**Báo cáo BTL**

**ĐỀ TÀI: XÂY DỰNG WEBSITE BÁN SÁCH CHO NHÀ SÁCH TRÍ TUỆ**

[CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN 2](#_Toc177942091)

[**1.1.** **Lý do chọn đề tài** 2](#_Toc177942092)

[**1.2.** **Giới thiệu chung** 2](#_Toc177942093)

[**1.3.** **Giới thiệu về công nghệ và ngôn ngữ sử dụng** 2](#_Toc177942094)

[***1.3.1.*** ***Giới thiệu về ASP.NET Core Api*** 2](#_Toc177942095)

[***1.3.2.*** ***Giới thiệu về ReactJS*** 2](#_Toc177942096)

[***1.3.3.*** ***Giới thiệu về SQL Server*** 2](#_Toc177942097)

[CHƯƠNG 2: KẾT QUẢ NGHIÊN CỨU 3](#_Toc177942098)

[**2.1. Phân tích hệ thống** 3](#_Toc177942099)

[**2.2. Tác nhân hệ thống** 3](#_Toc177942100)

[**2.3. Biểu đồ usecase** 3](#_Toc177942101)

[***2.3.1. Biểu đồ usecase tổng quát*** 3](#_Toc177942102)

[***2.3.2. Biểu đồ usecase phân rã*** 3](#_Toc177942103)

[**2.4. Biểu đồ chi tiết các usecase** 3](#_Toc177942104)

[**2.5. Thiết kế cơ sở dữ liệu** 3](#_Toc177942105)

[***2.5.1. Sơ đồ dữ liệu quan hệ*** 3](#_Toc177942106)

[***2.5.2. Chi tiết các bảng dữ liệu*** 3](#_Toc177942107)

[**2.6. Kết quả một số giao diện** 3](#_Toc177942108)

[***2.6.1. Giao diện của khách hang*** 3](#_Toc177942109)

[***2.6.2. Giao diện của quản trị viên*** 3](#_Toc177942110)

[**2.7. Kiểm thử ứng dụng** 3](#_Toc177942111)

[CHƯƠNG 3: KẾT LUẬN VÀ BÀI HỌC KINH NGHIỆM 4](#_Toc177942112)

[**3.1. Kết quả** 4](#_Toc177942113)

[**3.2. Bài học kinh nghiệm** 4](#_Toc177942114)

[Tài liệu tham khảo 5](#_Toc177942115)

**CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN**

* 1. **Lý do chọn đề tài**

Trong thời đại số hóa ngày nay, thương mại điện tử đang phát triển mạnh mẽ, mang lại nhiều tiện ích cho người dùng trong việc mua sắm trực tuyến. Một trong những lĩnh vực nổi bật hiện nay là kinh doanh sách trực tuyến, khi mà nhu cầu đọc và sở hữu sách ngày càng cao, song song với việc người dùng muốn tiếp cận các sản phẩm một cách nhanh chóng và tiện lợi. Việc xây dựng một website bán sách không chỉ đáp ứng nhu cầu của người tiêu dùng mà còn mở ra cơ hội kinh doanh hiệu quả trong lĩnh vực này.

Chọn đề tài này là vì tính thực tiễn và tiềm năng ứng dụng cao trong cuộc sống. Qua việc phát triển website bán sách, chúng tôi có thể áp dụng các kiến thức về lập trình web, quản lý cơ sở dữ liệu, cũng như các kỹ năng về thiết kế giao diện và trải nghiệm người dùng. Đề tài này cũng là cơ hội để chúng tôi nghiên cứu sâu hơn về cách thức vận hành của một hệ thống thương mại điện tử, từ quản lý sản phẩm, kho hàng cho đến tích hợp các phương thức thanh toán và chăm sóc khách hàng.

Hơn nữa, sách không chỉ là một sản phẩm có giá trị văn hóa mà còn là cầu nối tri thức. Phát triển một website bán sách sẽ giúp người dùng dễ dàng tiếp cận với nhiều thể loại sách khác nhau, từ sách giáo dục, giải trí đến nghiên cứu học thuật. Đây chính là động lực quan trọng khiến chúng tôi quyết định lựa chọn đề tài này.

* 1. **Giới thiệu chung**

Dự án bán sách điện tử nhằm xây dựng một nền tảng trực tuyến tiên tiến, nơi người dùng có thể dễ dàng tìm kiếm, mua và đánh giá các cuốn sách điện tử mà họ yêu thích. Với sự phát triển của công nghệ và nhu cầu đọc sách ngày càng gia tăng, dự án hướng đến việc cung cấp một dịch vụ thuận tiện, nhanh chóng và an toàn cho người dùng.

