ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO BÀI TẬP LỚN CUỐI KỲ

<u>ĐỀ TÀI:</u> Website quản lý thư viện

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU INT3202_7

Nhóm 20: **21020282 - Trịnh Kiều Anh**

21020011 - Nguyễn Trần Đạt

21020309 - Hồ Thu Giang

21020335 - Nguyễn Việt Hưng

20020106 - Bùi Hữu Việt Hùng

Hà Nội - 2023

Mục Lục

Chương 1

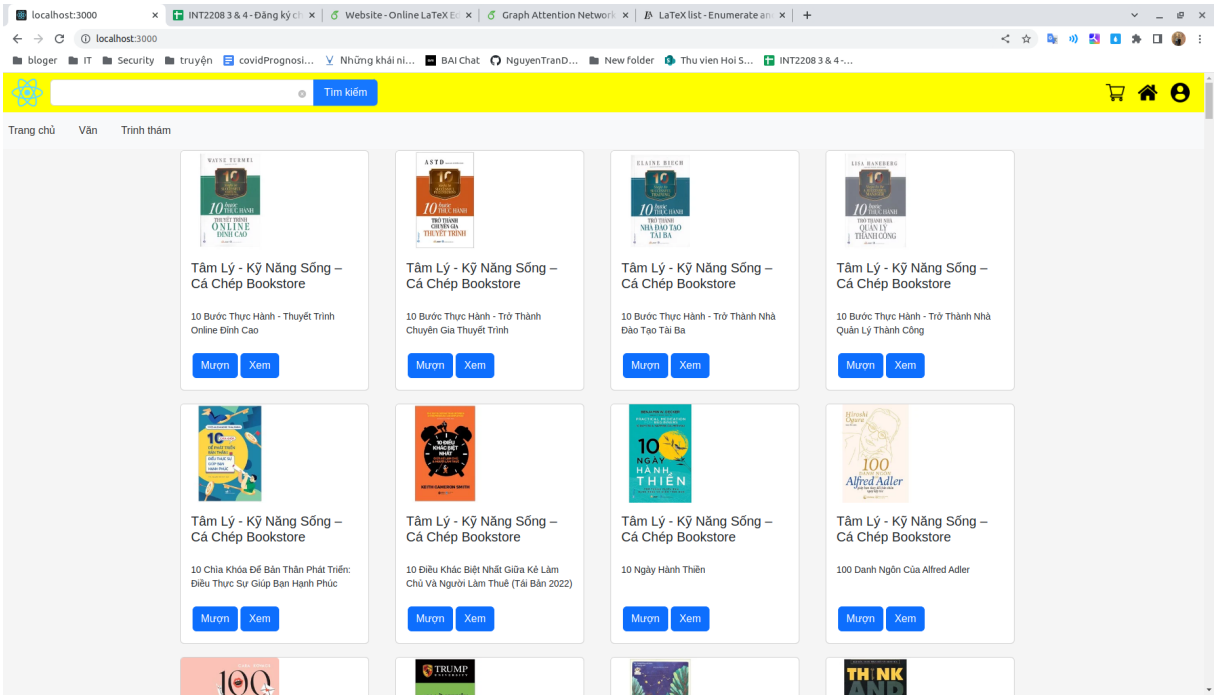
Đặt vấn đề

Trong xã hội ngày nay, nhu cầu đọc sách và tìm kiếm thông tin là điều không thể thiếu đối với mọi người. Tuy nhiên, việc tìm kiếm và quản lý các tài liệu đôi khi gặp nhiều khó khăn, đặc biệt là đối với các tổ chức như trường học, thư viện hay các công ty. Vì vậy, việc sử dụng một trang web quản lý thư viện trở nên cần thiết hơn bao giờ hết.

Một trang web quản lý thư viện sẽ giúp cho việc quản lý các tài liệu dễ dàng và hiệu quả hơn, từ việc lưu trữ, tìm kiếm cho đến việc theo dõi các mượn/trả sách. Bên cạnh đó, người dùng có thể tìm kiếm, đặt mượn và đặt trước các tài liệu một cách thuận tiện và nhanh chóng, giúp tiết kiệm thời gian và công sức.

Với sự phát triển không ngừng của công nghệ thông tin, việc xây dựng một trang web quản lý thư viện trở nên dễ dàng hơn bao giờ hết. Nếu được sử dụng đúng cách và phát triển thường xuyên, trang web quản lý thư viện sẽ giúp cho việc quản lý các tài liệu trở nên dễ dàng và thuận tiện hơn, từ đó giúp nâng cao chất lượng dịch vụ của các tổ chức và mang lại sự hài lòng cho người dùng.

Chính vì những lý do đó, nhóm chúng em đã thiết kế một website "Quản lý thư viện"với một loạt các tính năng hữu ích. Trang web này cung cấp cho người dùng khả năng hiển thị sách, thực hiện việc mượn sách, xem danh sách sách đã mượn, và kiểm tra lịch sử mượn sách. Ngoài ra, người dùng cũng có thể tìm kiếm sách, tạo tài khoản cá nhân, xem thông tin cá nhân, thay đổi mật khẩu và trải nghiệm tính năng đọc sách trực tuyến (hiện chỉ có sẵn trong bản demo). Đối với nhân viên thư viện, website còn giúp thông kê 10 cuốn sách được mượn nhiều nhất.



Hình 1.1: Hình ảnh trang chủ của website.

Chương 2

Mô tả kiến thức nền tảng

2.1 Tổng quan về quản lý thư viện

Quản lý thư viện là quá trình tổ chức, bảo quản, phục vụ và cho mượn tài liệu trong thư viện. Nó cũng bao gồm việc tạo ra các chính sách, quy trình và tiêu chuẩn để đảm bảo việc quản lý tài liệu được thực hiện hiệu quả và đáp ứng nhu cầu của người dùng.

Trong quản lý thư viện, các tài liệu được phân loại và đánh số để có thể tìm kiếm và truy cập một cách dễ dàng. Các tài liệu này bao gồm sách, báo, tạp chí, tài liệu điện tử, phim và âm nhạc.

Các thư viện có thể hoạt động trên nhiều cấp độ, bao gồm thư viện cá nhân, thư viện học đường, thư viện công cộng, thư viện đại học và thư viện quốc gia. Mỗi cấp độ thư viện có quy mô và chức năng khác nhau, với các yêu cầu và thách thức riêng.

Với sự phát triển của công nghệ, quản lý thư viện đã có sự thay đổi đáng kể trong thời gian gần đây. Việc sử dụng các phần mềm quản lý thư viện và các hệ thống tự động hóa giúp quản lý thư viện trở nên dễ dàng hơn và hiệu quả hơn.

2.2 Lợi ích của mô hình thư viện số

Mô hình thư viện số (digital library) mang lại nhiều lợi ích cho cả người quản lý thư viện và người sử dụng. Sau đây là một số lợi ích của mô hình thư viện số:

1. Tiết kiệm chi phí: Mô hình thư viện số cho phép tiết kiệm chi phí về không gian lưu trữ, vận chuyển tài liệu, và tiết kiệm chi phí cho các hoạt động liên quan đến việc bảo quản sách, tạp chí, tài liệu.

2. Tiện lợi: Người dùng có thể truy cập thư viện số 24/7 bất kể nơi đâu, không cần phải đến trực tiếp thư viện để mượn sách.

3. Khả năng tìm kiếm tài liệu nhanh chóng: Mô hình thư viện số cho phép người dùng tìm kiếm tài liệu nhanh chóng và chính xác hơn bằng cách sử dụng các công cụ tìm kiếm mạnh mẽ, đồng thời cung cấp thêm các thông tin phụ trợ như nhận xét, đánh giá từ người dùng khác, giúp người dùng có được sự lựa chọn phù hợp với nhu cầu của mình.

4. Dễ dàng chia sẻ và tái sử dụng tài liệu: Tài liệu trong thư viện số có thể được dễ dàng chia sẻ với những người dùng khác, đồng thời tài liệu có thể được tái sử dụng nhiều lần mà không cần phải sản xuất lại.

5. Tính bảo mật cao: Mô hình thư viện số có thể cung cấp tính bảo mật cao đối với các tài liệu nhạy cảm và thông tin cá nhân của người dùng.

