MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1 TỔNG QUAN

1.1 Lý do chọn đề tài

Trong thời đại công nghệ số phát triển mạnh mẽ như hiện nay, việc ứng dụng hệ thống quản lý thông tin đã trở thành một yêu cầu tất yếu. Đặc biệt trong môi trường giáo dục, nơi mà nhu cầu tra cứu tài liệu, quản lý sách, và theo dõi hoạt động mượn – trả tài liệu diễn ra thường xuyên, xử lý vi phạm diễn ra hằng ngày, thì việc xây dựng một hệ thống quản lý thư viện hiệu quả ngày càng trở nên cấp thiết.

Các thư viện hiện nay sở hữu một lượng lớn đầu sách, tài liệu tham khảo phục vụ công tác học tập, giảng dạy và nghiên cứu khoa học. Tuy nhiên, việc quản lý thông tin thư viện bằng phương pháp thủ công hoặc bán tự động đang tồn tại nhiều bất cập như: dữ liệu phân tán, khó kiểm soát tình trạng sách, thao tác tra cứu chậm, nguy cơ thất lạc thông tin cao. Những hạn chế này không chỉ ảnh hưởng đến hiệu quả hoạt động của thư viện mà còn gây bất tiện cho sinh viên, giảng viên khi tiếp cận nguồn tài liệu học tập từ đó làm giảm quá trình học tập và làm việc của sinh viên, giảng viên, các nhân viên của trường…

Xuất phát từ thực tiễn đó, nhóm chúng em quyết định thực hiện đề tài "Xây dựng ứng dụng web quản lý thư viện". Thông qua đề tài này, nhóm mong muốn ứng dụng những kiến thức đã học vào việc giải quyết các bài toán thực tế, xây dựng hệ thống có tính thực tiễn và khoa học, góp phần hiện đại hóa công tác quản lý thư viện, tạo tiền đề cho việc số hóa toàn bộ hoạt động thư viện trong tương lai. Hệ thống đảm bảo giải quyết triệt để những khó khăn mà các thư viện hiện nay đang gặp phải giúp cho việc quản lý thư viện trở nên thuận tiện cho người dùng của thư viện.

1.2 Mục tiêu của đề tài

Mục tiêu của đề tài là thiết kế và xây dựng thành công một hệ thống quản lý thư viện hiệu quả , đáp ứng nhu cầu của người dùng.

Cụ thể, nhóm đề ra các mục tiêu chính như sau:

- Phân tích các yêu cầu thực tế của thư viện để xác định các nhóm người dùng và các chức năng của hệ thống.
- Thiết kế giao diện cho hệ thống bằng Figma

- Lập trình giao diện bằng Reactjs, xây dựng các logic bên dưới cho thống bằng Node JS, tương tác thông qua các api.
- Xây dựng cơ sở dữ liệu bằng Mongo DB
- Quản lý dự án hiệu quả với Jira
- Quản lý và lưu trữ hiệu quả mã nguồn với GitHub
- Triển khai ứng dụng với Docker

1.3 Phương pháp nghiên cứu

- Phương pháp nghiên cứu lý thuyết: Phân tích logic nghiệp vụ của hệ thống quản lý thư viện, nghiên cứu các tài liệu về React Js , Node Js, Mongo DB,...
- Phương pháp nghiên cứu thực nghiệm: Tiến hành thiết kế giao diện trên Figma, lập trình frontend bằng React Js , backend bằng Node Js , Mongo DB cho hệ thống,...

1.4 Đối tượng nghiên cứu

Hệ thống quản lý thư viện được xây dựng trên nền tảng web với các chức năng như: quản lý thêm xóa sửa, tìm kiếm người dùng, sách, phiếu mượn,vi phạm, đăng nhập, đăng ký tài khoản. Phân quyền hệ thống với các nhóm người dùng khác nhau.

1.5 Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài là xây dựng hệ thống quản lý thư viện cho Trường đại học Trà Vinh với các chức năng cơ bản mà một hệ thống thư viện cần thiết. Đề tài chỉ nằm ở mức học thuật , chưa triển khai thực tế.

1.6 Bảng phân công

Thành viên	Công việc	Thời gian
Đào Công Hoàng Lam	Phân chia công việc trên Jira, khởi tạo dự án, triển khai ứng dụng trên Docker , xây dựng trang giao diện người dùng, xây dựng các chức năng quản lý	1/7 - 25/72025

Trần Dinh Hiển	Thiết kế UI,UX trên figma, phát triển chức năng mượn trả sách, duyệt sách, xem lịch sử mượn,...	1/7 - 20/72025
Nguyễn Trường Vũ	Vẽ sơ đồ Use Case, viết báo cáo, phát triển chức năng đăng nhập đăng ký, tra cứu sách ,điều hướng trang, xem chi tiết sách,..	1/7 - 20/72025

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

2.1 Công nghệ phần mềm Agile

Alige là một phương thức phát triển phần mềm linh hoạt, đưa sản phẩm đến tay người dùng càng nhanh càng tốt. Agile gần như là một phương pháp luận, một triết lý dựa trên hơn nguyên tắc phân đoạn vòng lặp (iterative) và tăng trưởng (incremental). Agile trở thành một phương thức quản lý dự án phổ biến nhất hiện nay với nhiều đại diện được gọi là các phương pháp "họ Agile".

Bốn giá trị cốt lõi của tuyên ngôn Alige:

- Cá nhân và tương tác hơn là quy trình và công cụ
- Phần mềm hoạt động tốt hơn là tài liệu đầy đủ
- Hợp tác với khách hàng hơn là đàm phán hợp đồng
- Ứng phó, phản hồi với các thay đổi hơn là làm theo kế hoạch

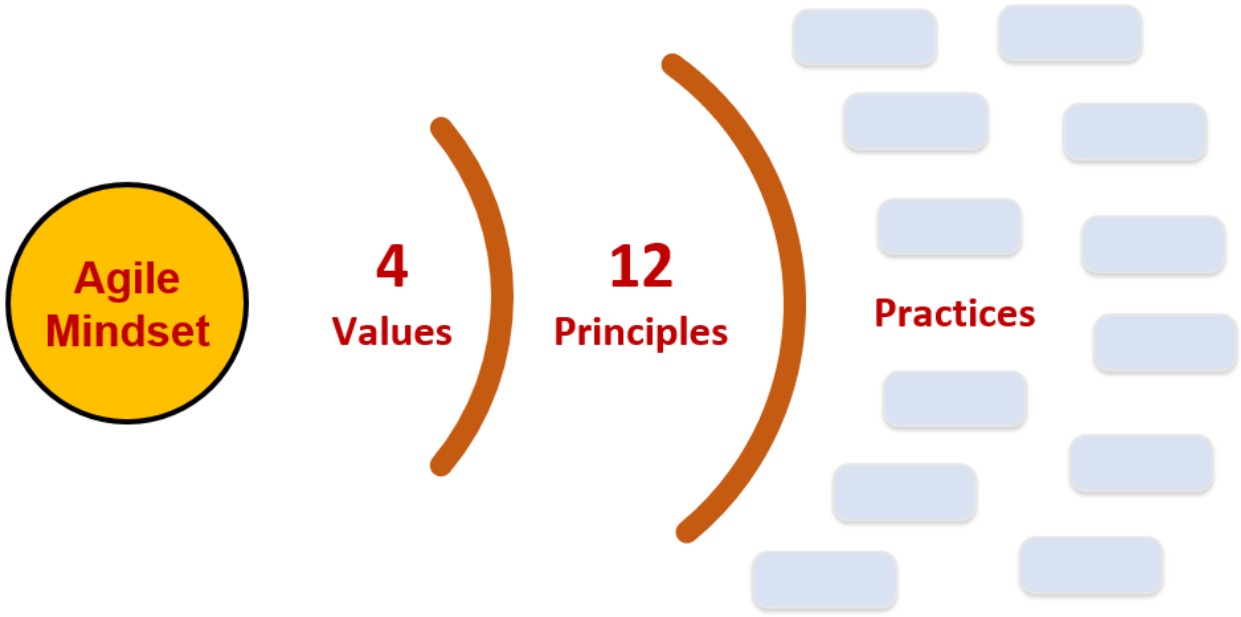
12 nguyên tắc của tuyên ngôn Alige:

- Ưu tiên sự hài lòng của khách hàng thông qua việc trao đổi và bàn giao liên tục, khách hàng sẽ cảm thấy hài lòng khi họ nhận được sản phẩm làm việc đều đặn thay vì chờ đợi thời gian kéo dài giữa các lần releases
- Chào đón việc thay đổi trong suốt quá trình phát triển. Tránh sự chậm trễ khi có yêu cầu thay đổi requirements.
- Chuyển giao sản phẩm thường xuyên. Chuyển giao phần mềm sử dụng được định kỳ, từ một vài tuần đến một vài tháng, với một thời gian ngắn hơn.
- Khách hàng và đội phát triển phải làm việc cùng nhau hàng ngày trong suốt dự án.
- Xây dựng các dự án xoay quanh các cá nhân có động lực. Tạo cho họ một môi trường và hỗ trợ họ những thứ cần thiết và tin tưởng họ để công việc được hoàn thành.
- Phương pháp hiệu quả nhất là truyền tải thông tin đến vào bên trong đội phát triển là hội thoại mặt-đối-mặt.

- Phần mềm làm việc được là thước đo của quá trình.
- Các quy trình Agile thúc đẩy sự phát triển bền vững. Các nhà tài trợ cho dự án, các nhà phát triển và người dùng cuối có thể duy trì một tốc độ vô hạn định.
- Liên tục quan tâm đến kỹ thuật xuất sắc và thiết kế tốt giúp nâng cao tính linh hoạt.
- Tính đơn giản – nghệ thuật tối đa hoá khối lượng công việc chưa hoàn thành – là điều thiết yếu.
- Các kiến trúc, yêu cầu và thiết kế tốt nhất xuất hiện từ các nhóm tự tổ chức.
- Ở thời điểm kết thúc sprint, các team nên xem lại làm thế nào để hiệu quả hơn, sau đó cùng tự cải thiện ứng xử, cải tiến quy trình, kỹ thuật của mình sao cho phù hợp

Các nguyên tắc phát triển Agile:

- Tương tác với khách hàng: Vai trò của khách hàng là cung cấp và ưu tiên các yêu cầu hệ thống mới và đánh giá từng bước gia tăng của hệ thống. Mời gọi khách hàng tham gia chặt chẽ với nhóm phát triển phần mềm
- Chấp nhận thay đổi: Yêu cầu về tính năng của sản phẩm và chi tiết của các tính năng này sẽ thay đổi khi nhóm phát triển và người product manager tìm hiểu thêm về nó. Điều chỉnh phần mềm để đối phó với những thay đổi.
- Phát triển và chuyển giao tăng trưởng: Luôn phát triển các sản phẩm phần mềm theo từng bước. Kiểm tra và đánh giá từng phần gia tăng khi nó được phát triển và phản hồi những thay đổi cần thiết cho nhóm phát triển.



Hình 2.1 Các giá trị, nguyên tắc và thông lệ chung của tuyên ngôn Agile

2.2 Mô hình 3 lớp (three-layer)

Mô hình 3 lớp là một kiến trúc kiểu client/server mà trong đó giao diện người dùng (UI-user interface), các quy tắc xử lý (BR-business rule hay BL-business logic), và việc lưu trữ dữ liệu được phát triển như những module độc lập, và hầu hết là được duy trì trên các nền tảng độc lập, và mô hình 3 tầng được coi là một kiến trúc phần mềm và là một mẫu thiết kế.

Mô hình 3-layer gồm có 3 phần chính:

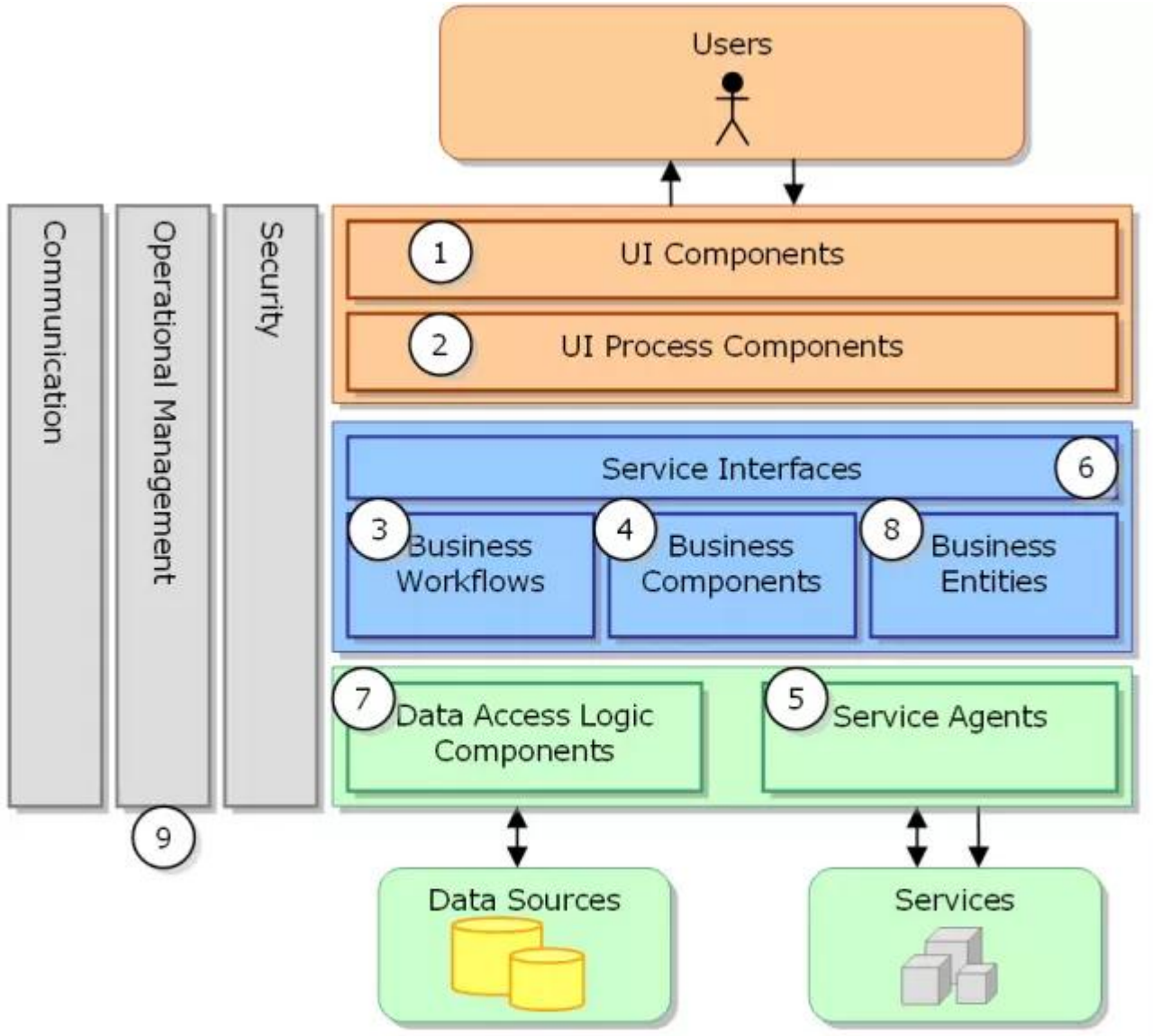
- Presentation Layer (GUI) : Lớp này có nhiệm vụ chính giao tiếp với người dùng. Nó gồm các thành phần giao diện (winform, webform,…) và thực hiện các công việc như nhập liệu, hiển thị dữ liệu, kiểm tra tính đúng đắn dữ liệu trước khi gọi lớp Business Logic Layer (BLL).
- Business Logic Layer (BLL) : Layer này phân ra 2 thành nhiệm vụ : Đây là nơi đáp ứng các yêu cầu thao tác dữ liệu của GUI layer, xử lý chính nguồn dữ liệu từ Presentation Layer trước khi truyền xuống Data Access Layer và lưu xuống hệ quản trị CSDL. Đây còn là nơi kiểm tra các ràng buộc, tính toàn vẹn và hợp lệ

dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về Presentation Layer.

- Data Access Layer (DAL) : Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu (tìm kiếm, thêm, xóa, sửa,…).

Ưu điểm của mô hình 3-layer:

- Việc phân chia thành từng lớp giúp cho code được tường minh hơn. Nhờ vào việc chia ra từng lớp đảm nhận các chức năng khác nhau và riêng biệt như giao diện, xử lý, truy vấn thay vì để tất cả lại một chỗ. Nhằm giảm sự kết dính.
- Dễ bảo trì khi được phân chia, thì một thành phần của hệ thống sẽ dễ thay đổi. Việc thay đổi này có thể được cô lập trong 1 lớp, hoặc ảnh hưởng đến lớp gần nhất mà không ảnh hưởng đến cả chương trình.
- Dễ phát triển, tái sử dụng: khi chúng ta muốn thêm một chức năng nào đó thì việc lập trình theo một mô hình sẽ dễ dàng hơn vì chúng ta đã có chuẩn để tuân theo. Và việc sử dụng lại khi có sự thay đổi giữa hai môi trường (Winform sang Webform) thì chỉ việc thay đổi lại lớp GUI.
- Dễ bàn giao, nếu mọi người đều theo một quy chuẩn đã được định sẵn, thì công việc bàn giao, tương tác với nhau sẽ dễ dàng hơn và tiết kiệm được nhiều thời gian.
- Dễ phân phối khối lượng công việc. Mỗi một nhóm, một bộ phận sẽ nhận một nhiệm vụ trong mô hình 3 lớp. Việc phân chia rõ ràng như thế sẽ giúp các lập trình viên kiểm soát được khối lượng công việc của mình.



Hình 2.2 Mô hình kiến trúc 3 lớp

2.3 ReactJS

ReactJS là một thư viện JavaScript mạnh mẽ giúp xây dựng giao diện người dùng (UI) động và phức tạp một cách hiệu quả và linh hoạt, cho phép tạo ra các ứng dụng web tương tác và trải nghiệm người dùng mượt mà hơn. Với khả năng tái sử dụng component và cơ chế Virtual DOM, ReactJS hỗ trợ phát triển các ứng dụng web nhanh chóng và hiệu suất cao, giúp cung cấp trải nghiệm tốt hơn cho người dùng.

React nổi bật với kiến trúc linh hoạt, đơn giản nhờ component và cộng đồng hỗ trợ mạnh mẽ, giúp người đọc hiểu rõ những ưu điểm chính của React trong phát triển ứng dụng.

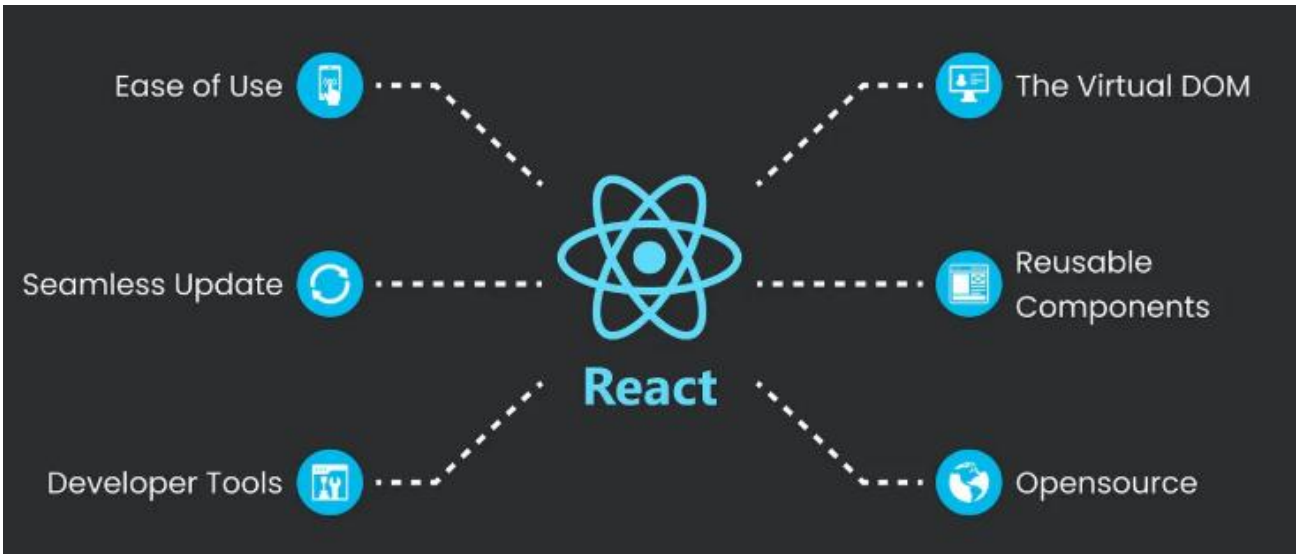
ReactJS được xây dựng dựa trên các thành phần cốt lõi như JSX, Virtual DOM, kiến trúc Component, Props & State, Lifecycle và liên kết dữ liệu một chiều, giúp người đọc nắm bắt nền tảng cơ bản để xây dựng ứng dụng React hiệu quả.

ReactJS có nhiều ưu điểm như tái sử dụng component, hỗ trợ ứng dụng di động, hiệu suất cao và SEO tốt, nhưng cũng có nhược điểm như chỉ là View Library, kích thước lớn và độ phức tạp ban đầu, giúp người đọc cân nhắc lựa chọn ReactJS phù hợp với dự án của mình.

ReactJS sử dụng JSX để tích hợp HTML vào JavaScript và DOM ảo để tối ưu hóa việc cập nhật giao diện.

ReactJS cho phép tự do thiết lập quy ước mã và cấu trúc thư mục, đồng thời có thể tích hợp linh hoạt vào các ứng dụng hiện có, giúp người đọc hiểu được tính linh hoạt của ReactJS trong việc xây dựng và tùy chỉnh ứng dụng.

Code React JS được viết bằng JavaScript theo cấu trúc component-based, giúp người đọc hình dung rõ hơn về cách tổ chức và tái sử dụng code trong React JS.



Hình 2.3 Cách hoạt động của ReactJS

2.4 Node Js

Node.js ra đời khi các developer đời đầu của JavaScript mở rộng nó từ một thứ bạn chỉ chạy được trên trình duyệt thành một thứ bạn có thể chạy trên máy của mình dưới dạng ứng dụng độc lập.

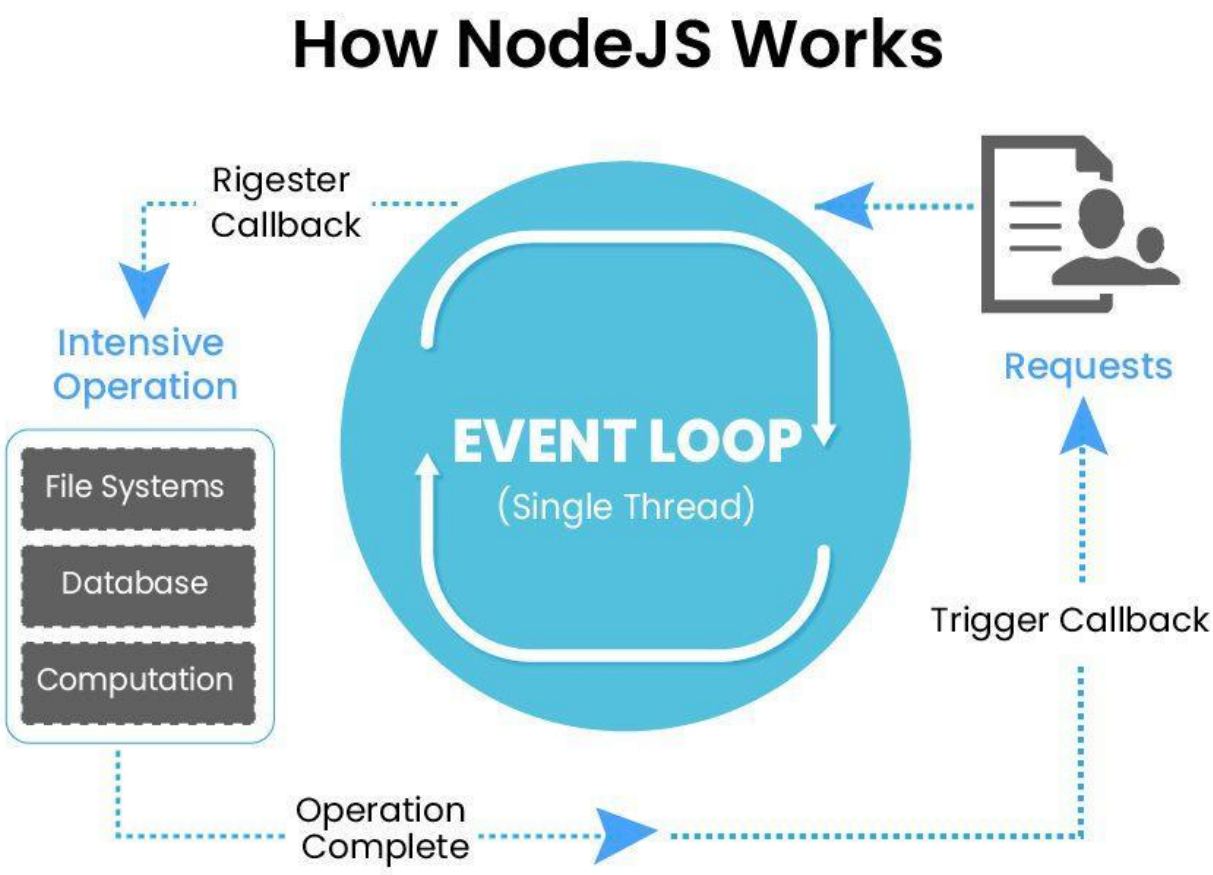
Node.js đã mở rộng khả năng của JavaScript từ việc chỉ phát triển front-end trong trình duyệt để bao gồm cả phát triển back-end. Điều này có nghĩa là các lập trình viên có thể sử dụng cùng một ngôn ngữ lập trình, JavaScript, để phát triển toàn bộ ứng dụng, từ front-end đến back-end, qua đó tạo điều kiện cho việc học tập và phát triển ứng dụng nhanh chóng và hiệu quả hơn.

Node.js hoạt động dựa trên một số nguyên tắc cơ bản giúp nó hiệu quả trong việc xử lý các ứng dụng có nhiều hoạt động nhập/xuất (I/O) mà không bị chặn, đồng thời giảm đáng kể sự phức tạp trong quản lý các luồng thực thi. Dưới đây là một số thành phần chính giải thích cách thức hoạt động của Node.js:

- Kiến trúc Non-blocking I/O và Event-Driven: Node.js sử dụng một mô hình non-blocking I/O (input/output) và event-driven, nghĩa là các hoạt động như đọc file, truy vấn cơ sở dữ liệu, hoặc giao tiếp mạng được thực hiện mà không chặn tiến trình chính. Điều này cho phép xử lý nhiều yêu cầu cùng lúc mà không cần tạo nhiều luồng (thread), giúp giảm bớt chi phí liên quan đến quản lý luồng và tối ưu hóa hiệu suất. Khi một hoạt động I/O được khởi tạo, nó sẽ được gửi đến thực thi trong hệ thống hoặc cơ sở dữ liệu mà không làm chậm tiến trình chính. Sau khi hoạt động hoàn tất, một sự kiện sẽ được phát đi và xử lý bằng các hàm gọi lại (callback).
- V8 JavaScript Engine Node.js được xây dựng trên động cơ JavaScript V8 của Google Chrome, đây là một động cơ rất nhanh cho phép biên dịch mã JavaScript thành mã máy để thực thi trực tiếp trên phần cứng, làm tăng hiệu suất thực thi.
- Single-Threaded Mặc dù Node.js hoạt động trên một luồng duy nhất cho logic ứng dụng của người dùng, nó vẫn sử dụng nhiều luồng ở tầng thấp hơn thông qua thư viện libuv để xử lý các hoạt động I/O. Tuy nhiên, những chi tiết này được ẩn giấu khỏi người dùng, giúp việc lập trình đơn giản hơn mà vẫn đảm bảo hiệu suất.

- Event Loop Trái tim của Node.js là “event loop”. Đây là vòng lặp sự kiện mà ở đó Node.js tiếp tục lắng nghe sự kiện và thực hiện các hàm gọi lại khi một sự kiện được kích hoạt. Vòng lặp sự kiện cho phép Node.js xử lý hàng nghìn kết nối đồng thời mà không cần phải tạo ra chi phí quản lý luồng.
- Trigger Callback Khi thao tác I/O hoàn tất, hệ điều hành thông báo cho Node.js, và Nodejs sau đó thực thi hàm callback tương ứng để xử lý kết quả hoặc tiếp tục xử lý logic.
- NPM (Node Package Manager) NPM là hệ thống quản lý gói cho Node.js, cho phép các nhà phát triển dễ dàng chia sẻ và sử dụng mã nguồn từ nhau. NPM là một trong những kho lưu trữ mã nguồn mở lớn nhất thế giới và chứa hàng ngàn module có thể được tích hợp vào ứng dụng của bạn.

Tổng hợp lại, Nodejs mang đến một mô hình hiệu quả và mạnh mẽ cho các ứng dụng web và máy chủ, nhờ khả năng xử lý đồng thời nhiều hoạt động I/O mà không bị chặn, và qua đó tối ưu hóa việc sử dụng tài nguyên và cải thiện hiệu suất.



Hình 2.4 Cách hoạt động của NodeJS

2.5 Mongo DB

MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở, là CSDL thuộc NoSql và được hàng triệu người sử dụng.

MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như CSDL quan hệ nên truy vấn sẽ rất nhanh.