Hệ thống cung cấp kho sách phong phú, đa dạng về thể loại, ngôn ngữ, phục vụ mọi đối tượng. Giao diện người dùng được thiết kế thân thiện, dễ sử dụng, giúp người dùng dễ dàng tìm kiếm và mua sách một cách nhanh chóng. Ngoài ra, hệ thống cũng sẽ tích hợp các tính năng như đánh giá và bình luận sách cho phép người dùng chia sẻ ý kiến và trải nghiệm của mình với cộng đồng.

Dự án cũng chú trọng đến việc bảo vệ bản quyền tác giả và đảm bảo quyền lợi của người mua. Hệ thống thanh toán an toàn, bảo mật sẽ đảm bảo thông tin cá nhân và tài khoản của người dùng không bị xâm phạm. Mục tiêu cuối cùng là xây dựng một nền tảng bán sách điện tử uy tín và chất lượng, thúc đẩy văn hóa đọc và tạo điều kiện cho người dùng tiếp cận tri thức một cách dễ dàng và hiệu quả nhất.

* 1. **Giới thiệu về công nghệ và ngôn ngữ sử dụng**
     1. ***Giới thiệu về ASP.NET Core Api***

ASP.NET Core là một framework mã nguồn mở và đa nền tảng (cross-platform) của Microsoft, được thiết kế để phát triển các ứng dụng web hiện đại, bao gồm các dịch vụ Web API RESTful. ASP.NET Core API được sử dụng để xây dựng các dịch vụ API có thể truy cập từ các ứng dụng khác thông qua giao thức HTTP.

1. Kiến trúc của ASP.NET Core API

Kiến trúc của ASP.NET Core API dựa trên mô hình MVC (Model-View-Controller), nhưng khi xây dựng API, chỉ cần tập trung vào hai thành phần chính là Controller và Model:

Model: Đại diện cho dữ liệu hoặc các đối tượng mà API xử lý.

Controller: Tiếp nhận và xử lý các yêu cầu HTTP, trả về phản hồi dưới dạng JSON, XML hoặc các định dạng dữ liệu khác.

1. Đặc điểm nổi bật của ASP.NET Core API

Đa nền tảng (Cross-Platform): Có thể chạy trên nhiều hệ điều hành như Windows, macOS và Linux.

Hiệu suất cao (High Performance): Tối ưu về tốc độ và hiệu suất so với các framework trước đây.

Dependency Injection: Hỗ trợ tích hợp sẵn, giúp quản lý các dependencies dễ dàng.

Modular: Các tính năng chỉ được thêm khi cần, giúp giảm thiểu kích thước và tối ưu hóa hiệu suất ứng dụng.

Middleware Pipeline: Sử dụng các middleware để quản lý và xử lý các yêu cầu HTTP.

Tích hợp OpenAPI (Swagger): Giúp tạo và kiểm thử tài liệu API một cách dễ dàng và tự động.

1. Cấu trúc của một dự án ASP.NET Core API

Một dự án ASP.NET Core API điển hình thường có cấu trúc như sau:

Controllers: Chứa các controller xử lý các yêu cầu từ client.

Models: Chứa các lớp mô tả dữ liệu mà ứng dụng xử lý.

Startup.cs: Nơi cấu hình các dịch vụ và middleware cho ứng dụng.

Program.cs: Tệp điểm vào của ứng dụng, nơi khởi tạo web host và chạy ứng dụng.

appsettings.json: Chứa các cấu hình ứng dụng, như kết nối cơ sở dữ liệu và các thông số hệ thống.

1. Các thành phần quan trọng trong ASP.NET Core API

Routing: Định nghĩa các đường dẫn (route) để ánh xạ các yêu cầu HTTP đến các phương thức trong controller.

Action Methods: Các phương thức trong controller xử lý các yêu cầu HTTP với các verb như GET, POST, PUT, DELETE.

Middleware: Xử lý các yêu cầu HTTP trước khi chúng đến controller, chẳng hạn như middleware về xác thực, ủy quyền, và xử lý lỗi.

Dependency Injection: Tích hợp sẵn, giúp quản lý và cung cấp các dịch vụ trong toàn bộ ứng dụng.

5. OpenAPI/Swagger trong ASP.NET Core API

ASP.NET Core API hỗ trợ Swagger (OpenAPI), giúp tài liệu hóa các endpoint API và cho phép kiểm thử trực tiếp từ giao diện Swagger UI. Điều này rất hữu ích cho việc phát triển và kiểm thử API, đặc biệt là khi làm việc với các dịch vụ phức tạp.