6. Tính khả dụng cao: Mô hình thư viện số cho phép thư viện mở rộng phạm vi phục vụ người dùng, bao gồm cả những người dùng ở xa, bị khuyết tật hoặc không có thời gian để đến thư viện.

2.3 Tổng quan về công nghệ và hệ quản trị đã sử dụng

2.3.1 NodeJS

Node.js là một môi trường chạy mã JavaScript ở phía máy chủ (server-side runtime environment). Nó được xây dựng dựa trên Chrome V8 JavaScript engine và cho phép chạy mã JavaScript bên ngoài trình duyệt.

Node.js cho phép lập trình viên sử dụng JavaScript để phát triển ứng dụng máy chủ, không chỉ giới hạn trong việc xây dựng ứng dụng web mà còn cả các ứng dụng mạng và ứng dụng dòng lệnh. Với Node.js, JavaScript có thể được sử dụng để xây dựng các API, xử lý yêu cầu HTTP, quản lý các kết nối đến cơ sở dữ liệu, xử lý tác vụ hàng đợi và nhiều tác vụ khác liên quan đến phía máy chủ. Node.js có một hệ sinh thái phong phú của các module và thư viện mở rộng, cho phép lập trình viên sử dụng các module có sẵn hoặc tạo các module riêng để mở rộng khả năng của Node.js. Nó đã trở thành một công cụ phổ biến cho phát triển ứng dụng web hiệu năng cao, ứng dụng realtime, ứng dụng IoT và nhiều lĩnh vực phát triển phần mềm khác.

2.3.2 ExpressJS

Express.js là một framework phát triển ứng dụng web bằng ngôn ngữ JavaScript và chạy trên môi trường Node.js. Nó được xây dựng trên cơ sở HTTP module của Node.js và cung cấp một cách tiếp cận đơn giản và mạnh mẽ để xây dựng các ứng dụng web và API.

Express.js giúp đơn giản hóa việc xây dựng và quản lý các tác vụ phổ biến trong phát triển web, bao gồm định tuyến (routing), xử lý yêu cầu và phản hồi (request/response), quản lý middleware, xử lý lỗi, và nhiều chức năng khác. Nó giúp tổ chức mã nguồn dễ dàng và linh hoạt, giúp lập trình viên xây dựng ứng dụng web nhanh chóng và hiệu quả.

Express.js cũng hỗ trợ việc tích hợp các module và thư viện khác để bổ sung các tính năng cho ứng dụng, chẳng hạn như middleware PassportJS để xác thực người dùng, Sequelize để làm việc với cơ sở dữ liệu, Socket.IO để xây dựng ứng dụng realtime, và nhiều module khác.

Với cú pháp đơn giản và cộng đồng phát triển sôi nổi, Express.js đã trở thành một trong những framework phát triển ứng dụng web phổ biến nhất trong hệ sinh thái Node.js.

2.3.3 Axios

Axios là một thư viện JavaScript mã nguồn mở được sử dụng để gửi các yêu cầu HTTP từ phía client tới một máy chủ và xử lý các phản hồi từ máy chủ. Nó cung cấp một cách dễ dàng và linh hoạt để tương tác với các API từ phía client trong các ứng dụng web và mobile.

Axios hỗ trợ các phương thức HTTP như GET, POST, PUT, DELETE và PATCH và cung cấp các tính năng như xử lý các yêu cầu và phản hồi bất đồng bộ, quản lý các tiêu đề yêu cầu và phản hồi, xử lý các lỗi mạng, và các chức năng liên quan đến việc tương tác với API.

Axios là một thư viện JavaScript phổ biến để gửi các yêu cầu HTTP và xử lý phản hồi từ máy chủ, giúp tương tác với các API một cách dễ dàng và hiệu quả trong các ứng dụng web và mobile.

2.3.4 ReactJS

ReactJS là một thư viện JavaScript mã nguồn mở được sử dụng để xây dựng giao diện người dùng (UI) trong các ứng dụng web. Nó được phát triển bởi Facebook và cộng đồng React và đã trở thành một trong những công cụ phát triển phổ biến nhất cho việc xây dựng giao diện người dùng động và tương tác.

ReactJS tập trung vào việc xây dựng các thành phần UI tái sử dụng (reusable UI components), giúp tách giao diện thành các phần nhỏ, độc lập và dễ quản lý. Nó sử dụng cú pháp JSX để kết hợp HTML và JavaScript để mô tả giao diện.

ReactJS là một thư viện JavaScript mạnh mẽ cho việc xây dựng giao diện người dùng linh hoạt, dễ quản lý và hiệu suất cao trong các ứng dụng web.

Ngoài ra, để giúp thao tác nhanh hơn, nhóm chúng em còn sử dụng lazy loading cho trang web, hiển thị theo thời gian loading của trang web.

2.3.5 MySQL

MySQL sử dụng ngôn ngữ truy vấn cấu trúc (Structured Query Language - SQL) để tương tác với cơ sở dữ liệu. Nó cung cấp một hệ thống quản lý cơ sở dữ liệu mạnh mẽ và linh hoạt, hỗ trợ các tính năng như tạo, xóa, cập nhật và truy vấn dữ liệu. Nó hỗ trợ đa người dùng, đa luồng, cung cấp tính bảo mật và ổn định.

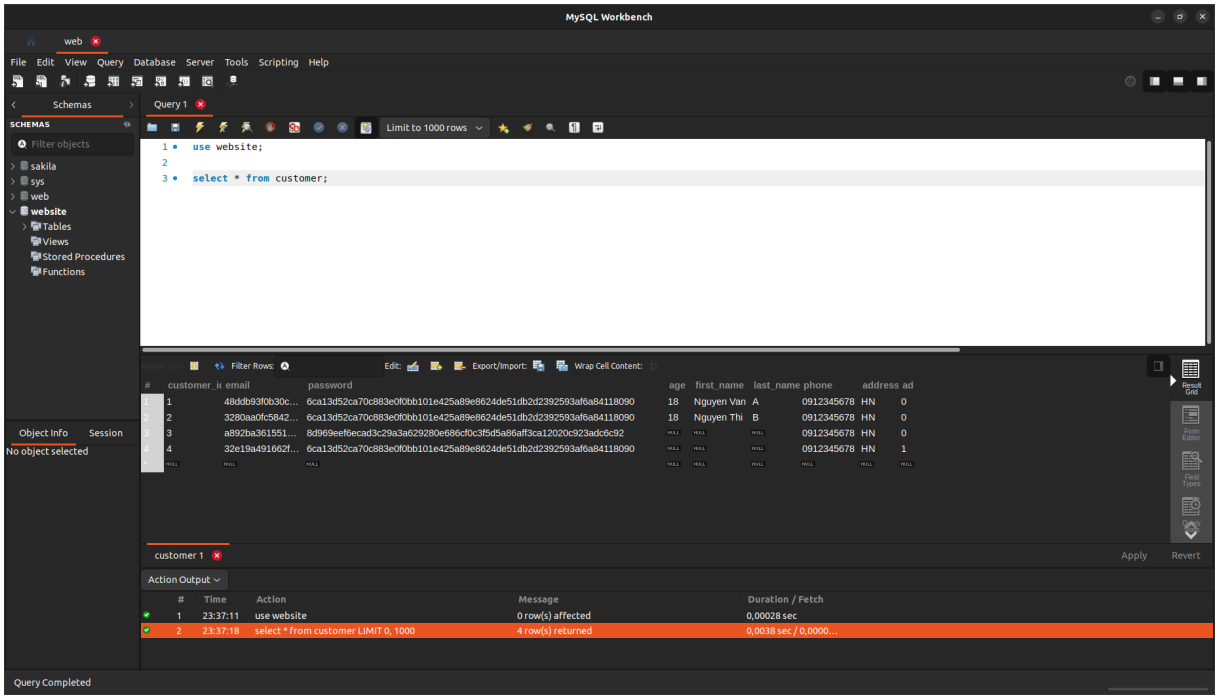
MySQL có thể được triển khai trên nhiều hệ điều hành khác nhau, bao gồm Windows, macOS và các phiên bản của Linux. Ngoài ra, nó cũng hỗ trợ các ngôn ngữ lập trình phổ biến như PHP, Python, Java và nhiều ngôn ngữ khác để tương tác với cơ sở dữ liệu.

