

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH  
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**--------------------------------



**XÂY DỰNG WEBSITE THƯƠNG MẠI ĐIỆN TỬ SỬ DỤNG**

**CÔNG NGHỆ MERN STACK**

**Môn học: Các công nghệ phần mềm mới**

**GVHD: Nguyễn Minh Đạo**

**Sinh viên thực hiện: Nguyễn Đình Hiếu**

**MSSV: 20142498**

**HỌC KỲ 1 - NĂM HỌC: 2023-2024**

**TP. HỒ CHÍ MINH – 2023**

**ĐIỂM SỐ**

|  |  |  |  |
| --- | --- | --- | --- |
| TIÊU CHÍ | NỘI DUNG | TRÌNH BÀY | TỔNG |
| ĐIỂM |  |  |  |

**NHẬN XÉT CỦA GIÁO VIÊN:**

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

TP. Hồ Chí Minh, tháng 11 năm 2023

Giáo viên chấm điểm

**Nguyễn Minh Đạo**

**LỜI CẢM ƠN**

Để hoàn thành tốt đề tài và bài báo cáo này, em xin gửi lời cảm ơn chân thành đến giảng viên, thầy Nguyễn Minh Đạo, người đã trực tiếp hỗ trợ chúng em trong suốt quá trình làm báo cáo. Em cảm thầy đã đưa ra những lời khuyên từ kinh nghiệm thực tiễn của mình để định hướng cho em đi đúng với yêu cầu của báo cáo, luôn giải đáp thắc mắc và đưa ra những góp ý, chỉnh sửa kịp thời giúp em khắc phục nhược điểm và hoàn thành tốt bài báo cáo.

Em cũng xin gửi lời cảm ơn chân thành các quý thầy cô trong khoa công nghệ thông tin nói chung và ngành Công Nghệ Thông Tin nói riêng đã tận tình truyền đạt những kiến thức cần thiết giúp em có nền tảng để làm thực tập, đã tạo điều kiện để em có thể hoàn thành tốt thực tập và báo cáo này.

Bài báo cáo được em thực hiện trong khoảng thời gian ngắn, với những kiến thức và ngôn từ còn hạn chế cùng nhiều hạn chế khác. Do đó, trong quá trình làm có những thiếu sót là điều không thể tránh khỏi nên em rất mong nhận được những ý kiến đóng góp quý báu của các quý thầy cô để kiến thức của em được hoàn thiện hơn và em có thể làm tốt hơn nữa trong những lần sau. Em xin chân thành cảm ơn.

Cuối lời, em kính chúc quý thầy, quý cô luôn dồi dào sức khỏe và thành công hơn nữa trong sự nghiệp trồng người. Một lần nữa em xin chân thành cảm ơn.

**MỤC LỤC**

# **CHƯƠNG 1: MỞ ĐẦU**

* 1. **Giới thiệu về chủ đề:**

Thương mại điện tử (E-commerce) là một trong những lĩnh vực phát triển mạnh mẽ nhất trong những năm gần đây. Với sự phát triển của công nghệ thông tin, người tiêu dùng ngày càng có xu hướng mua sắm trực tuyến nhiều hơn. Điều này dẫn đến nhu cầu xây dựng website thương mại điện tử ngày càng tăng cao. MERN Stack là một trong những công nghệ được sử dụng phổ biến nhất trong việc xây dựng website thương mại điện tử. MERN là viết tắt của MongoDB, Express.js, React.js và Node.js. Mỗi công nghệ trong MERN Stack đều có những ưu điểm và tính năng riêng, giúp tạo nên một giải pháp toàn diện cho việc xây dựng website thương mại điện tử. Với những ưu điểm của MERN Stack, việc xây dựng website thương mại điện tử trở nên dễ dàng và nhanh chóng hơn. Các lập trình viên chỉ cần nắm vững các công nghệ trong MERN Stack là có thể xây dựng một website thương mại điện tử hoàn chỉnh. Sau khi được học môn các công nghệ phần mềm mới, em đã được tìm hiểu về các công nghệ trên chính vì thế trong đồ án này em xin phép làm về đề tài “***Xây dựng website thương mại điện tử sử dụng công nghệ mern stack***”. Website bao gồm có user và admin. User hay người dùng có thể đăng ký, đăng nhập vào xem các sản phẩm và có thể mua sản phẩm, có thể thanh toán online hoặc không. Còn đối với admin sẽ quản lý toàn bộ hệ thống bao gồm người dùng, sản phẩm, các đơn đặt hàng.

