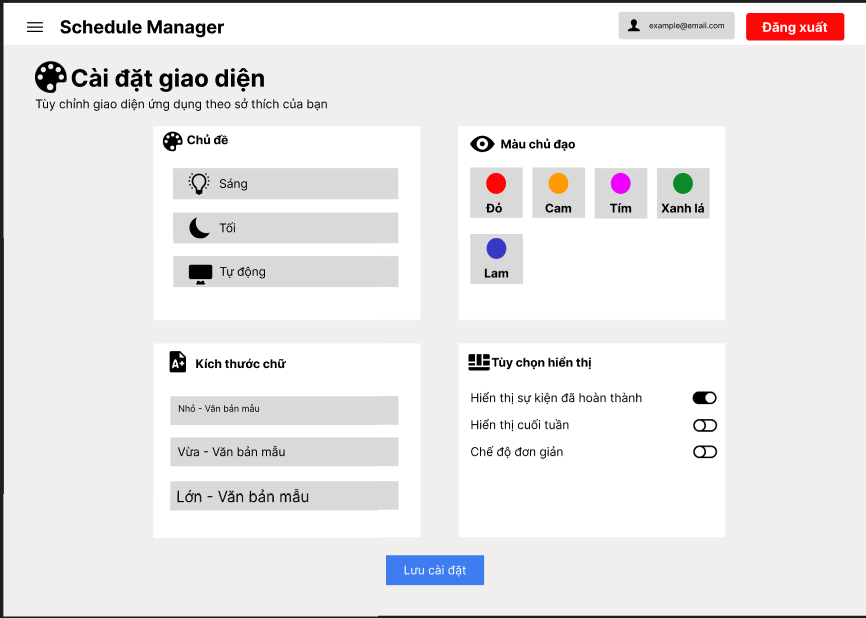
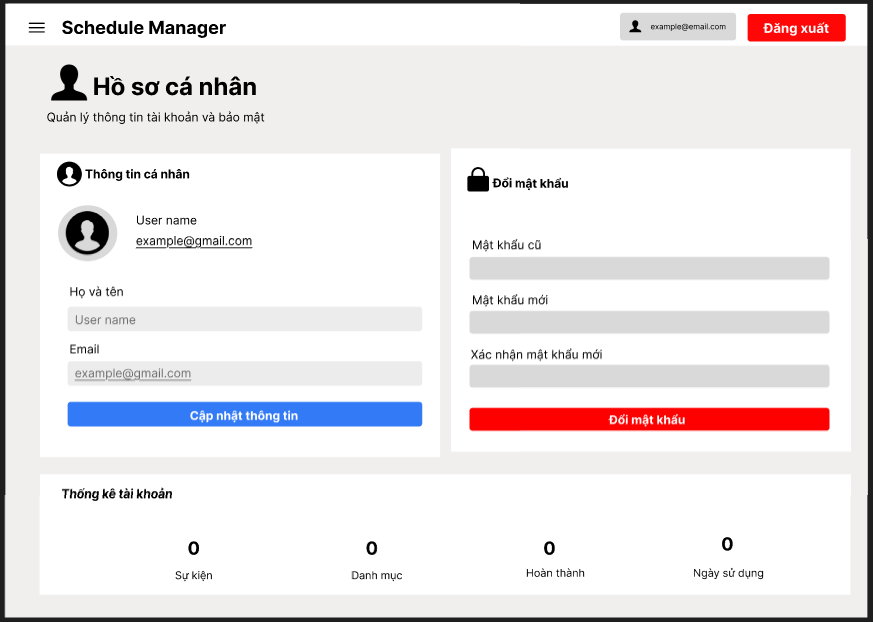
**Giao diện trang Setting:**