Với CSDL quan hệ chúng ta có khái niệm bảng, các cơ sở dữ liệu quan hệ (như MySQL hay SQL Server...) sử dụng các bảng để lưu dữ liệu thì với MongoDB chúng ta sẽ dùng khái niệm là collection thay vì bảng

So với RDBMS thì trong MongoDB collection ứng với table, còn document sẽ ứng với row , MongoDB sẽ dùng các document thay cho row trong RDBMS.

Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định.

Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB

Ưu điểm:

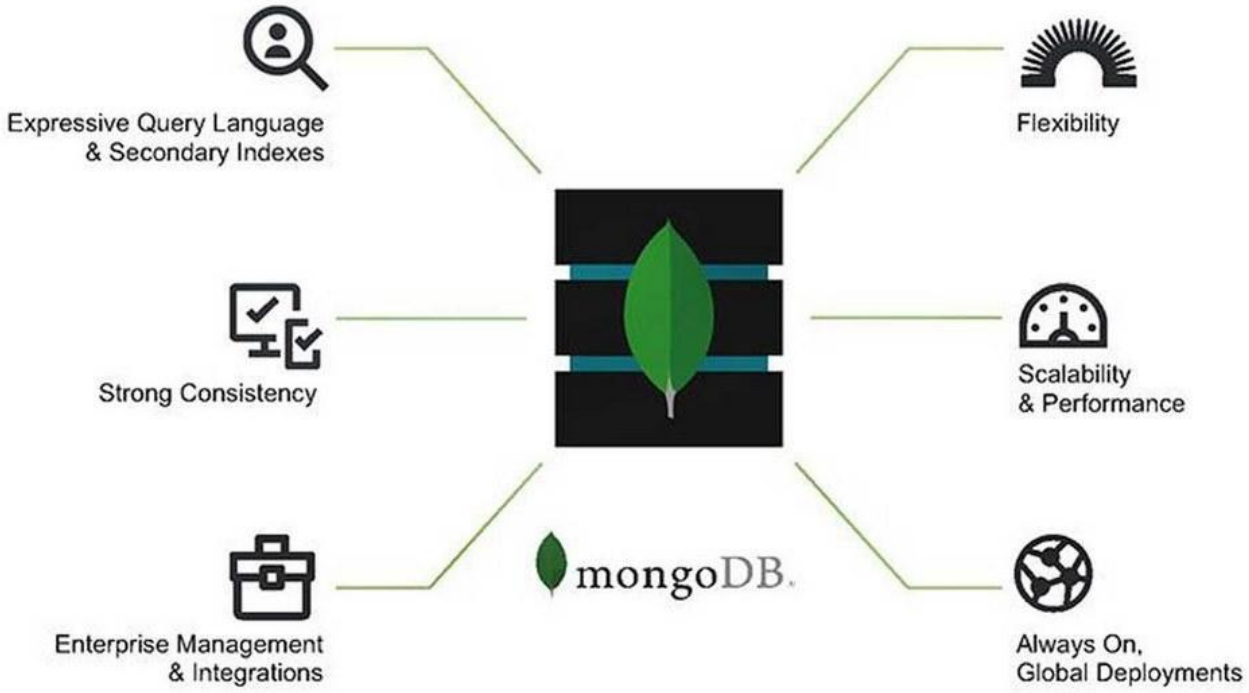
- Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ có các kích cỡ và các document khác nhau, linh hoạt trong việc lưu trữ dữ liệu.
- Dữ liệu trong MongoDB không có sự ràng buộc lẫn nhau, không có join như trong RDBMS nên khi insert, xóa hay update nó không cần phải mất thời gian kiểm tra xem có thỏa mãn các ràng buộc dữ liệu như trong RDBMS.
- MongoDB rất dễ mở rộng (Horizontal Scalability). Trong MongoDB có một khái niệm cluster là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node với vào cluster:
- Trường dữ liệu "_id" luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.
- Khi có một truy vấn dữ liệu, bản ghi được cached lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.
- Hiệu năng cao: Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Với một lượng dữ liệu đủ lớn thì thử nghiệm cho thấy tốc độ insert của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.

Nhược điểm:

- Một ưu điểm của MongoDB cũng chính là nhược điểm của nó. MongoDB không có các tính chất ràng buộc như trong RDBMS nên khi thao tác với mongoDB thì phải hết sức cẩn thận.

- Tốn bộ nhớ do dữ liệu lưu dưới dạng key-value, các collection chỉ khác về value do đó key sẽ bị lặp lại. Không hỗ trợ join nên dễ bị dư thừa dữ liệu.
- Khi insert/update/remove bản ghi, MongoDB sẽ chưa cập nhật ngay xuống ổ cứng, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng điều này sẽ là nhược điểm vì sẽ có nguy cơ bị mất dữ liệu khi xảy ra các tình huống như mất điện...

Với các hệ thống realtime (thời gian thực) yêu cầu phản hồi nhanh, các hệ thống bigdata với yêu cầu truy vấn nhanh hay các hệ thống có lượng request lớn thì MongoDB sẽ là sự lựa chọn ưu tiên hơn CSDL quan hệ. Tùy theo dự án và trường hợp cụ thể để sử dụng CSDL quan hệ hay sử dụng MongoDB đem lại hiệu quả cao.



Hình 2.5 Cách tính năng của MongoDB

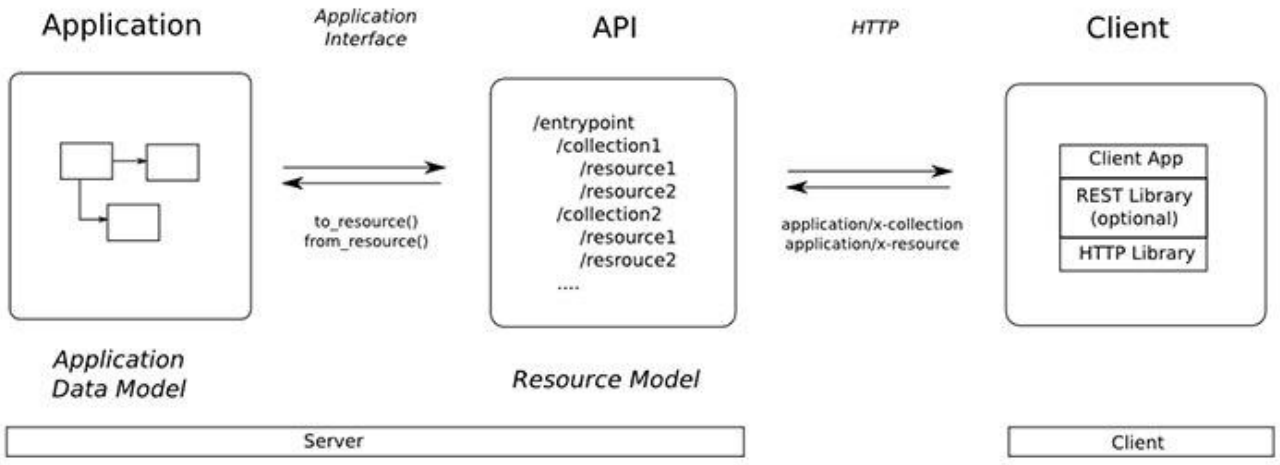
2.6 RESTful API

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web 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.

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.



Hình 2.6 Cách hoạt động của RESTful API

2.7 Postman

Postman là một ứng dụng mã nguồn mở dùng để phát triển và kiểm thử các API . Công nghệ cung cấp môi trường cho các nhà phát triển thực hiện hoạt động tạo, chia sẻ, kiểm thử và quản lý các API.

Nền tảng Postman có các tính năng như tạo yêu cầu HTTP, kiểm thử tự động, quản lý biến môi trường và biến toàn cục. Những tiện ích này sẽ giúp nhà phát triển tiết kiệm thời gian trong việc phát triển và kiểm thử API. Đồng thời, Postman còn có chế độ tương tác thông minh với API, giúp người dùng hiểu rõ hơn về cách API hoạt động và phản hồi của chúng.

Lập trình viên nên sử dụng Postman vì công nghệ này có thể cung cấp nhiều lợi ích quan trọng trong quá trình phát triển và kiểm thử API.

- Phát triển nhanh chóng: Postman cho phép lập trình viên tạo và chia sẻ các yêu cầu HTTP một cách nhanh chóng, giúp tiết kiệm thời gian trong quá trình phát triển.
- Kiểm thử hiệu quả: Với Postman, lập trình viên có thể tạo các bộ kiểm thử tự động để kiểm tra tính đúng đắn và hiệu suất của API một cách hiệu quả.
- Quản lý biến môi trường: Postman cho phép lập trình viên quản lý và sử dụng các biến môi trường và biến toàn cục, giúp dễ dàng thay đổi các thiết lập môi trường khi cần thiết.
- Tương tác thông minh với API: Postman cung cấp các tính năng tương tác thông minh với API, giúp lập trình viên hiểu rõ hơn về cách API hoạt động và phản hồi của chúng.
- Quản lý tập tin và tài nguyên: Postman cung cấp các công cụ quản lý tập tin và tài nguyên liên quan đến API, giúp lập trình viên dễ dàng tổ chức và quản lý dự án.

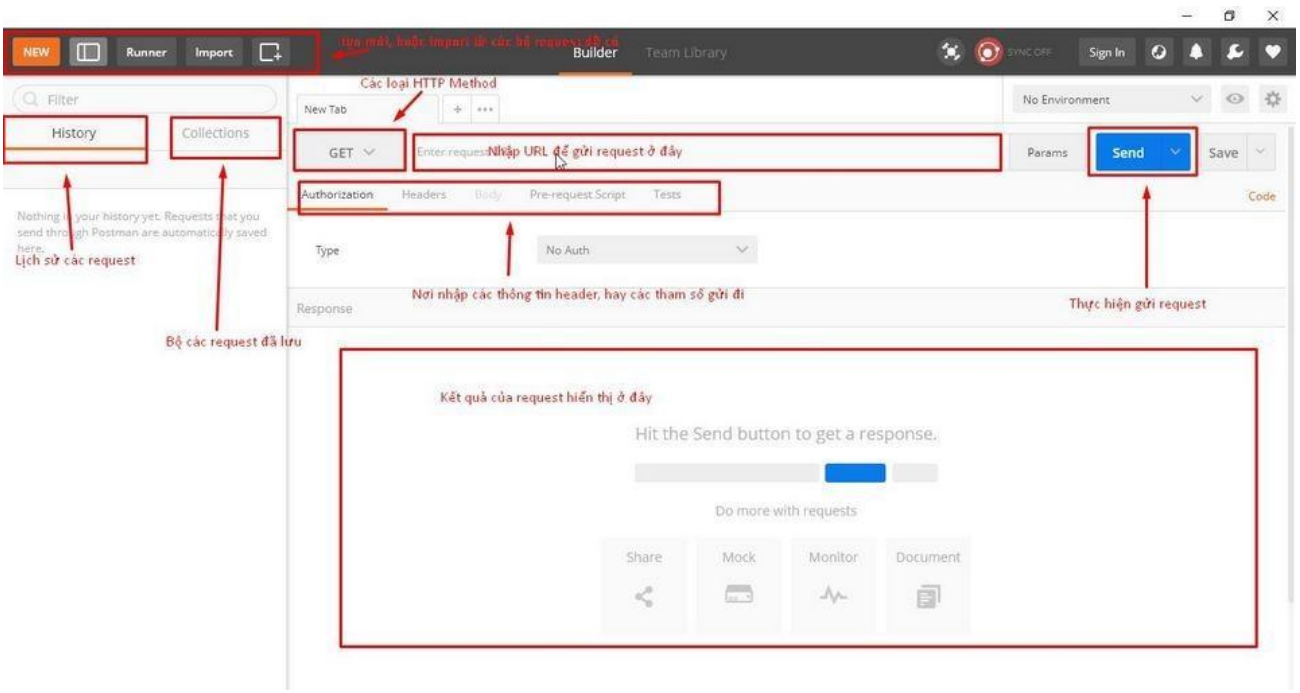
Postman có nhiều ưu điểm quan trọng khi sử dụng trong quá trình phát triển và kiểm thử API, bao gồm:

- Giao diện thân thiện: Postman cung cấp giao diện người dùng dễ sử dụng và thân thiện, giúp người dùng tạo và quản lý các yêu cầu API một cách hiệu quả.
- Tự động hóa kiểm thử: Cung cấp khả năng tạo các bộ kiểm thử tự động để kiểm tra API, giúp giảm thiểu các lỗi và tối ưu hóa hiệu suất của API

- Quản lý biến môi trường: Cho phép lập trình viên quản lý và sử dụng các biến môi trường và biến toàn cục, dễ dàng thay đổi các thiết lập môi trường khi cần thiết.
- Chia sẻ dễ dàng: Cung cấp khả năng chia sẻ các collection và bộ kiểm thử với đồng nghiệp hoặc cộng đồng, tạo điều kiện cho việc hợp tác và kiểm thử chung.
- Tương tác mạnh mẽ với API: Postman cung cấp các tính năng tương tác mạnh mẽ với API, giúp người dùng hiểu rõ cách API hoạt động và phản hồi của chúng.
- Quản lý dự án thông minh: Cung cấp các công cụ quản lý tập tin và tài nguyên liên quan đến API, giúp dễ dàng tổ chức và quản lý dự án.

Mặc dù Postman là một công cụ mạnh mẽ và phổ biến trong quá trình kiểm thử nhưng người dùng nên lưu ý một số vấn đề dưới đây:

- Giới hạn trong phiên bản miễn phí: Phiên bản miễn phí của Postman có giới hạn về số lượt gửi yêu cầu và các tính năng cao cấp, điều này có thể tạo ra hạn chế đối với các dự án lớn.
- Tính tương thích: Mặc dù Postman hỗ trợ nhiều loại yêu cầu API và các kiểu dữ liệu phản hồi nhưng vẫn xuất hiện một số hạn chế khi tương tác với các loại API phức tạp và cần nhiều xử lý.
- Quản lý phiên bản và quy trình: Trong môi trường phát triển phần mềm lớn, việc quản lý phiên bản và quy trình thường trở nên phức tạp khi dự án mở rộng và có nhiều người tham gia.
- Hiệu suất khi thực hiện kiểm thử lớn: Trong trường hợp thực hiện kiểm thử lớn và phức tạp, việc quản lý và theo dõi các bộ kiểm thử sẽ không hiệu quả.
- Yêu cầu kỹ năng: Để sử dụng Postman một cách hiệu quả, người dùng cần phải có kiến thức vững về API và quá trình phát triển phần mềm. Đây sẽ là một thử thách lớn đối với người dùng mới.



Hình 2.7 Giao diện Postman

2.8 JWT (JSON Web Token)

JWT là các ký hiệu viết tắt của từ JSON Web Token, đây là một tiêu chuẩn mở. JWT giúp người dùng tạo ra chuỗi mã hóa dữ liệu để trao đổi thông tin giữa các hệ thống khác nhau dưới dạng JSON object một cách an toàn và tin cậy.

Các chuỗi thông tin trao đổi được xác minh thông qua chữ ký điện tử sử dụng thuật toán HMAC hoặc cặp khóa public/private key. Nhờ đó, thông tin đáp ứng các tiêu chuẩn RSA hoặc ECDSA về tính bảo mật và an toàn. Nhờ đó, chuỗi thông tin đáp ứng tiêu chuẩn JWT luôn được đánh giá cao về tính bảo vệ, tránh đánh cắp thông tin dữ liệu tối đa.

JWT gồm 3 thành phần chính: Header, Payload, Signature và chúng được ngăn cách với nhau bằng dấu ".". Vì vậy, cấu trúc của JWT sẽ theo format sau: "header.payload.signature".

Header chứa kiểu dữ liệu và các thuật toán giúp mã hóa ra chuỗi JWT một cách hoàn hảo. Header bao gồm 2 phần chính, đó là:

- Typ (Type): Đây là token và được mặc định là một JWT.
- ALG (Algorithm): Được dùng để mã hóa nhanh chóng (thuật toán chữ ký được sử dụng phổ biến như HMAC, SHA256, RSA).

Payload đóng một vai trò rất quan trọng trong JWT, đây là nơi chứa các nội dung của thông tin (claim) mà người sử dụng muốn truyền đi ở bên trong chuỗi. Các thông tin này góp phần mô tả thực thể một cách đơn giản và nhanh chóng hoặc cũng có thể là các thông tin bổ sung thêm cho phần Header.

Signature là phần chữ ký bí mật và phần này được tạo ra bởi mã hóa phần Header cùng với phần Payload kết hợp với một chuỗi secret (khóa bí mật). Khi 3 phần này được kết hợp lại với nhau, chúng ta sẽ có một chuỗi JWT hoàn chỉnh nhất. Nhờ đó, JWT đảm bảo có thể trợ giúp hiệu quả cao cho công việc của một người lập trình viên.

Các ưu điểm nổi bật của JWT:

- Tính bảo mật cao: JWT nổi bật với tính bảo mật tốt. Người dùng thực hiện đăng nhập vào hệ thống bắt buộc dùng mã JWT cho những yêu cầu tiếp theo. Không chỉ vậy, việc này sẽ giúp cho người dùng thuận tiện hơn trong việc click vào những server, resource và url một cách khá là dễ dàng bởi họ dễ dàng xác nhận quyền truy cập với một JWT duy nhất. Để tránh tình huống JWT bị đánh cắp và kẻ gian sử dụng, JWT luôn luôn có thời hạn sử dụng ở khoảng thời gian ngắn. Mã JWT sẽ không có giá trị sử dụng nếu ngoài thời gian sử dụng. Tuy nhiên, người dùng cần bảo mật mật mã này để đảm bảo an toàn thông tin tối đa.
- Dễ dàng truyền đạt và trao đổi thông tin Cũng bởi đặc tính JWT có độ an toàn cao nên các thành viên dễ dàng trao đổi, nhận dạng nhau theo phần chữ ký. Thông qua đó, người dùng dễ dàng nhận biết được ai là người gửi thông tin, đảm bảo tính chính xác và phòng ngừa tình trạng giả mạo.
- Mang lại nhiều lợi ích cho người dùng: Thuật toán mã hóa HMAC giúp thông tin bảo mật và dễ dàng chia sẻ. Quen thuộc với các ngôn ngữ lập trình khi chúng tạo ra một bản đồ trực tiếp hướng đến các đối tượng khác nhau. JWT nhỏ gọn hơn nhiều XML khác nhờ được mã hóa một cách rất thông minh. Phù hợp với mọi thiết bị, kể cả các thiết bị di động cá nhân.

Hình 2.8 Cấu trúc của JWT

2.9 Github

GitHub là một mạng xã hội đặc biệt dành cho lập trình viên, là một hệ thống quản lý dự án, lưu trữ source code, theo dõi và cộng tác trong các dự án phần mềm.

Ngoài ra, GitHub là một dịch vụ nổi tiếng cung cấp kho lưu trữ mã nguồn Git cho các dự án phần mềm. Github có đầy đủ những tính năng của Git, ngoài ra nó còn bổ sung những tính năng về social để các developer tương tác với nhau.

GitHub có 2 phiên bản: miễn phí và trả phí. Với phiên bản có phí thường được các doanh nghiệp sử dụng để tăng khả năng quản lý team cũng như phân quyền bảo mật dự án. Còn lại thì phần lớn chúng ta đều sử dụng Github với tài khoản miễn phí để lưu trữ source code.

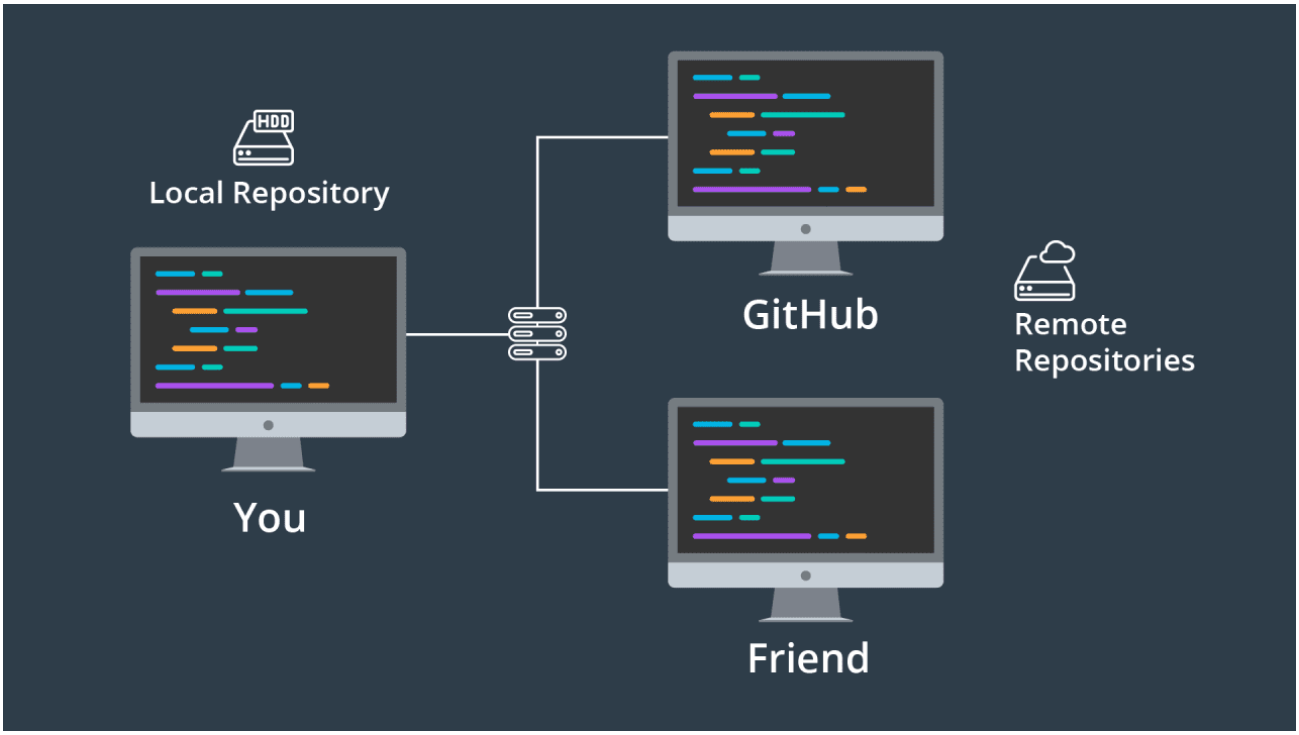
Github cung cấp các tính năng social networking như feeds, followers, và network graph để các developer học hỏi kinh nghiệm của nhau thông qua lịch sử commit.

Nếu một comment để mô tả và giải thích một đoạn code. Thì với Github, commit message chính là phần mô tả hành động mà bạn thực hiện trên source code.

Github trở thành một yếu tố có sức ảnh hưởng lớn trong cộng động nguồn mở. Cùng với Linkedin, Github được coi là một sự thay thế cho CV của bạn. Các nhà tuyển dụng cũng rất hay tham khảo Github profile để hiểu về năng lực coding của ứng viên.

Các tính năng chính của Github:

- GitHub tập trung mã nguồn và tài liệu trong một nơi duy nhất: GitHub cho phép tất cả mã nguồn và tài liệu dự án được lưu trữ tập trung trong các "repository" (kho lưu trữ). Điều này đảm bảo rằng bất kỳ ai muốn đóng góp vào dự án đều có quyền truy cập vào những tài nguyên cần thiết, giảm thiểu các vấn đề về truy cập thông tin. Mỗi repository thường đi kèm với các hướng dẫn cụ thể, giúp người tham gia hiểu được mục tiêu và quy tắc của dự án.
- Quản lý xung đột mã: Lập trình không chỉ đơn giản là viết mã; nó còn đòi hỏi sự sáng tạo và linh hoạt. Ví dụ, hai lập trình viên có thể đang làm việc trên các phần mã khác nhau, nhưng chúng cần phải hoạt động hài hòa với nhau. Đôi khi, một phần mã có thể khiến phần mã khác bị lỗi, hoặc có thể có tác động không mong muốn lên cách mà phần khác hoạt động. GitHub giải quyết vấn đề này bằng cách hiển thị rõ ràng những thay đổi từ hai lập trình viên trước khi họ đẩy (push) mã lên "branch" (nhánh chính) của dự án. Điều này giúp phát hiện và khắc phục các lỗi tiềm ẩn trước khi chúng ảnh hưởng đến toàn bộ dự án.
- GitHub giúp theo dõi và khôi phục phiên bản mã nguồn: Một trong những tính năng quan trọng nhất của GitHub là khả năng quản lý phiên bản. Điều này cho phép bạn theo dõi mọi thay đổi trong dự án và quay lại các phiên bản trước đó nếu cần. GitHub dựa trên công nghệ Git, một hệ thống kiểm soát phiên bản (version control system), giúp lưu lại các thay đổi qua từng "commit" (lần lưu mã). Bạn có thể quay lại phiên bản trước đó nếu phát hiện lỗi, hoặc theo dõi ai đã thay đổi những gì và khi nào.



Hình 2.9 Cách hoạt động của GitHub

2.10 Figma

Figma là một ứng dụng web chứa đựng nhiều công cụ thiết kế mạnh mẽ. Khi sử dụng Figma, bạn có khả năng sáng tạo không giới hạn cho giao diện người dùng (UI/UX), tạo mẫu thiết kế, và tạo nội dung đăng trên các mạng xã hội, cũng như thực hiện nhiều dự án thiết kế khác.

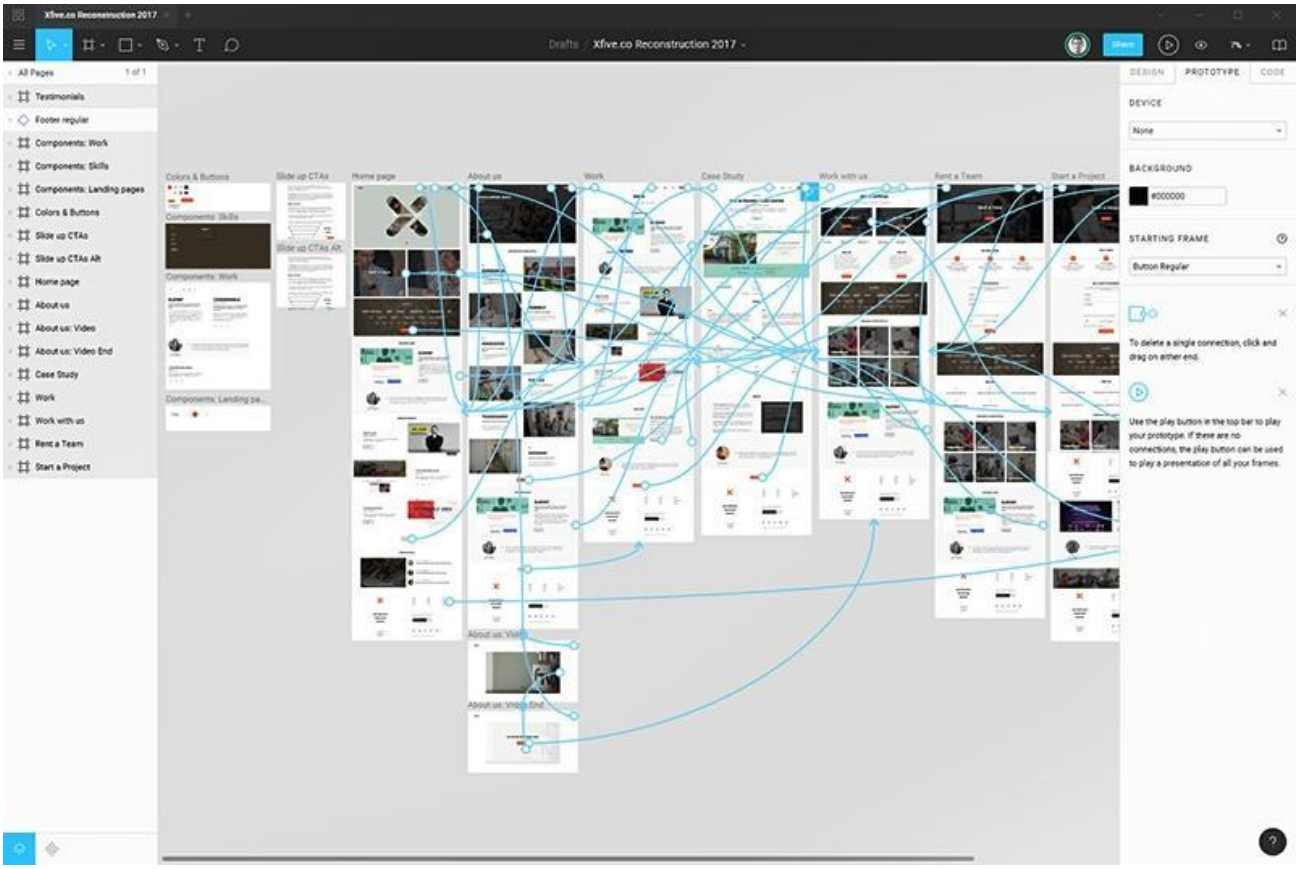
Figma được ví như là một công cụ đa năng, phù hợp cho nhiều mục đích thiết kế từ sản phẩm số đến các thiết kế UX/UI. Các ứng dụng phổ biến của Figma bao gồm thiết kế tạo mẫu (Prototype design), thiết kế wireframe cho website, thiết kế giao diện cho ứng dụng di động và tạo ra các bài đăng trên các mạng xã hội.