MySQL cũng được đánh giá cao về khả năng mở rộng, với khả năng hỗ trợ hàng triệu lượt truy cập một ngày và có thể mở rộng lên đến hàng trăm máy chủ. Điều này làm cho MySQL trở thành một giải pháp lý tưởng cho các ứng dụng web có lưu lượng truy cập lớn.

Ngoài ra, MySQL còn có tính năng bảo mật tốt, bao gồm các cơ chế xác thực và phân quyền cấp độ người dùng, giúp bảo vệ dữ liệu khỏi các cuộc tấn công độc hại.

Vì dữ liệu thông tin người dùng là một dữ liệu vô cùng quan trọng. Nó đòi hỏi sự bảo mật cao. Chính vì vậy nhóm chúng em quyết định sử dụng MySQL làm một database lưu thông tin tài khoản người dùng. Trong MySQL chúng em sử dụng một số kiểu dữ liệu như INT, VARCHAR. Và để hỗ trợ tìm kiếm nhanh hơn, mạnh hơn, tối ưu hơn nhóm

chúng em sử dụng tìm kiếm dựa trên index (customer_index). Ngoài ra, để tăng tính bảo mật, nhóm còn sử dụng hàm SHA2 của MySQL để mã hóa email và mật khẩu của người dùng.



Hình 2.1: Hình ảnh lưu trữ tài khoản người dùng trên MySQL

2.3.6 MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu phi quan hệ (non-relational database management system - NoSQL DBMS) phổ biến và mã nguồn mở. Nó được phát triển bởi MongoDB Inc. và là một phần của phong cách cơ sở dữ liệu không cần lược đồ (schemaless).

MongoDB lưu trữ dữ liệu dưới dạng tài liệu (document), sử dụng định dạng JSON-like gọi là BSON (Binary JSON). Mỗi tài liệu trong MongoDB được lưu trữ trong một collection, tương tự như bảng trong cơ sở dữ liệu quan hệ. Tuy nhiên, trong MongoDB, các tài liệu trong cùng một collection không cần phải có cấu trúc dữ liệu giống nhau.

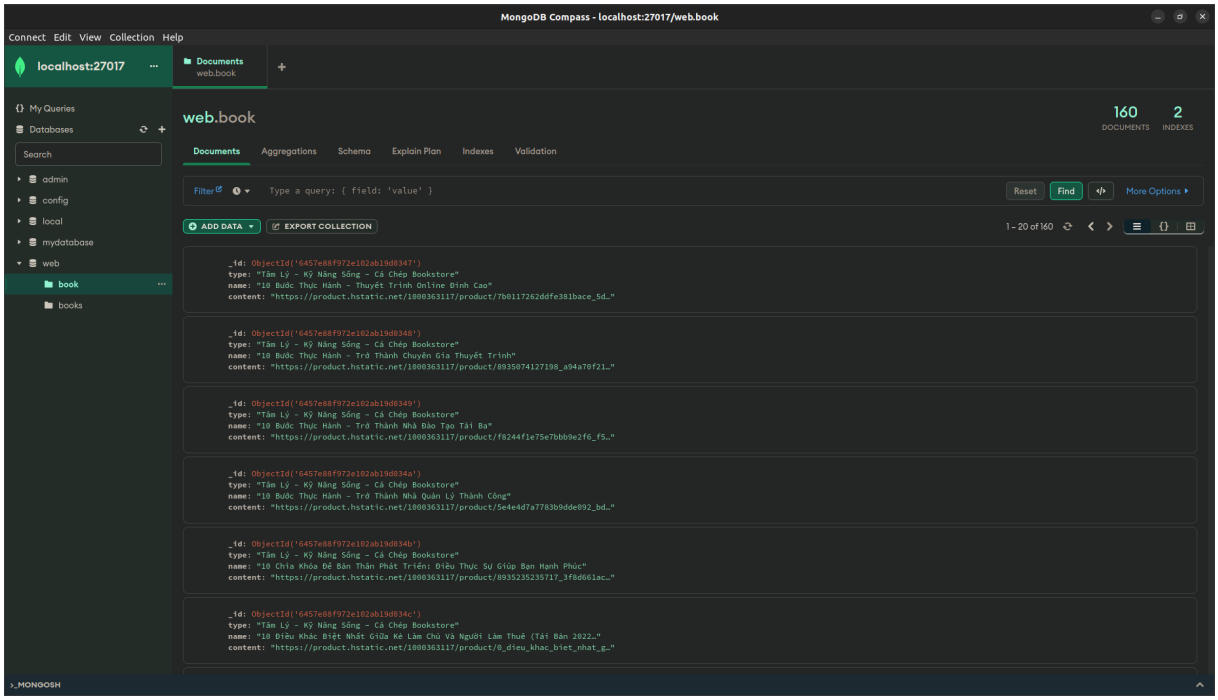
Một số đặc điểm và lợi ích của MongoDB bao gồm:

- Linh hoạt: MongoDB cho phép thay đổi cấu trúc dữ liệu một cách dễ dàng, không cần phải thay đổi lược đồ trước.
- Mở rộng: Nó hỗ trợ mô hình mở rộng ngang (horizontal scaling), cho phép mở rộng hệ thống dựa trên khả năng xử lý của nhiều máy chủ.

- Tính dự phòng và phục hồi: MongoDB cung cấp các tính năng dự phòng và phục hồi dữ liệu, như repliaction và sharding, để đảm bảo tính sẵn sàng và bảo mật dữ liệu.
- Truy vấn linh hoạt: Nó hỗ trợ các truy vấn phong phú và mạnh mẽ, bao gồm các truy vấn tìm kiếm, so sánh, phân tích và kết hợp dữ liệu.

MongoDB được sử dụng rộng rãi trong các ứng dụng web, mobile và Internet of Things (IoT), nơi linh hoạt và khả năng mở rộng của nó là điểm mạnh.

Vì dữ liệu sách có thể thay đổi linh hoạt, và để tiện cho việc thêm/sửa/xóa, nhóm em quyết định sử dụng MongoDB để lưu thông tin về sách. Trong MongoDB, nhóm em sử dụng một số kiểu dữ liệu như: ObjectID, text. Nhóm em cũng hỗ trợ tìm kiếm sách dựa trên tên và thể loại của sách. Ngoài ra, để hỗ trợ tìm kiếm, nhóm em cũng sử dụng tìm kiếm trên 2 id là (_id và name_text_type_text) để giúp tìm kiếm sách một cách nhanh chóng và hiệu quả. Không những thế, nhóm cũng đang phát triển tìm kiếm dựa trên từ gợi ý để hỗ trợ người dùng truy xuất dựa trên những từ đã nhập vào nhưng bị thiếu hoặc sai dựa trên thuật toán levenshtein.



Hình 2.2: Hình ảnh lưu trữ sách trên MongoDB

2.3.7 Cassandra

Cassandra là một hệ quản trị cơ sở dữ liệu phi quan hệ (non-relational database management system - NoSQL DBMS) phân tán và mã nguồn mở. Nó được thiết kế để xử lý dữ liệu lớn và đảm bảo tính sẵn sàng và khả năng mở rộng cao.

Cassandra được phát triển bởi Apache Software Foundation và được xây dựng dựa trên các nguyên tắc của mô hình cột (column-oriented model) và phân tán trên nhiều máy chủ (distributed architecture). Nó sử dụng mô hình dữ liệu cấu trúc dạng cột để lưu trữ và truy vấn dữ liệu.