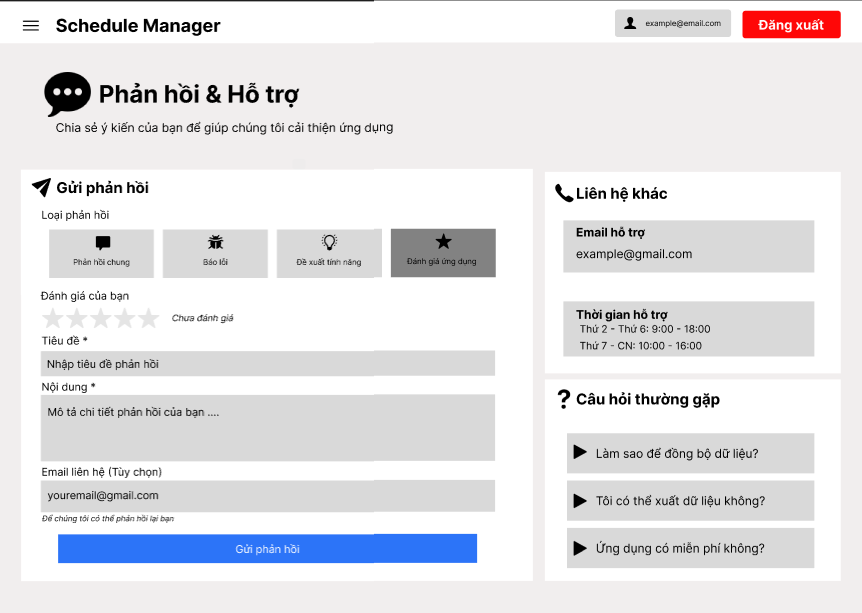
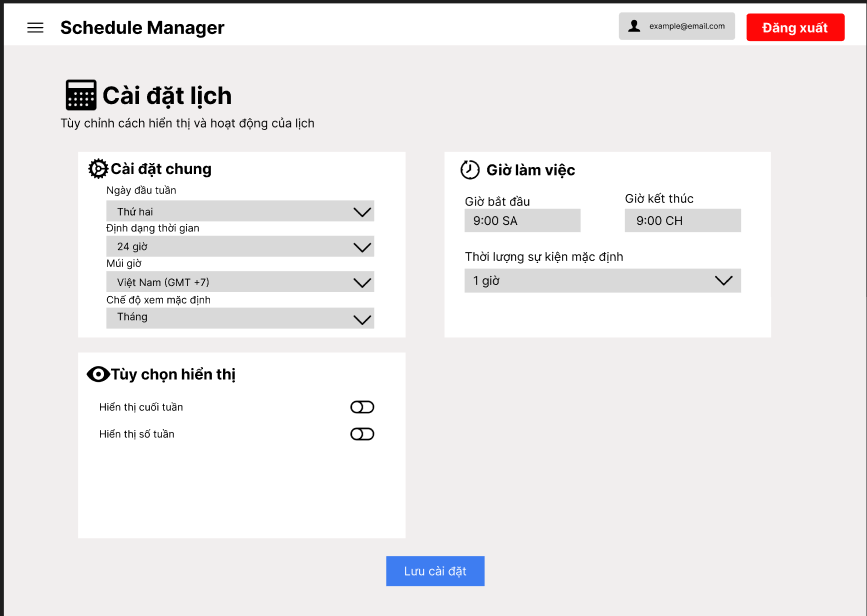
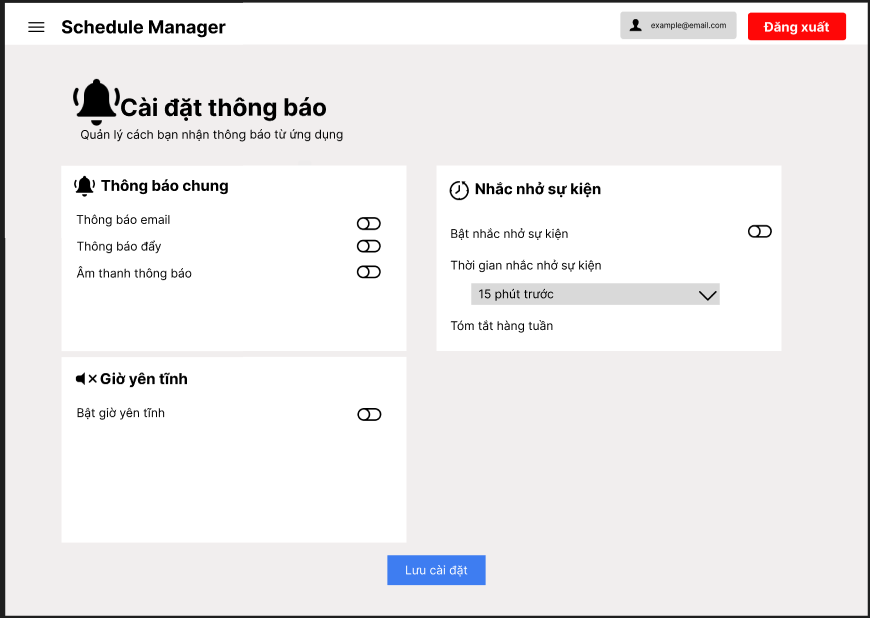
- Giao diện thiết lập trực quan