* 1. **Mục đích, yêu cầu thực hiện:**

Đề tài “***Xây dựng website thương mại điện tử sử dụng công nghệ mern stack***” đặt ra các vấn đề trọng tâm về mục tiêu mà sinh viên nghiên cứu cần thực hiện được bao gồm:

Vấn đề đầu tiên được đưa ra về yếu tố lý thuyết, sinh viên thực hiện cần nghiên cứu chuyên môn, hiểu được tổng quan và khả năng ứng dụng những kiến thức đã học ở trường vào trong đề tài này. Song song với công nghệ được lựa chọn để sử dụng, sinh viên cần tìm hiểu thêm cách xây dựng giao diện đẹp, thân thiện với người dùng.

Vấn đề thứ hai là nâng cao khả năng làm việc độc lập, tìm hiểu và tìm hướng giải quyết về đề tài. Xây dựng thái tích cực trong việc học, làm đồ án.

* 1. **Đối tượng và phạm vi nghiên cứu:**
     1. **Đối tượng nghiên cứu:**

Xử dụng ReactJs và Redux để xây dựng giao diện và các logic về frontend và sử dụng NodeJs và Expressjs để giải quyết bài toán đề ra. Kết hợp với đó là các thư viện javascript hỗ trợ. Sử dụng cơ sở dữ liệu mongoDB.

* 1. **Kết quả mong đợi:**

Chủ đề “***Xây dựng website thương mại điện tử sử dụng công nghệ mern stack***” phục vụ các mục đích sau:

* Xây dựng website để khách hàng có thể vào đặt hàng và mua hàng, kèm theo đó là có thể thanh toán.
* Xây dựng website admin để quản lý các sản phẩm, đơn hàng, người dùng.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

1. **Reactjs:**
2. **Tổng quan về Reactjs:**

ReactJS là một thư viện JavaScript phía người dùng (frontend) phát triển bởi Facebook và được sử dụng rộng rãi để xây dựng giao diện người dùng động trên các ứng dụng web.

ReactJS là một thư viện JavaScript được sử dụng để xây dựng giao diện người dùng (UI) cho các ứng dụng web. Nó tập trung vào việc tạo ra các thành phần UI tái sử dụng và quản lý sự thay đổi của UI dựa trên dữ liệu.

1. **Các tính năng chính:**

Component-Based: ReactJS sử dụng kiến trúc dựa trên thành phần (component-based), cho phép bạn chia giao diện người dùng thành các thành phần độc lập, tái sử dụng và quản lý chúng một cách dễ dàng.

Virtual DOM: React sử dụng một Virtual DOM để cải thiện hiệu suất. Thay vì cập nhật trực tiếp DOM mỗi khi có thay đổi, React tạo ra một bản sao ảo của DOM và so sánh nó với DOM thực tế, sau đó chỉ cập nhật những phần cần thiết, giúp giảm tải cho trình duyệt.

One-Way Data Binding: React sử dụng luồng dữ liệu một chiều (one-way data binding), giúp quản lý dữ liệu dễ dàng và tránh tình trạng rơi vào vòng lặp vô tận (infinite loop).

JSX: JSX là một phần quan trọng của React, cho phép bạn viết mã HTML dưới dạng JavaScript, giúp dễ dàng tạo các thành phần UI.