* + 1. ***Giới thiệu về ReactJS***
       1. ReactJS

a) Tổng quan

React còn được gọi là ReactJS hoặc React.js, là 1 thư viện JavaScript mã nguồn mở được phát triển bởi đội ngũ kỹ sư đến từ Facebook; nó được giới thiệu vào năm 2011, tuy nhiên đến năm 2013 mới được giới thiệu cho cộng đồng lập trình viên.

React là một thư viện UI phát triển tại Facebook để hỗ trợ việc xây dựng những thành phần (components) UI có tính tương tác cao, có trạng thái và có thể sử dụng lại được. React được sử dụng tại Facebook trong production, và www.instagram.com được viết hoàn toàn trên React.

Một trong những điểm hấp dẫn của React là thư viện này không chỉ hoạt động trên phía client, mà còn được render trên server và có thể kết nối với nhau. React so sánh sự thay đổi giữa các giá trị của lần render này với lần render trước và cập nhật ít thay đổi nhất trên DOM. Trươc khi đến cài đặt và cấu hình, chúng ta sẽ đi đến một số khái niệm cơ bản:

React.js là một thư viện Javascript đang nổi lên trong những năm gần đây với xu hướng Single Page Application. Trong khi những framework khác cố gắng hướng đến một mô hình MVC hoàn thiện thì React nổi bật với sự đơn giản và dễ dàng phối hợp với những thư viện Javascript khác. Nếu như AngularJS là một Framework cho phép nhúng code javasscript trong code html thông qua các attribute như ng-model, ng-repeat...thì với react là một library cho phép nhúng code html trong code javascript nhờ vào JSX, bạn có thể dễ dàng lồng các đoạn HTML vào trong JS. Tích hợp giữa javascript và HTML vào trong JSX làm cho các component dễ hiểu hơn

React JS sử dụng Virtual DOM (DOM ảo) để cải thiện vấn đề này.Virtual DOM là một object Javascript, mỗi object chứa đầy đủ thông tin cần thiết để tạo ra một DOM, khi dữ liệu thay đổi nó sẽ tính toán sự thay đổi giữa object và tree thật, điều này sẽ giúp tối ưu hoá việc re-render DOM tree thật.

React sử dụng kiến trúc dựa trên component, trong đó giao diện người dùng được chia thành các thành phần nhỏ, độc lập và có thể tái sử dụng. Mỗi component trong React là một lớp hoặc một hàm JavaScript, chứa logic và giao diện riêng biệt. Các component có thể lồng vào nhau để tạo ra cấu trúc UI phức tạp.

b) Nguyên lý xây dựng

Nguyên lý xây dựng của React dựa trên components (component-based approach), có thể tái sử dụng và phù hợp với ứng dụng 1 trang (Single Page Application – SPA). React giúp lập trình viên xây dựng giao diện người dùng dựa trên JSX (môt cú pháp mở rộng của JavaScript), tạo ra các DOM ảo (virtual DOM) để tối ưu việc render 1 trang web.

React sử dụng kiến trúc dựa trên component, trong đó giao diện người dùng được chia thành các thành phần nhỏ, độc lập và có thể tái sử dụng. Mỗi component trong React là một lớp hoặc một hàm JavaScript, chứa logic và giao diện riêng biệt. Các component có thể lồng vào nhau để tạo ra cấu trúc UI phức tạp.

JSX:

JSX là một dạng ngôn ngữ cho phép viết các mã HTML trong Javascript. Đặc điểm: Faster: Nhanh hơn. JSX thực hiện tối ưu hóa trong khi biên dịch sang mã Javacsript. Các mã này cho thời gian thực hiện nhanh hơn nhiều so với một mã tương đương viết trực tiếp bằng Javascript. Safer: an toàn hơn. Ngược với Javascript, JSX là kiểu statically-typed, nghĩa là nó được biên dịch trước khi chạy, giống như Java, C++. Vì thế các lỗi sẽ được phát hiện ngay trong quá trình biên dịch. Ngoài ra, nó cũng cung cấp tính năng gỡ lỗi khi biên dịch rất tốt. Easier: Dễ dàng hơn. JSX kế thừa dựa trên Javascript, vì vậy rất dễ dàng để cho các lập trình viên Javascripts có thể sử dụng

Components:

React được xây dựng xung quanh các component, chứ không dùng template như các framework khác. Trong React, chúng ta xây dựng trang web sử dụng những thành phần (component) nhỏ. Chúng ta có thể tái sử dụng một component ở nhiều nơi, với các trạng thái hoặc các thuộc tính khác nhau, trong một component lại có thể chứa thành phần khác. Mỗi component trong React có một trạng thái riêng, có thể thay đổi, và React sẽ thực hiện cập nhật component dựa trên những thay đổi của trạng thái. Mọi thứ React đều là component. Chúng giúp bảo trì mã code khi làm việc với các dự án lớn. Một react component đơn giản chỉ cần một method render. Có rất nhiều methods khả dụng khác, nhưng render là method chủ đạo.

c) Ưu nhược điểm

Ưu điểm:

Dễ học và sử dụng: ReactJS có cú pháp đơn giản và dễ hiểu, đặc biệt là đối với những người đã quen thuộc với JavaScript. JSX, một phần mở rộng của JavaScript, cho phép lập trình viên viết mã HTML trong JavaScript, làm cho quá trình phát triển trở nên trực quan và dễ dàng hơn. Điều này giúp các lập trình viên mới dễ dàng tiếp cận và bắt đầu sử dụng React nhanh chóng.

Viết component dễ dàng hơn: Cấu trúc component-based của ReactJS giúp việc viết và quản lý các thành phần UI trở nên dễ dàng và hiệu quả hơn. Lập trình viên có thể tách biệt các phần của giao diện người dùng thành các component nhỏ hơn, dễ dàng kiểm thử và bảo trì. Việc viết các component này trở nên đơn giản nhờ JSX, cho phép kết hợp JavaScript và HTML một cách tự nhiên và dễ hiểu.

Hiệu suất tốt hơn với Virtual DOM: ReactJS sử dụng Virtual DOM để tối ưu hóa hiệu suất ứng dụng. Thay vì cập nhật toàn bộ DOM thật mỗi khi có thay đổi, React sẽ cập nhật Virtual DOM trước, sau đó so sánh với DOM thật để chỉ thay đổi những phần cần thiết. Điều này giúp giảm thiểu số lượng thao tác trên DOM thật, từ đó cải thiện tốc độ và hiệu suất của ứng dụng.

Thân thiện với SEO: Một trong những thách thức lớn đối với các ứng dụng JavaScript truyền thống là khả năng thân thiện với SEO. ReactJS giải quyết vấn đề này bằng cách cho phép render trên server (Server-Side Rendering – SSR), giúp các công cụ tìm kiếm dễ dàng lập chỉ mục các trang web. Điều này làm tăng khả năng hiển thị và xếp hạng của trang web trên các công cụ tìm kiếm, từ đó thu hút lượng truy cập lớn hơn.

Dễ dàng bảo trì và mở rộng: Cấu trúc dựa trên component của ReactJS giúp mã nguồn dễ dàng quản lý, bảo trì và mở rộng. Mỗi component có thể được phát triển, kiểm thử và bảo trì độc lập, giúp việc sửa lỗi và thêm tính năng mới trở nên đơn giản hơn. Điều này cũng giúp các nhóm phát triển có thể làm việc song song trên các phần khác nhau của ứng dụng mà không gây ra xung đột.

Hỗ trợ đa nền tảng cả web và mobile: Với React Native, một framework dựa trên React, lập trình viên có thể phát triển các ứng dụng di động cho cả iOS và Android sử dụng cùng một codebase. Điều này giúp tiết kiệm thời gian và nguồn lực, đồng thời đảm bảo tính nhất quán của ứng dụng trên các nền tảng khác nhau.

Nhược điểm:

Tốc độ phát triển nhanh: React liên tục được cập nhật, dẫn đến việc các lập trình viên phải theo dõi và cập nhật kiến thức thường xuyên. Điều này có thể khiến việc duy trì dự án lâu dài trở nên khó khăn.

Tài liệu không đầy đủ cho một số phiên bản mới: Mặc dù cộng đồng React rất lớn, nhưng với các phiên bản mới, tài liệu hoặc hướng dẫn có thể không theo kịp, dẫn đến khó khăn trong việc tra cứu thông tin.

Cấu trúc phức tạp với dự án lớn: React chỉ là một thư viện UI, do đó, khi dự án lớn, việc tổ chức và quản lý code trở nên phức tạp. Các công cụ và thư viện bổ trợ như Redux, React Router có thể làm tăng sự phức tạp.

Chỉ tập trung vào UI: React chỉ xử lý giao diện người dùng, vì vậy khi phát triển các ứng dụng phức tạp, bạn cần sử dụng thêm các công cụ khác như Redux hoặc MobX để quản lý state, hay Next.js để xử lý server-side rendering (SSR).