Các tiện ích nổi bật của figma

- Khả năng tương thích cao: Figma là một ứng dụng dựa trên nền tảng website, có thể truy cập và áp dụng trên rất nhiều các hệ điều hành phổ biến như Windows, MacOS, Linux,.. với điều kiện thiết bị của bạn đang kết nối Internet.
- Tính cộng tác cao: khả năng cộng tác ngay trong khoảng thời gian thực. Nhờ Figma mà vấn đề làm việc nhóm từ xa đã không còn trở ngại, mà trở nên cực kỳ dễ dàng bởi

ứng dụng cho phép các thành viên trong nhóm có thể cùng nhau thiết kế, phản hồi, kiểm tra tiến độ và đặt cờ các tình huống có thể xuất hiện ngay trong thời gian thực.

- Trang bị nhiều công cụ plugin mạnh mẽ: Figma cung cấp một kho plugin rộng lớn và vô cùng hữu ích, giúp bạn giải quyết các vấn đề và nâng cao hiệu suất làm việc. Quản lý màu sắc, thay đổi nội dung, hình ảnh và hoạt ảnh trở nên đơn giản hơn bao giờ hết thông qua các plugin này.
- Thiết kế nhiều layout trong một sản phẩm Figma tích hợp tính năng quản lý nhiều artboard đồng thời, cho phép bạn tạo nhiều bố cục với kích thước khác nhau trên cùng trên một sản phẩm mà không cần phải tạo thêm bất cứ tệp mới nào.
- Xuất được đa dạng file ảnh cực sắc: Khả năng xuất ra hình ảnh với độ sắc nét được giữ nguyên, bạn có thể lưu dưới nhiều định dạng khác nhau như SVG, PDF, PNG, JPG. Điều này giúp linh hoạt trong việc sử dụng hình ảnh và phục vụ đa dạng trong các tình huống khác nhau.
- Hỗ trợ lưu trữ đám mây: Figma tận dụng dịch vụ đám mây để thuận tiện cho việc lưu trữ và chỉnh sửa dữ liệu, mô hình này tương đồng với việc chỉnh sửa nội dung trực tuyến trên Google Docs. Điều này giúp loại bỏ lo ngại về việc ổ đĩa bị đầy hoặc quên sao lưu, bởi Figma sẽ thực hiện tự động các công đoạn này.



Hình 2.10 Giao diện figma

2.11 Jira

Jira là ứng dụng làm việc cho phép các nhóm theo theo dõi các vấn đề (issue) cần xử lí, quản lý dự án và tự động hóa quy trình làm việc, do công ty phần mềm Atlassian của Úc phát triển.

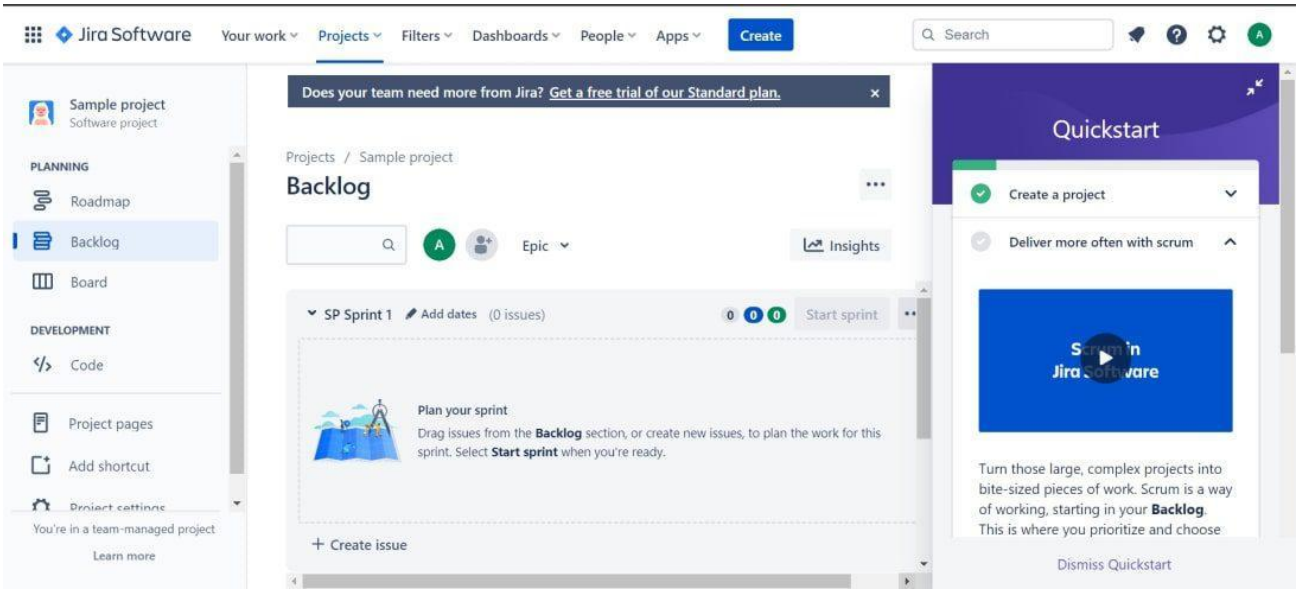
Đây là một công cụ phổ biến trong các đội ngũ phát triển phần mềm để lập kế hoạch, theo dõi và phát hành các dự án phần mềm. JIRA cung cấp một nền tảng tập trung để quản lý các nhiệm vụ, lỗi và các loại vấn đề khác, giúp các đội ngũ tổ chức sắp xếp và ưu tiên công việc của họ.

Các thành phần cơ bản của Jira:

- Roles: Xác lập các role của dự án, Mục này xác nhận ai tham gia vào dự án, những người add vào role thì mới có thể tạo Resource Allocation và project team sau này. Nhiều người có thể vào 1 role.
- Project: Chức năng này dùng để phân quyền approve worklog cho thành viên của dự án. Ai là team lead của group nào thì sẽ được approve worklog cho member của group đó. Project management được quyền approve cho toàn bộ thành viên dự án
- Issue là thành phần cơ bản để theo dõi các công việc, lỗi, nhiệm vụ hoặc bất kỳ công việc nào cần được quản lý. Các issue có thể được phân loại thành nhiều loại khác nhau như Bug (lỗi), Task (nhiệm vụ), Story (câu chuyện), Epic (tính năng lớn), v.v.
- Component là sản phẩm của dự án. Ở đây sẽ nhập tất cả sản phẩm của dự án lấy từ file kế hoạch doanh số. Nếu dự án làm theo Scrum thì sẽ là Product của Sprint tương ứng.
- Workflow là chuỗi các trạng thái và chuyển đổi mà một issue có thể trải qua trong suốt vòng đời của nó. Workflows có thể được tùy chỉnh để phản ánh quy trình làm việc cụ thể của từng dự án.
- Priority: Là mức độ ưu tiên của một defect. Có 4 mức, chọn theo datalist
- Status: Đại diện cho các vị trí của vấn đề trong workflow
- Resolution (Giải quyết): Kết quả cuối cùng của một issue, ví dụ: Fixed (Đã sửa), Won't Fix (Không sửa), Duplicate (Trùng lặp).
- Custom Fields (Trường tùy chỉnh): Jira cho phép người dùng tạo các trường tùy chỉnh để lưu trữ các thông tin đặc thù của dự án hoặc nhóm làm việc. Các trường này có thể được thêm vào các issue để cung cấp thêm thông tin cần thiết.
- Reports and Dashboards (Báo cáo và Bảng điều khiển): Jira cung cấp nhiều loại báo cáo và bảng điều khiển để giúp theo dõi tiến độ, hiệu suất và tình trạng của dự án. Người dùng có thể tạo các báo cáo tùy chỉnh và bảng điều khiển để hiển thị các số liệu quan trọng và các thông tin cần thiết.

Jira là phần mềm với khá nhiều thuật ngữ công nghệ:

- Sprint: Một vòng lặp ngắn hạn (lý tưởng là 2-4 tuần) mà đội phát triển thực hiện đầy đủ các công việc cần thiết như lập kế hoạch, phân tích yêu cầu, thiết kế, triển khai để cho ra các phần nhỏ của sản phẩm.
- Backlog: Danh sách tập hợp các user stories, bugs và tính năng cho một sản phẩm hoặc sprint.
- Scrum: Một phương pháp Agile, nơi sản phẩm được xây dựng theo các lần lặp đi lặp lại trong một sprint.
- Scrum of Scrums: Một kỹ thuật để mở rộng quy mô Scrum, các dự án đa đội – theo truyền thống gọi là program management.
- Board: Công cụ dùng để hiển thị hoạt động công việc trong một quy trình làm việc cụ thể. Nó có thể thay đổi thích ứng với các phương pháp Agile khác nhau (ví dụ, một bảng Scrum sẽ hiển thị các công việc được di chuyển từ product backlog đến sprint backlog, trong khi đó một bảng Kanban thường có một quy trình làm việc ba bước: To do, In Progress, và Done).



Hình 2.11 Giao diện Jira

2.12 Docker

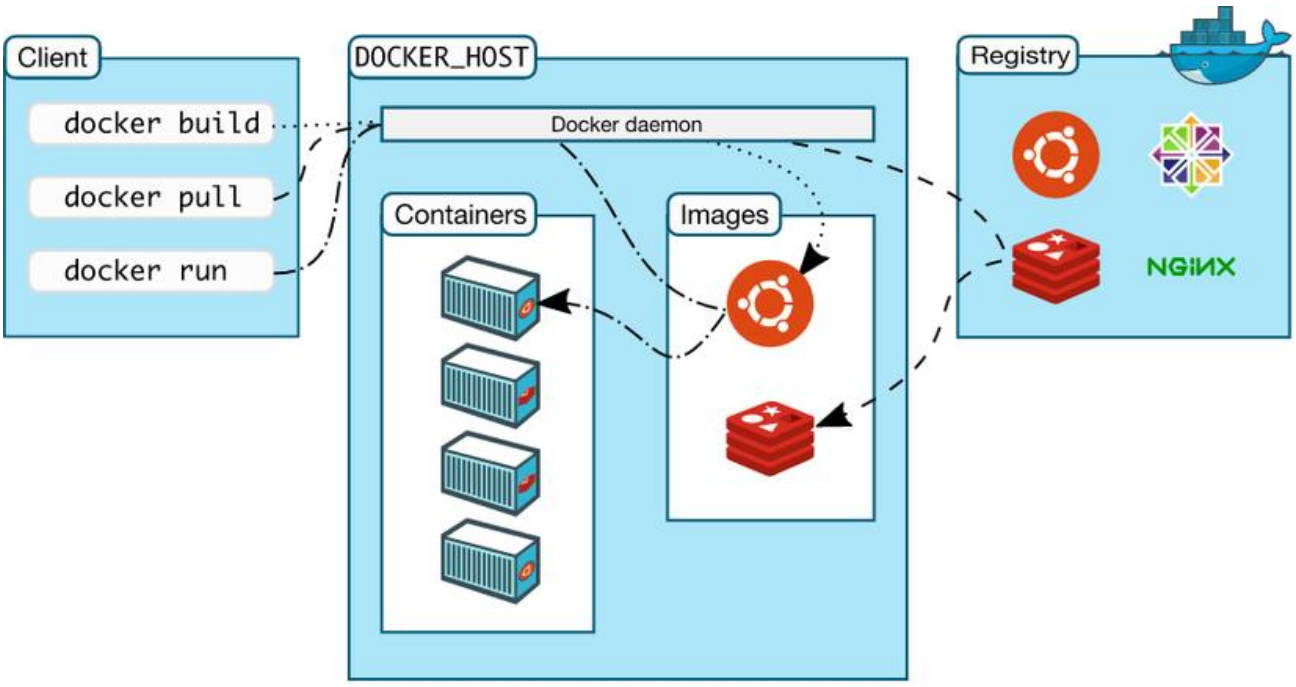
Docker là một nền tảng phần mềm giúp bạn building, deploying và running ứng dụng dễ dàng hơn bằng cách sử dụng các containers (trên nền tảng ảo hóa).

Docker đóng gói phần mềm thành các container tiêu chuẩn hóa, chứa đựng tất cả những thứ cần thiết để phần mềm hoạt động như thư viện, công cụ hệ thống, mã nguồn và thời gian chạy. Khi cần deploy app lên bất kỳ server nào, bạn chỉ cần run container của Docker thì app của bạn sẽ được khởi chạy ngay lập tức.

Container là các gói phần mềm nhỏ gọn chứa tất cả các thành phần cần thiết của một ứng dụng như mã nguồn, thư viện, và các công cụ, giúp đảm bảo ứng dụng có thể chạy đồng nhất trên mọi môi trường. Bằng cách đó, nhờ vào container, ứng dụng sẽ chạy trên mọi máy Linux khác bất kể mọi cài đặt tùy chỉnh mà máy có thể khác với máy được sử dụng để viết code.

Các ưu điểm khi triển khai ứng dụng trên Docker:

- Tính dễ ứng dụng: Docker rất dễ cho mọi người sử dụng từ lập trình viên, sysadmin… nó tận dụng lợi thế của container để build, test nhanh chóng. Có thể đóng gói ứng dụng trên laptop của họ và chạy trên public cloud, private cloud…
- Tốc độ: Docker container rất nhẹ và nhanh, bạn có thể tạo và chạy docker container trong vài giây.
- Môi trường chạy và khả năng mở rộng: Bạn có thể chia nhỏ những chức năng của ứng dụng thành các container riêng lẻ. Ví dụ Database chạy trên một container và Redis cache có thể chạy trên một container khác trong khi ứng dụng Node.js lại chạy trên một cái khác nữa. Với Docker, rất dễ để liên kết các container với nhau để tạo thành một ứng dụng, làm cho nó dễ dàng scale, update các thành phần độc lập với nhau.
- Hiệu suất cao: Docker containers chia sẻ cùng một hệ điều hành kernel, giúp giảm thiểu tài nguyên hệ thống so với các máy ảo (VMs), dẫn đến hiệu suất cao hơn và chi phí thấp hơn.
- Quản lý phụ thuộc dễ dàng: Docker giúp quản lý tất cả các phụ thuộc của ứng dụng trong một container, đảm bảo rằng không có sự mâu thuẫn phiên bản giữa các thư viện và công cụ.



Hình 2.12 Thực thi Docker

Như vậy, chương 2 đã trình bày toàn bộ cơ sở lý thuyết và các khái niệm quan trọng làm nền tảng cho quá trình phát triển hệ thống, bao gồm kiến trúc web, mô hình 3 lớp, nguyên lý hoạt động của React, Node.js cũng như cơ chế xác thực bằng JWT. Thiết kế giao diện với Figma, quản lý dự dự án với Jira, lưu trữ và quản lý mã nguồn với Github. Những kiến thức này đóng vai trò định hướng cho việc phân tích, thiết kế và xây dựng hệ thống được trình bày trong các chương tiếp theo. Chương 3 sẽ đi vào phân tích cụ thể yêu cầu hệ thống, các chức năng chính và mô hình hóa kiến trúc nhằm đảm bảo hệ thống đáp ứng đúng mục tiêu đề ra.

CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU

Dựa trên các cơ sở lý thuyết đã được trình bày ở chương 2 cơ sở lý thuyết, chương này sẽ tập trung vào việc hiện thực hóa hệ thống thông qua việc phân tích, thiết kế hệ thống, lựa chọn công nghệ phù hợp, xây dựng kiến trúc tổng thể, và triển khai các chức năng chính của ứng dụng. Việc hiện thực hóa sẽ bám sát theo các yêu cầu đã phân tích, đồng thời đảm bảo tính khả thi, mở rộng và bảo trì lâu dài. Trong chương này, nhóm sẽ trình bày quy trình phát triển hệ thống, sơ đồ hoạt động, các thành phần chính của hệ thống (frontend, backend, cơ sở dữ liệu), cũng như quy trình triển khai các chức năng từ thiết kế đến lập trình chi tiết.

3.1 Quản lý dự án với Jira

Kế hoạch phát triển phần mềm, xây dựng các Sprint chính của hệ thống:

- Sprint 1: Đăng nhập đăng ký, tra cứu, xem chi tiết sách
  Mục tiêu: Phát triển tính năng đăng ký tài khoản, đăng nhập/đăng xuất, tra cứu sách, xem chi tiết sách.
  Người thực hiện: Nguyễn Trường Vũ
  Thời gian thực hiện: 1/7 – 7/7/2025
  Kết quả: Giao diện và logic đăng nhập hoạt động ổn định, tra cứu nhanh chóng.
- Sprint 2: Đặt mượn sách
  Mục tiêu: Cho phép mượn sách, tập trung vào người dùng độc giả, quản lý sách, quản lý mượn trả
  Người thực hiện: Trần Dinh Hiển
  Thời gian thực hiện: 8/7 – 12/7/2025
  Kết quả: cho phép mượn sách, quản lý sách, quản lý mượn trả hiệu quả
- Sprint 3: Duyệt sách, xem lịch sử mượn
  Mục tiêu: Thủ thư duyệt yêu cầu mượn, hiển thị lịch sử mượn của người dùng
  Thời gian thực hiện: 13/7- 18/7/2025
  Người thực hiện: Trần Dinh Hiển

Kết quả: Duyệt yêu cầu mượn trả, lập phiếu mượn nhanh chóng, cần kiểm tra điều kiện.

- Sprint 4: Nghiệp vụ quản lý

  Mục tiêu:Xây dựng các chức năng quản lý người dùng sách , quản lý mượn trả, quản lý vi phạm,...

  Người thực hiện: Đào công Hoàng Lam

  Thời gian thực hiện: 15/7- 22/7/2025

  Kết quả: Giúp quản trị viên, thủ thư quản lý hiệu quả hệ thống.

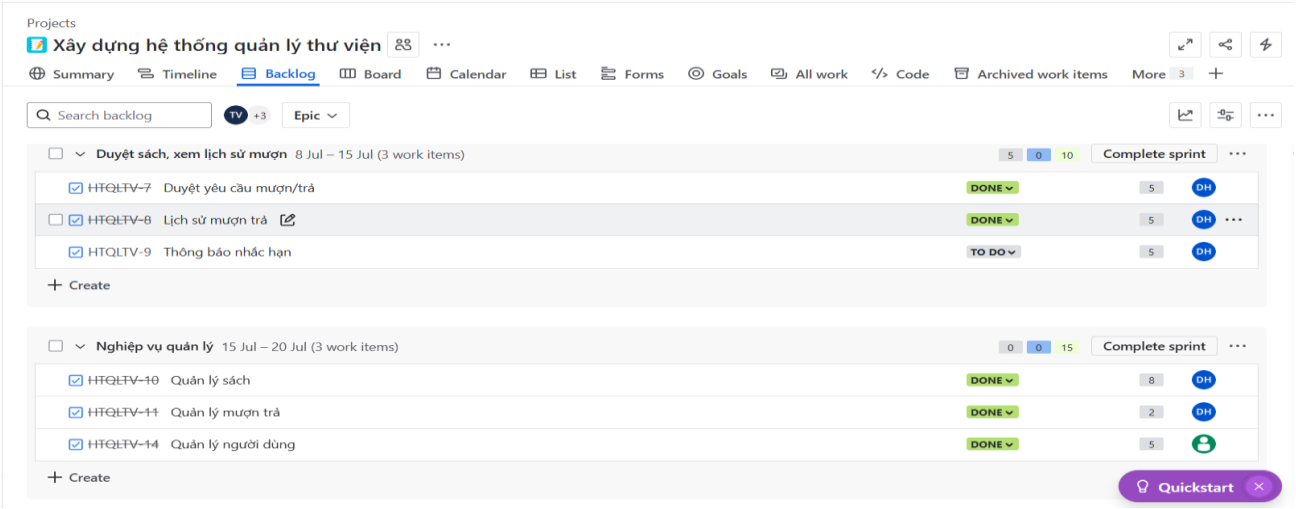
- Sprint 5: Hoàn thiện UI/UX

  Mục tiêu: Hoàn thiện giao diện cho hệ thống

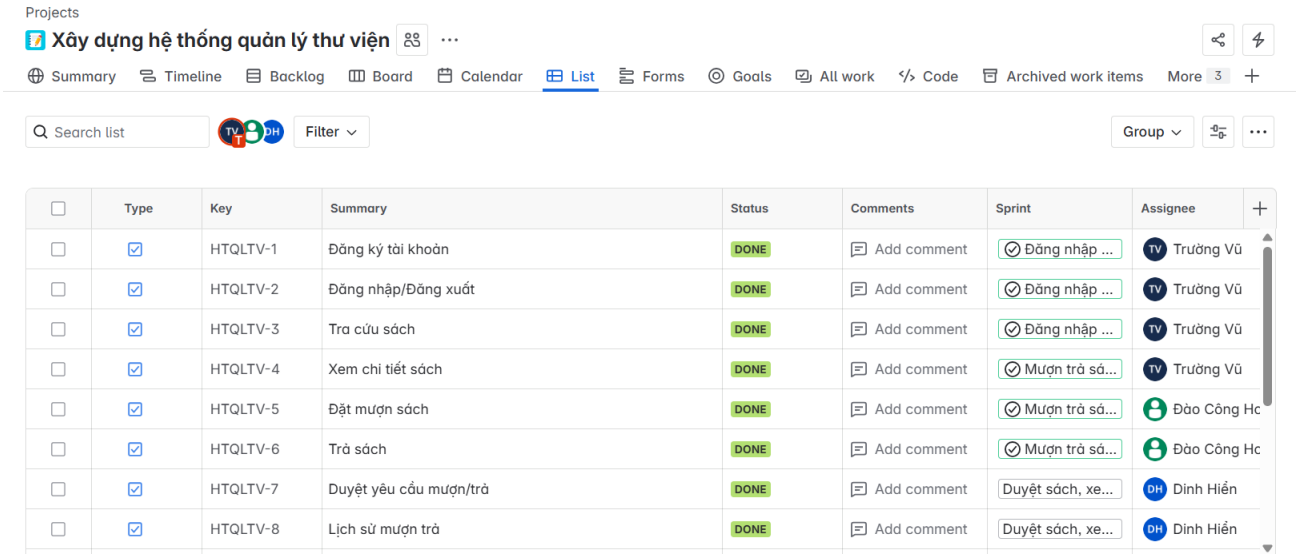
  Người thực hiện: Đào Công Hoàng Lam, Nguyễn Trường Vũ

  Thời gian thực hiện:18/7 – 22/7/2025

  Kết quả: Giao diện thân thiện với người dùng, dễ thao tác.



Hình 3.1 Sprint tồn đọng



Hình 3.2 Danh sách công việc

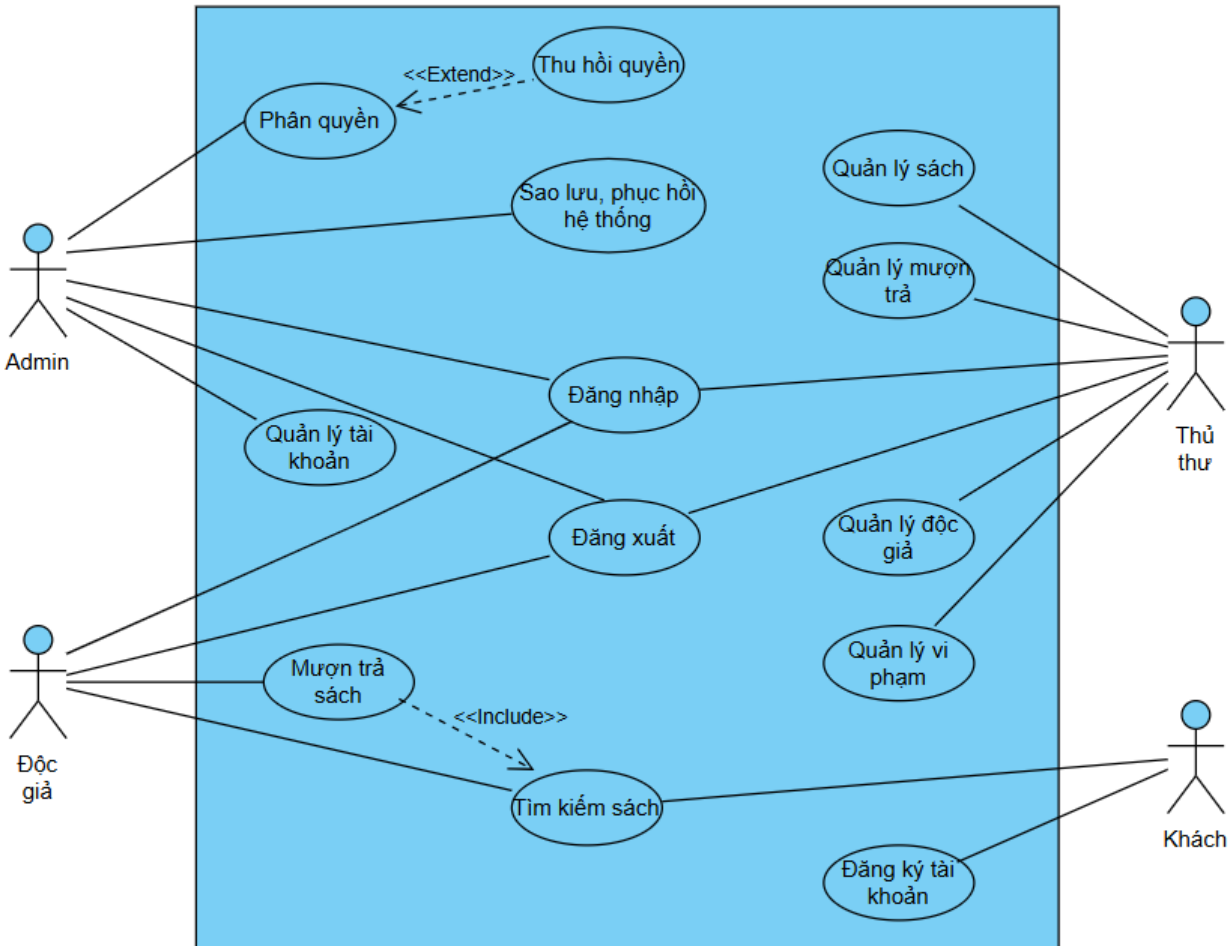
Sau khi lập kế hoạch , đưa các công việc vào các Sprint, ta bấm Start sprint để bắt đầu làm việc. Sau khi hoàn thành công việc , chọn từ "TO DO" sang "DONE" để hoàn thành công việc.

Tương tự với các công việc khác cho đến khi hoàn thành tất cả công việc có trong Sprint, sau đó các sprint đã hoàn thành công việc sẽ di chuyển khỏi Backlog.

Jira giúp quản lý công việc rõ ràng, chia nhỏ công việc thành các task cụ thể, theo dõi trạng thái của từng công việc. Lập kế hoạch sprint dễ dàng, hỗ trợ kế hoạch theo chu kỳ, tính toán khối lượng công việc bằng Story point giúp ước lượng chính xác. Hỗ trợ làm việc nhóm hiệu quả, nhiều thành viên có thể làm cùng trên 1 dự án, giao việc bình luận, trao đổi ngay trong từng task. Jira là một công cụ mạnh mẽ trong việc quản lý dự án.

3.2 Vẽ lược đồ Use Case của hệ thống

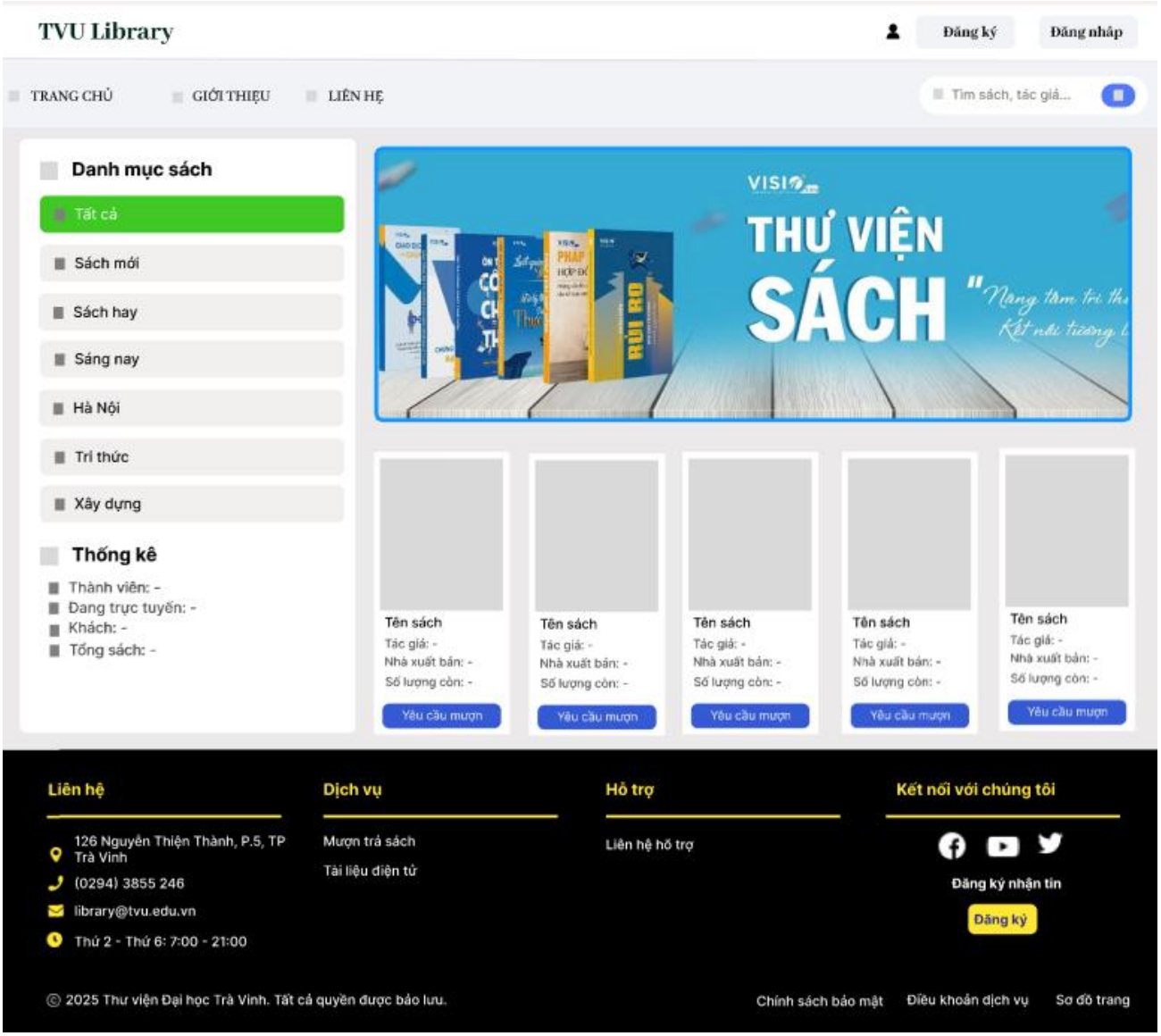
Lược đồ Use Case của hệ thống quản lý thư viện có các nhóm người dùng cơ bản như quản trị viên, thủ thư, độc giả, khách cùng tương tác với các Use Case của hệ thống .