Hỗ trợ cộng đồng lớn: React có một cộng đồng lớn và phong phú, điều này đồng nghĩa với việc có nhiều tài liệu, thư viện và công cụ hỗ trợ để xây dựng các ứng dụng React.

1. **Tại sao chúng ta sử dụng ReactJS:**

Hiệu suất tốt: Nhờ vào Virtual DOM và cách React quản lý các sự thay đổi trên giao diện người dùng, React có hiệu suất tốt và đáng tin cậy trong việc xây dựng các ứng dụng web động.

Tái sử dụng thành phần: React cho phép bạn tạo các thành phần UI tái sử dụng, giúp tiết kiệm thời gian và công sức trong phát triển ứng dụng.

Thư viện và cộng đồng lớn: Sự phát triển nhanh chóng và sự hỗ trợ từ cộng đồng đã làm cho React trở thành một trong những công cụ phát triển frontend phổ biến nhất, với nhiều tài liệu và công cụ bổ trợ.

JavaScript nổi bật: ReactJS sử dụng JavaScript, một ngôn ngữ phát triển phổ biến, giúp các nhà phát triển dễ dàng tiếp cận và làm việc với nó.

Hỗ trợ từ các công ty lớn: React được phát triển và duy trì bởi Facebook, và được sử dụng rộng rãi trong các ứng dụng của Facebook và Instagram, cho thấy tính đáng tin cậy và hỗ trợ dài hạn.

1. **Nodejs:**
2. **Tổng quan:**

Node.js là một môi trường chạy phía máy chủ mã nguồn mở được xây dựng trên công cụ JavaScript V8, đúng như công cụ V8 mà Google Chrome sử dụng. Nó cho phép các nhà phát triển chạy mã JavaScript trên máy chủ, giúp xây dựng các ứng dụng mạng có khả năng mở rộng và hiệu suất cao.

1. **Các đặc điểm chính:**

Kiến trúc Phi Chặn và Sự Kiện Định Hướng: Node.js sử dụng mô hình xử lý I/O phi chặn và định hướng sự kiện. Điều này có nghĩa rằng nó có thể xử lý nhiều kết nối đồng thời mà không chặn việc thực thi mã. Điều này làm cho nó rất hiệu quả cho việc xây dựng ứng dụng thời gian thực và xử lý một lượng lớn yêu cầu đồng thời.

JavaScript Ở Mọi Nơi: Node.js cho phép các nhà phát triển sử dụng JavaScript cả ở phía máy chủ và phía máy khách. Sự đồng nhất trong ngôn ngữ lập trình giúp đơn giản hóa việc phát triển, vì các nhà phát triển có thể sử dụng cùng một ngôn ngữ trong toàn bộ ngăn xếp ứng dụng.

Hệ Sinh Thái Phong Phú Của Gói (npm): Node.js có một hệ sinh thái phong phú các gói và mô-đun thông qua registry npm (Node Package Manager). Các nhà phát triển có thể dễ dàng tìm và sử dụng các thư viện để mở rộng chức năng của ứng dụng của họ, từ đó tăng tốc quá trình phát triển.

Khả Năng Tương Thích Đa Nền Tảng: Node.js là đa nền tảng và chạy trên nhiều hệ điều hành khác nhau, bao gồm Windows, macOS và Linux. Điều này làm cho nó trở thành một lựa chọn linh hoạt cho việc phát triển ứng dụng có thể triển khai trên nhiều nền tảng khác nhau.

Khả Năng Mở Rộng: Node.js rất thích hợp để xây dựng các ứng dụng có khả năng mở rộng nhờ vào kiến trúc phi chặn của nó. Các nhà phát triển có thể dễ dàng mở rộng ứng dụng bằng cách thêm các phiên bản Node.js khác để phân phối công việc.