SEO không tối ưu (với React Client-side Rendering): Các ứng dụng React thường render trên client-side, điều này có thể không tối ưu cho SEO. Tuy nhiên, điều này có thể được cải thiện bằng cách sử dụng các framework như Next.js để thực hiện server-side rendering.

JSX có thể khó hiểu với người mới: JSX, cú pháp kết hợp giữa HTML và JavaScript, có thể khó hiểu và không quen thuộc với những người mới học.

* + - 1. Material UI

1. Tổng quan:

Material UI là một thư viện các React Component đã được tích hợp thêm cả Google's Material Design. Theo như giới thiệu trên trang chủ thì được xây dựng nhờ tình cảm với React và Google's Material Design =)) Do đó mà phần hướng dẫn trên trang chủ của Material UI cũng đã nói nên sử dụng Material UI với React.

Material UI đem đến cho bạn và trang web của bạn một giao diện hoàn toàn mới, với những button, textfield, toogle... được design theo một phong cách mới lạ, thay vì việc nhà nhà người người dùng Bootstrap như hiện nay.

Cách cài đặt:

Cài đặt bằng npm rất đơn giản chỉ với một câu lệnh: npm install material-ui

Và để sử dụng cũng rất đơn giản chỉ cần 1 dòng require: var mui = require("material-ui");

Components:

1. Một số components thông dụng

Layout

Layout, tạm dịch là bố cục, là cách bố trí các đối tượng trên 1 trang web. Material-UI cung cấp nhiều component tiện dụng hỗ trợ việc sắp xếp bố cục tr nen dễ dàng hơn như: Box, Container, Grid, Stack, Image List, Stack, …

Navigations

Material-UI cung cấp sẵn 1 số component phục vụ cho mục đích điều hướng (đối tượng hoặc trang web) như: Bottom Navigation, Breadcrumbs, Drawer, Link, Menu, Stepper, Tabs, Pagination, Speed dial, …

Surfaces

Material-UI cung cấp 1 số component dưới dạng khung với bố cục có sẵn, chỉ cần truyền giá trị cần hiển thị lên giao diện như: Accordion, App Bar, Card, Paper, …

Feedback

Material-UI cung cấp các component để thể hiện sự phản hồi thao tác của người dùng.

Có một số components thuộc nhóm này như: Alert, Skeleton, Progress, Dialog, Snackbar, Backdrop, …

Cách sử dụng

Để sử dụng bất kỳ component nào từ material UI, bước đầu tiên cần là phải import chúng ra từ thư viện, sau đó có thể gọi trực tiếp vào phần return của function.

1. Ưu nhược điểm

Ưu điểm

Giao diện đẹp: thư viện cung cấp các component tùy chỉnh một cách tỉ mỉ, đảm bảo rằng mọi thành phần đều đáp ứng các tiêu chuẩn cao nhất về hình thức và chức năng.

Khả năng tùy chỉnh: Material UI cung cấp khả năng tùy chỉnh cao từ giao diện cho đến chức năng của mỗi component.

Cộng đồng: thư viện có một cộng đồng sử dụng và hỗ trợ lớn nhất trong hệ sinh thái React. Nó có thể coi là lâu đời gần bằng React – từ năm 2014 và luôn được tin tưởng cho đến tận nay.

Bên cạnh đó, MUI cũng ghi nhận hơn 2500 các nhà đóng góp về mã nguồn của họ – một con số rất lớn cho một open-source.

Nhược điểm

Kích thước lớn: MUI có kích thước tương đối lớn do chứa nhiều thành phần, dẫn đến việc tăng kích thước của gói JavaScript tổng thể. Điều này có thể ảnh hưởng đến hiệu suất của ứng dụng, đặc biệt khi không sử dụng tree-shaking hiệu quả.

Thời gian tải trang chậm: Do kích thước lớn và việc tải nhiều thành phần CSS, ứng dụng có thể gặp vấn đề về tốc độ tải trang, đặc biệt là khi không được tối ưu hóa.

Độ phức tạp khi tùy chỉnh: Mặc dù MUI cung cấp các công cụ tùy chỉnh mạnh mẽ (chủ yếu qua hệ thống theme), việc tùy chỉnh theo đúng ý đôi khi có thể phức tạp. Cấu trúc của theme và overrides đôi khi khó hiểu đối với những người chưa quen.

Phụ thuộc vào CSS-in-JS: MUI sử dụng CSS-in-JS để tạo style động. Điều này có thể làm giảm hiệu suất trong các ứng dụng lớn và phức tạp do phải tạo CSS runtime. Nó cũng khiến một số lập trình viên không quen hoặc không thích làm việc với CSS-in-JS gặp khó khăn.