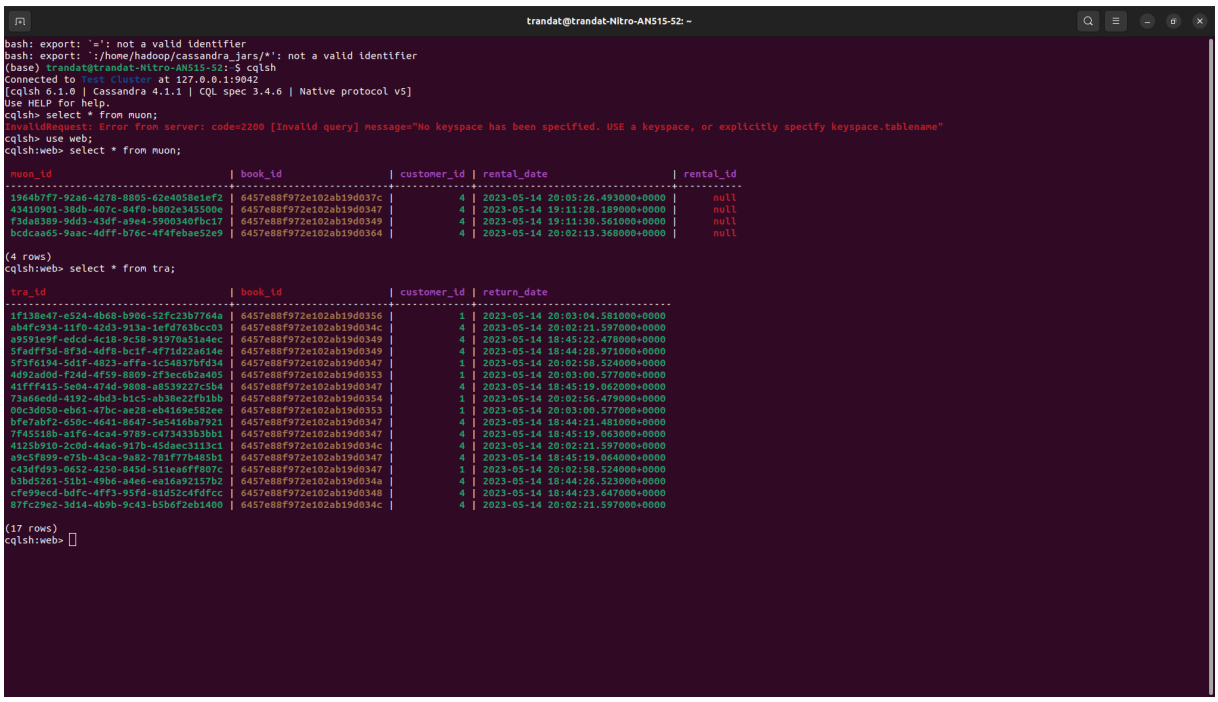
Một số đặc điểm và lợi ích của Cassandra bao gồm:

- Tính mở rộng ngang: Cassandra được thiết kế để mở rộng theo chiều ngang với khả năng thêm máy chủ một cách dễ dàng để xử lý tải lớn và dữ liệu phân tán.
- Tính sẵn sàng và bền vững: Nó cung cấp tính sẵn sàng cao thông qua việc sao chép dữ liệu trên nhiều máy chủ và có khả năng tự phục hồi khi xảy ra sự cố.
- Hỗ trợ truy vấn linh hoạt: Cassandra cung cấp ngôn ngữ truy vấn CQL(Cassandra Query Language) để truy vấn dữ liệu, hỗ trợ các truy vấn phức tạp và linh hoạt.
- Tích hợp dễ dàng với ứng dụng: Cassandra cung cấp các giao diện và API cho phép lập trình viên dễ dàng tích hợp và tương tác với cơ sở dữ liệu.

Cassandra được sử dụng trong nhiều lĩnh vực, bao gồm các ứng dụng web với dữ liệu lớn, hệ thống IoT, cơ sở hạ tầng mạng xã hội, và các hệ thống phân tán khác đòi hỏi tính mở rộng và khả năng xử lý dữ liệu phân tán.

Vì dữ liệu mượn/trả sách là một dữ liệu lớn, tăng theo từng ngày. Vậy nên, chúng em quyết định sử dụng Cassandra để lưu thông tin lịch sử mượn/trả sách. Trong Cassandra, nhóm em sử dụng một số kiểm dữ liệu như: uuid, text, int, timestamp để lưu trữ. Để tối ưu truy vấn, nhóm chúng em sử dụng tìm kiếm dựa trên 2 index là book_id, và customer_id. Nhóm em đang cố gắng phát triển để thống kê số lần mượn của một cuốn sách theo tùy chọn của người quản trị (hiện đã có thống kê top10 cuốn sách được mượn nhiều nhất) và tự động tính tiền khi khách hàng mượn quá số ngày quy định.

Ngoài những framework trên, nhóm em còn sử dụng một số framework khác như SCSS/CSS để hỗ trợ hiển thị website một cách trực quan và giúp người dùng thao tác dễ dàng hơn.



Hình 2.3: Hình ảnh lưu trữ lịch sử mượn trả trên Cassandra

Tổng kết chương: Với sự kết hợp của các framework, thư viện và hệ quản trị cơ sở dữ liệu, website quản lí thư viện được xây dựng để giúp người quản lí có thể quản lí được hàng trăm đầu sách, những ca mượn, trả sách một cách có hệ thống và đảm bảo được bảo mật, tránh tình trạng mất sách.

Việc sử dụng nhiều database giúp cho thông tin khách hàng và sản phẩm phân tán. Điều đó giúp ứng dụng của chúng em bảo mật hơn. Đồng thời, việc sử dụng nhiều database cũng hỗ trợ được nhược điểm của việc sử dụng một database gây nên, và cũng hỗ trợ cải thiện, nâng cấp khi có thay đổi. Không những thế chúng ta còn tận dụng được ưu điểm của các database khi lưu trữ. Tuy nhiên, việc sử dụng nhiều database cũng gây ra hiện tượng dữ liệu không đồng nhất. Đây cũng là vấn đề nhóm đang hướng tới trong tương lai.

Chương 3

Phân tích đặc tả yêu cầu

3.1 Phân tích yêu cầu

3.1.1 Phân tích yêu cầu chung của hệ thống

Công việc quản lý thư viện là một công việc phức tạp, đòi hỏi người quản lý phải linh hoạt, nhạy bén. Một số ứng dụng được tạo nên để hỗ trợ người quản lý một phần công việc. Website quản lý thư viện cũng là một ứng dụng như thế. Một số yêu cầu về Website quản lý thư viện bao gồm:

1. Đăng ký thành viên: Hệ thống cần cung cấp đăng ký thành viên để cho phép người dùng đăng nhập và sử dụng các chức năng của thư viện như mượn sách, đặt sách, tìm kiếm sách, và quản lý thông tin cá nhân (Đã làm được).

2. Quản lý thông tin sách: Hệ thống cần có chức năng quản lý thông tin về sách, bao gồm tên sách, tác giả, mã ISBN, năm xuất bản, nhà xuất bản, thể loại và số lượng sách trong thư viện (Đã làm được).

3. Quản lý mượn sách: Hệ thống cần cho phép người dùng mượn sách và lưu trữ thông tin về sách đã mượn, thời gian mượn (Đã làm được).

4. Tìm kiếm sách: Hệ thống cần có công cụ tìm kiếm sách để cho phép người dùng tìm kiếm sách theo tiêu chí khác nhau, bao gồm tên sách, thể loại (Đã làm được).

5. Quản lý thông tin cá nhân: Hệ thống cần cho phép người dùng quản lý thông tin cá nhân, bao gồm đổi mật khẩu, cập nhật thông tin liên lạc và xem lịch sử mượn sách (Đã làm được).

6. Thống kê: Hệ thống cần thống kê các quyển sách có số lượng người dùng nhiều nhất để hệ thống có thể xác định những cuốn sách có số lượng người dùng nhiều nhất để có thể đưa ra các quyết định chính xác về việc quản lý sách. Bên cạnh đó, thông kê sách cũng giúp cho quản lý thư viện nắm bắt được xu hướng đọc của độc giả và đưa ra các kế hoạch phù hợp để đáp ứng nhu cầu của độc giả (Đã làm được).

- Và một số những yêu cầu khác như đa ngôn ngữ, quản lý nhân viên, quản lý hoạt động, v.v. Nhóm mà nhóm cũng đang hướng tới trong tương lai.