- Chức năng trang setting(đang phát triển)

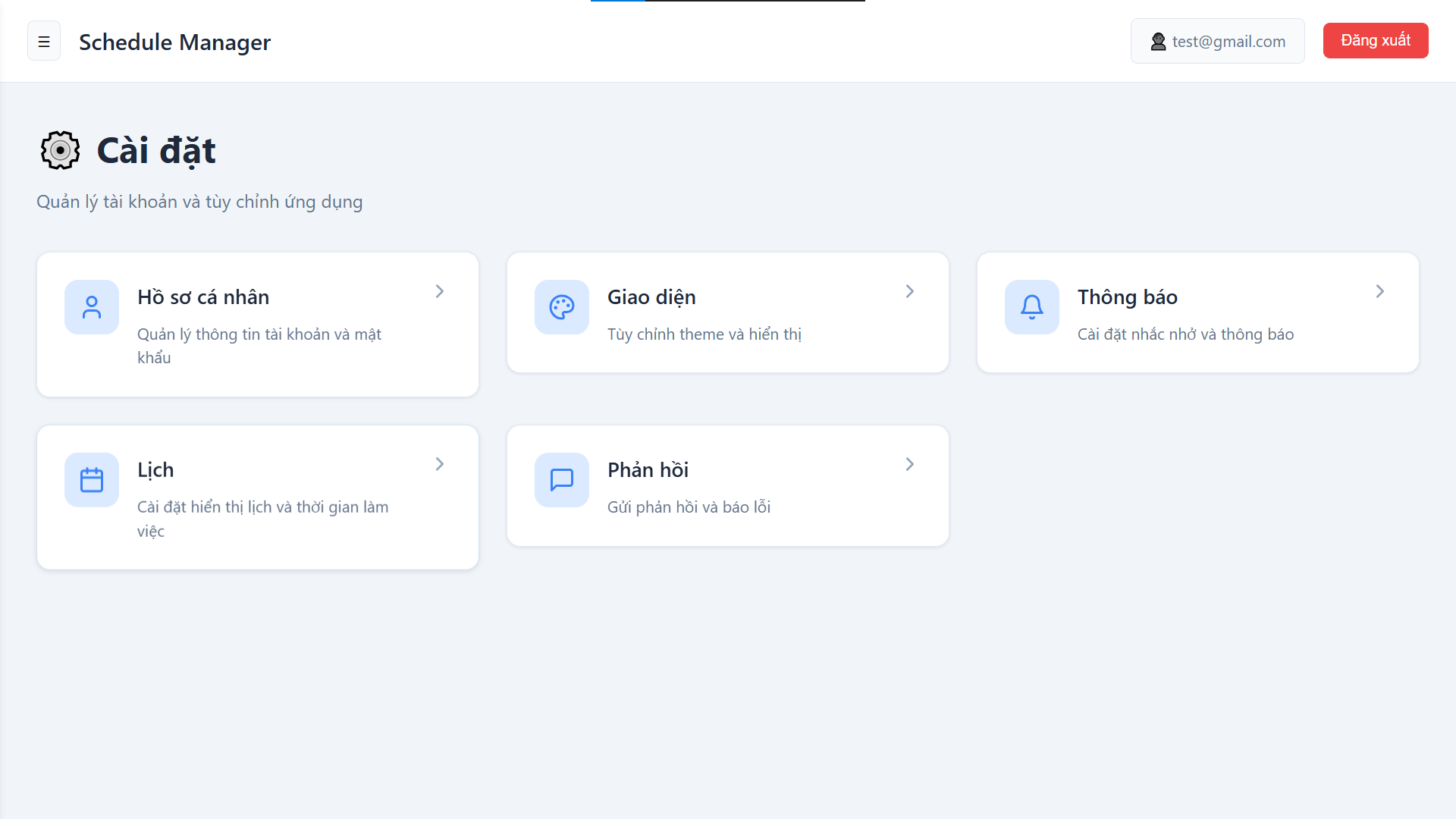
- Chức năng cho các trang con của setting(đang phát triển)