Tính đồng nhất trong thiết kế: MUI tuân thủ nghiêm ngặt Material Design, vì vậy nếu bạn cần một giao diện thiết kế hoàn toàn khác biệt, việc tùy chỉnh toàn bộ giao diện có thể mất thời gian và không dễ dàng.

Phụ thuộc vào React: MUI chỉ hỗ trợ cho React, điều này làm hạn chế sử dụng nếu bạn đang làm việc với các framework hoặc thư viện khác ngoài React.

* + 1. ***Giới thiệu về SQL Server***

SQL là một công cụ quản lý dữ liệu được sử dụng phổ biến ở nhiều lĩnh vực. Đầu tiên SQL được sử dụng trong các ngôn ngữ quản lý CSDL và chạy trên máy đơn lẻ. Do sự phát triển nhanh chóng của nhu cầu xây dựng những CSDL lớn theo mô hình khách/chủ (Client/Server), nhiều phần mềm sử dụng ngôn ngữ SQL đã ra đời mà điển hình là MS SQL Server, Oracle, Sybase... Trong mô hình khách/chủ, toàn bộ CSDL được tập trung lưu trữ trên máy chủ (Server), mọi thao tác xử lý dữ liệu được thực hiện trên máy chủ bằng các lệnh SQL. Máy trạm (Client) chỉ dùng để cập nhật dữ liệu cho máy chủ hoặc lấy thông tin từ máy chủ. Trên lĩnh vực đang phát triển hiện nay là Internet, ngôn ngữ SQL lại càng đóng vai trò quan trọng hơn. Nó được sử dụng để nhanh chóng tạo ra các trang Web động (Dynamic Web Page). Trang Web động thường có nội dung được lấy ra từ CSDL. SQL có thể được sử dụng như một chất keo kết dính giữa CSDLvà trang Web. Khi người dùng yêu cầu, SQL sẽ thực hiện việc truy cập thông tin trong CSDL trên máy chủ và hiển thị kết quả trên trang Web. Và SQL cũng là công cụ để cập nhật thông tin cho CSDLđó.

* Chức năng chính của SQL Server:

Lưu trữ dữ liệu: SQL Server lưu trữ dữ liệu trong các bảng, với khả năng liên kết dữ liệu giữa các bảng thông qua khóa chính và khóa ngoại.

Quản lý giao dịch (ACID): SQL Server đảm bảo các giao dịch có tính Atomicity, Consistency, Isolation, và Durability, giúp duy trì tính toàn vẹn và nhất quán của dữ liệu.

Phân tích dữ liệu: Cung cấp các công cụ phân tích và tạo báo cáo mạnh mẽ như SQL Server Reporting Services (SSRS) và SQL Server Analysis Services (SSAS).

Bảo mật: SQL Server tích hợp các cơ chế bảo mật mạnh mẽ như mã hóa, kiểm soát truy cập, và các chính sách phân quyền.

* Ưu điểm của SQL Server:

Hiệu suất cao: SQL Server hỗ trợ các công nghệ tối ưu như In-Memory OLTP và Columnstore Index, giúp tăng tốc độ xử lý dữ liệu lớn.

Bảo mật mạnh mẽ: SQL Server cung cấp các công cụ bảo mật toàn diện như Transparent Data Encryption (TDE), mã hóa dữ liệu và kiểm soát quyền truy cập chi tiết.

Khả năng tích hợp cao: Dễ dàng tích hợp với các sản phẩm khác của Microsoft như Power BI, Azure, SharePoint và .NET, giúp xây dựng hệ sinh thái ứng dụng hoàn chỉnh.

Khả năng mở rộng: SQL Server có thể mở rộng từ các ứng dụng nhỏ trên SQL Server Express đến các hệ thống lớn với hàng triệu giao dịch mỗi ngày qua phiên bản Enterprise.

Dễ sử dụng: SQL Server Management Studio (SSMS) cung cấp giao diện đồ họa thân thiện giúp quản trị và thao tác dữ liệu dễ dàng hơn.

* Nhược điểm của SQL Server:

Chi phí cao: Các phiên bản cao cấp như SQL Server Enterprise có chi phí bản quyền đắt đỏ, đặc biệt là đối với các doanh nghiệp nhỏ.

Yêu cầu tài nguyên cao: SQL Server cần một lượng tài nguyên phần cứng đáng kể (bộ nhớ, CPU) để vận hành hiệu quả, đặc biệt khi xử lý khối lượng dữ liệu lớn.