3.2 Thu thập yêu cầu

3.2.1 Yêu cầu chức năng

Yêu cầu người đọc:

- Người dùng có thể mượn sách
- Người dùng có thể xem sách
- Người dùng có thể tìm kiếm sách qua thanh tìm kiếm, có thể tìm kiếm qua các từ gợi ý.
- Người dùng có thể quản lí thông tin đăng nhập tên, mật khẩu
- Người dung có thể xem được sách đang mượn
- Người dung có thể trả sách

Yêu cầu người quản lí:

- Người quản lí có thể thêm/ sửa/ xóa sách
- Người quản lí có thể quản lí người đọc bao gồm : thêm, xóa, sửa
- Người quản lí có các chức năng còn lại như người đọc.
- Người quản lý có thể xem được thống kê sách nào mượn nhiều nhất.

3.2.2 Yêu cầu phi chức năng

- Hệ thống hoạt động 24/7

- Hệ thống đưa ra phản hồi nhanh , dưới 1s cho thao tác xem sách và dưới 2s với các thao tác khác.
- Bảo mật thông tin khách hàng tốt theo tiêu chuẩn OWASP
- Giao diện dễ dùng thân thiện đảm bảo có thể để người dùng tự khám phá hết chức năng trong 5 phút.

Từ những yêu cầu trên, nhóm em đã xây dựng và phát triển ứng dụng nhằm đáp ứng những yêu cầu tối thiểu của người dùng cũng như người quản lý. Tuy nhiên còn một số yêu cầu chúng em chưa thể hoàn thành như thêm sửa xóa nhóm đang phát triển có thể lưu được ảnh, bảo mật thông tin khách hàng chưa được tốt. Ngoài ra, nhóm cũng đang phát triển để có thể chia thành nhiều thư viện thành viên, có thể tra ở trong các thư viện thành viên thay vì chỉ hỗ trợ một thư viện cố định, phân nhiều quyền hơn để hạn chế truy cập dữ liệu trái phép.

Chương 4

Phân tích thiết kế và xây dựng hệ thống

4.1 Mô hình MICROSERVICE

Mô hình microservice là một kiến trúc phần mềm được sử dụng để phát triển các ứng dụng lớn, phức tạp và dễ mở rộng. Trong mô hình này, ứng dụng được phân chia thành nhiều dịch vụ nhỏ hơn, được gọi là microservices, mỗi dịch vụ chịu trách nhiệm thực hiện một chức năng cụ thể trong hệ thống. Mỗi microservice có thể được phát triển, triển khai và quản lý độc lập với nhau, và có thể được viết bằng các ngôn ngữ và công nghệ khác nhau.

Mô hình microservice giúp phân tách ứng dụng thành các thành phần nhỏ hơn, dễ quản lý và mở rộng hơn. Bằng cách phân tách ứng dụng thành các dịch vụ độc lập, các nhà phát triển có thể tập trung vào việc phát triển một phần của hệ thống mà không ảnh hưởng đến các phần khác. Mô hình này cũng giúp giảm thiểu rủi ro khi có lỗi xảy ra trong một dịch vụ, vì các dịch vụ khác vẫn có thể tiếp tục hoạt động bình thường.

Từ những ưu điểm đó, nhóm chọn sử dụng mô hình microservice để hỗ trợ phát triển và bảo trì sản phẩm một cách tốt nhất có thể. Không những thế, mô hình còn giúp phân chia nhiệm vụ một cách dễ dàng hơn khi mỗi một thành viên sẽ phụ trách một cơ sở dữ liệu nhất định.

- API: cung cấp giao thức để hỗ trợ truy xuất cơ sở dữ liệu dựa trên hoạt động của khách hàng.

- View: hiển thị thông tin, tương tác trực tiếp với người dùng, chuyển tiếp các yêu

cầu tới hệ thống để hiển thị ra màn hình.

- Controller: có nhiệm vụ nhận yêu cầu của người dùng và gọi các phương thức phù hợp để xử lý và thao tác với cơ sở dữ liệu.

4.2 Thiết kế API

Trong quá trình sử dụng trang web, người dùng sẽ thực hiện các thao tác để yêu cầu hiển thị thông tin, trong khi chỉ gửi đi các yêu cầu tương ứng đến hệ thống. Để thực hiện điều này, hệ thống sẽ sử dụng một API để tiếp nhận và xử lý các yêu cầu từ người dùng, thực hiện các truy vấn dữ liệu cần thiết lên cơ sở dữ liệu và trả về kết quả phù hợp để hiển thị trên giao diện.

Đáng chú ý là, hệ thống được thiết kế nhằm đảm bảo tính bảo mật của thông tin. Các thông tin nhạy cảm như mật khẩu sẽ không được hiển thị cho người ngoài. Thay vào đó, người dùng chỉ có thể nhập mật khẩu dưới dạng ký tự "* "và gửi mật khẩu tới hệ thống một cách an toàn.

Hơn nữa, chức năng thêm sản phẩm mới và thống kê chỉ sẽ được sử dụng bởi những người có quyền hạn như thủ thư hoặc quản lý thư viện. Điều này giúp đảm bảo rằng chỉ những người được ủy quyền mới có thể thêm mới các sản phẩm vào hệ thống.

Tổng thể, mục đích chính của hệ thống là bảo vệ tính bảo mật cho toàn bộ hệ thống. Điều này bao gồm việc ngăn chặn truy cập trái phép vào cơ sở dữ liệu và đảm bảo an toàn cho thông tin. Bằng cách này, hệ thống giúp đảm bảo sự bảo mật và tránh mất mát thông tin quan trọng.

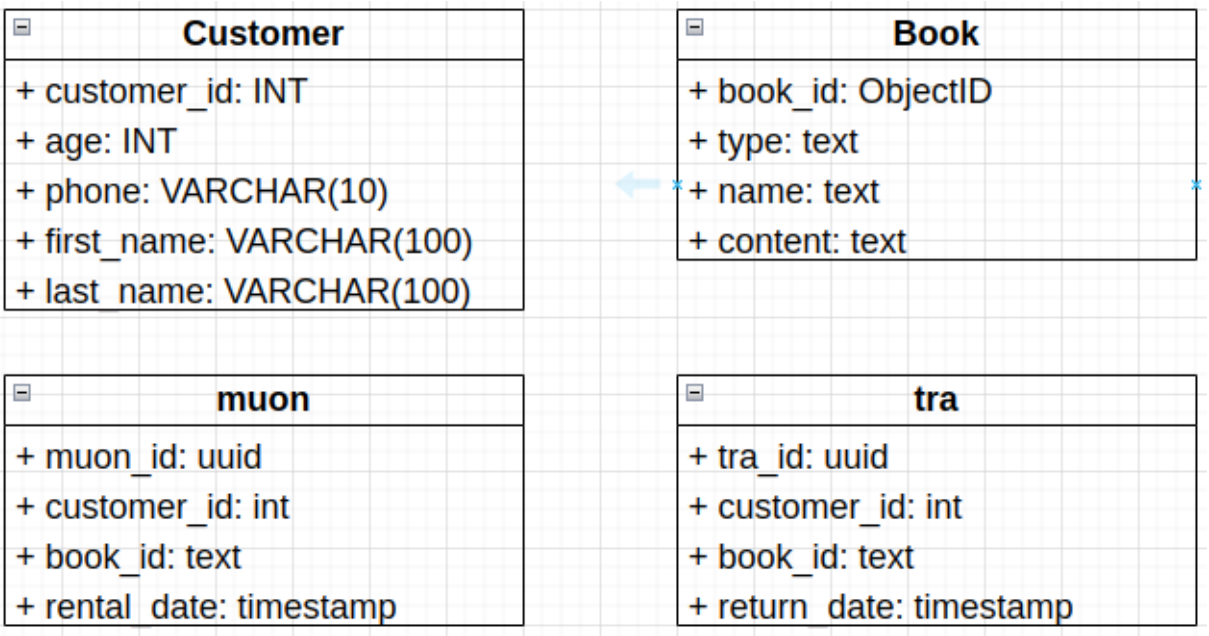
4.3 Thiết kế cơ sở dữ liệu

Nhóm em chia nhỏ các cơ sở dữ liệu và dựa vào ưu điểm của các hệ quản trị cơ sở dữ liệu để lưu trữ và truy xuất. Điều này giúp chúng em tận dụng được điểm mạnh của các hệ quản trị cơ sở dữ liệu khác nhau, đồng thời cũng giúp dữ liệu của chúng em phân tán, giúp chúng em truy xuất nhanh hơn, và hơn hết, nó cũng giúp chúng em nâng cấp, quản lý, phát triển và bảo trì dễ dàng hơn. Tuy nhiên, việc chia thành nhiều cơ sở dữ liệu khác nhau cũng gây ra cho chúng em một số vấn đề về đồng bộ dữ liệu và tăng tốc truy xuất trên các hệ quản trị cơ sở dữ liệu khác nhau. Hiện tại, nhóm đang cố gắng tăng tốc truy xuất trên các hệ quản trị bằng cách sử dụng truy xuất đồng thời nhiều câu lệnh SQL (song song hóa).