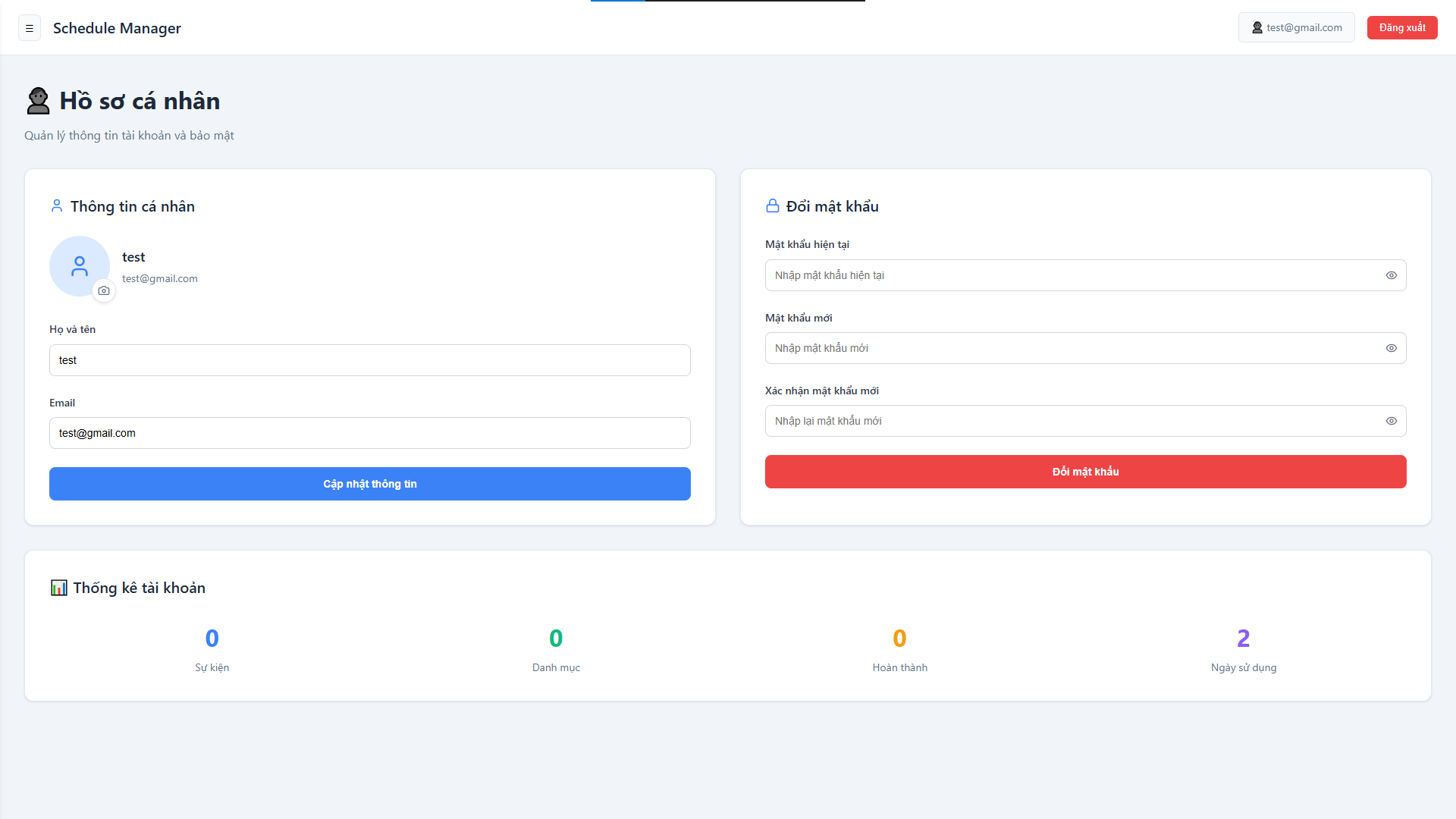