1. **Tại sao chúng ta lại sử dụng nodejs:**

Xây Dựng Máy Chủ Web: Node.js thường được sử dụng để tạo máy chủ web. Các nhà phát triển có thể nhanh chóng xây dựng máy chủ web hiệu quả, nhẹ nhàng có khả năng xử lý nhiều kết nối đồng thời, làm cho nó lý tưởng cho các ứng dụng web thời gian thực.

Phát Triển API: Node.js thường được chọn để xây dựng các API (Giao diện Lập Trình Ứng dụng) do tốc độ và khả năng xử lý một lượng lớn yêu cầu API.

Ứng Dụng Thời Gian Thực: Nó rất thích hợp để xây dựng các ứng dụng thời gian thực như ứng dụng trò chuyện, máy chủ trò chơi trực tuyến và các nền tảng truyền trực tiếp, nơi độ trễ thấp là quan trọng.

Kiến Trúc Microservices: Node.js là lựa chọn phổ biến cho việc phát triển kiến trúc microservices, nhờ tính nhẹ nhàng và khả năng mở rộng của nó.

IoT (Internet of Things): Node.js có thể được sử dụng trong các ứng dụng IoT để xử lý giao tiếp thiết bị và xử lý dữ liệu, nhờ vào kiến trúc sự kiện của nó.

# **CHƯƠNG 3: CƠ SỞ DỮ LIỆU VÀ PHÂN TÍCH CÁC CHỨC NĂNG CHÍNH**

1. **Cơ sở dữ liệu:**

**Bảng 1.1. Bảng user**

|  |  |  |  |
| --- | --- | --- | --- |
| **SST** | **Field** | **Type** | **Mô tả** |
| 1 | \_id | ObjectId | Id user |
| 2 | email | string | gmail |
| 3 | name | string | Tên người dùng |
| 4 | password | string | Mật khẩu |
| 5 | isAdmin | Boolean | Có phải là admin |
| 6 | createdAt | Date | Thời gian tạo |
| 7 | updatedAt | Date | Thời gian update |

**Bảng 1.2. Bảng products**

|  |  |  |  |
| --- | --- | --- | --- |
| **SST** | **Field** | **Type** | **Mô tả** |
| 1 | \_id | ObjectId | Id user |
| 2 | user | ObjectId | Id người dùng |
| 3 | name | string | Tên sản phẩm |
| 4 | image | string | Đường dẫn đến hình ảnh |
| 5 | brand | string | Thương hiệu |
| 6 | category | string | Doanh mục |
| 7 | description | string | Mô tả sản phẩm |
| 8 | rating | Int32 | Số sao |
| 9 | numReviews | Int32 | Số lượng review |
| 10 | price | double | Giá |
| 11 | countInStock | Int32 | Số lượng sản phẩm |
| 12 | reviews | Array | Danh sách review |
| 13 | createdAt | Date | Thời gian tạo |
| 14 | updatedAt | Date | Thời gian update |

**Bảng 1.3. Bảng review trong product**

|  |  |  |  |
| --- | --- | --- | --- |
| **SST** | **Field** | **Type** | **Mô tả** |
| 1 | name | string | Tên người comment |
| 2 | rating | string | Số sao |
| 3 | comment | string | Nội dung |
| 4 | User\_id | ObjectId | Id người comment |
| 5 | \_id | ObjectId | Id comment |
| 6 | createdAt | Date | Thời gian tạo |
| 7 | updatedAt | Date | Thời gian update |

**Bảng 1.4. Bảng orders**

|  |  |  |  |
| --- | --- | --- | --- |
| **SST** | **Field** | **Type** | **Mô tả** |
| 1 | \_id | ObjectId | Tên người comment |
| 2 | user | string | Số sao |
| 3 | OrderItems | string | Nội dung |
| 4 | shippingAddress | ObjectId | Id người comment |
| 5 | paymentMethod | ObjectId | Id comment |
| 6 | itemsPrice |  |  |
| 7 | taxPrice |  |  |
| 8 | shippingPrice |  |  |
| 9 | totalPrice |  |  |
| 10 | isPaid |  |  |
| 11 | isDelivered |  |  |
| 12 | paidAt |  |  |
| 13 | paymentResult |  |  |
| 14 | deliveredAt |  |  |
| 15 | createdAt | Date | Thời gian tạo |
| 16 | updatedAt | Date | Thời gian update |