- Actor Admin có toàn quyền với hệ thống
- Actor thử thư có quyền thao tác với các Use case như quản lý sách, quản lý mượn trả, quản lý độc giả, quản lý vi phạm. Actor thủ thư cần đăng nhập trước khi sử dụng hệ thống quản lý thư viện.
- Actor độc giả có quyền thao tác với các Use case mượn trả sách, tìm kiếm sách. Actor độc giả cần đăng nhập trước khi mượn trả sách của hệ thống.
- Actor khách có quyền tìm kiếm sách, nhưng không thể mượn sách. Có thể đăng ký tài khoản để trở thành độc giả để có các quyền của độc giả.

3.3 Thiết kế giao diện với Figma

3.3.1 Giao diện người dùng

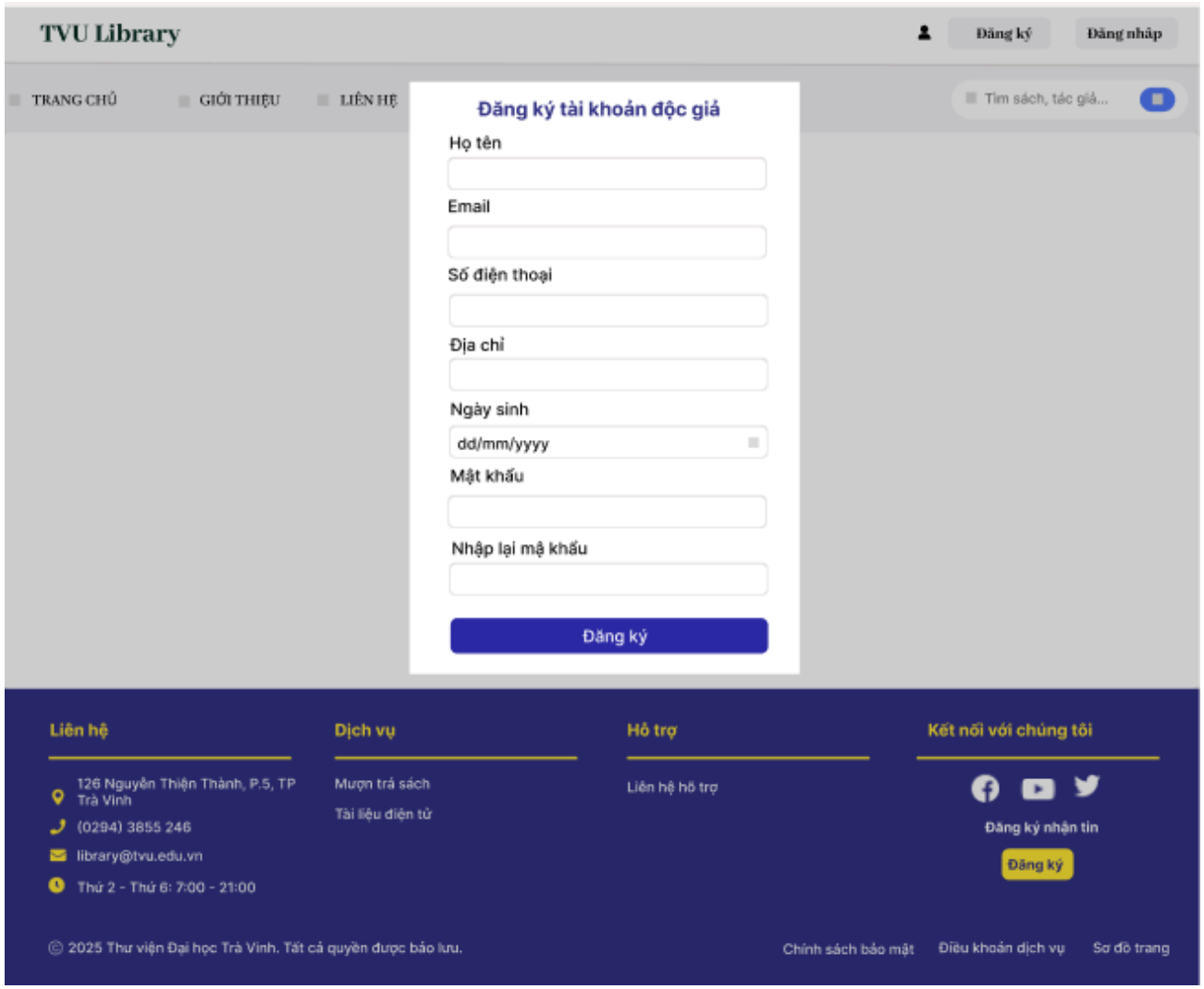


Hình 3.3 Giao diện trang chủ

Trang chủ hiển thị các thông tin cơ bản của thư viện, hiển thị menu, banner, các thông báo chính từ thư viện. Phần chân trang hiển thị các thông tin liên hệ, địa chỉ,...

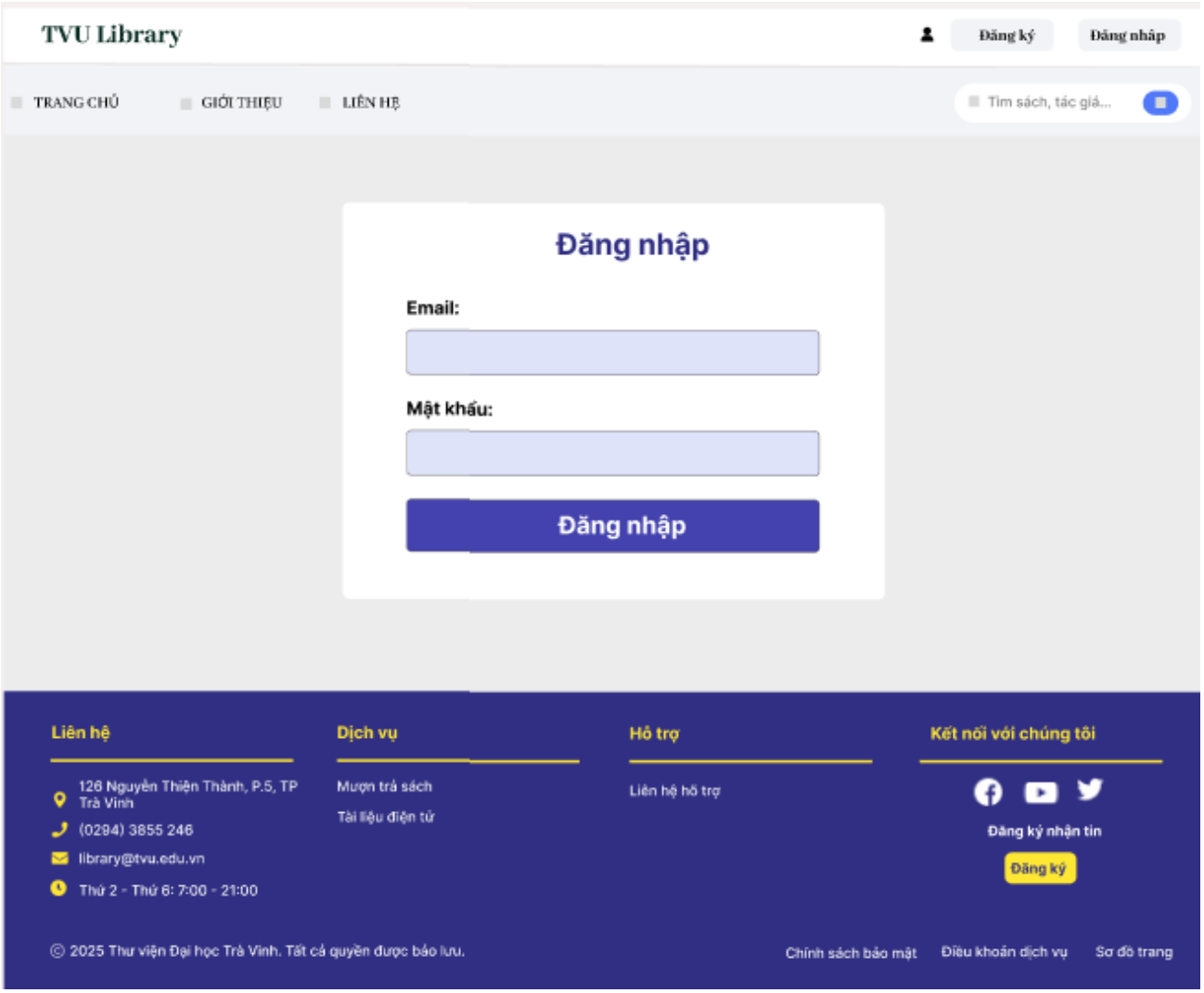
Trang chủ hiển thị danh mục sách có trong hệ thống, độc giả có thể yêu cầu mượn từ trang này.Trước khi mượn sách yêu cầu đăng nhập, nếu không có tài khoản người dùng có thể đăng ký với hệ thống.

3.3.2 Giao diện trang đăng nhập, đăng ký



Hình 3.4 Trang đăng ký tài khoản

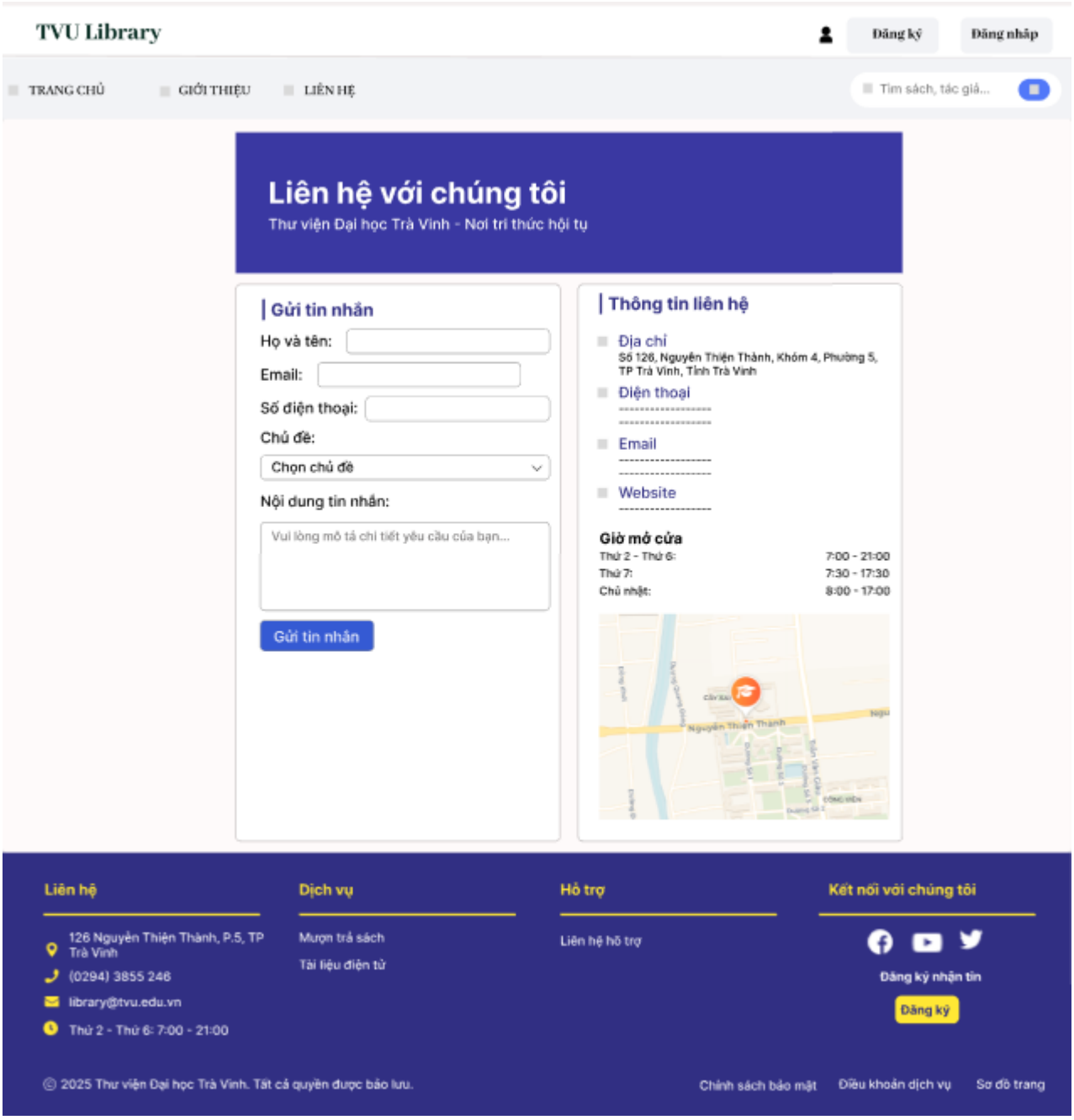
Người dùng phải khai báo các thông tin như: họ tên, email, số điện thoại, địa chỉ, ngày sinh, mật khẩu, và nhập lại mật khẩu, sau đó nhấn nút đăng ký để tạo tài khoản



Hình 3.5 Trang đăng nhập

Người nhập email và mật khẩu, sau đó nhấn nút đăng nhập.

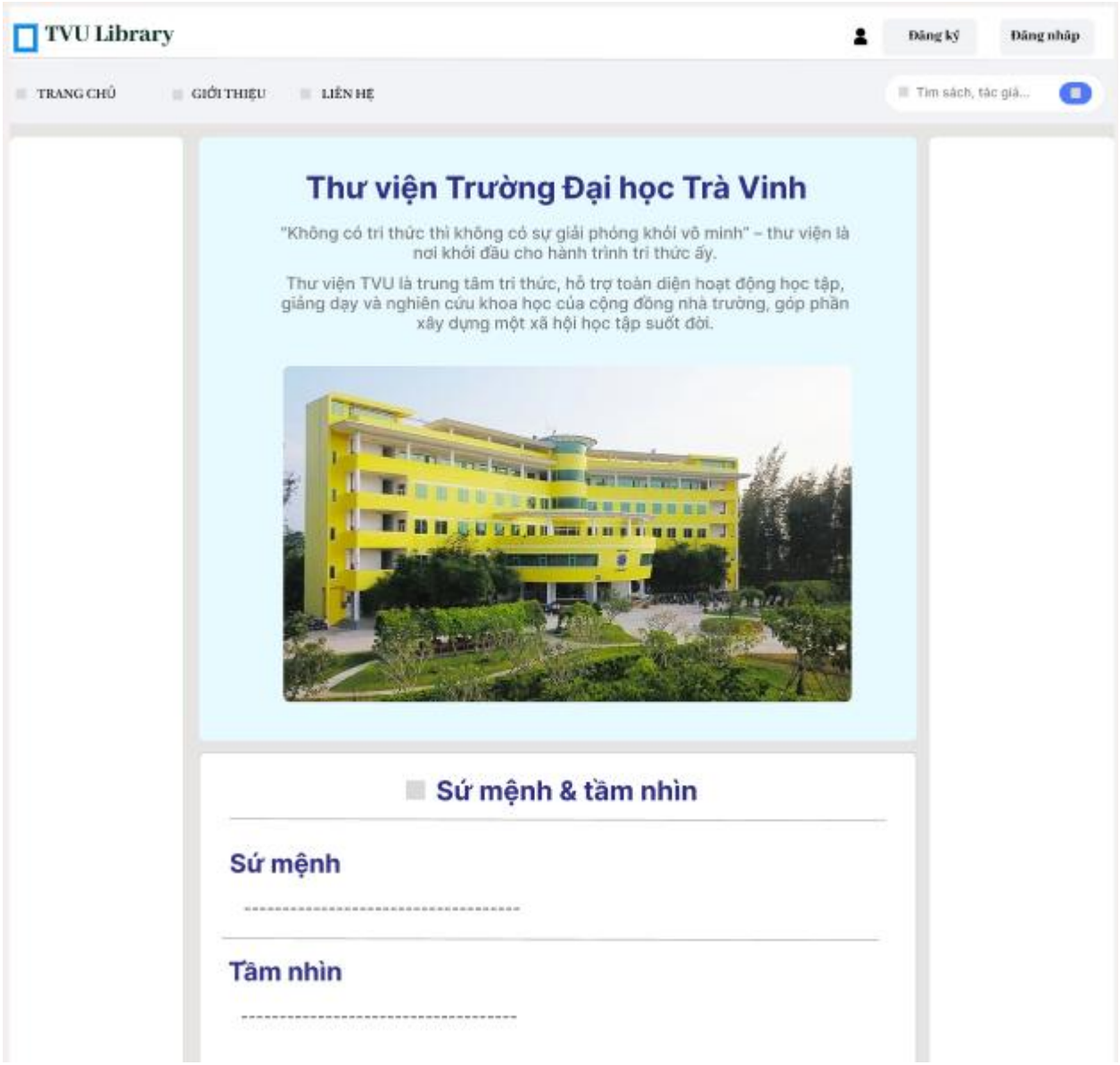
3.3.3 Giao diện trang liên hệ



Hình 3.6 Trang liên hệ

Giao diện trang Liên hệ được thiết kế rõ ràng, gồm hai phần chính: biểu mẫu gửi tin nhắn và thông tin liên hệ của Thư viện Đại học Trà Vinh. Người dùng có thể dễ dàng nhập họ tên, email, nội dung liên hệ và gửi đi. Bên cạnh đó là thông tin địa chỉ, giờ mở cửa và bản đồ tích hợp giúp tra cứu thuận tiện.

3.3.4 Giao diện trang giới thiệu

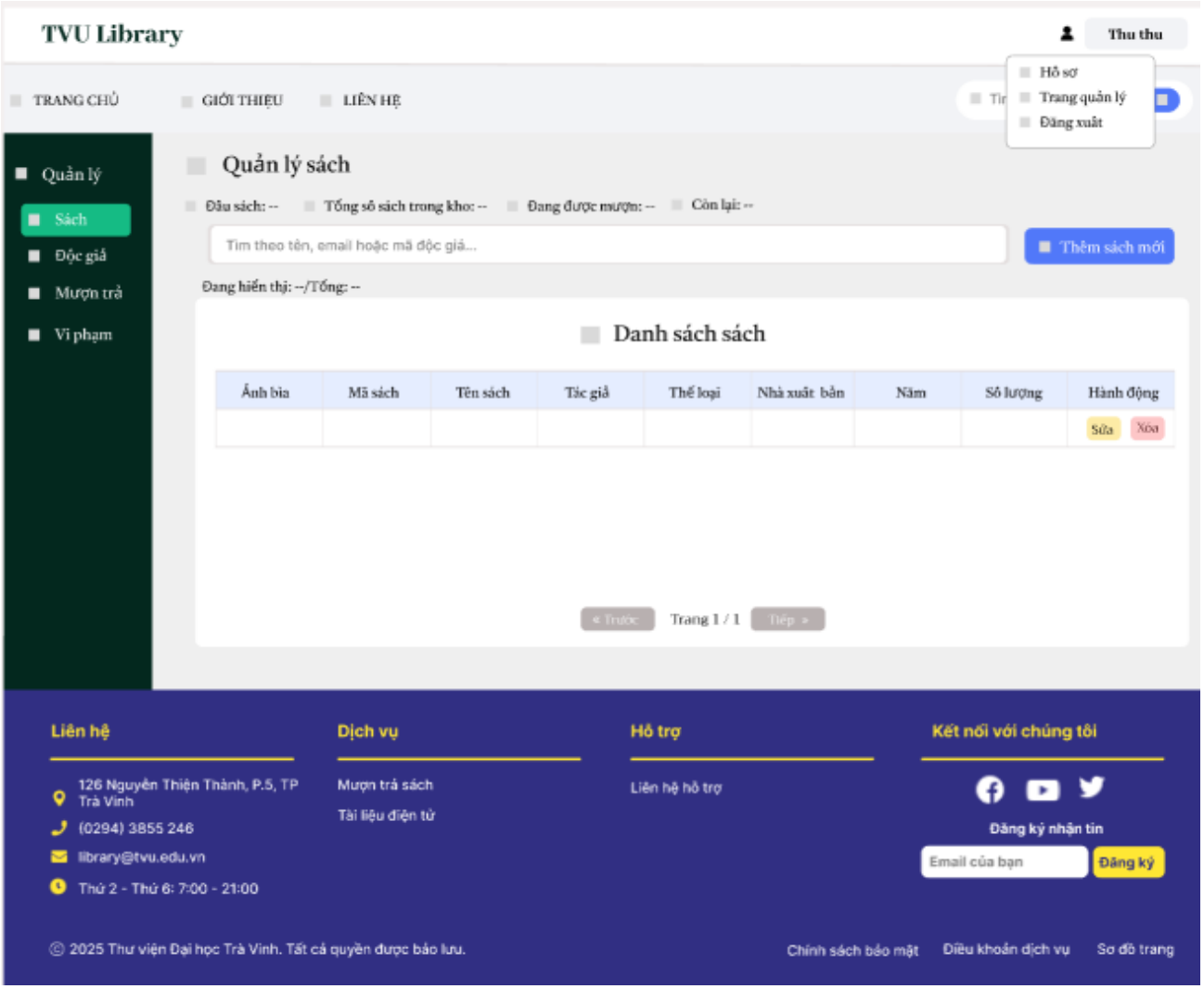


Hình 3.7 Trang giới thiệu

Trang giới thiệu rõ nét về thư viện giúp người dùng có cái nhìn rõ nét về thư viện.

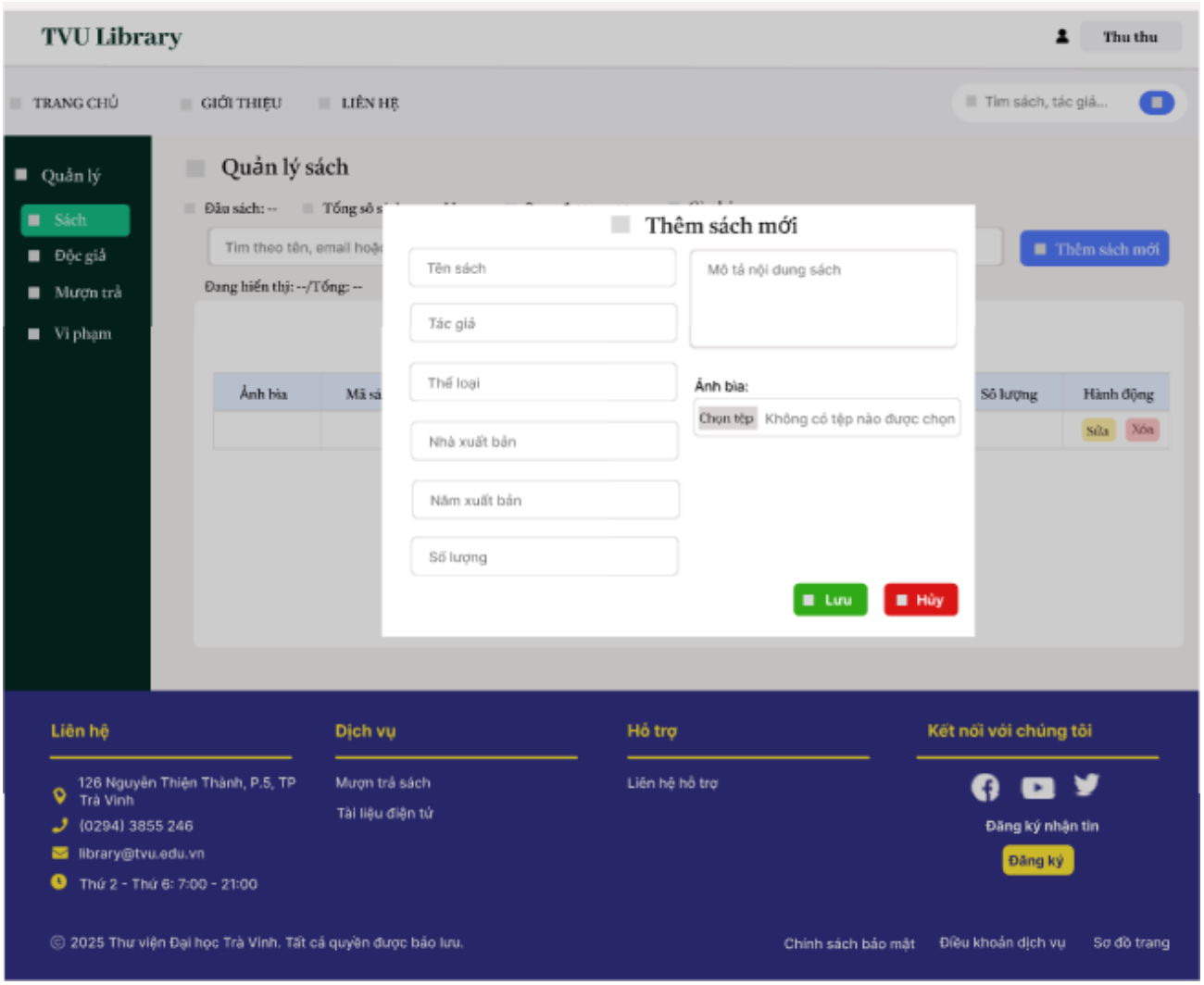
3.3.5 Giao diện các trang quản lý

Trang quản lý có các giao diện như quản lý sách, quản lý độc giả, quản lý mượn trả, quản lý vi phạm, được trình bày dưới dạng bảng, người dùng thủ thư có thể thực hiện các thao tác thêm, xóa, sửa.



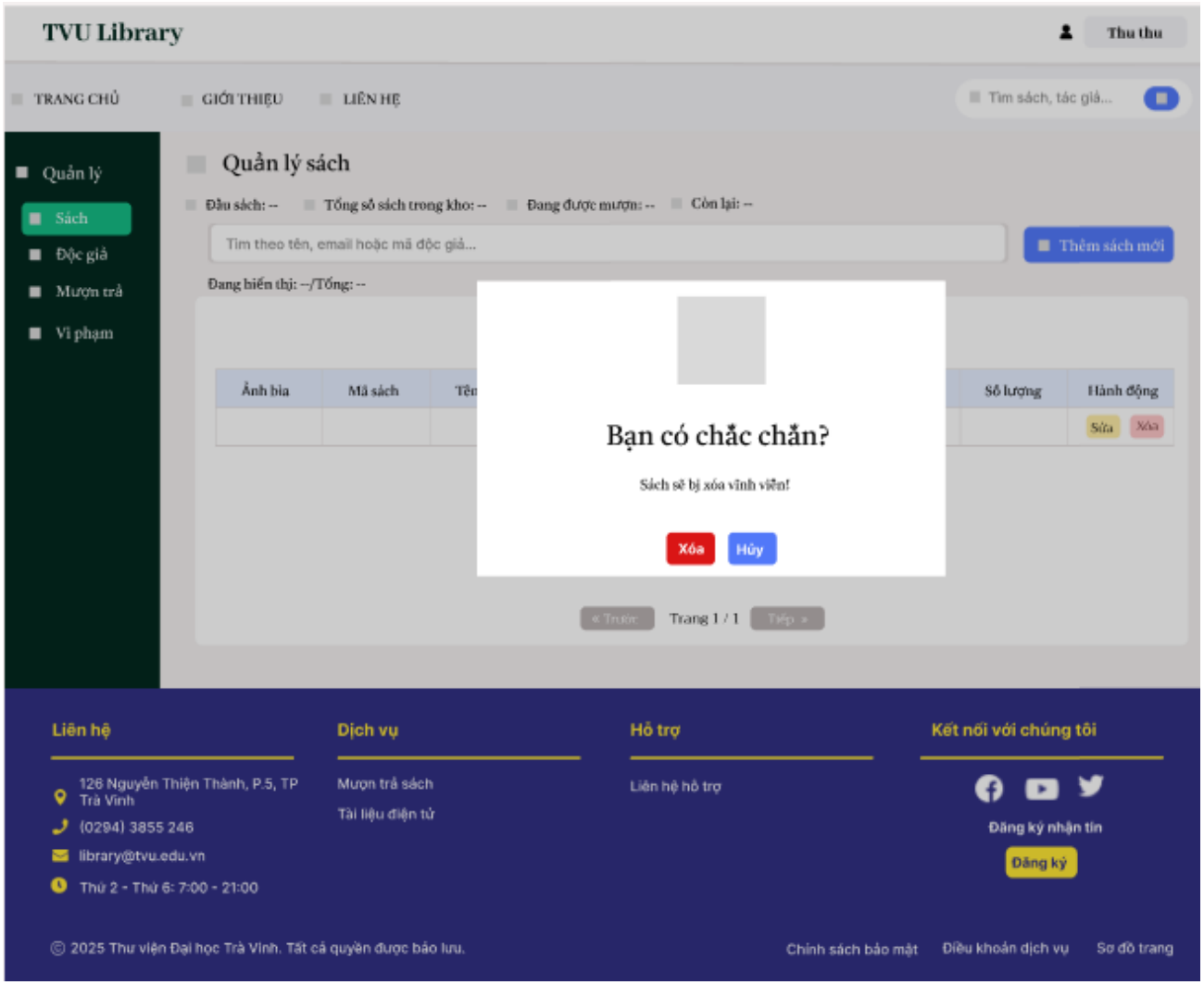
Hình 3.8 Trang quản lý sách

Trang quản lý sách quản lý các thông tin sách như ảnh bìa, mã sách, tên sách, tác giả, thể loại, nhà xuất bản, năm, số lượng, và có thể thêm , xóa , sửa thông tin sách.