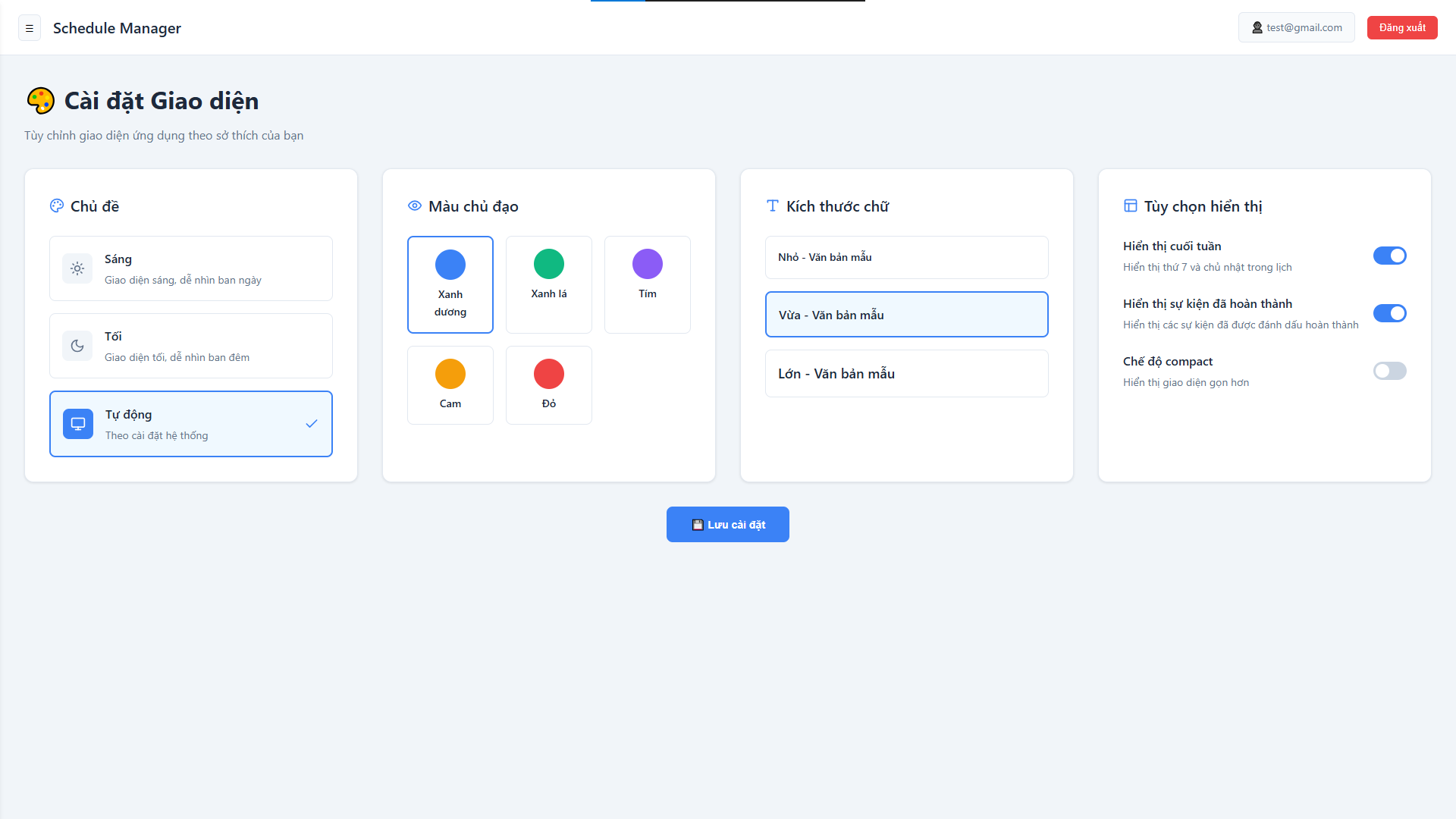


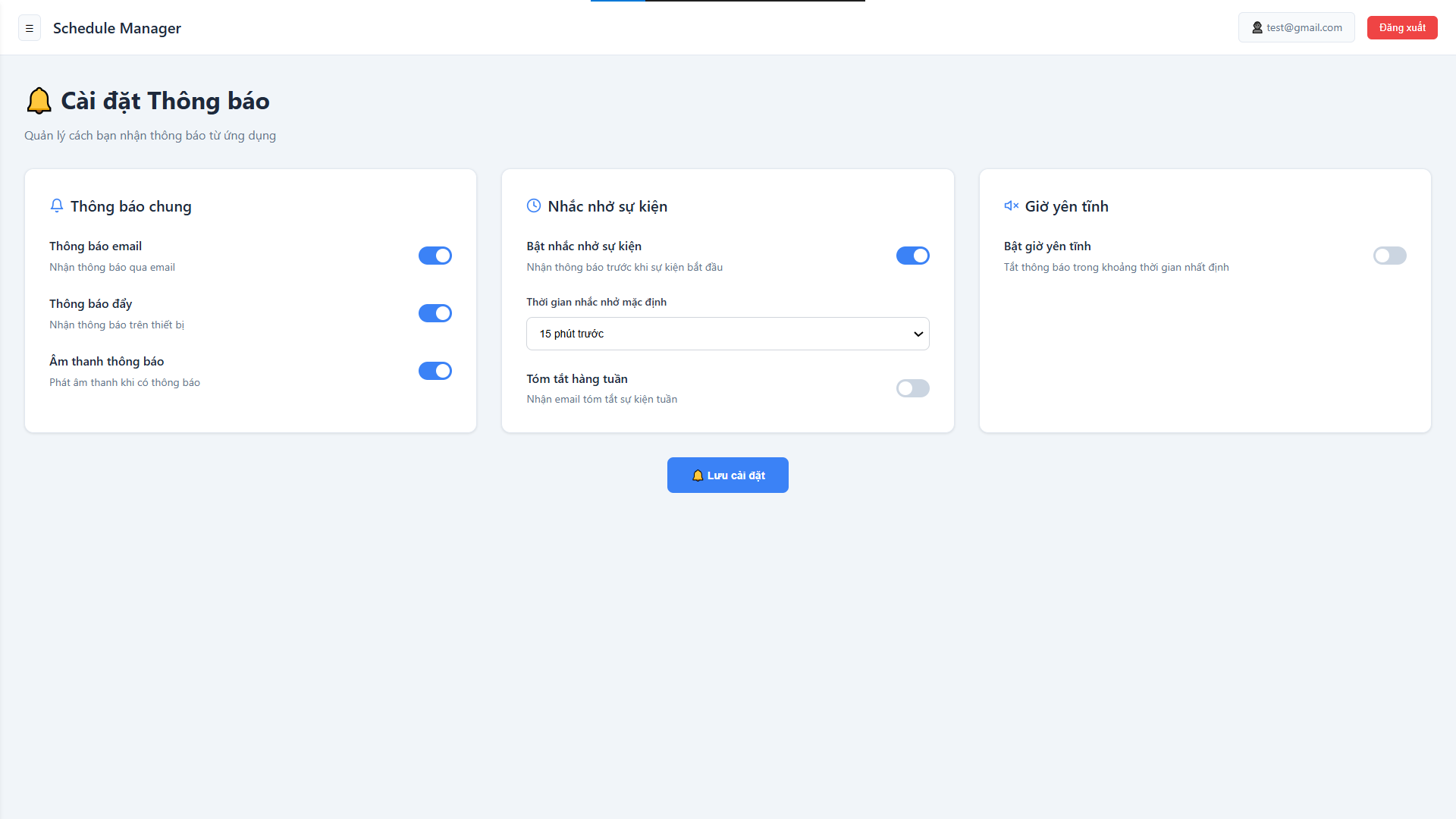