- MySQL: có tính bảo mật tốt, bao gồm cơ chế xác thực và phân quyền cấp độ người dùng, giúp bảo vệ dữ liệu khỏi các cuộc tấn công độc hại. MySQL sẽ lưu trữ thông tin của các user (người đọc, admin) bao gồm : email, mật khẩu, tên tuổi, số điện thoại, địa chỉ, phân quyền. Nhóm còn sử dụng mã hóa SHA2 của MySQL để mã hóa mật khẩu (Hình 5.1).

- MongoDB: cho phép lưu trữ và truy xuất dữ liệu dễ dàng hơn so với các hệ quản trị cơ sở dữ liệu quan hệ truyền thống và có khả năng mở rộng dễ dàng. Từ những ưu điểm trên, nhóm em đã quyết định sử dụng MongoDB để lưu thông tin về sách. Trong MongoDB, nhóm em sử dụng một số kiểu dữ liệu như: ObjectID, text. Nhóm em cũng hỗ trợ tìm kiếm sách dựa trên tên và thể loại của sách. Ngoài ra, để hỗ trợ tìm kiếm, nhóm em cũng sử dụng tìm kiếm trên id để giúp tìm kiếm sách một cách nhanh chóng và hiệu quả. Nhóm cũng đang phát triển tìm kiếm dựa trên từ gợi ý để hỗ trợ truy xuất được nhanh chóng hơn. (Hình 5.2).

- Cassandra: cho phép mở rộng một cách linh hoạt và dễ dàng để xử lý các tải trọng dữ liệu lớn. Vì Cassandra hỗ trợ nhiều tính năng hữu ích như khả năng sao lưu và khôi phục dữ liệu tự động, khả năng sao chép dữ liệu qua các máy chủ và khả năng đồng bộ dữ liệu giữa các trung tâm dữ liệu khác nhau nên rất phù hợp để lưu thông tin lịch sử mượn/trả sách (đây là thông tin cần đc truy xuất nhiều). Trong Cassandra, nhóm em sử dụng một số kiểm dữ liệu như: uuid, text, int, timestamp để lưu trữ. Nhóm em đang cố gắng phát triển để thống kê số lần mượn của một cuốn sách và tự động tính tiền khi khách hàng mượn quá số ngày quy định. (Hình 2.3)



Hình 4.1: Hình ảnh cơ sở dữ liệu

Chương 5

Phân tích chi tiết thiết kế dữ liệu

5.1 MySQL

Hệ quản trị cơ sở dữ liệu MySQL được sử dụng để hỗ trợ lưu trữ thông tin người dùng trong hệ thống quản lý thư viện. MySQL là một hệ quản trị cơ sở dữ liệu phổ biến và mạnh mẽ, cung cấp các tính năng và khả năng cần thiết để quản lý và truy xuất dữ liệu một cách hiệu quả. Dưới đây là một số khía cạnh chi tiết về cách MySQL hỗ trợ lưu trữ thông tin người dùng:

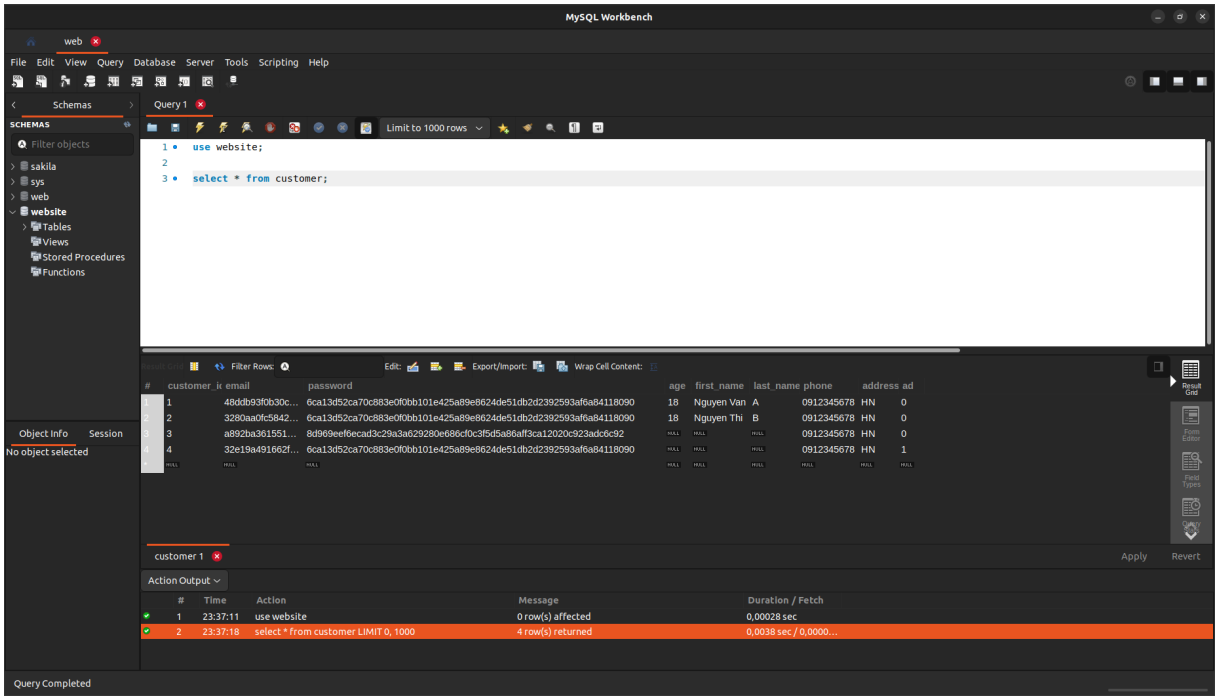
- Bảng (Table): để lưu trữ thông tin người dùng. Mỗi bảng bao gồm các cột (columns) và hàng (rows), trong đó mỗi hàng đại diện cho một bản ghi (record) của dữ liệu thông tin người dùng.

- Cột (Column): Mỗi cột trong bảng định nghĩa một loại dữ liệu cụ thể, chẳng hạn như tên người dùng, địa chỉ email, mật khẩu và thông tin cá nhân khác. Để đảm bảo tính duy nhất của dữ liệu, chúng em sử dụng PRIMARY KEY cho cột customer_id và UNIQUE cho cột email để đảm bảo một người dùng có duy nhất một email.

  Câu lệnh tạo cơ sở dữ liệu:

```
CREATE TABLE `customer` (
  `customer_id` int NOT NULL AUTO_INCREMENT,
  `email` varchar(100) DEFAULT 'example@gmail.com',
  `password` varchar(100) DEFAULT '12345678',
  `age` int DEFAULT NULL,
  `first_name` varchar(100) DEFAULT NULL,
```

```
  `last_name` varchar(100) DEFAULT NULL,
  `phone` varchar(100) DEFAULT '0912345678',
  `address` varchar(100) DEFAULT 'HN',
  `ad` int DEFAULT '0',
  PRIMARY KEY (`customer_id`),
  UNIQUE KEY `email` (`email`)
)
```



Hình 5.1: Hình ảnh lưu trữ tài khoản người dùng trên MySQL

Câu lệnh truy vấn cơ sở dữ liệu để đăng nhập bằng NodeJS:

```
app.post('/api/login', async (req, res) => {
    const { username, password } = req.body;
    connectionMySQL.query(`SELECT * FROM customer WHERE email = SHA2('
        ${username}', 256) AND password = SHA2('${password}', 256)`,
        async (error, results) => {
         if (error) {
             res.status(500).json({ message: 'SQL error' });
             return;
         }
         if (results.length === 0) {
             res.status(401).json({ message: 'sai email hoac mat khau'
                 });
             return;
```

```
        }
        const user = results[0];

        res.status(200).json({ customer_id: user.customer_id, ad: user
            .ad });
    });
});
```