1. **Chức năng chính của phía client**

*a) Xem danh sách sản phẩm trang home:*

* Sử dụng component ProductList để hiển thị danh sách các sản phẩm có sẵn trong cơ sở dữ liệu.
* Giao diện hiển thị tên, hình ảnh, mô tả, số lượng đánh giá sao và giá cả của mỗi sản phẩm.
* Người dùng có thể nhấp vào sản phẩm để xem chi tiết..

*b) Chi tiết sản phẩm:*

* Sử dụng component ProductScreen để hiển thị thông tin chi tiết về một sản phẩm cụ thể.
* Giao diện hiển thị hình ảnh, tên, giá cả và mô tả sản phẩm.
* Người dùng có thể điều chỉnh số lượng sản phẩm và thêm vào giỏ hàng.

*c) Quản lý giỏ hàng:*

* Sử dụng component CartScreen để hiển thị giỏ hàng và các mục đã được thêm vào giỏ hàng.
* Giao diện hiển thị thông tin về sản phẩm, số lượng, giá tiền và tổng giá trị của giỏ hàng.
* Người dùng có thể điều chỉnh số lượng sản phẩm, xóa sản phẩm khỏi giỏ hàng và tiến hành thanh toán.

*d) Xác nhận đặt hàng:*

* Sử dụng component PlaceOrderScreen để hiển thị thông tin xác nhận đặt hàng.
* Giao diện hiển thị thông tin về địa chỉ giao hàng, phương thức thanh toán và các sản phẩm trong giỏ hàng.
* Người dùng có thể xác nhận đặt hàng và gửi yêu cầu đặt hàng lên server.

*e) Đăng ký và đăng nhập:*

* Sử dụng component RegisterScreen và LoginScreen để cho phép người dùng đăng ký và đăng nhập vào ứng dụng.
* Giao diện hiển thị các trường nhập liệu như tên, email, mật khẩu để người dùng đăng ký tài khoản.
* Người dùng có thể đăng nhập bằng email và mật khẩu đã đăng ký để truy cập vào các chức năng của ứng dụng.

*f) Xem và chỉnh sửa thông tin cá nhân:*

* Sử dụng component ProfileScreen để hiển thị thông tin cá nhân của người dùng.
* Giao diện hiển thị thông tin như tên, email, và các thông tin cá nhân khác của người dùng.
* Người dùng có thể chỉnh sửa thông tin cá nhân và cập nhật lên server.

*g) Xem và quản lý đơn hàng:*

* Sử dụng component OrderScreen để hiển thị danh sách các đơn hàng của người dùng.
* Giao diện hiển thị thông tin về đơn hàng bao gồm ngày đặt hàng, tổng giá trị, và trạng thái đơn hàng.
* Người dùng có thể xem chi tiết từng đơn hàng và cập nhật trạng thái đơn hàng.

## **Tương tác với backend**

* Phía client giao tiếp với backend thông qua việc gọi các API từ phía server bằng cách sử dụng các thư viện như axios hoặc sử dụng các hooks được cung cấp bởi Redux Toolkit hoặc RTK Query.
* Các API từ phía server cung cấp các endpoint để lấy dữ liệu, thực hiện các yêu cầu CRUD và xử lý các tác vụ logic phía server.
* Phía client sử dụng các hooks như useGetUsersQuery, useGetOrdersQuery, useDeleteUserMutation, useUpdateProductMutation, useUploadProductImageMutation, useGetProductsQuery, useGetUserDetailsQuery, useUpdateUserMutation từ các slices hoặc APIs được định nghĩa để gọi các API tương ứng và thực hiện các tác vụ liên quan đến dữ liệu.
* Khi phía client gọi các API, nó sẽ gửi yêu cầu đến phía server và nhận lại các phản hồi (response) từ server để hiển thị dữ liệu hoặc xử lý các tác vụ logic khác.