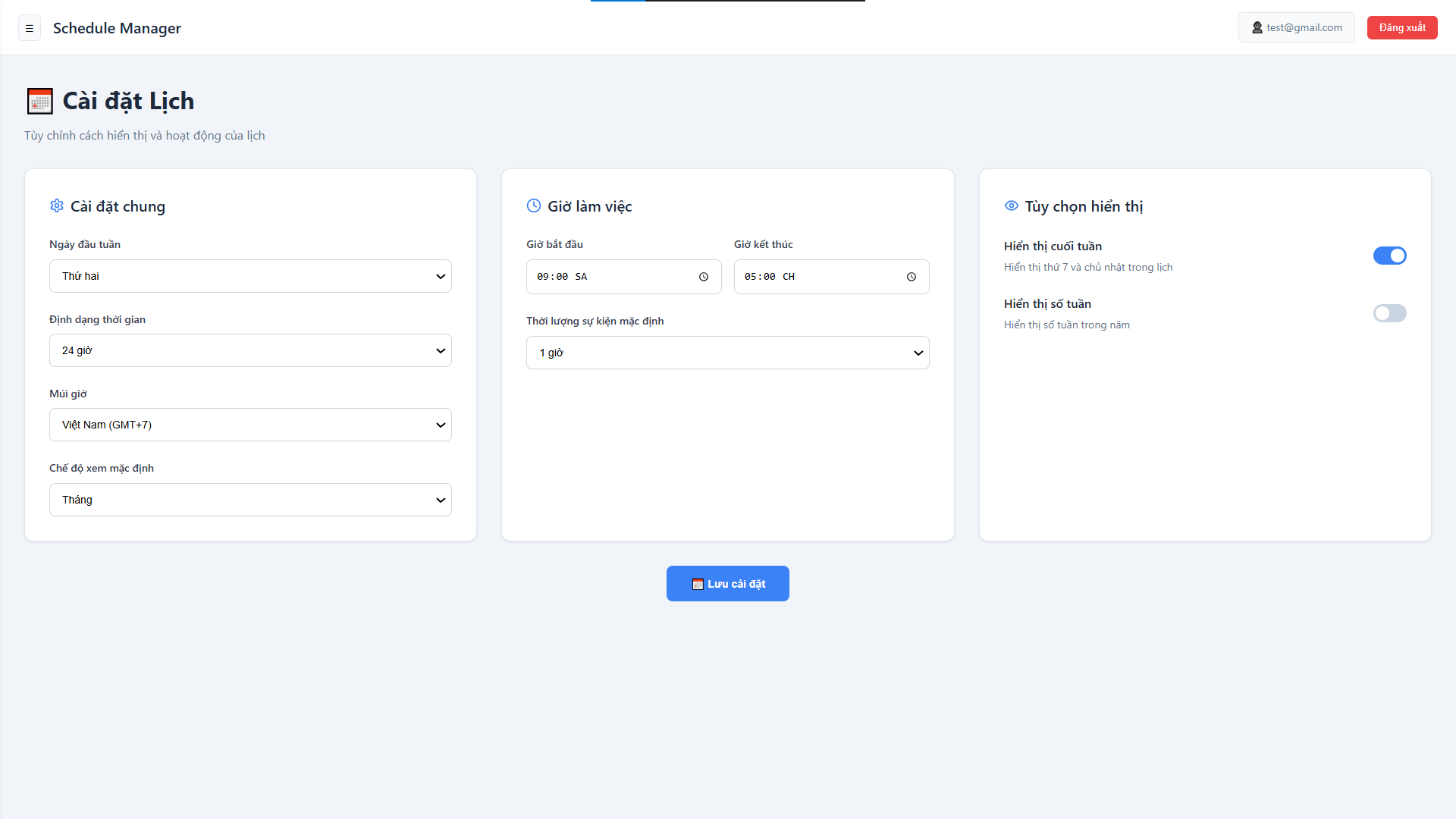
Hình 6.19 Giao diện trang setting trên figma