Hạn chế trên các nền tảng phi Windows: Mặc dù SQL Server đã hỗ trợ Linux, nhưng hiệu năng và tính năng đôi khi không được tối ưu bằng phiên bản trên Windows.

**CHƯƠNG 2: KẾT QUẢ NGHIÊN CỨU**

## **2.1. Phân tích hệ thống**

2.1.1. Sơ đồ kiến trúc tổng quan

A diagram of a server layer

Description automatically generated

Client Layer:

- Người dùng tương tác với giao diện web (ReactJs).

- Gửi yêu cầu và nhận phản hồi từ Server Layer.

Server Layer:

- Xử lý logic nghiệp vụ.

- Xác thực và phân quyền.

- Kết nối với Database Layer để lấy hoặc lưu trữ dữ liệu.

Database Layer:

- Lưu trữ dữ liệu về người dùng, sách, đơn hàng, ...

- Sử dụng SQL Server làm cơ sở dữ liệu.

**2.2 Sơ đồ luồng dữ liệu (Data Flow Diagram - DFD)**

A diagram of a web server

Description automatically generated

## **2.2. Tác nhân hệ thống**

Các tác nhân hệ thống bao gồm:  
Người dùng

Quản trị viên

CSDL

Các usecase phần frontend:

Đăng ký

Đăng nhập

Tìm kiếm sách

Xem chi tiết sách

Đánh giá và bình luận sách

Xem giỏ hàng

Xóa sách khỏi giỏ hàng

Thêm sách vào giỏ hàng

Thanh toán

Xem lịch sử đơn hàng

Đặt hàng

Hủy đơn hàng

Các usecase phía backend

Đăng nhập

Quản lý đơn hàng

QUản lý người dùng

Quản lý khuyến mãi

Quản lý sách

Quản lý đánh giá và bình luận

Quản lý phan quyền

## **2.3. Biểu đồ usecase**

### ***2.3.1. Biểu đồ usecase tổng quát***



### ***2.3.2. Biểu đồ usecase phân rã***

a) Biểu đồ usecase phía frontend



b) Biểu đồ usecase phía backend



## **2.4. Mô tả chi tiết các usecase**

2.4.1. Mô tả chi tiết usecase đăng nhập

* Tên của Use Case: Đăng nhập
* Mô tả vắn tắt : Use case này cho phép Khách hàng dùng tài khoản để đăng nhập vào hệ thống.
* Luồng các sự kiện
* Luồng cơ bản

1. Use case này bắt đầu khi khách hàng lần đầu đăng nhập vào hệ thống và kích vào Login, hệ thống sẽ hiển thị ô nhập dữ liệu như tên tài khoản và mật khẩu.
2. Khách hàng nhập tên tài khoản và mật khẩu xong click đăng nhập. Use case kết thúc.

* Luồng rẽ nhánh

1. Tại bước 3.1.1 trong luồng cơ bản nếu người dùng nhập sai tên tài khoản hoặc mật khẩu thì hệ thống sẽ thông báo lỗi và yêu cầu nhập lại. Khách hàng có thể nhập lại để tiếp tục.
2. Tại bất kì thời điểm nào trong quá trình thực thi use case nếu không thể kết nối với cơ sở dữ liệu thì hệ thống sẽ hiển thị thống báo lỗi và use case kết thúc.

* Các yêu cầu đặc biệt: Không có
* Tiền điều kiện: Không có.
* Hậu điều kiện: Nếu use case kết thúc thành công, đăng nhập vào hệ thống.
* Các điểm mở rộng: Không có.

2.4.1. Mô tả chi tiết usecase đăng ký

* Tên Use Case: Đăng ký tài khoản người dùng
* Mô tả vắn tắt: Use case này mô tả quá trình đăng ký tài khoản người dùng trên ứng dụng. Người dùng có thể tạo một tài khoản mới bằng cách cung cấp thông tin cá nhân cần thiết và xác nhận tài khoản của mình.
* Luồng các sự kiện:
* Luồng cơ bản:

1. Use case này bắt đầu khi người dùng nhấp vào nút “Đăng Ký” trên màn hình đăng nhập. Hệ thống hiển thị ra màn hình đăng ký cho khách hàng.
2. Người dùng điền đầy đủ các thông tin yêu cầu trên màn hình sau đó nhấn “ĐĂNG KÝ”.
3. Hệ thống lưu thông tin người dùng vào bảng TAIKHOAN và bảng KHACHHANG sau đó hiển thị thông báo “Đăng ký thành công” và chuyển sang màn hình đăng nhập.