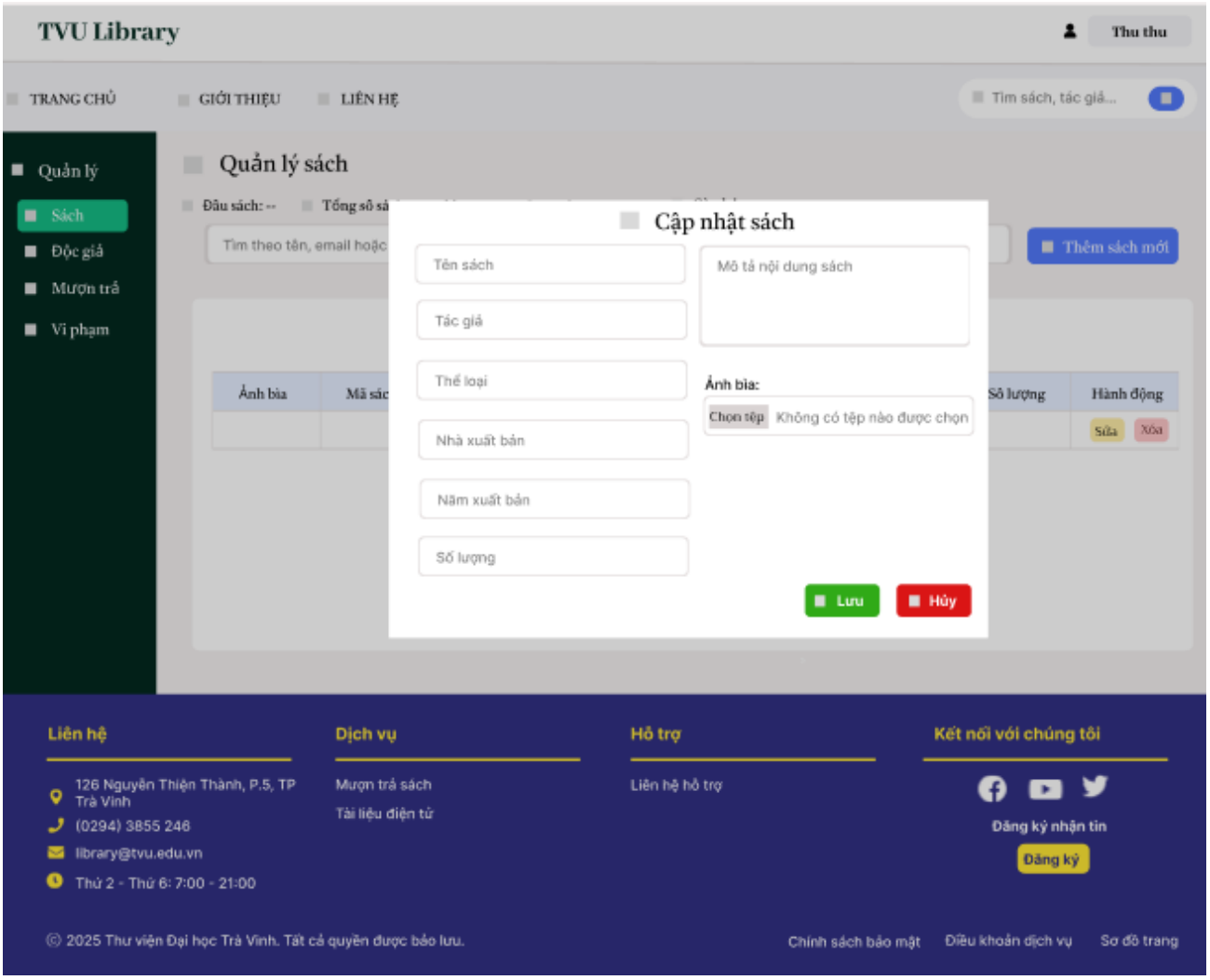
Hình 3.9 Giao diện thêm mới sách

Trang thêm mới sách sau khi người dùng nhập các thông tin sách như ảnh bìa, mã sách, tên sách, tác giả, thể loại, nhà xuất bản, năm, số lượng, mô tả sách nếu các thông tin được nhập đúng thì khi nhấn nút lưu sách mới sẽ được lưu còn khi nhấn nút hủy sẽ loại bỏ thao tác thêm.



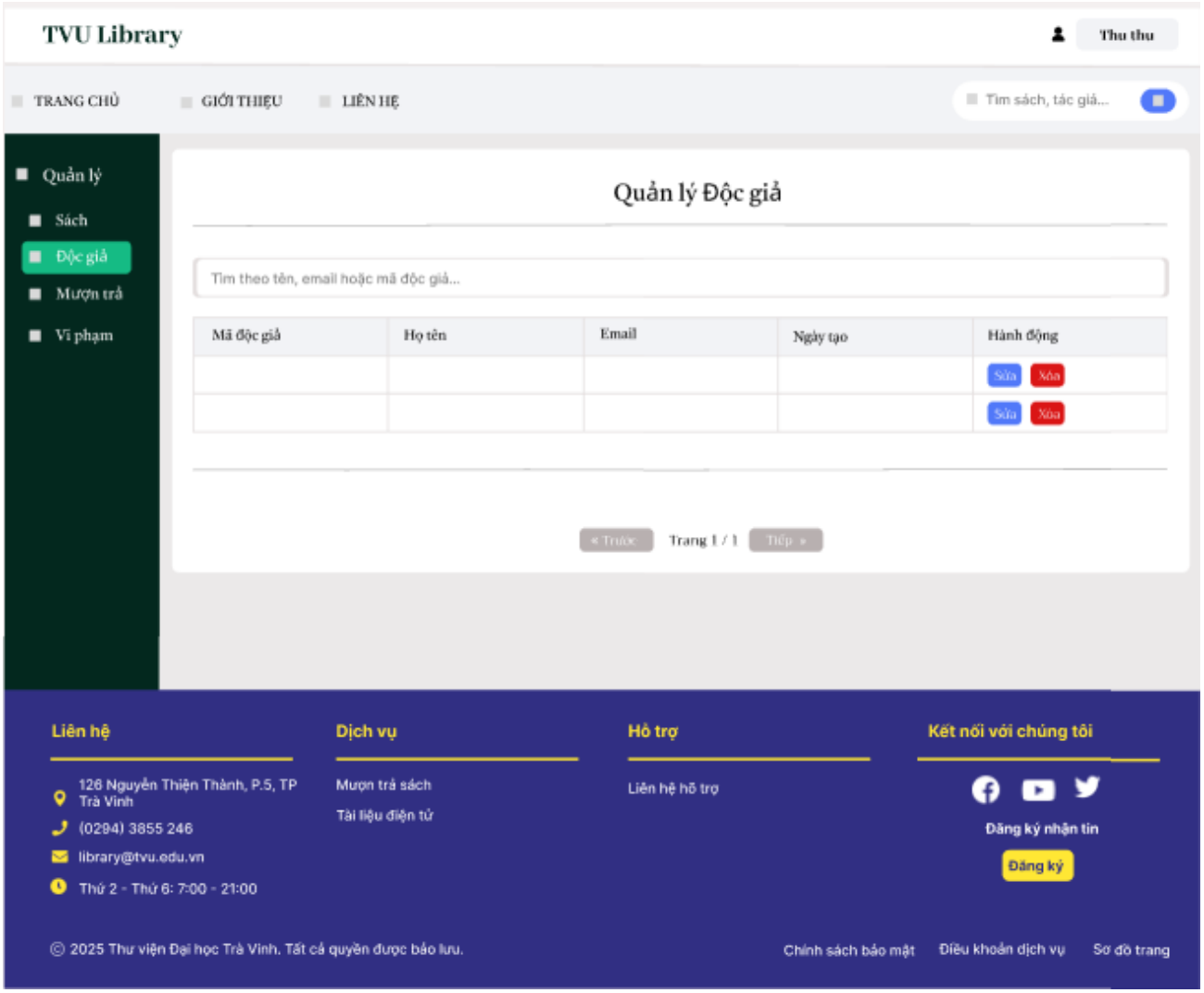
Hình 3.10 Giao diện xóa sách

Người dùng xác nhận một lần nữa có muốn chắc chắn muốn xóa sách hay không.



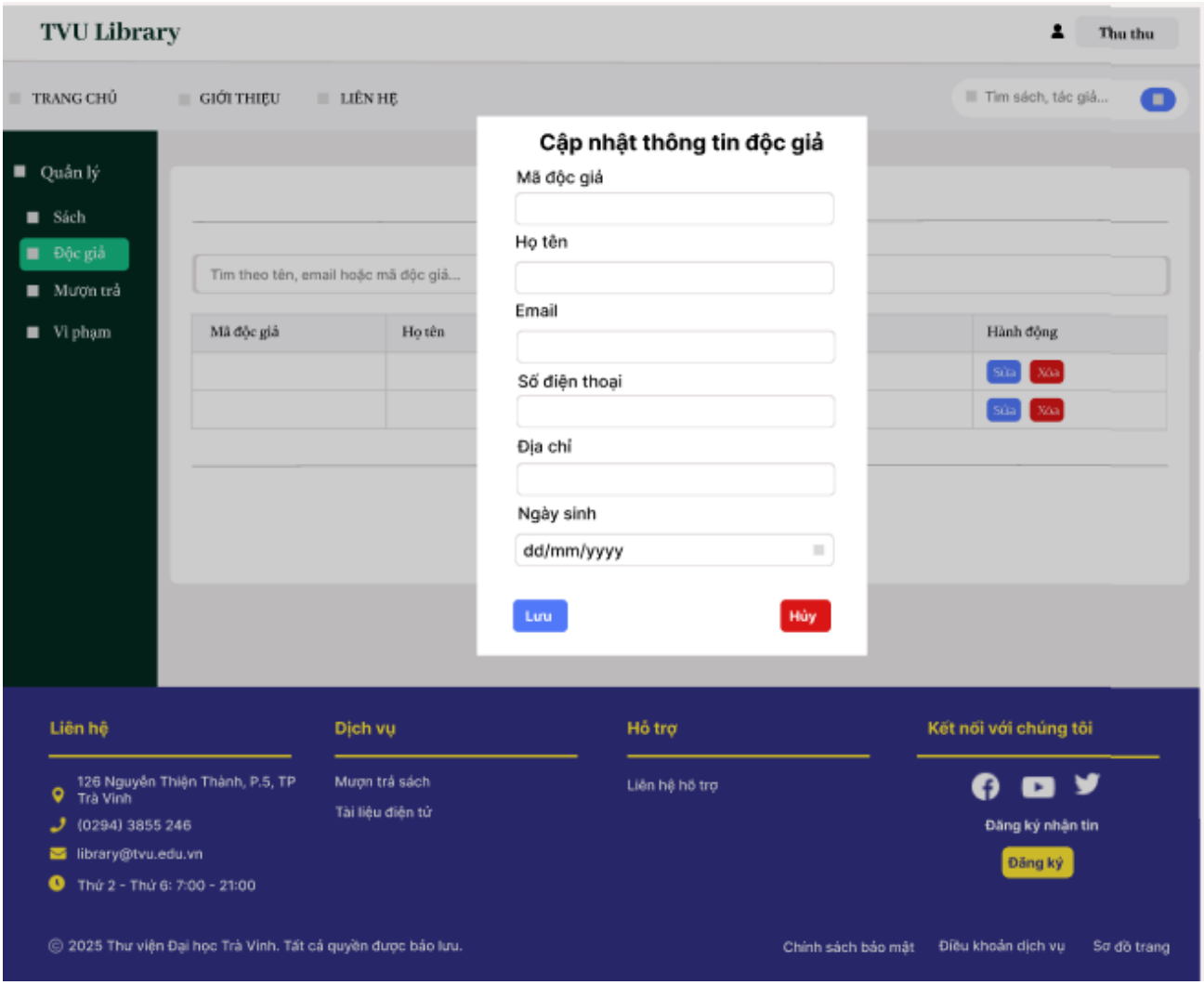
Hình 3.11 Giao diện cập nhật sách

Giao diện sửa đổi các thông tin của sách khi bị nhập sai hoặc có cập nhật mới về số lượng hay ảnh bìa,...



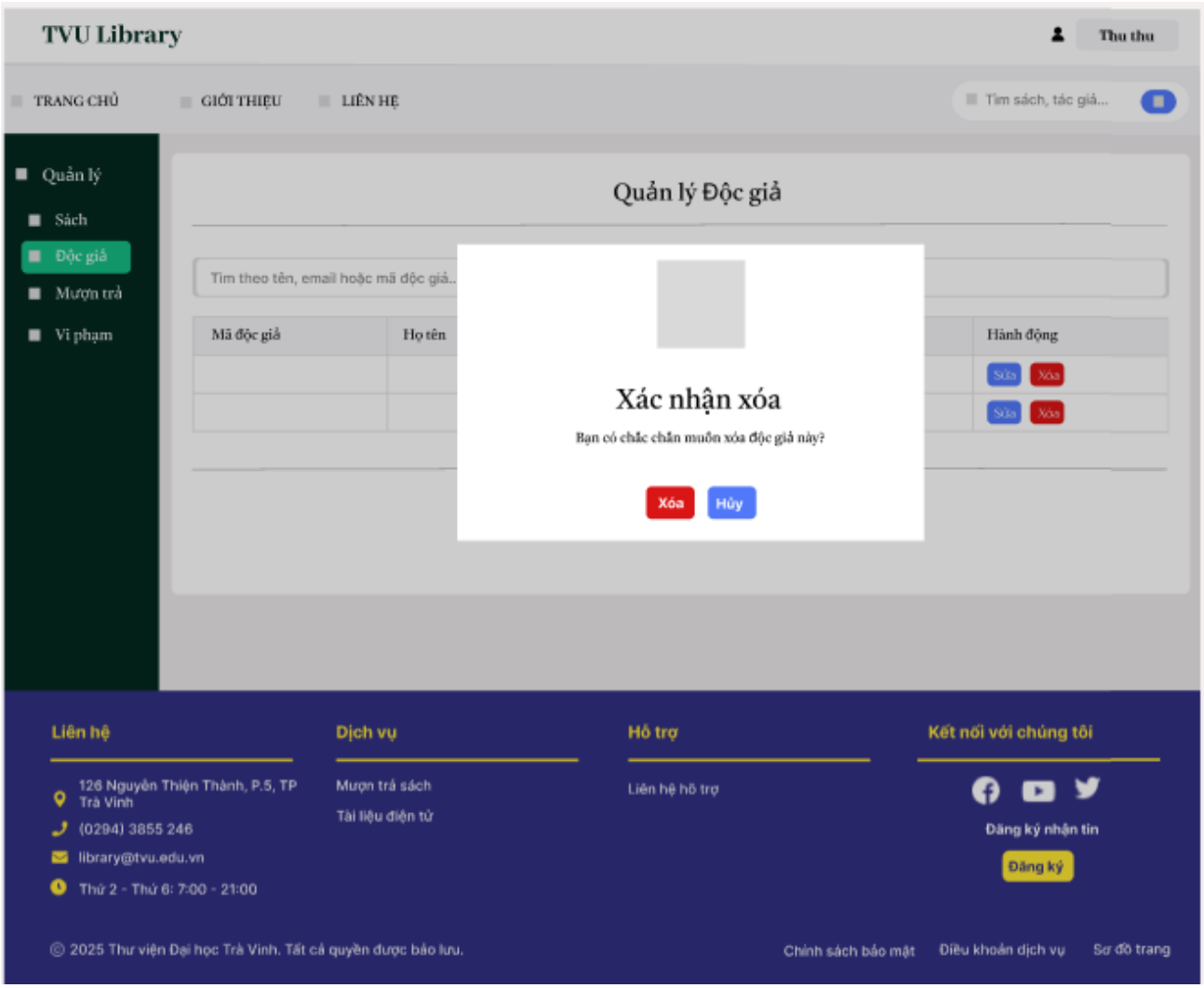
Hình 3.12 Trang quản lý độc giả

Trang quản lý độc giả quản lý các thông tin độc giả như mã độc giã, họ tên, email, ngày tạo, thủ thư có thể thêm xóa sửa các thông tin của độc giả.



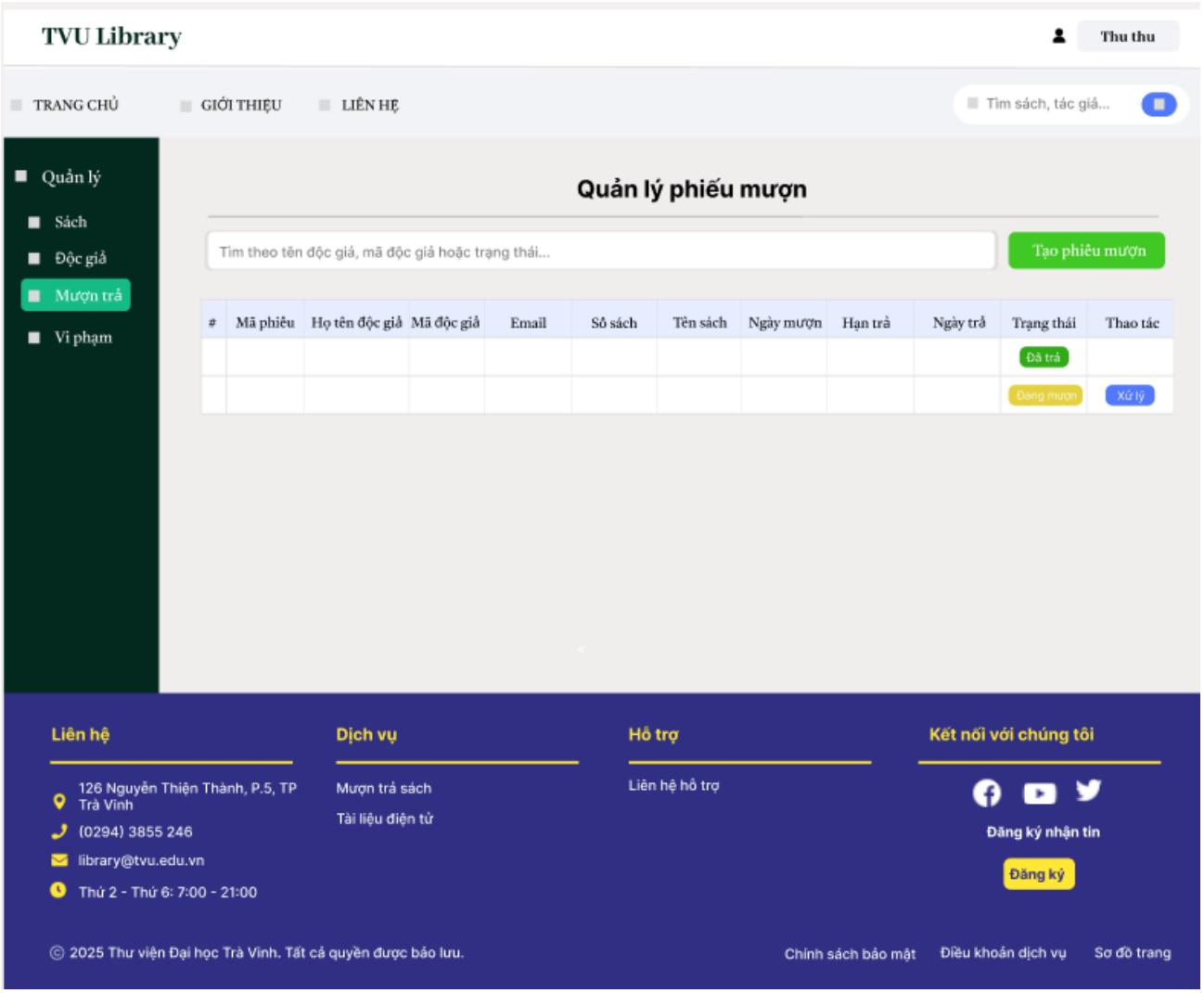
Hình 3.13 Giao diện cập nhật độc giả

Giao diện sửa đổi các thông tin của độc giả khi bị nhập sai hoặc có cập nhật mới về thông tin độc giả.



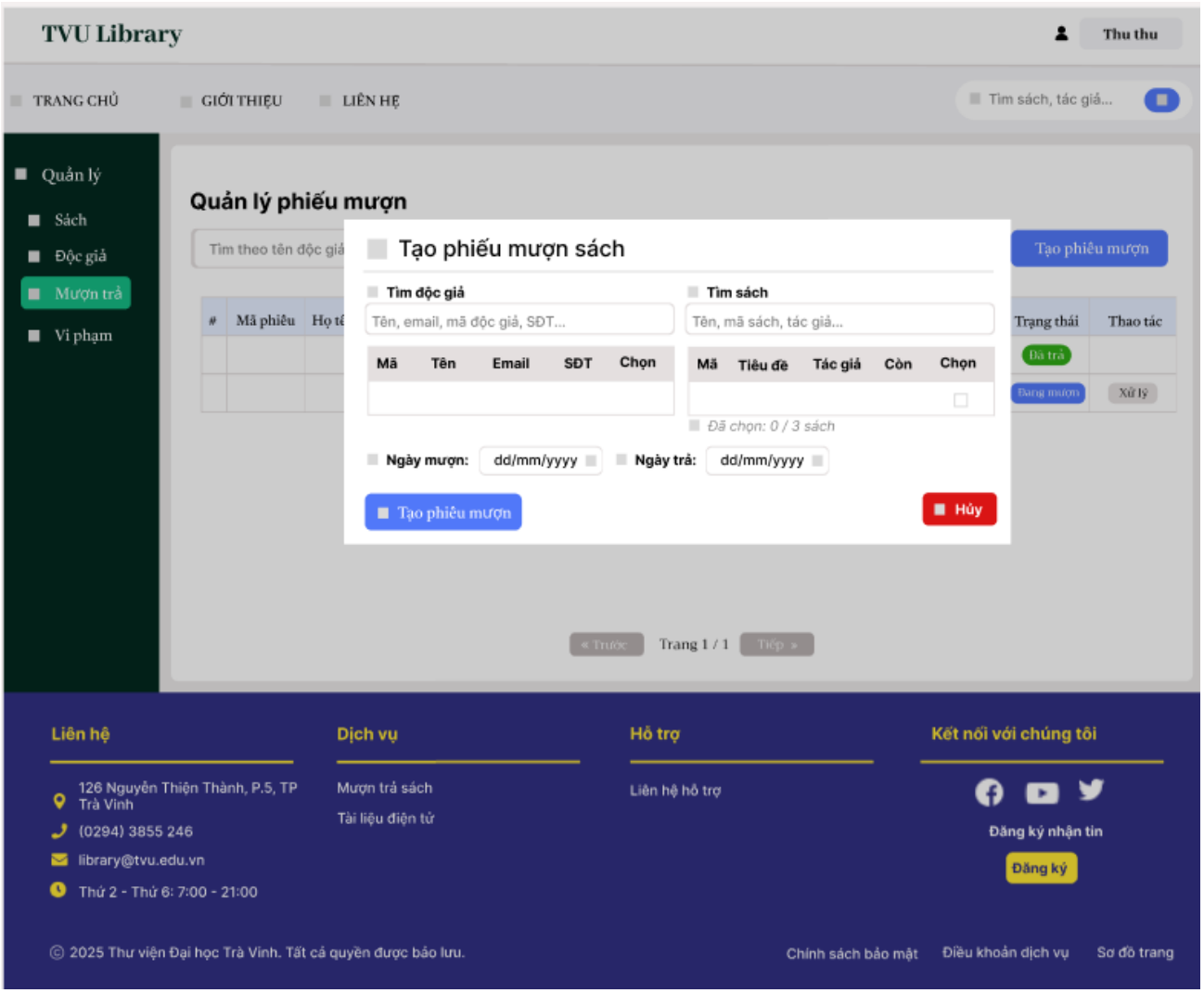
Hình 3.14 Giao diện xóa độc giả

Xác nhận chắc chắn xóa vĩnh viễn độc giả khỏi danh sách.



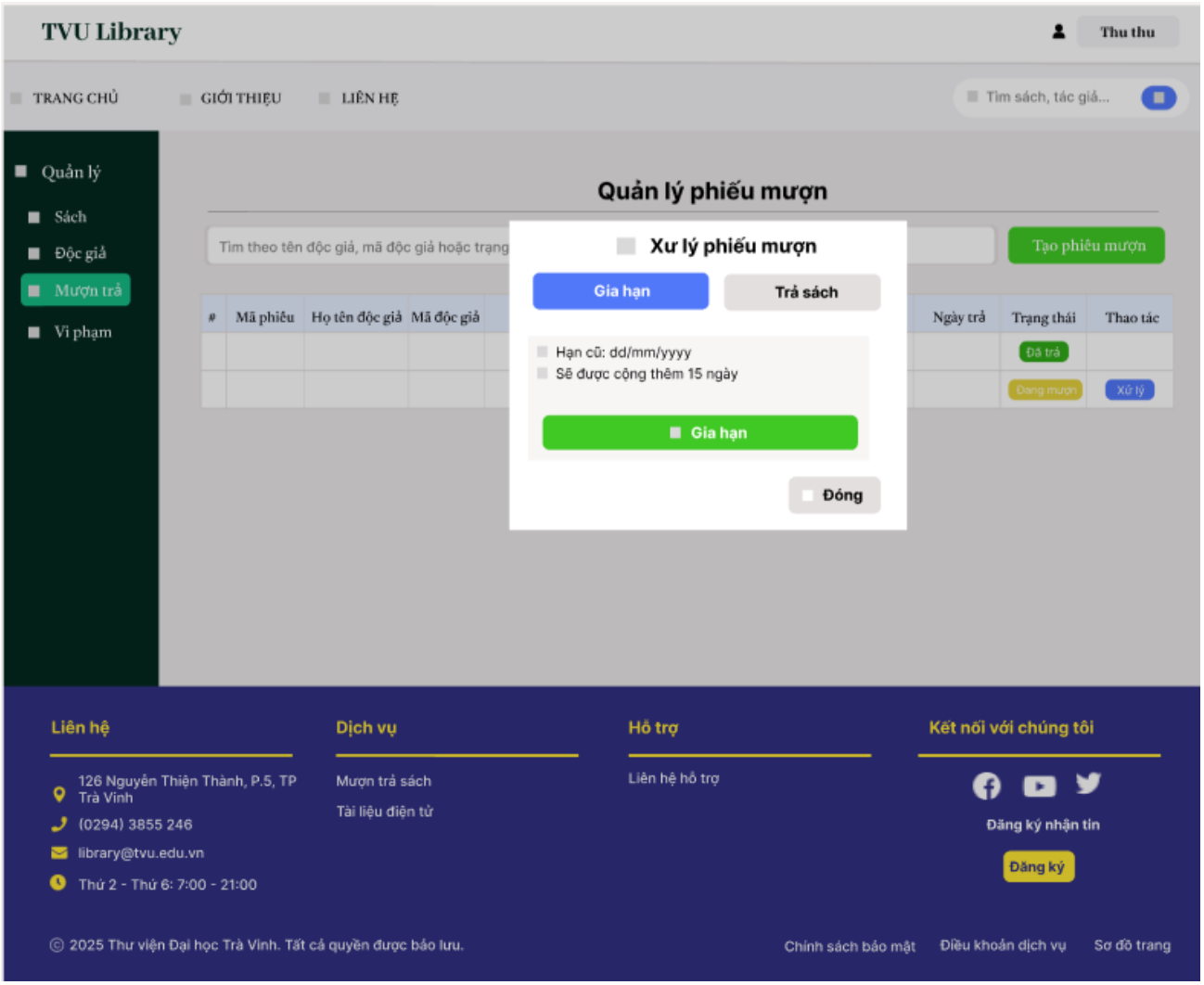
Hình 3.15 Trang quản lý phiếu mượn

Trang quản lý phiếu mượn quản lý các thông tin như mã phiếu mượn, họ tên độc giả, mã độc giả, email, số sách, tên sách, ngày mượn, hạn trả, ngày trả, trạng thái,.. thủ thư có thể thêm xóa sửa phiếu mượn

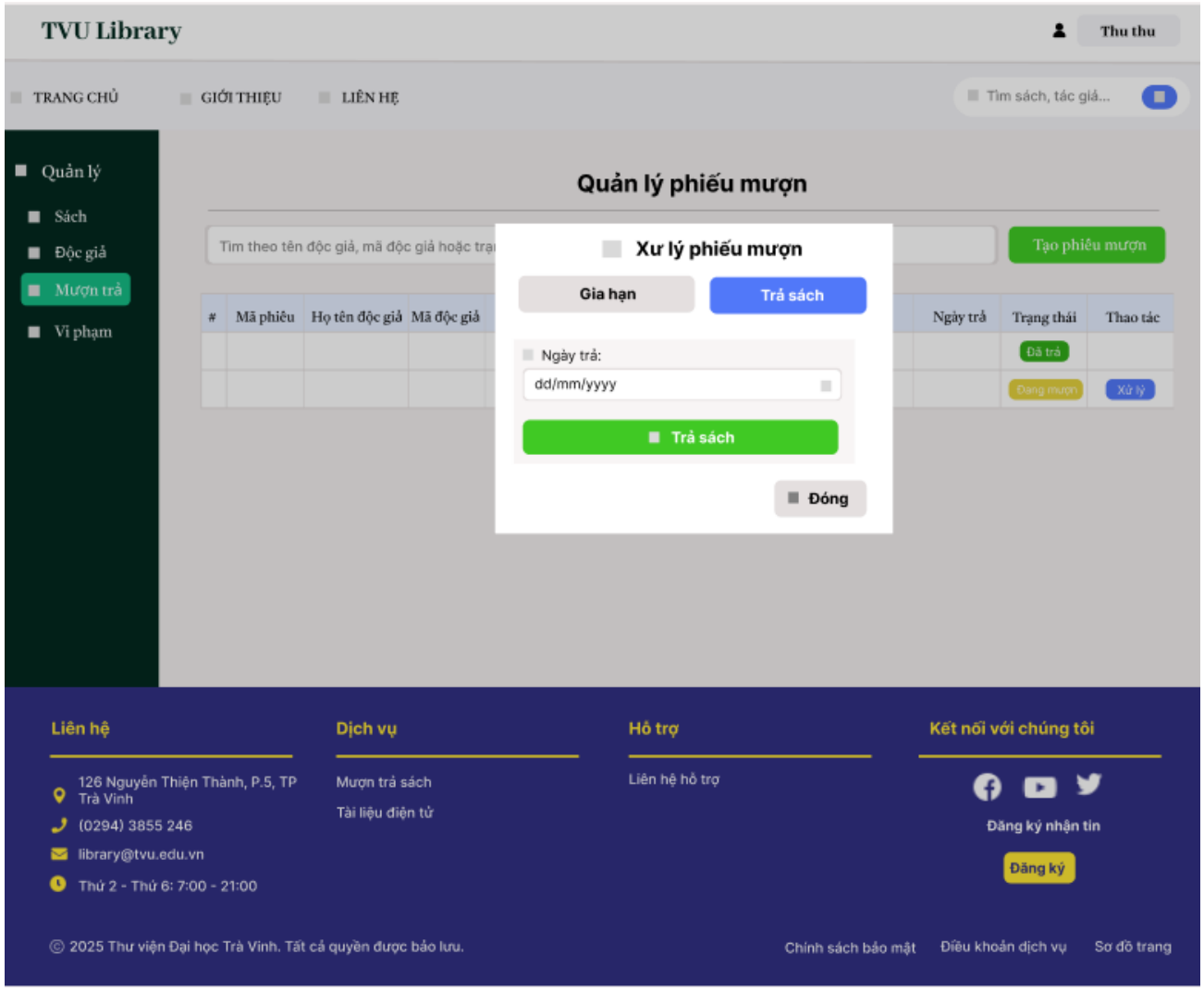


Hình 3.16 Giao diện tạo phiếu mượn

Giao diện quản lý phiếu mượn sách cho phép thủ thư dễ dàng tìm kiếm độc giả và sách, sau đó tạo phiếu mượn với thông tin ngày mượn – ngày trả. Giao diện hiển thị rõ ràng danh sách người đọc, sách được chọn và hỗ trợ thao tác nhanh với nút xác nhận hoặc hủy bỏ.