# **CHƯƠNG 4: PHÂN TÍCH GIAO DIỆN**

1. **CartScreen**

* Chức năng: Hiển thị giao diện giỏ hàng của ứng dụng. Dựa vào dữ liệu giỏ hàng từ Redux store, màn hình này hiển thị các phần tử JSX tương ứng.
* Trong phần khai báo:
* Sử dụng hook useNavigate để lấy đối tượng navigate dùng để chuyển hướng giữa các trang trong ứng dụng.
* Sử dụng hook useSelector để truy cập state trong Redux store và lấy giá trị của cart và cartItems.
* Hàm addToCartHandler được sử dụng để xử lý thêm sản phẩm vào giỏ hàng khi nhấn nút "Add to Cart". Nó gửi một action addToCart đến store với thông tin sản phẩm và số lượng.
* Hàm removeFromCartHandler được sử dụng để xử lý xóa sản phẩm khỏi giỏ hàng khi nhấn nút "Remove from Cart". Nó gửi một action removeFromCart đến store với id của sản phẩm.
* Hàm checkoutHandler được sử dụng để xử lý chuyển hướng đến trang đăng nhập khi nhấn nút "Proceed To Checkout".
* Trong phần render:
* Sử dụng các component của react-bootstrap và component tùy chỉnh như Message để hiển thị giao diện người dùng.
* Nếu giỏ hàng trống, hiển thị một thông báo và một liên kết để quay lại trang chủ.
* Nếu giỏ hàng không trống, hiển thị danh sách các sản phẩm trong giỏ hàng mỗi sản phẩm có hình ảnh, tên, giá, số lượng, và nút xóa.
* Tính toán tổng số lượng sản phẩm và tổng giá trị giỏ hàng và hiển thị chúng trong một component Cart.
* Sử dụng nút "Proceed To Checkout" để chuyển hướng đến trang thanh toán khi nhấn nút đó.
* Cuối cùng, export CartScreen như một default export để có thể sử dụng nó trong các file khác.

1. **HomeScreen**

* Chức năng: Hiển thị giao diện trang chủ, tải và hiển thị danh sách sản phẩm từ API, và xử lý các trạng thái tải và lỗi tương ứng.
* Trong phần khai báo:
* Sử dụng useParams để lấy giá trị của pageNumber và keyword từ định tuyến URL hiện tại.
* Sử dụng useGetProductsQuery để gửi yêu cầu API để lấy danh sách sản phẩm. Biến data chứa dữ liệu trả về, isLoading là trạng thái xác định xem dữ liệu đang được tải hay không và error chứa bất kỳ lỗi nào trong quá trình tải dữ liệu.
* Trong phần render:
* Nếu keyword không tồn tại, hiển thị ProductCarousel, ngược lại hiển thị một liên kết trở lại trang chủ.
* Nếu đang tải dữ liệu, hiển thị Loader, nếu có lỗi hiển thị Message và thông báo lỗi, nếu không có lỗi hiển thị danh sách sản phẩm và cuối cùng hiển thị component Paginate để phân trang dữ liệu.
* Các component Meta, Product, Loader, Message, Paginate, ProductCarousel được sử dụng trong phần trả về đều là component tùy chỉnh được import từ các file tương ứng.
* Cuối cùng, export HomeScreen như một default export để có thể sử dụng nó trong các file khác.