* Các luồng rẽ nhánh :

1. Tại bước b trong luồng cơ bản nếu thông tin cung cấp không hợp lệ hoặc thiếu, hệ thống hiển thị thông báo lỗi và yêu cầu người dùng cung cấp lại thông tin cần thiết.
2. Tại bước b nếu email đã được sử dụng cho tài khoản khác, hệ thống hiển thị thông báo lỗi và yêu cầu người dùng sử dụng một địa chỉ email khác.
3. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được với có sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

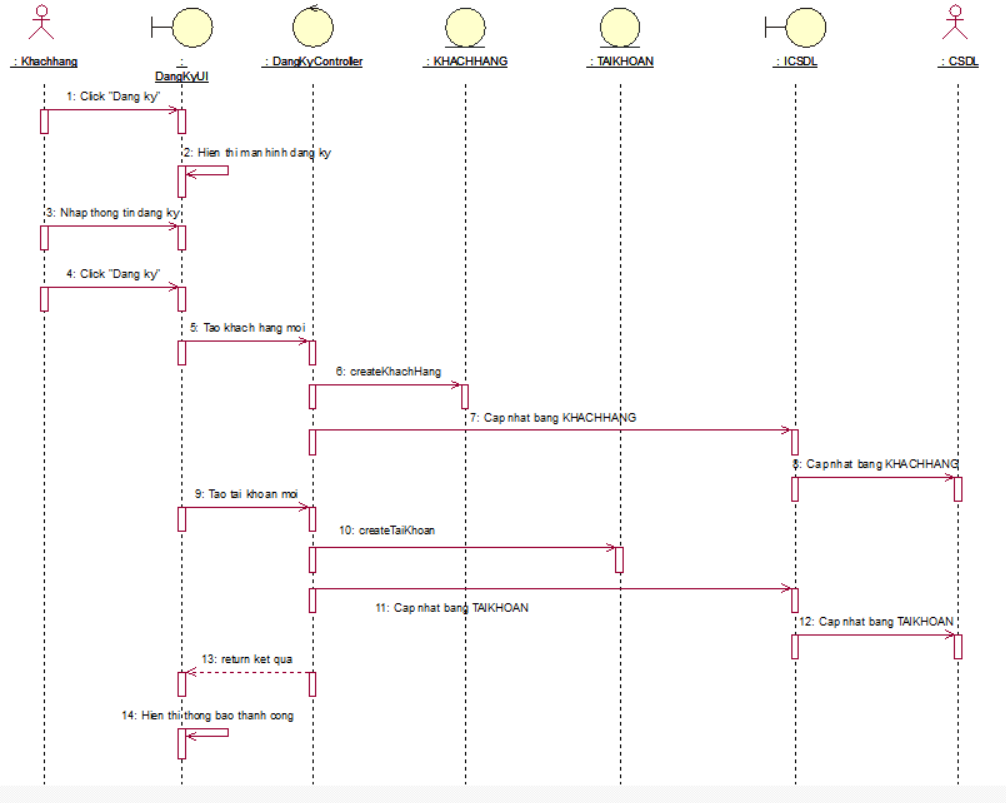
* Các yêu cầu đặc biệt: Yêu cầu xác thực email: Hệ thống cần gửi một email xác nhận đến địa chỉ email được cung cấp để đảm bảo tính duy nhất của email.
* Tiền điều kiện:
* Hệ thống đăng ký tài khoản phải được triển khai và sẵn sàng hoạt động.
* Giao diện đăng ký tài khoản phải có sẵn cho người dùng truy cập.
* Hậu điều kiện: Use case kết thúc thành công thì thông tin sẽ được lưu lại trong cơ sở dữ liệu.
* Điểm mở rộng: Không có

2.5. Biểu đồ trình tự các usecase

2.5.1. Biểu đồ trình tự usecase đăng nhập



2.5.2. Biểu đồ trình tự usecase đăng ký



## **2.5. Thiết kế cơ sở dữ liệu**

### ***2.5.1. Sơ đồ dữ liệu quan hệ***

### ***2.5.2. Chi tiết các bảng dữ liệu***

## **2.6. Kết quả một số giao diện**

### ***2.6.1. Giao diện của khách hang***

### ***2.6.2. Giao diện của quản trị viên***

## **2.7. Kiểm thử ứng dụng**

**CHƯƠNG 3: KẾT LUẬN VÀ BÀI HỌC KINH NGHIỆM**

## **3.1. Kết quả**

## **3.2. Bài học kinh nghiệm**

**Tài liệu tham khảo**