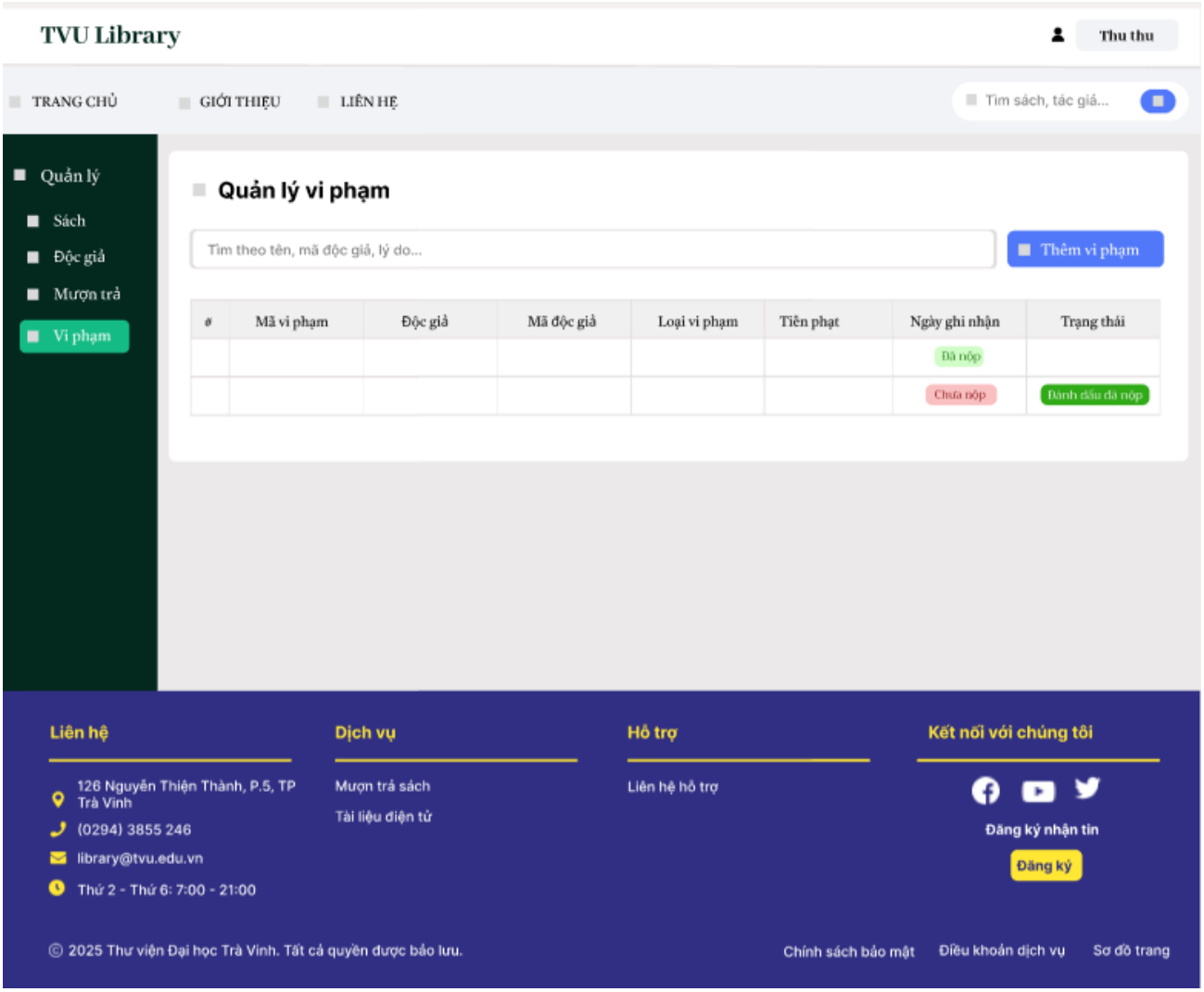


Hình 3.17 Giao diện gia hạn phiếu mượn



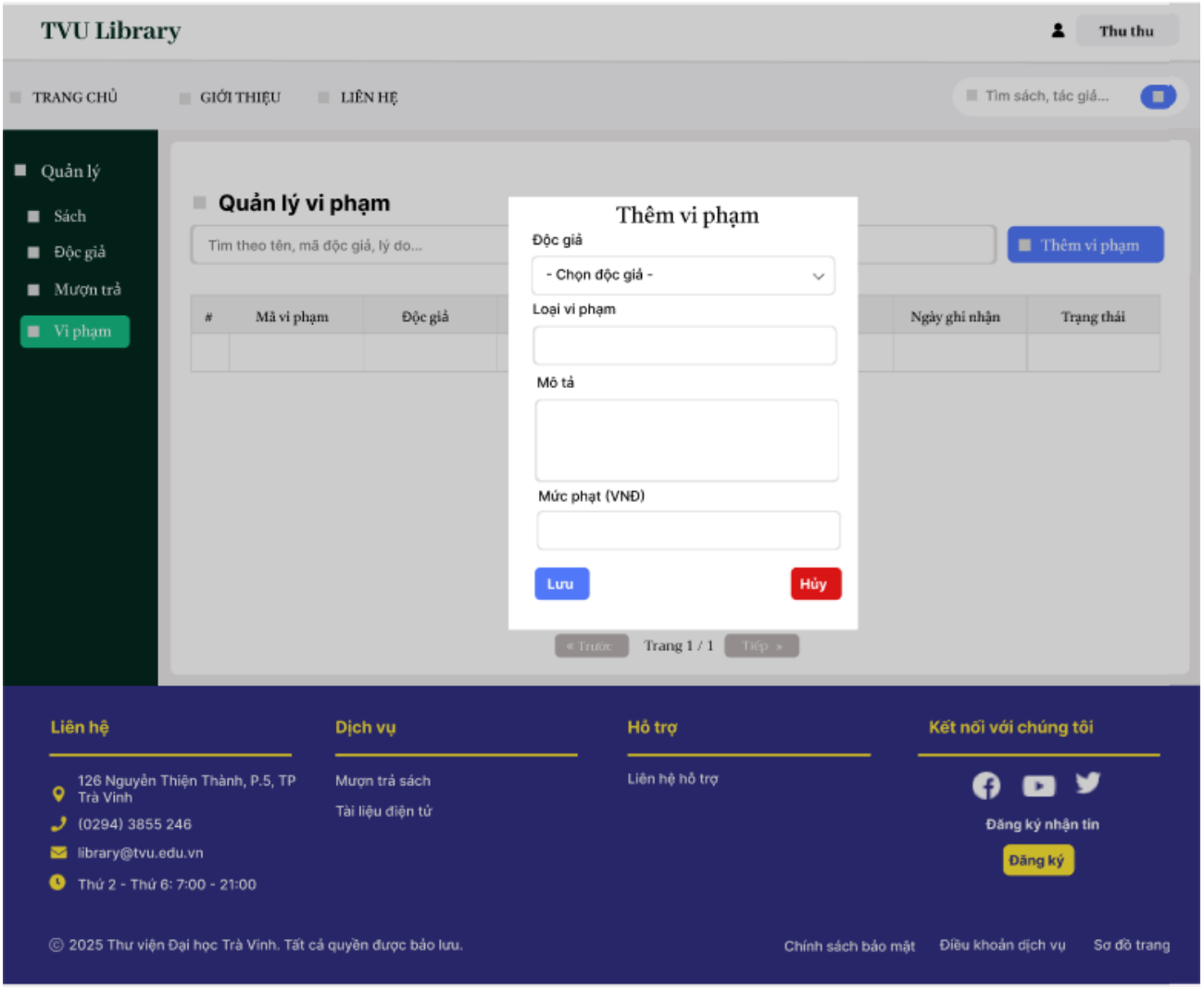
Hình 3.18 Giao diện xử lý phiếu mượn trả sách

Giao diện xử lý phiếu mượn hỗ trợ thủ thư thực hiện thao tác gia hạn hoặc trả sách nhanh chóng. Khi chọn gia hạn, hệ thống sẽ tự động cộng thêm 15 ngày vào hạn cũ. Giao diện đơn giản, dễ sử dụng, giúp quản lý mượn trả hiệu quả hơn.



Hình 3.19 Trang quản lý vi phạm

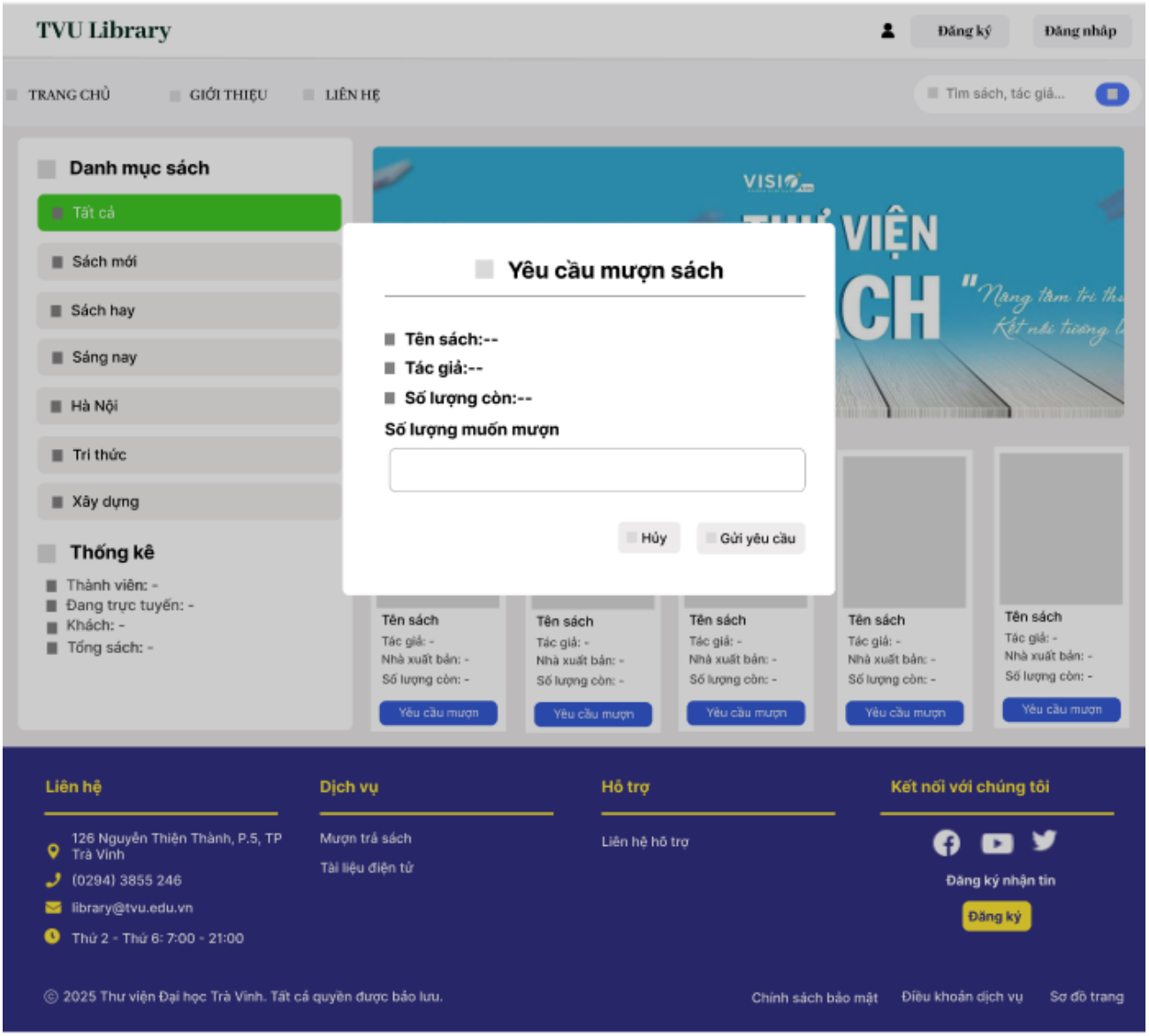
Trang quản lý vi phạm quản lý các thông tin như mã vi phạm, độc giả, mã độc giả, loại vi phạm, tiền phạt, ngày ghi nhận, trạng thái,... thủ thư có thể thêm xóa sửa vi phạm.



Hình 3.20 Giao diện thêm vi phạm

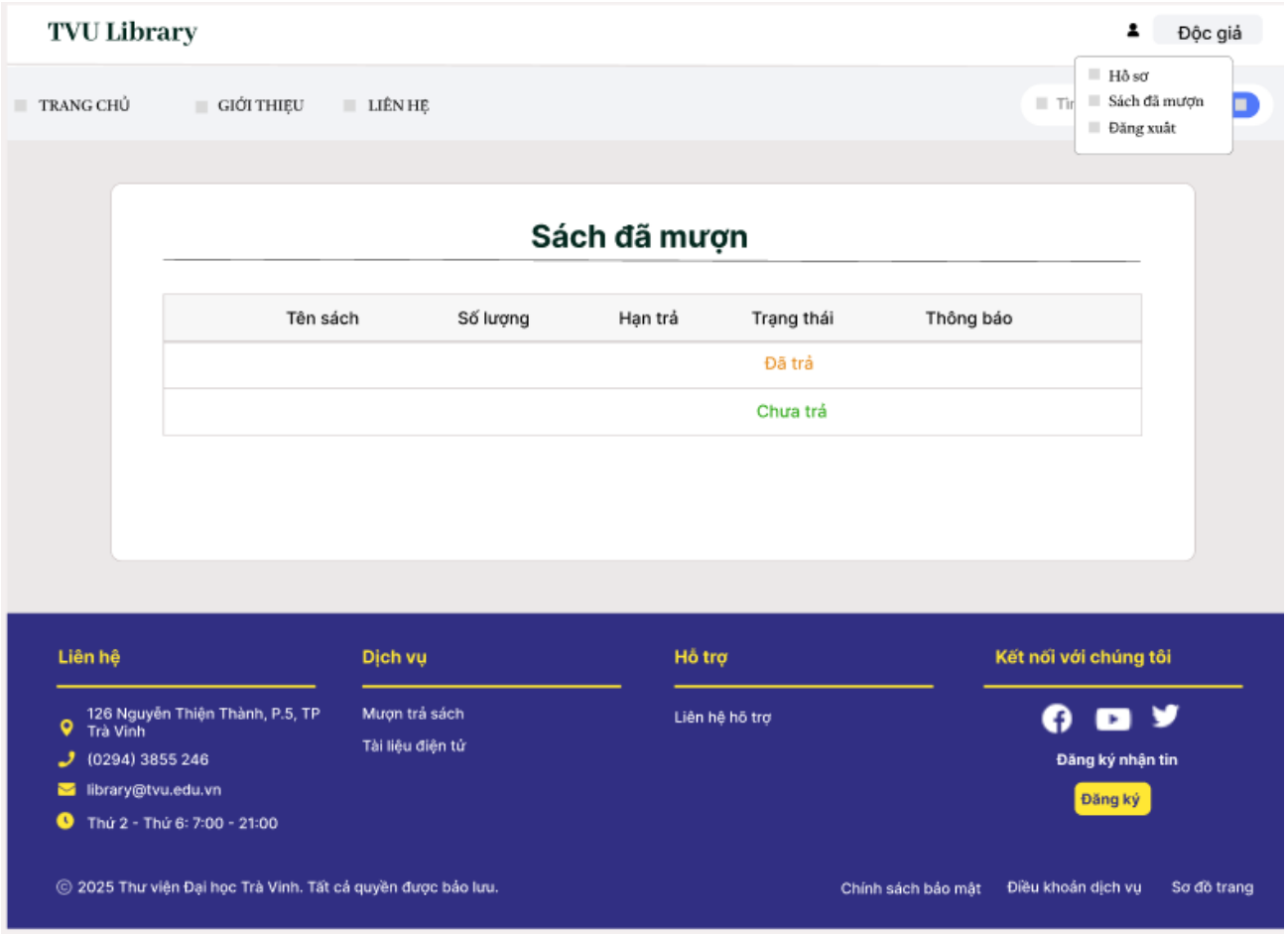
Giao diện quản lý vi phạm cho phép thủ thư ghi nhận các trường hợp vi phạm của độc giả. Biểu mẫu gồm các trường như: chọn độc giả, loại vi phạm, mô tả chi tiết và mức phạt (VNĐ). Giao diện đơn giản, dễ thao tác giúp việc quản lý vi phạm minh bạch và thuận tiện hơn.

3.3.6 Các giao diện mượn sách



Hình 3.21 Giao diện yêu cầu mượn sách

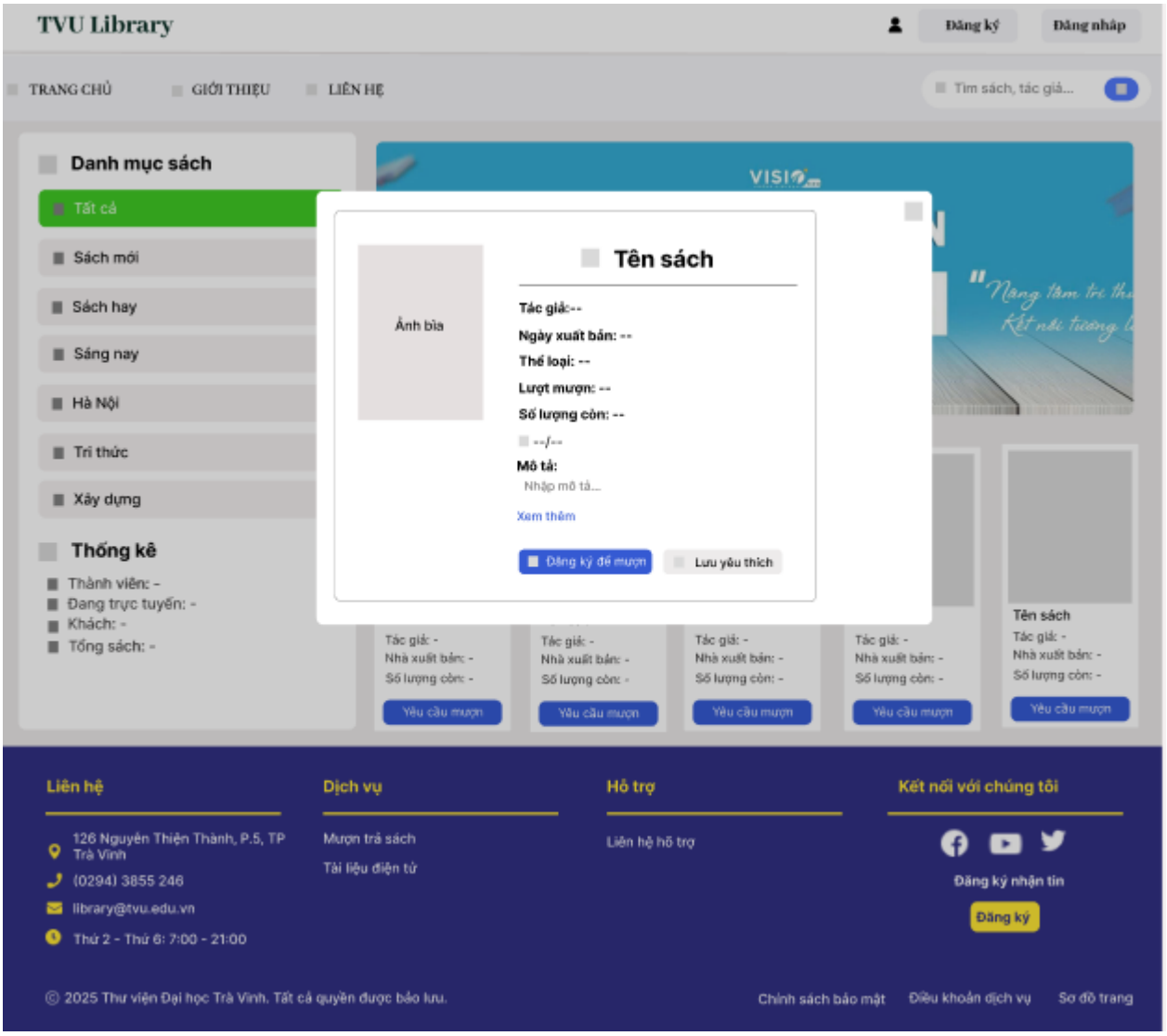
Giao diện yêu cầu mượn sách cho phép người dùng lựa chọn sách từ danh mục và gửi yêu cầu mượn. Thông tin hiển thị gồm tên sách, tác giả, số lượng còn lại và số lượng muốn mượn. Giao diện đơn giản, hỗ trợ thao tác nhanh và dễ sử dụng cho độc giả.



Hình 3.22 Giao diện sách đã mượn

Giao diện sách đã mượn giúp độc giả theo dõi danh sách các sách mình đã mượn, bao gồm tên sách, số lượng, hạn trả, trạng thái (đã trả/chưa trả) và thông báo kèm theo. Giao diện trực quan, hỗ trợ người dùng quản lý lịch mượn sách hiệu quả hơn.

3.3.7 Giao diện chi tiết sách



Hình 3.23 Giao diện chi tiết sách

Giao diện cung cấp một cái nhìn tổng quát về sách như tác giả, thể loại, ngày xuất bản,…giúp người dùng dễ dàng tìm sách phù hợp.

CHƯƠNG 4 XÂY DỰNG VÀ TRIỂN KHAI HỆ THỐNG

Hệ thống quản lý thư viện được xây dựng nhằm hỗ trợ việc tổ chức và quản lý sách, người dùng, phiếu mượn, và các hoạt động mượn trả trong thư viện một cách hiệu quả. Giao diện người dùng được phát triển bằng Reactjs, mang lại trải nghiệm mượt mà, hiện đại và thân thiện với người sử dụng. Phía backend sử dụng Nodejs kết hợp với Expressjs để xử lý logic nghiệp vụ và xây dựng các API RESTful. Dữ liệu được lưu trữ trong cơ sở dữ liệu MongoDB, phù hợp với cấu trúc dữ liệu linh hoạt và dễ mở rộng. Toàn bộ hệ thống hoạt động dựa trên mô hình client-server, trong đó frontend và backend giao tiếp với nhau thông qua các API HTTP, đảm bảo tính phân tách rõ ràng và dễ bảo trì.

Hệ thống quản lý thư viện được thiết theo mô hình 3 lớp rõ ràng, dựa trên kiến trúc client-server

- Tầng giao diện: có chức năng hiển thị giao diện người dùng, thu thập input , gửi yêu cầu đến server, sử dụng framework React
- Tầng xử lý nghiệp vụ: xử lý logic, bảo mật, xác thực điều phối dữ liệu, gọi cơ sở dữ liệu, sử dụng framework Node.js
- Tầng truy cập cơ sở dữ liệu: kết nối, truy vấn, lưu trữ dữ liệu trong cơ sở dữ liệu, sử dụng MongoDB

4.1 Frontend

Phần giao diện người dùng (Frontend) được xây dựng bằng React.js, một thư viện JavaScript phổ biến dùng để xây dựng giao diện động và hiện đại, sử dụng React Router để điều hướng trang

Giao diện sẽ lấy dữ liệu bằng cách gửi các yêu cầu HTTP đến server thông qua phương thức GET , hiển thị lên danh mục sách, thông tin người dùng,...

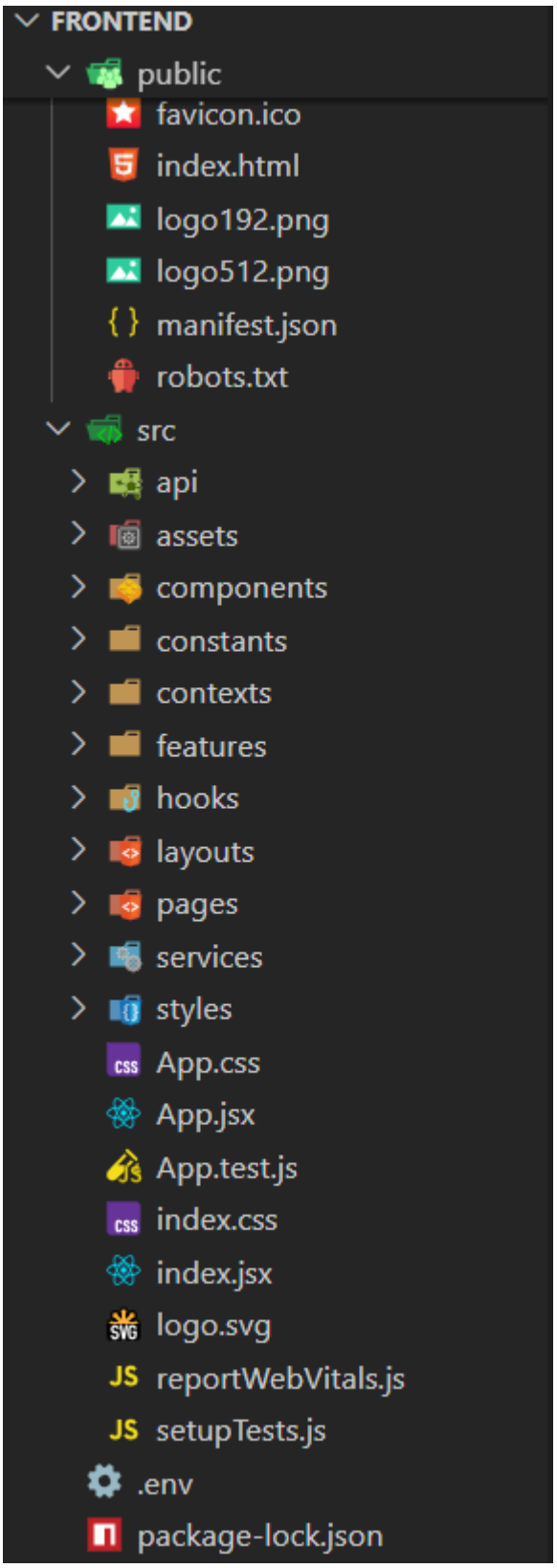
Cấu trúc thư mục Frontend:

Thư mục public/:

- index.html: File HTML chính, nơi React sẽ inject vào div#root.
- manifest.json File cấu hình để biến ứng dụng thành Progressive Web App (PWA).
- favicon.ico: Icon hiển thị trên tab trình duyệt.
- logo192.png, logo512.png Logo ứng dụng với các kích thước khác nhau.