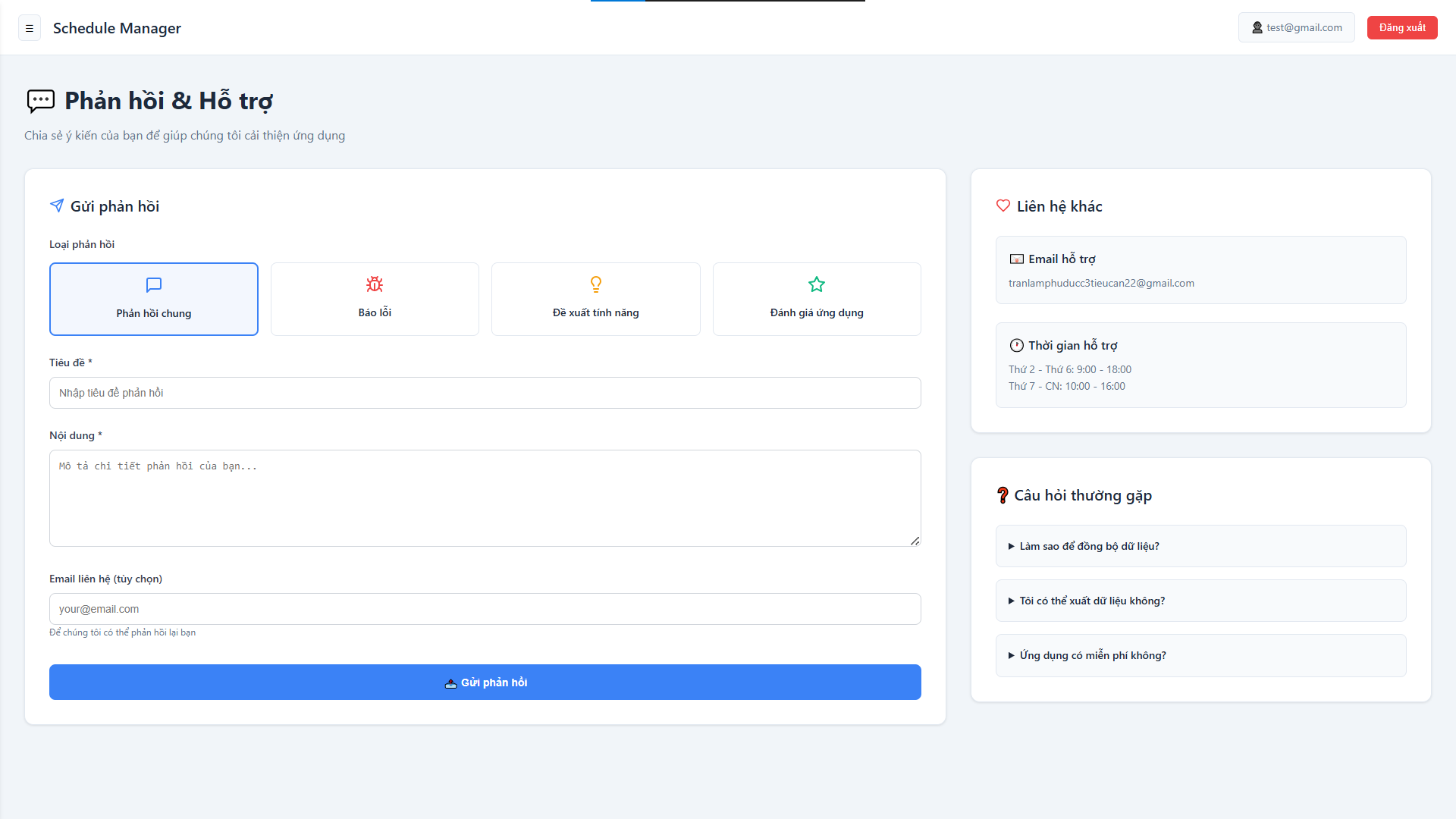












Hình 6.20 Giao diện trang Setting

**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

│   │   ├── PrivacySettings.tsx            // Privacy và security settings

│   │   ├── SupportForm.tsx                // Support ticket form

│   │   ├── ThemeSelector.tsx              // Theme selection component

│   │   ├── LanguageSelector.tsx           // Language selection

│   │   └── ExportImportData.tsx           // Data export/import tools

│   └── ui/

│       ├── Switch.tsx                     // Toggle switch component

│       ├── Select.tsx                     // Select dropdown

│       └── Slider.tsx                     // Range slider

├── store/

│   ├── settingsStore.ts                   // Zustand store cho settings

│   └── themeStore.ts                      // Store cho theme management

├── lib/

│   ├── settingsApi.ts                     // API calls cho settings

│   ├── themeUtils.ts                      // Theme utilities

│   └── exportUtils.ts                     // Data export utilities

└── types/

    └── settings.ts                        // TypeScript definitions cho settings

Hình 6.21 Mã nguồn

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/

6.2.2. Triển khai lên host

- Sử dụng Vercel để triển khai web lên host

- Sử dụng Supabase để triển khai database

- Fronend 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)