Câu lệnh truy vấn cơ sở dữ liệu để đăng ký bằng NodeJS:

```
app.post('/api/register', async (req, res) => {
    const { username, password, firstName, lastName, address, phone }
        = req.body;
    let query = `SELECT * FROM customer WHERE email = SHA2('${username
        }', 256)`;
    connectionMySQL.query(query, (err, result) => {
        if (err) throw err;
        if(result.length > 0) {
            res.status(400).send('Email da ton tai.');
        } else {
            query = `INSERT INTO customer (email, password, first_name
                , last_name, phone, address) VALUES (SHA2('${username}
                ',256), SHA2('${password}', 256), '${firstName}', '${
                lastName}', '${phone}', '${address}')`;
            connectionMySQL.query(query, (err, result) => {
                if (err) throw err;
                console.log(`Insert data ${username}, ${password} done
                    `);
                res.sendStatus(200);
            });
        }
    });
});
```

Câu lệnh truy vấn cơ sở dữ liệu để cập nhật password bằng NodeJS:

```
app.post('/update/password', async (req, res) => {
    const { password, newPassword, userID } = req.body;
    let query = `SELECT password FROM customer WHERE customer_id = ${
        userID}`;

    connectionMySQL.query(query, (err, result) => {
        if (err) throw err;
        if(result[0].password == password) {
```

```
            query = `UPDATE customer SET password = SHA2('${
                newPassword}', 256) WHERE customer_id = ${userID}`;
            connectionMySQL.query(query, (err, result) => {
                if (err) throw err;
                res.sendStatus(200);
            });
        } else {
            res.status(400).send({ message: 'change password done.' })
                ;
        }
    });
});
```

Câu lệnh truy vấn cơ sở dữ liệu để xem thông tin khách hàng bằng NodeJS:

```
app.post('/user/info', async (req, res) => {
    const { userID } = req.body;
    connectionMySQL.query(`SELECT concat(first_name, last_name) as
        name, phone, email, age, address, ad
                    FROM customer where customer_id = ${userID}`, (
                        error, results) => {
        if (error) {
            console.log(error);
            res.status(500).json({ message: 'SQL error' });
            return;
        }
        if (results.length === 0) {
            res.status(404).json({ message: 'not found' });
            return;
        }
        const userInfo = results[0];
        res.status(200).json({ userInfo });
    });
});
```

5.2 MongoDB

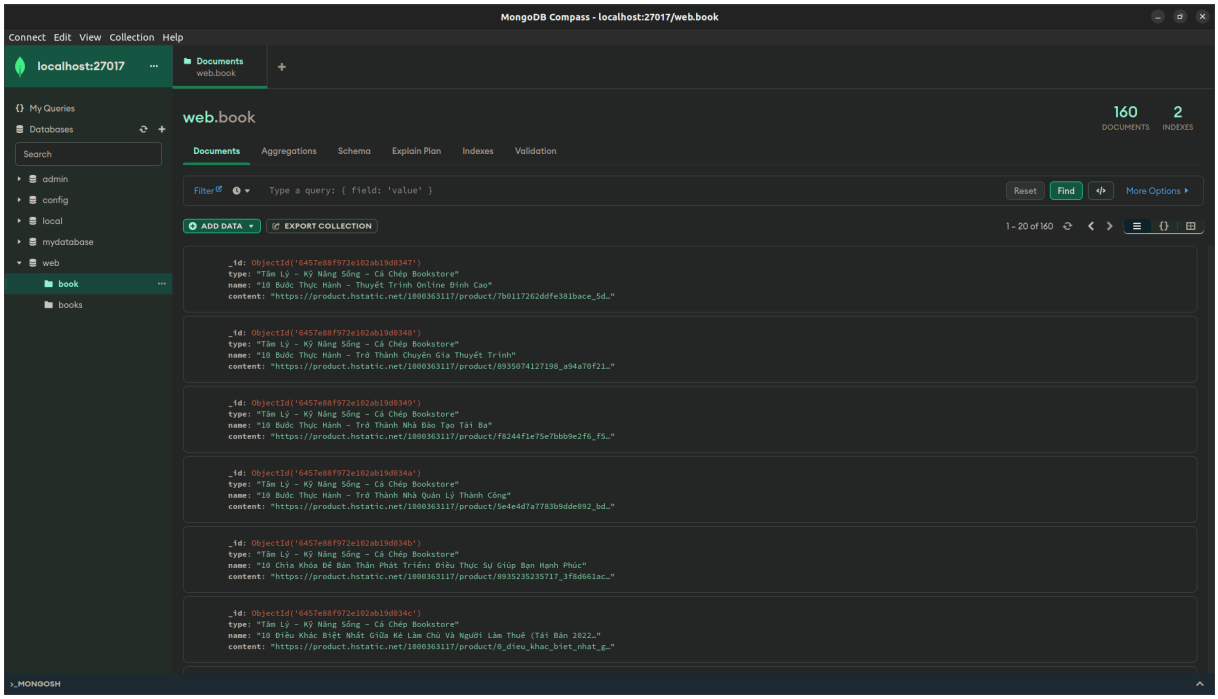
Khi thiết kế dữ liệu MongoDB để lưu trữ thông tin sách, nhóm em có thể sử dụng các tài liệu (documents) để đại diện cho từng cuốn sách. Dưới đây là một phân tích chi tiết về việc thiết kế dữ liệu MongoDB cho việc lưu trữ sách:

- Collection: Trong MongoDB, dữ liệu được tổ chức thành các collection, tương tự

như các bảng trong hệ thống quan hệ. Mỗi collection đại diện cho một tập hợp các tài liệu có cấu trúc tương tự. Trong trường hợp này, chúng em đã tạo một collection với tên "Books"để lưu trữ thông tin sách.

- Document: Mỗi cuốn sách sẽ được đại diện bởi một tài liệu (document) trong collection "Books". Tài liệu này chứa các cặp khóa-giá trị (key-value pairs) để lưu trữ thông tin chi tiết về sách. Các trường thông tin có thể bao gồm tiêu đề (type), tên cuốn sách (name), link ảnh của sách (content).

- ID: Mỗi tài liệu sẽ có một trường duy nhất được gọi là "_id"để định danh duy nhất cho sách đó trong collection. MongoDB cung cấp một giá trị _id tự động nếu không được cung cấp.



Hình 5.2: Hình ảnh lưu trữ sách trên MongoDB

Câu lệnh truy vấn cơ sở dữ liệu để lấy thông tin sách bằng NodeJS:

```
app.post('/book', async function (req, res) {
    const { search } = req.body;
    try {
        const regex = new RegExp(search, 'i');
        const results = await Book.find({
            $or: [
                { name: { $regex: regex } },
                { type: { $regex: regex } }
```

```
            ]
        });
        if(results.length <= 0) {
            try {
                const books = await Book.find({
                    $or: [
                        { name: { $regex: `.*${search.split('').join('.*')}.*`, $options: 'i' } },
                        { type: { $regex: `.*${search.split('').join('.*')}.*`, $options: 'i' } }
                    ]
                });
                return res.send(books);
            } catch (err) {
                console.error(err);
                return res.status(500).send('Internal Server Error');
            }
        }
        return res.send(results);
    } catch (err) {
        console.error(err);
        return res.status(500).send('Internal Server Error');
    }
});
```

5.3 Cassandra

Khi thiết kế dữ liệu Cassandra để lưu trữ lịch sử mượn/trả sách, chúng em đã xem xét mô hình dữ liệu wide column và cấu trúc dữ liệu hướng cột của Cassandra và thấy nó phù hợp với dữ liệu lịch sử mượn trả. Dưới đây là một phân tích chi tiết về việc thiết kế dữ liệu Cassandra cho việc lưu trữ lịch sử mượn/trả sách:

- Bảng (Table): Trong Cassandra, dữ liệu được tổ chức thành các bảng (table). Mỗi bảng đại diện cho một tập hợp các dòng (rows) và các cột (columns) có tên. Trong trường hợp này, chúng em tạo hai bảng có tên "muon"và "tra"để lưu trữ lịch sử mượn/trả sách.