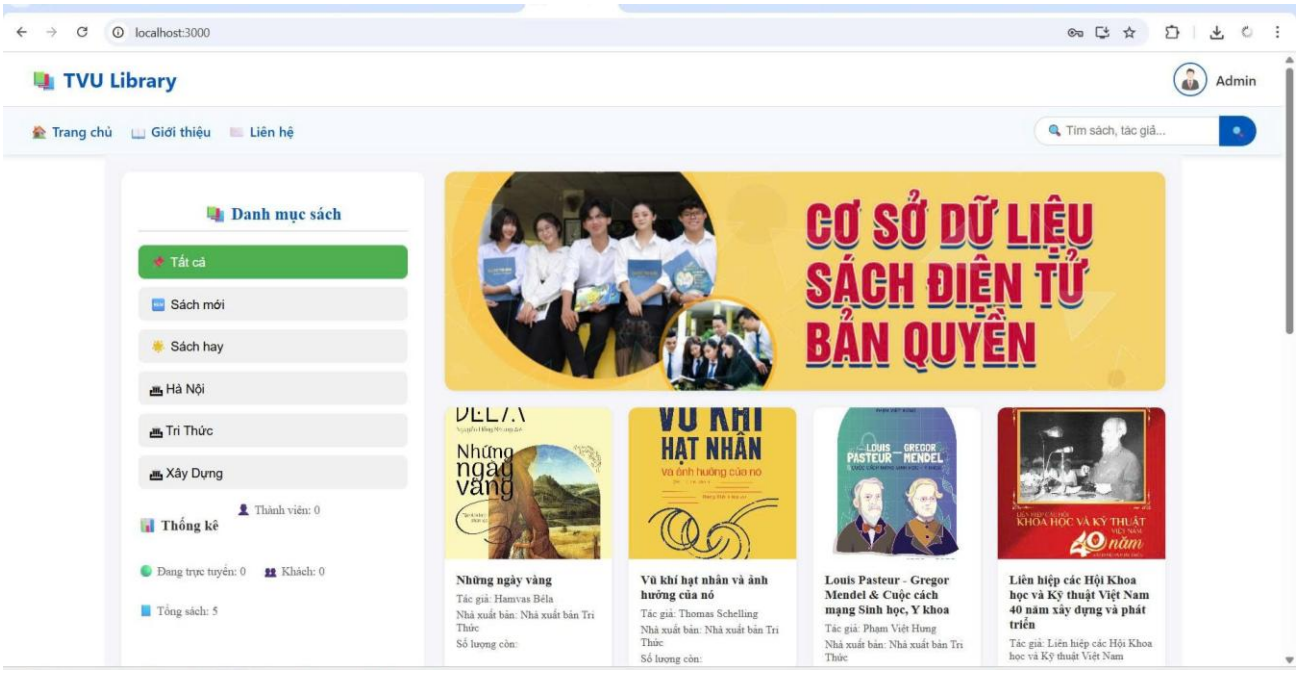
Thư mục src/: chứa toàn bộ mã nguồn chính của Frontend

- api/Chứa các hàm gọi API, thường dùng Axios để giao tiếp với backend.
- assets/Chứa tài nguyên tĩnh như hình ảnh, icon, font,...
- components/Các component dùng lại được, ví dụ: Button, Input, Modal,...
- constants/Chứa các hằng số được dùng nhiều nơi như các loại trạng thái, thông báo,...
- contexts/Cung cấp các context để quản lý state toàn cục qua React Context API.
- features/Chứa các chức năng chính như quản lý người dùng, quản lý sách, mượn trả,...
- hooks/Các custom hook phục vụ tái sử dụng logic như useFetch, useAuth,...
- layouts/Định nghĩa bố cục trang như Layout chính, Layout Admin,...
- pages/ Các trang cụ thể trong ứng dụng như Trang chủ, Trang quản lý, Trang đăng nhập,...
- services/Chứa logic xử lý dữ liệu trước khi hiển thị hoặc gửi đi, ví dụ: xử lý dữ liệu thống kê.
- styles/ Các file CSS hoặc SCSS dùng chung cho toàn ứng dụng.
- App.jsx: Component gốc chứa định tuyến toàn ứng dụng.
- index.jsx: Điểm khởi đầu của ứng dụng, render App vào DOM.
- .env:Chứa các biến môi trường như địa chỉ API, PORT,...
- package-lock.json:Ghi lại phiên bản chính xác của các thư viện để đồng bộ môi trường.



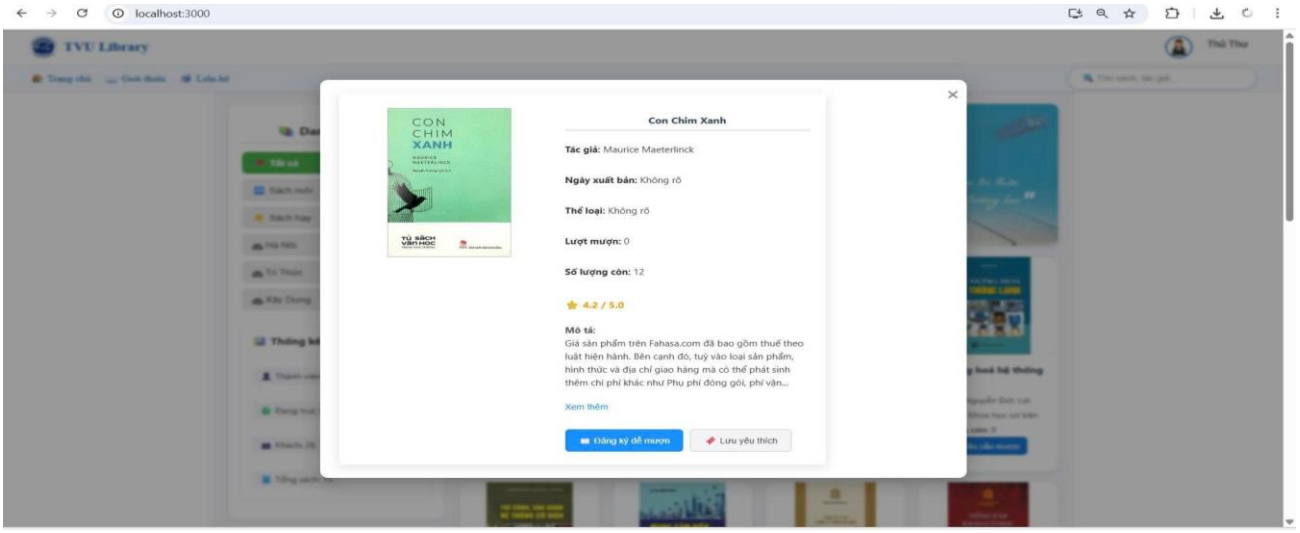
Hình 4.1 Cấu trúc thư mục Frontend

Giao diện người dùng bao gồm trang chủ, giới thiệu, liên hệ, thanh tìm kiếm,...



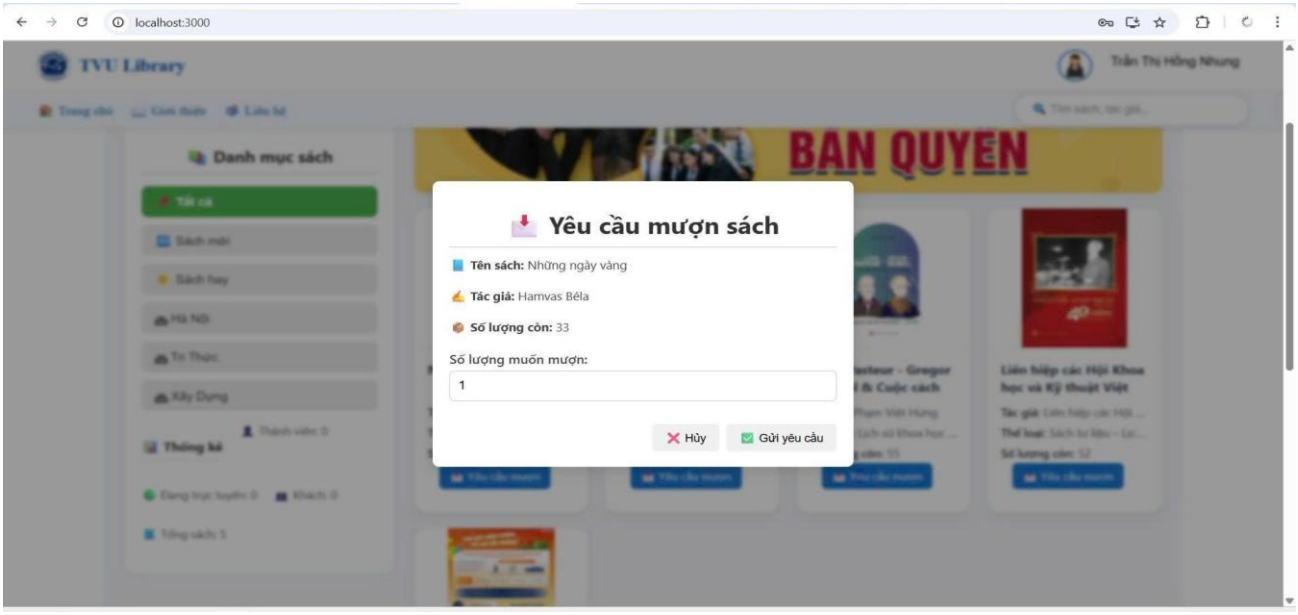
Hình 4.2 Giao diện trang chủ

Trang chủ chứa các danh mục sách của hệ thống, có banner chính, các sách được chia thành các box nhỏ có khoảng cách bằng nhau,người dùng độc giả có thể mượn bằng cách nhấn vào nút yêu cầu mượn, bắt buộc phải đăng nhập trước khi mượn, nếu người dùng chưa có tài khoản thì có thể đăng ký tài khoản, phần footer chứa các thông tin liên hệ, địa chỉ,..



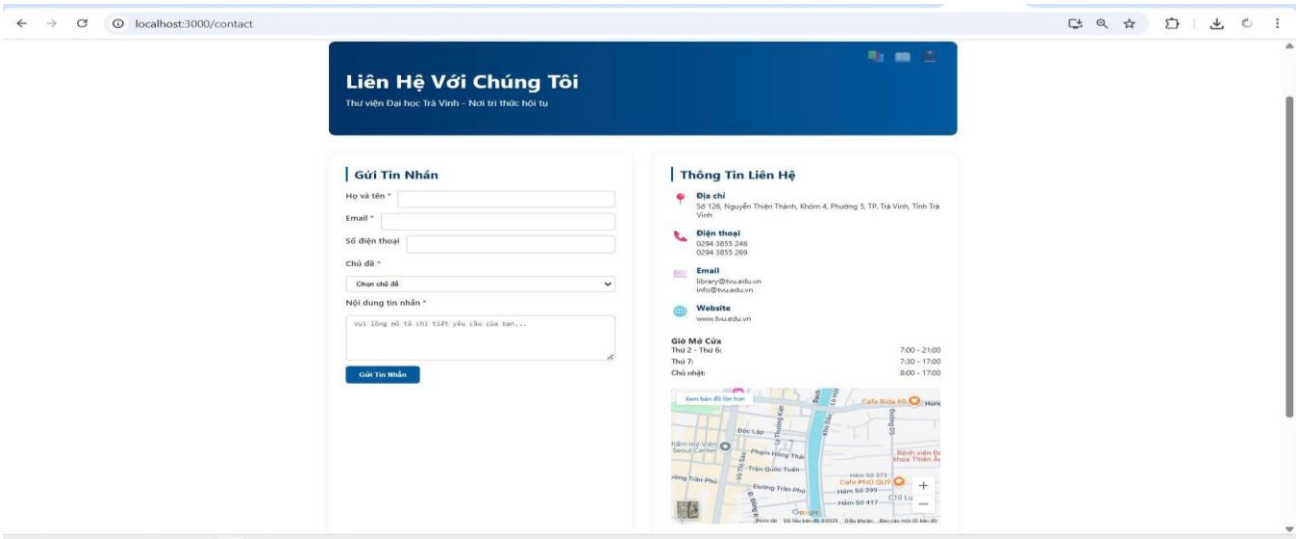
Hình 4.3 Giao diện chi tiết sách

Giao diện cung cấp một cái nhìn tổng quát về sách như tác giả, thể loại, ngày xuất bản,…giúp người dùng dễ dàng tìm sách phù hợp.



Hình 4.4 Giao diện yêu cầu mượn sách

Giao diện cho phép người dùng chọn số lượng sách muuons mượn và nhấn nút gửi yêu cầu để chờ phản hồi từ thủ thư.

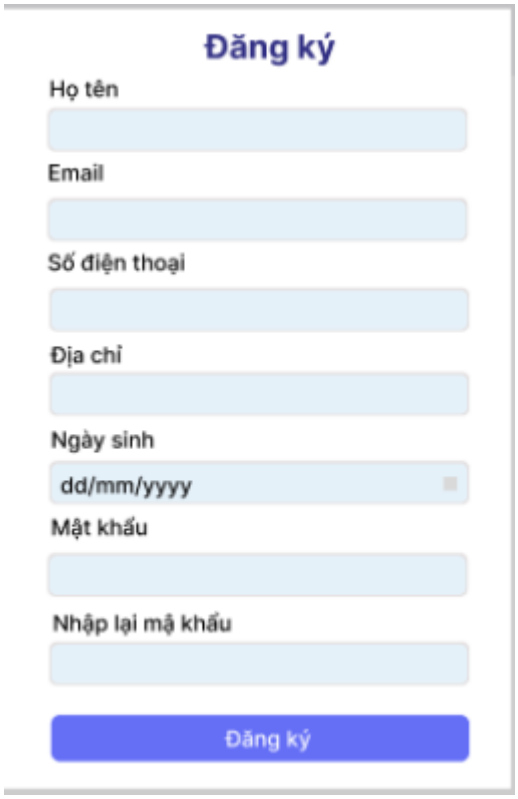


Hình 4.5 Trang liên hệ

Trang liên hệ cung cấp các thông tin liên hệ của hệ thống như: địa chỉ, số điện thoại,email.Ngoài ra, người dùng có thể gửi tin nhắn bằng cách nhập các thông tin liên hệ như

họ tên, email, số điện thoại, chủ đề, tin nhắn, nội dung sau đó nhấn nút gửi. Hệ thống sẽ phản hồi thông qua email của người dùng.

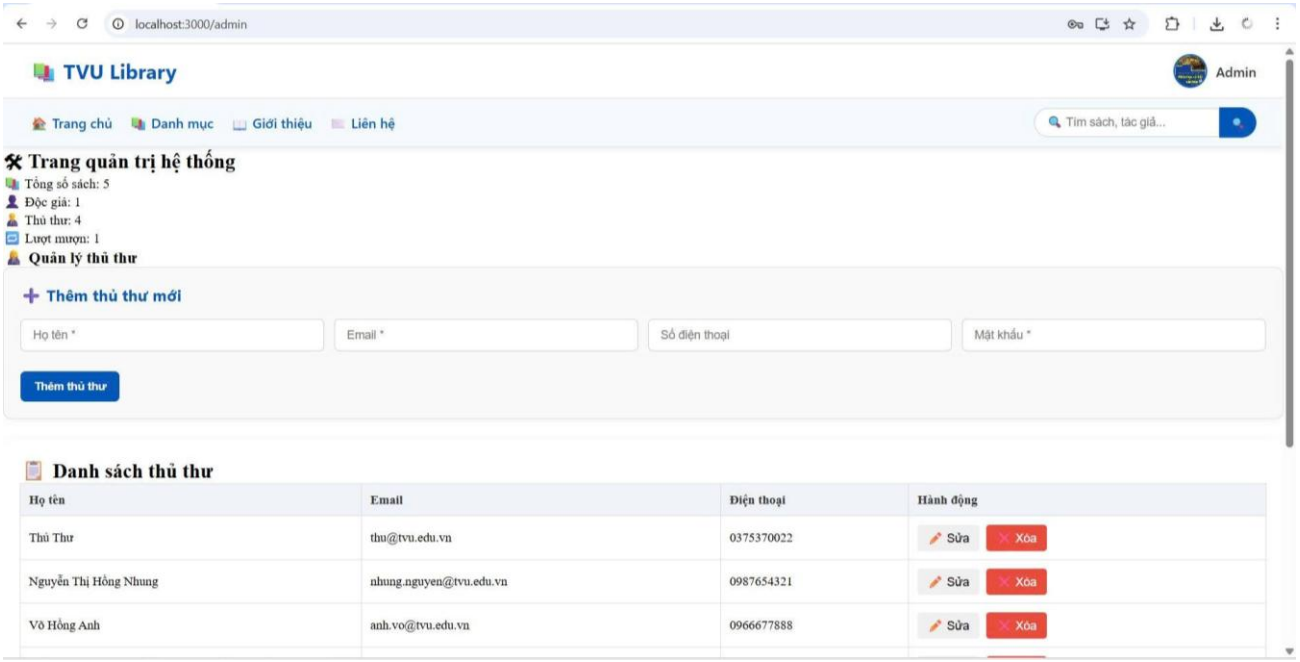
Hình 4.6 Trang giới thiệu



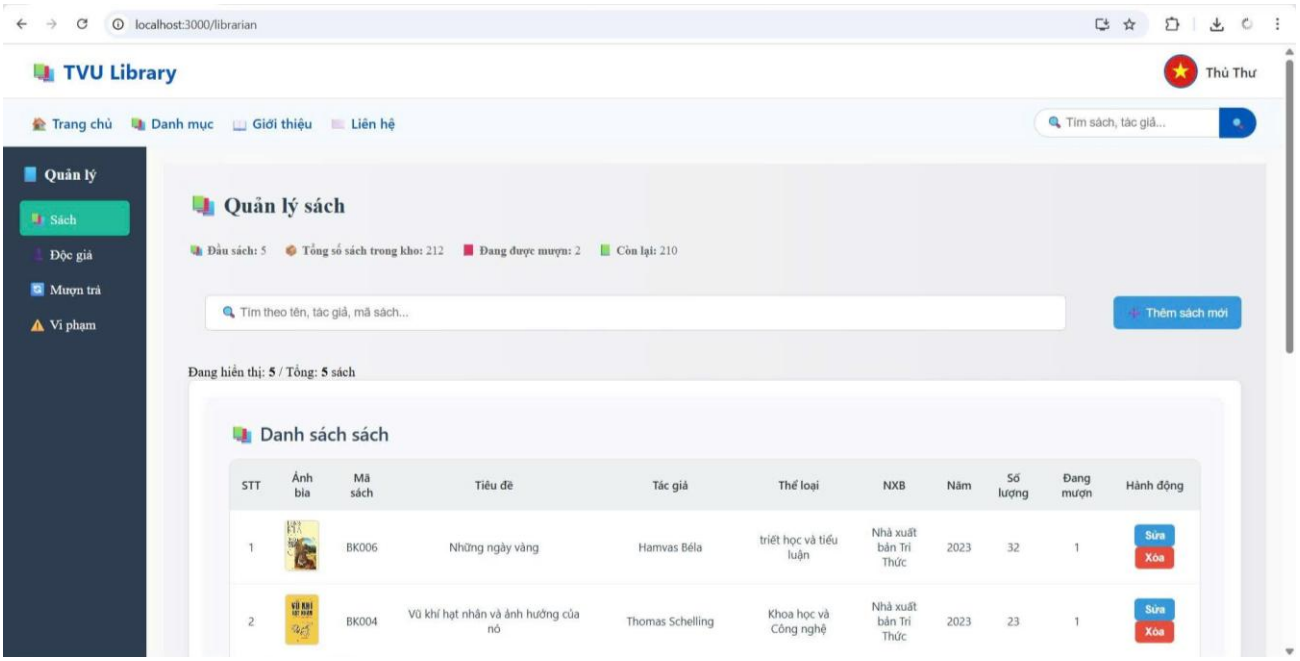
Hình 4.7 Form đăng ký

Hình 4.8 Form đăng nhập

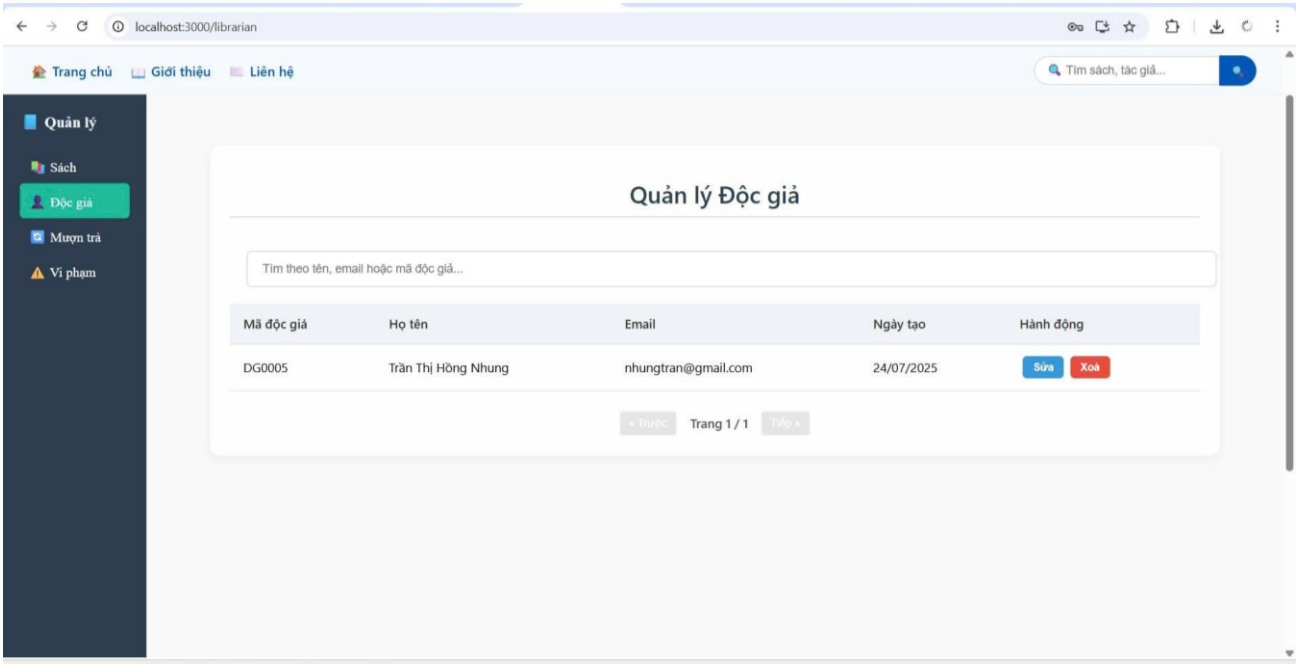
Giao diện trang quản lý: bao gồm các trang quản lý sách, quản lý độc giả, quản lý mượn trả, quản lý vi phạm, thử thư có thể thêm xóa sửa các thông tin. Ngoài các trang quản lý còn có các thông báo khi thêm xóa sửa.



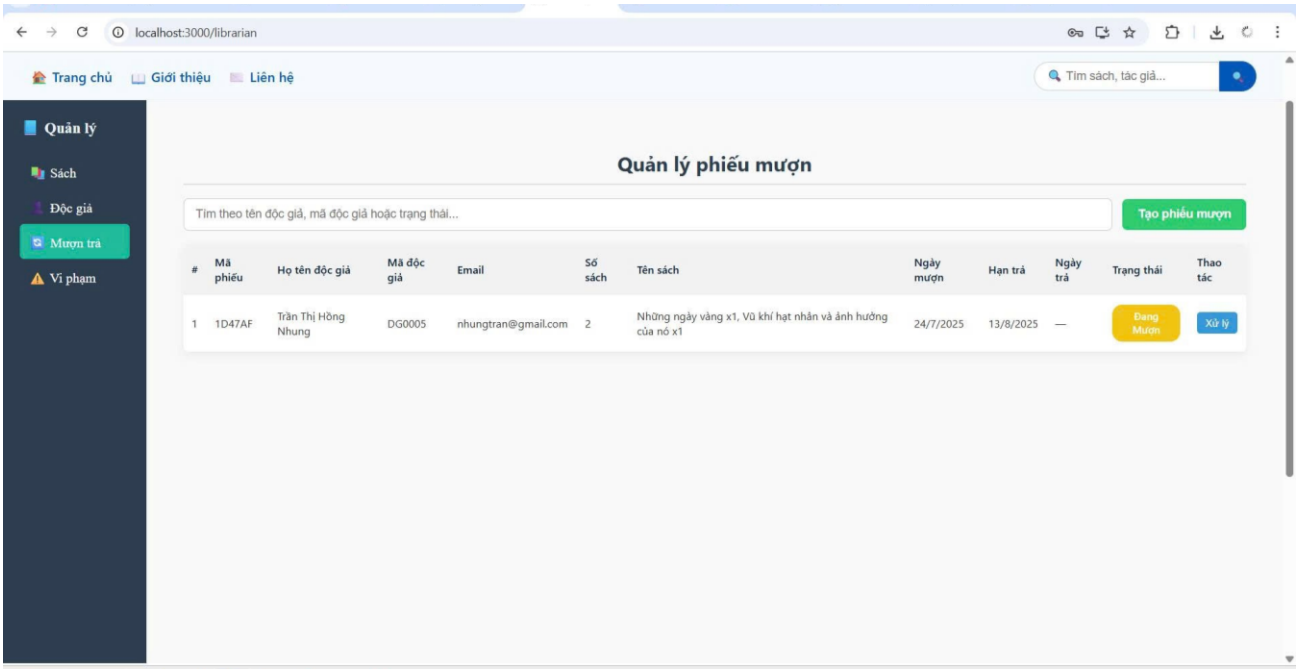
Hình 4.9 Giao diện quản lý thủ thư của admin



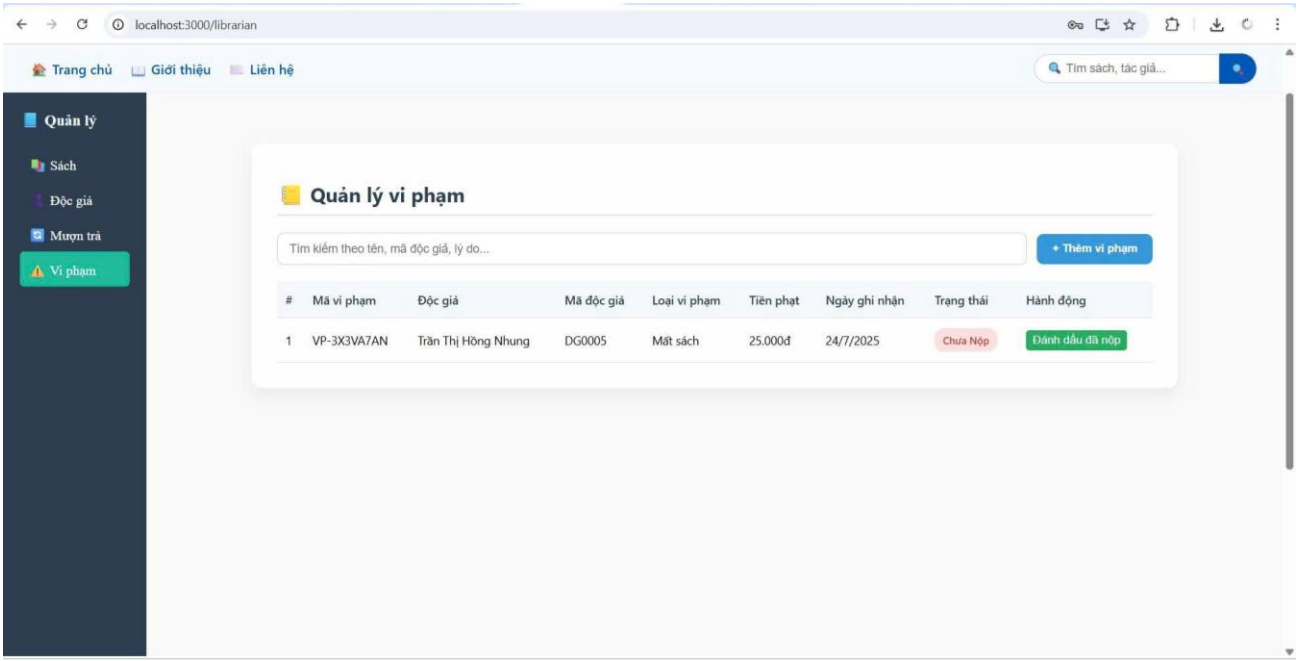
Hình 4.10 Trang quản lý sách



Hình 4.11 Trang quản lý độc giả



Hình 4.12 Trang quản lý phiếu mượn



Hình 4.13 Trang quản lý vi phạm

4.2 Backend

Phần Backend của hệ thống được xây dựng bằng Node.js kết hợp với Express.js, giúp tạo ra các API RESTful để xử lý các yêu cầu từ frontend. Dữ liệu được lưu trữ và truy xuất từ MongoDB, một cơ sở dữ liệu NoSQL linh hoạt, phù hợp với các ứng dụng web hiện đại.

Hệ thống backend có các chức năng chính như:

- Xử lý các yêu cầu GET, POST, PUT, DELETE để phục vụ các thao tác như: lấy danh sách sách, thêm/sửa/xóa sách, quản lý người dùng, xử lý mượn/trả sách,...
- Áp dụng mô hình MVC để tách biệt các phần xử lý logic, định tuyến và truy vấn dữ liệu.
- Sử dụng JWT (JSON Web Token) để xác thực người dùng, đảm bảo các tài nguyên được bảo vệ chỉ được truy cập bởi những người dùng hợp lệ.

Dữ liệu từ frontend được gửi đến backend thông qua API. Backend tiếp nhận yêu cầu, truy vấn cơ sở dữ liệu MongoDB, xử lý logic và trả về dữ liệu tương ứng cho frontend hiển thị .

Cấu trúc thư mục Backend:

Thư mục Public/: Chứa các file tĩnh như ảnh, CSS, JS client.

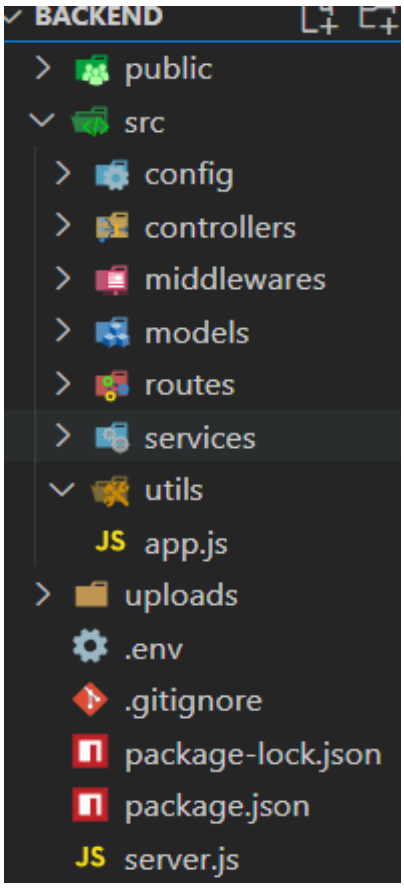
Thư mục Src/:

- config/: cấu hình hệ thống, kết nối mongoDB, cấu hình JWT, CORS, PORT, hoặc biến môi trường
- controllers/: Xử lý logic điều hướng yêu cầu từ người dùng, gọi services để xử lý nghiệp vụ, trả về response cho client.
- middlewares/: xác thực token, kiểm tra phân quyền, xử lý lỗi
- models/:Chứa các Schema của MongoDB định nghĩa cấu trúc dữ liệu.
- routes/: Định tuyến các đường dẫn API đến controllers
- services/: Chứa logic nghiệp vụ chính của ứng dụng, Controllers sẽ gọi đến đây để xử lý
- utils/: Mã hóa mật khẩu, sinh token JWT

Thư mục uploads/: Lưu trữ tạm thời các file người dùng tải lên

File app.js : File cấu hình app chính, gọi express, cấu hình middleware, routes.

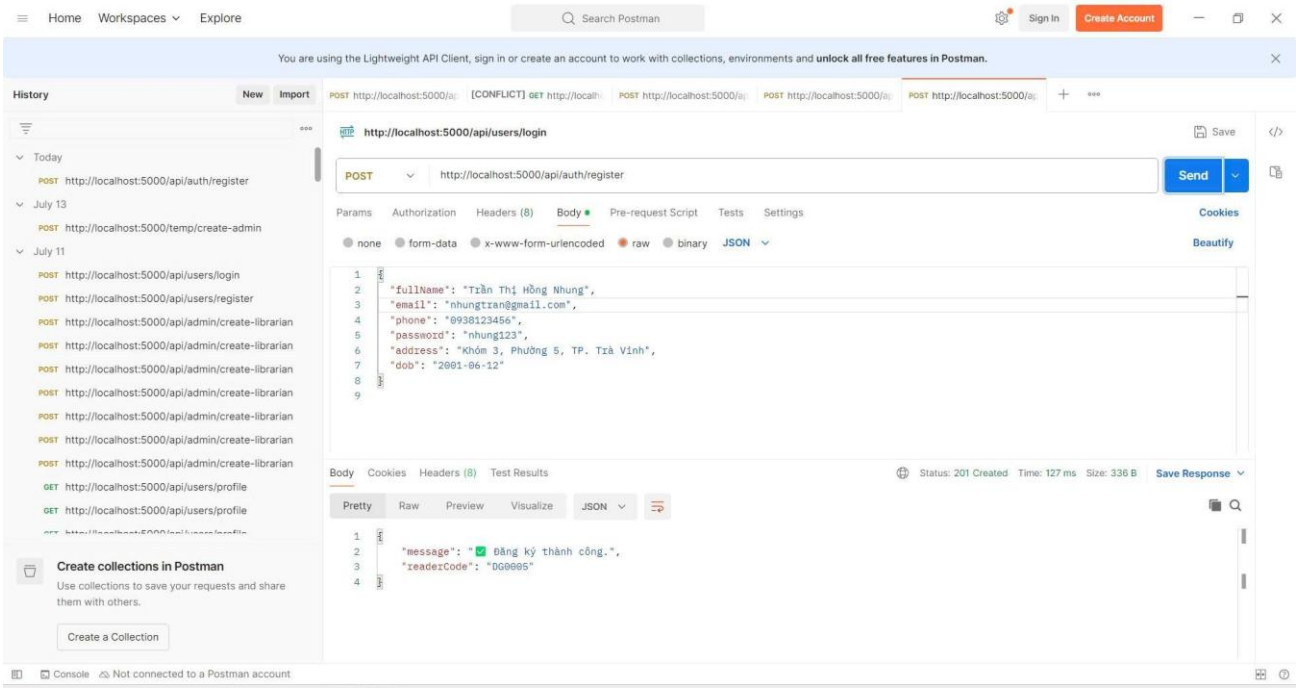
File server.js : File khởi chạy server



Hình 4.14 Cấu trúc thư mục Backend

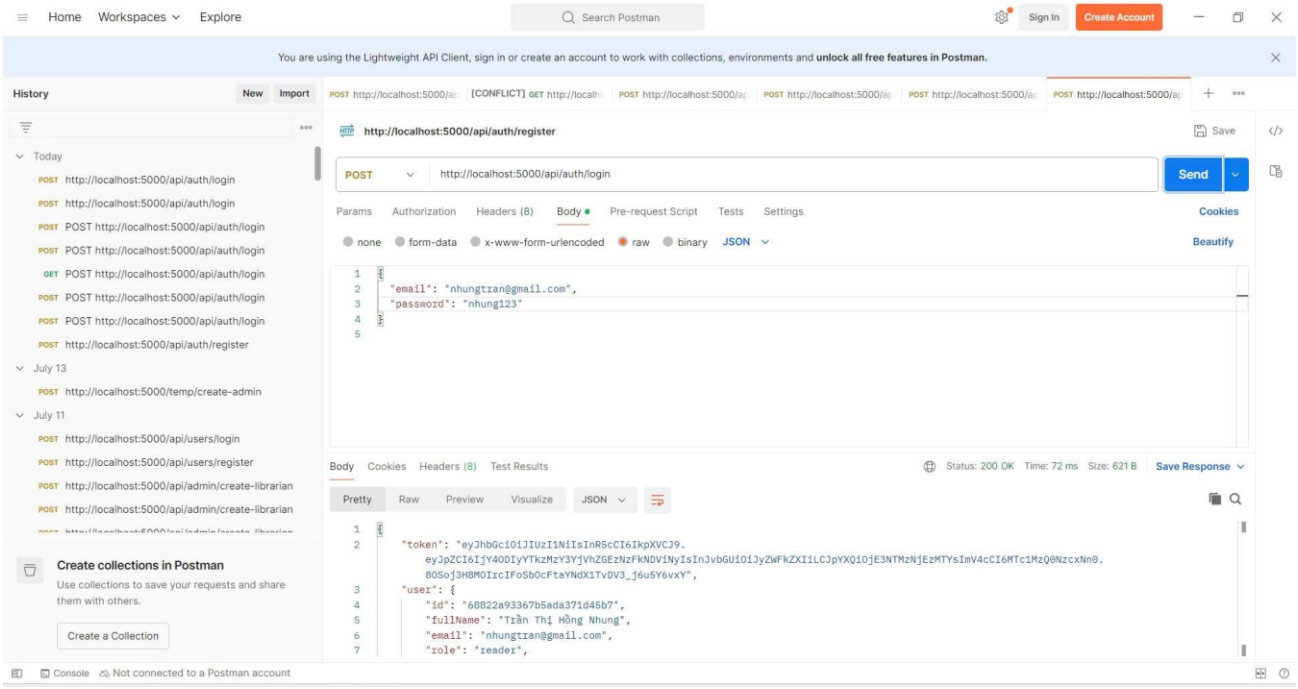
Để đảm bảo các API hoạt động đúng chức năng, nhóm đã sử dụng Postman – công cụ phổ biến để kiểm tra RESTful API.Kiểm thử API của backend thông qua các phương thức GET, POST, PUT, DELETE.

- Người dùng đăng ký tài khoản với phương thức POST, kết quả trả về đăng ký thành công , thông tin người dùng được tạo thành công.



Hình 4.15 Kiểm thử API đăng ký tài khoản

- Người dùng đăng nhập vào tài khoản với phương thức POST,kết quả đăng nhập thành công Server xác thực đúng thông tin đăng nhập, phản hồi lại JWT token và thông tin người dùng.



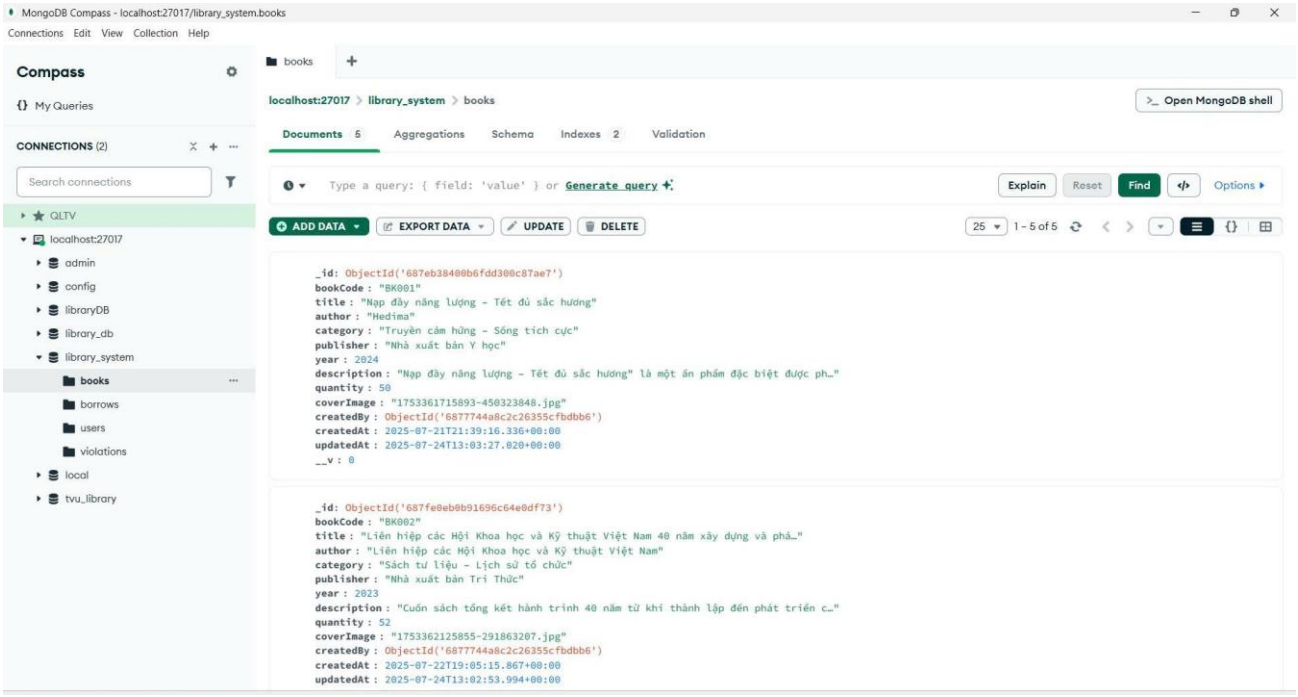
Hình 4.16 Kiểm thử API đăng nhập

4.3 MongoDB

MongoDB là hệ quản trị cơ sở dữ liệu NoSQL dạng tài liệu (document-oriented), sử dụng cấu trúc dữ liệu BSON (tương tự JSON) để lưu trữ.Trong MongoDB, dữ liệu được lưu trong các collection, tương tự như bảng trong SQL.Mỗi collection chứa nhiều document, tương tự như một dòng dữ liệu (record), nhưng có thể có cấu trúc linh hoạt hơn.

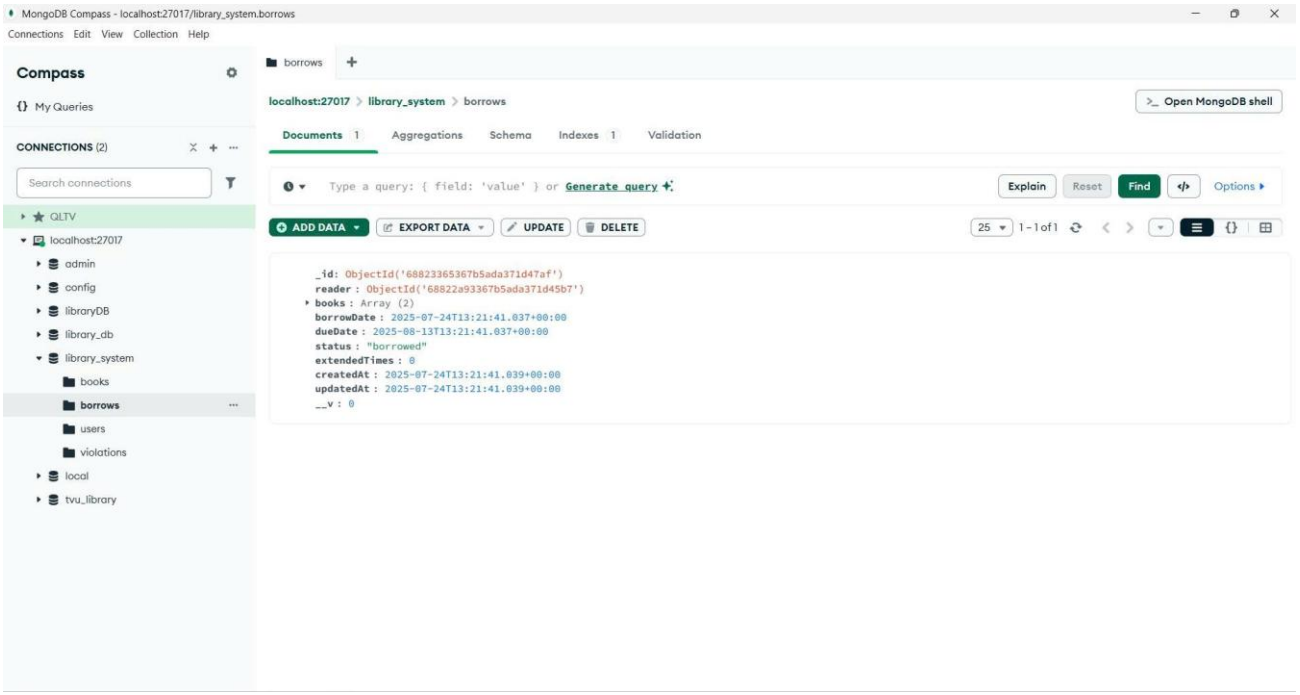
Các collection chính trong hệ thống quản lý thư viện:

- Books: Lưu thông tin sách, mã sách, tên sách, tác giả, nhà xuất bản, năm xuất bản,mô tả sách, số lượng tồn kho,đường dẫn bìa sách, thời gian tạo và cập nhật dữ liệu.



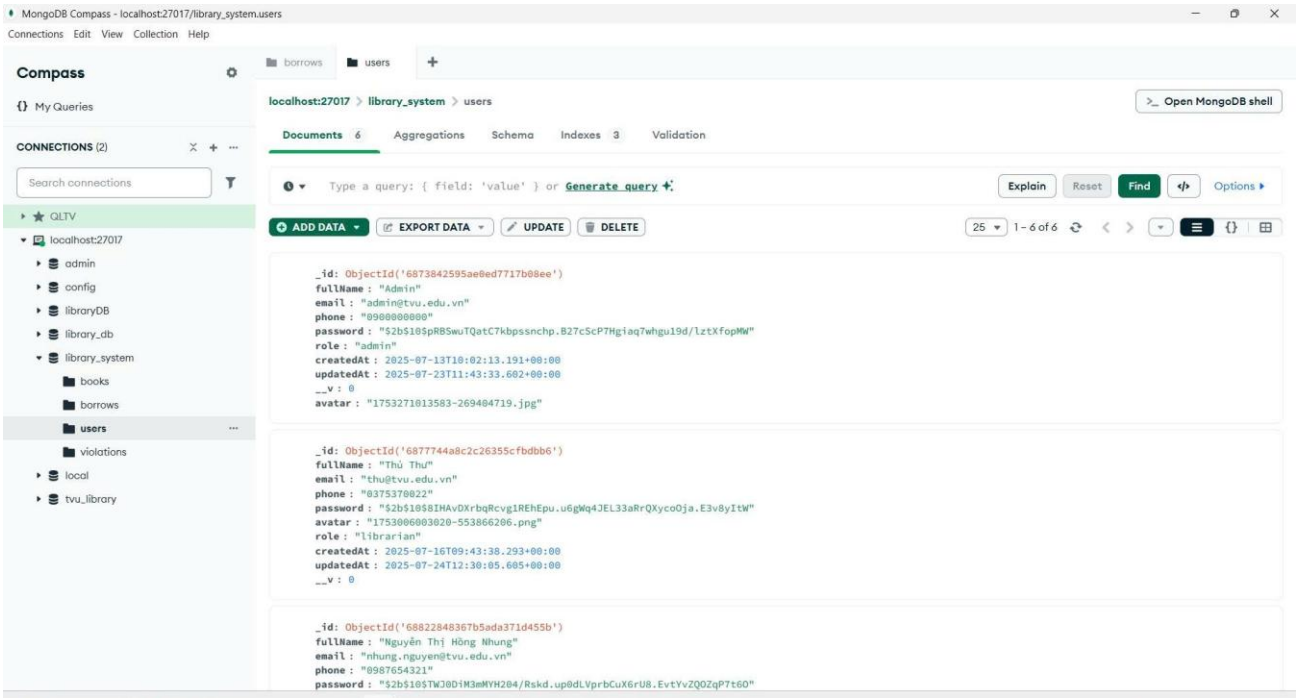
Hình 4.17 Collection books

- Borrows: Lưu thông tin mượn trả, ngày mượn, hạn trả sách, trạng thái, số lần gia hạn, thời gian tạo và cập nhật bản ghi.



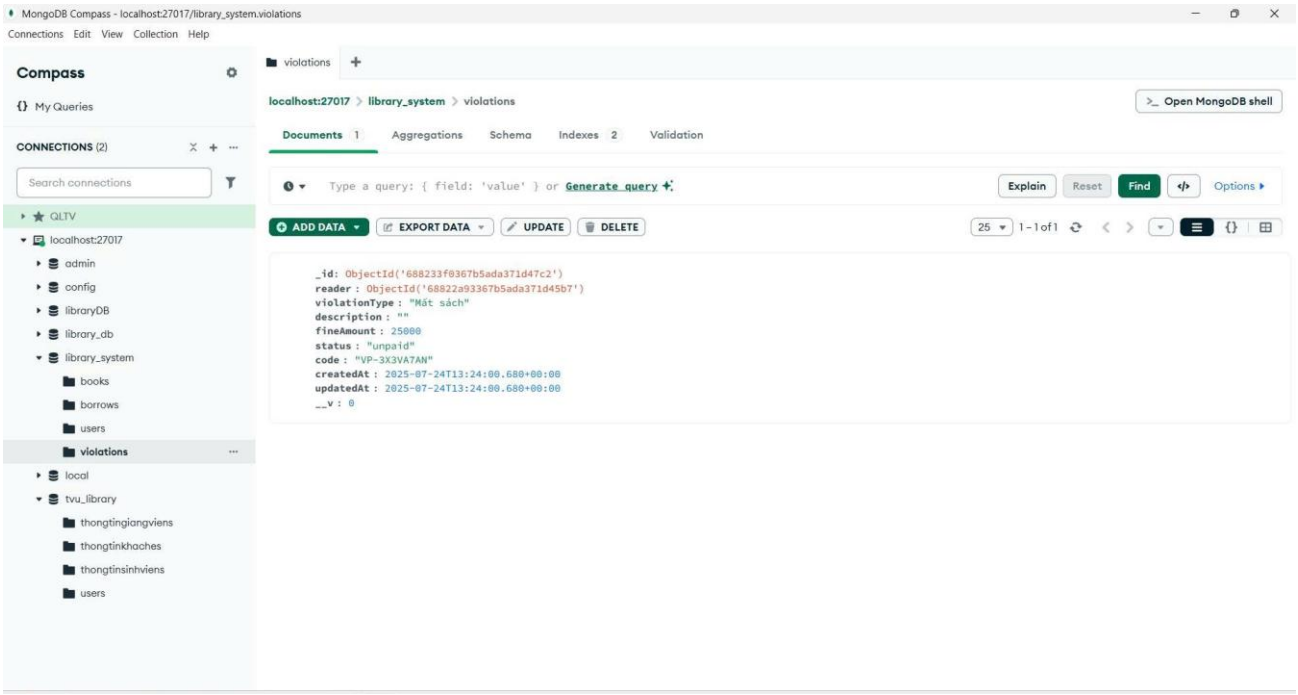
Hình 4.18 Collection borrows

- Users: Lưu thông tin người dùng, họ tên, email, số điện thoại, mật khẩu đã mã hóa, quyền, ảnh, thời gian tạo và cập nhật.



Hình 4.19 Collection users

- Violations: Lưu thông tin vi phạm,mã vi phạm, loại vi phạm, mô tả vi phạm, số tiền phạt, trạng thái thanh toán.



Hình 4.20 Collection violations

4.4 Chất lượng của hệ thống

Tính an toàn

- Xác thực người dùng bằng JWT , đảm bảo chỉ người dùng hợp lệ mới có thể truy cập hệ thống
- Phân quyền truy cập: Mỗi người dùng có các quyền khác nhau, quản trị viên có toàn quyền, thử thư có quyền quản lý, độc giả có quyền mượn trả sách
- Mật khẩu được mã hóa trước khi lưu xuống cơ sở dữ liệu , đảm bảo an toàn.

Tính riêng tư

- Dữ liệu cá nhân: Tên, địa chỉ email, lịch sử mượn sách phải được bảo vệ. Người dùng chỉ xem được thông tin của mình.
- Admin không thể xem mật khẩu tài khoản.

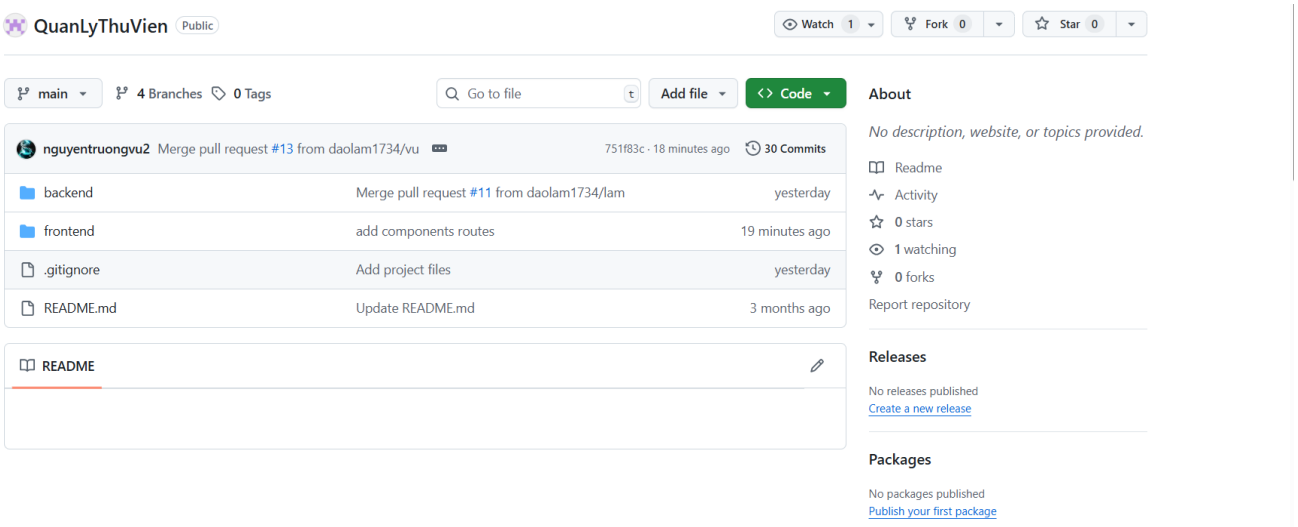
Đảm bảo chất lượng phần mềm

- Kiểm thử phần mềm: Test tất cả các chức năng của hệ thống

- Tài liệu thiết kế rõ ràng: Kiến trúc hệ thống, công nghệ sử dụng, được ghi chép rõ ràng chi tiết
- Giao diện thân thiện, dễ sử dụng.

4.5 Quản lý mã nguồn với GitHub

Toàn bộ mã nguồn của hệ thống được lưu trữ và quản lý thông qua nền tảng GitHub. Để đảm bảo việc phân công công việc hiệu quả và tránh xung đột mã nguồn, nhóm đã tạo 3 nhánh (branch) riêng biệt, tương ứng với 3 thành viên trong nhóm. Mỗi thành viên phát triển chức năng độc lập trên nhánh của mình. Sau khi hoàn tất và kiểm thử các chức năng, mã nguồn từ các nhánh cá nhân sẽ được kiểm tra, so sánh và gộp (merge) vào nhánh chính (main). Cách làm này giúp: Tăng tính độc lập và rõ ràng trong công việc nhóm. Giảm thiểu rủi ro xung đột mã khi làm việc song song. Dễ dàng quản lý, theo dõi lịch sử chỉnh sửa và trách nhiệm của từng thành viên.



Hình 4.21 Quản lý mã nguồn với github

CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết quả

- Xây dựng thành công ứng dụng web quản lý thư viện với các chức năng cơ bản như: quản lý sách, quản lý mượn trả, quản lý vi phạm, đăng nhập đăng ký,…
- Ứng dụng nhiều công nghệ hiện đại:React.js để xây dựng giao diện người dùng thân thiện, Node.js đảm nhận xử lý logic nghiệp vụ phía máy chủ, và MongoDB để lưu trữ dữ liệu phi quan hệ linh hoạt.
- Giao tiếp giữa frontend và backend được thực hiện thông qua RESTful API, giúp hệ thống hoạt động mạch lạc, dễ mở rộng và bảo trì.

5.2 Hạn chế

- Giao diện người dùng chưa tối cho thiết bị di động.
- Chưa có tính năng gửi email thông báo khi gần đến hạn trả sách.
- Hệ thống chưa kiểm soát được số lượng sách tồn kho theo thời gian thực.

5.3 Hướng phát triển

- Tối ưu giao diện theo hướng responsive design để hoạt động tốt hơn trên điện thoại, máy tính bảng.
- Thêm chức năng gửi email tự động khi gần đến hạn trả sách hoặc vi phạm.
- Xây dựng thống kê nâng cao: biểu đồ mượn sách theo tháng, thể loại sách được mượn nhiều,...

TÀI LIỆU THAM KHẢO

Vietnix. (n.d.). ReactJS là gì? Những điều bạn cần biết về ReactJS. https://vietnix.vn/react-js-la-gi/

TopDev. (n.d.). Node.js là gì? Ưu điểm của Node.js trong lập trình web. https://topdev.vn/blog/node-js-la-gi

Viblo. (n.d.). MongoDB là gì? Cơ sở dữ liệu phi quan hệ. https://viblo.asia/p/mongodb-la-gi-co-so-du-lieu-phi-quan-he-bJzKmgoPl9N

TopDev. (n.d.). RESTful API là gì? Tất tần tật về RESTful API cho lập trình viên. https://topdev.vn/blog/restful-api-la-gi

TopDev. (n.d.). RESTful API là gì? Tất tần tật về RESTful API cho lập trình viên. https://topdev.vn/blog/restful-api-la-gi

FPT Shop. (n.d.). Postman là gì? Tìm hiểu về phần mềm test API hiệu quả. https://fptshop.com.vn/tin-tuc/danh-gia/postman-la-gi-168171

FPT Shop. (n.d.). Figma là gì? Những điểm nổi bật của công cụ thiết kế giao diện này. https://fptshop.com.vn/tin-tuc/danh-gia/figma-la-gi-nhung-diem-noi-bat-150734

TopDev. (n.d.). GitHub là gì? Tổng quan về nền tảng lưu trữ và quản lý mã nguồn. https://topdev.vn/blog/github-la-gi

CodeGym. (2021, December 20). JWT là gì? Tìm hiểu về JSON Web Token và cách hoạt động của nó. CodeGym Blog. https://codegym.vn/blog/jwt-la-gi/

TopDev. (n.d.). Docker là gì? Tìm hiểu tất tần tật về Docker cho lập trình viên. TopDev. https://topdev.vn/blog/docker-la-gi/