- Cột (Column): Các bảng trong Cassandra có cấu trúc dữ liệu hướng cột (columnar). Mỗi cột được định nghĩa bởi tên cột và giá trị của nó. Trong bảng "muon", các cột bao gồm ID mượn sách (muon_id), ID sách (book_id), ngày mượn (rental_date).

Trong bảng "tra", các cột bao gồm ID mượn sách (muon_id), ID sách (book_id), ngày trả (return_date).

- Khóa chính (Primary Key): Trong Cassandra, chúng ta cần chọn một khóa chính (primary key) để đảm bảo tính duy nhất của mỗi dòng trong bảng. Trong trường hợp này, khóa chính bao gồm: muon_id và tra_id

Câu lệnh tạo cơ sở dữ liệu:

```
CREATE TABLE web.tra (
    tra_id uuid,
    book_id text,
    customer_id int,
    return_date timestamp,
    PRIMARY KEY ((tra_id, book_id))
);

CREATE TABLE web.muon (
    muon_id uuid PRIMARY KEY,
    book_id text,
    customer_id int,
    rental_date timestamp,
    rental_id uuid
)
```

Câu lệnh truy vấn cơ sở dữ liệu để lấy thông tin sách đã mượn bằng NodeJS:

```
app.post('/rental', async (req, res) => {
    const { userID } = req.body;
    let books = [];
    const query = `SELECT * FROM muon WHERE customer_id = ${userID}
       ALLOW FILTERING`;
    connectionCassandra.execute(query)
        .then(async (result) => {
            for (let i = 0; i < result.rows.length; ++i) {
                const book = await Book.findById(new mongoose.Types.
                   ObjectId(result.rows[i].book_id));
                if (book) {
                    books.push(book);
                } else {
```

```
                    console.log("not found");
                }
            }
            res.status(200).json({ books });
        })
        .catch((err) => {
            console.error('Failed to execute query', err);
            res.status(500).json({ message: 'Failed to execute query'
                });
        });
});
```

Câu lệnh truy vấn cơ sở dữ liệu để lấy thông tin sách đã trả bằng NodeJS:

```
app.post('/book/borrow', async (req, res) => {
    const { userID } = req.body;
    let books = [];
    const query = `SELECT * FROM tra WHERE customer_id = ${userID}
        ALLOW FILTERING`;
    connectionCassandra.execute(query)
        .then(async (result) => {
            for (let i = 0; i < result.rows.length; ++i) {
                const book = await Book.findById(new mongoose.Types.
                    ObjectId(result.rows[i].book_id));
                if (book) {
                    books.push(book);
                } else {
                    console.log("not found");
                }
            }
            res.status(200).json({ books });
        })
        .catch((err) => {
            console.error('Failed to execute query', err);
            res.status(500).json({ message: 'Failed to execute query'
                });
        });
});
```

Câu lệnh truy vấn cơ sở dữ liệu để cập nhật thông tin sách đã trả bằng NodeJS:

```
app.post('/update/return', async (req, res) => {
    const { _id, userID } = req.body;

```

```
    const query = `SELECT muon_id FROM muon WHERE book_id = ? AND
        customer_id = ? ALLOW FILTERING`;

    try {
        const result = await connectionCassandra.execute(query, [_id,
            userID], { prepare: true });

        const updateQueries = result.rows.map(row => {
            const deleteQuery = `DELETE FROM muon WHERE muon_id = ?`;
            const insertQuery = `INSERT INTO tra(tra_id, book_id,
                customer_id, return_date) VALUES (uuid(), ?, ?,
                toTimestamp(now()))`;

            return [
                { query: deleteQuery, params: [row.muon_id.buffer] },
                { query: insertQuery, params: [_id.toString(), userID]
                    }
            ];
        }).flat();

        await Promise.all(updateQueries.map(q => connectionCassandra.
            execute(q.query, q.params, { prepare: true })));

        res.status(200).json({ message: 'Success' });
    } catch (err) {
        console.error('Failed to execute queries', err);
        res.status(500).json({ message: 'Failed to execute queries' })
            ;
    }
});
```

Câu lệnh truy vấn cơ sở dữ liệu để cập nhật thông tin sách đã mươn bằng NodeJS:

```
app.post('/update/rental', async (req, res) => {
    const { _id, userID } = req.body;
    const query = `INSERT INTO muon(muon_id, book_id, customer_id,
        rental_date) VALUES (uuid(), ?, ?, toTimestamp(now()))`;
    connectionCassandra.execute(query, [_id.toString(), userID], {
        prepare: true }, function (err) {
        if (err) throw err;
        console.log(`Inserted book with ID ${_id} for ${userID} into
            Cassandra`);
    })
```

```
});
```

Câu lệnh truy vấn cơ sở dữ liệu để lấy thông tin 10 cuốn sách mượn nhiều nhất bằng NodeJS:

```
app.post('/static', async (req, res) => {
    const { limit } = req.body;
    const m = new Map();
    let query = `SELECT book_id FROM tra`;
    try {
        const result = await connectionCassandra.execute(query);
        for(let i = 0; i<result.rows.length; ++i) {
            const key = result.rows[i].book_id;
            if (m.has(key)) {
                m.set(key, { count: m.get(key).count + 1, uniqueCount:
                    m.get(key).uniqueCount });
            } else {
                m.set(key, { count: 1, uniqueCount: 1 });
            }
        }

        query = `SELECT book_id FROM muon`;
        const result1 = await connectionCassandra.execute(query);
        for(let i = 0; i<result1.rows.length; ++i) {
            const key = result1.rows[i].book_id;
            if (m.has(key)) {
                m.set(key, { count: m.get(key).count + 1, uniqueCount:
                    m.get(key).uniqueCount });
            } else {
                m.set(key, { count: 1, uniqueCount: 1 });
            }
        }

        const sortedArray = [...m.entries()].sort((a, b) => b[1].count
            - a[1].count);
        const top10 = sortedArray.slice(0, limit);
        let books = [];
        for (let i = 0; i < top10.length; ++i) {
            const book = await Book.findById(new mongoose.Types.
               ObjectId(top10[i][0]));
            if (book) {
                books.push(book);
            } else {
```

```
                console.log("not found");
            }
        }
        res.status(200).json({ books });
    } catch(err) {
        console.error('Failed to execute query', err);
    }
});
```

Chương 6

Kết luận và định hướng phát triển

6.1 Kết luận

Nhóm đã xây dựng được một số tính năng cơ bản một website quản lý sách cần có như: Qua việc thử nghiệm, nhóm em thấy website đã hoạt động tốt trên các tính năng đó và chưa gắp sự cố nào. Tuy nhiên, website của chúng em còn nhiều hạn chế mà chúng em sẽ khắc phục trong tương lai.

6.2 Định hướng phát triển

Nhóm em dự định phát triểm thêm một số tính năng như: để giúp website hoàn thiện hơn. Đồng thời cũng sẽ thêm một số chức năng bảo mật và sẽ tọa ra một số những ưu đãi khi người dùng mượn sách hoặc có đóng góp cho thư viện.

Tài liệu tham khảo

[1] https://www.w3schools.com/.

[2] https://mongoosejs.com/docs/.

[3] https://viblo.asia/p/su-dung-mysql-voi-node-js-express-6J3Zg2mWKmB.

[4] https://www.npmjs.com/package/mysql.

[5] https://hocwebchuan.com/tutorial/reactjs/reactjs_mysql.php.

[6] https://viblo.asia/p/su-dung-mongodb-voi-nodejs-ZDEeLXRoeJb.

[7] https://getbootstrap.com/docs/3.4/components/.

[8] https://react.dev/learn.

[9] https://cassandra.apache.org/doc/latest/cassandra/cql/index.html.

[10] https://www.instaclustr.com/support/documentation/cassandra/using-cassandra/connect-to-cassandra-with-node-js/.

[11] https://www.instaclustr.com/support/documentation/cassandra/using-cassandra/connect-to-cassandra-with-node-js/.

[12] https://www.npmjs.com/package/cassandra-driver.

[13] https://www.npmjs.com/package/mongoose.