1. **PaymentScreen**

* Chức năng: Hiển thị giao diện cho phần chọn phương thức thanh toán trong quá trình thanh toán.
* Import các thành phần và thư viện:
* useState, useEffect từ react để quản lý trạng thái và hiệu ứng.
* Form, Button, Col từ react-bootstrap để tạo các thành phần form và nút.
* useNavigate từ react-router-dom để điều hướng đến các đường dẫn khác.
* useDispatch, useSelector từ react-redux để truy cập và thay đổi trạng thái Redux.
* FormContainer từ '../components/FormContainer' để tạo khung chứa cho form.
* CheckoutSteps từ '../components/CheckoutSteps' để hiển thị bước tiến trình thanh toán.
* Trong phần khai báo:
* Sử dụng useNavigate để có thể chuyển hướng đến trang giao hàng nếu địa chỉ giao hàng chưa được cung cấp.
* Sử dụng useSelector để lấy thông tin về giỏ hàng từ state.cart.
* Sử dụng useEffect để kiểm tra xem địa chỉ giao hàng đã được cung cấp hay chưa. Nếu không có địa chỉ giao hàng (!shippingAddress.address), sử dụng navigate để chuyển hướng đến trang giao hàng ('/shipping').
* Sử dụng useState để quản lý trạng thái phương thức thanh toán:
* Khởi tạo trạng thái mặc định cho phương thức thanh toán là 'PayPal'.
* Sử dụng hàm setPaymentMethod để cập nhật trạng thái phương thức thanh toán khi người dùng chọn.
* Sử dụng useDispatch để gửi hành động lưu phương thức thanh toán:
* Khai báo biến dispatch để gửi hành động Redux.
* Trong hàm submitHandler, gửi hành động savePaymentMethod với trạng thái paymentMethod hiện tại.
* Phần render:
* Sử dụng FormContainer để tạo khung chứa cho form.
* Sử dụng CheckoutSteps để hiển thị tiến trình thanh toán.
* Hiển thị tiêu đề "Payment Method".
* Sử dụng Form để tạo form cho việc chọn phương thức thanh toán.
* Sử dụng Form.Check để tạo radio button cho phương thức thanh toán PayPal.
* Khi người dùng chọn phương thức thanh toán, cập nhật trạng thái paymentMethod thông qua hàm setPaymentMethod.
* Sử dụng Button để hiển thị nút "Continue" để tiếp tục quá trình thanh toán.
* Cuối cùng, export PaymentScreen như một default export để có thể sử dụng nó trong các file khác.

1. **LoginScreen**

* Chức năng: Hiển thị giao diện đăng nhập trong ứng dụng web. Người dùng có thể nhập email và mật khẩu của họ để đăng nhập.
* Import các thành phần và thư viện: useState, useEffect từ react, Link, useLocation, useNavigate từ react-router-dom, Form, Button, Row, Col từ react-bootstrap, useDispatch, useSelector từ react-redux, Loader từ ../components/Loader, FormContainer từ ../components/FormContainer.
* Trong phần khai báo:
* Sử dụng hook useState để tạo hai biến trạng thái email và password.
* Các biến dispatch và navigate được khởi tạo bằng cách sử dụng hook useDispatch và useNavigate.
* useLoginMutation là một hook tùy chỉnh được import và gán giá trị cho biến login và isLoading.
* useSelector được sử dụng để lấy thông tin từ trạng thái auth trong Redux store và gán giá trị cho biến userInfo.
* useLocation được sử dụng để lấy thông tin vị trí hiện tại, URLSearchParams được sử dụng để trích xuất thông tin từ tham số truy vấn URL.
* Trong useEffect, nếu userInfo có giá trị, mã sẽ sử dụng navigate để chuyển hướng đến trang được chỉ định.
* submitHandler là một hàm xử lý sự kiện cho việc đăng nhập. Nó gọi login để gửi yêu cầu đăng nhập đến máy chủ và xử lý kết quả trả về. Nếu thành công, thông tin đăng nhập được cập nhật trong Redux store và người dùng được chuyển hướng đến trang được chỉ định. Nếu không thành công, thông báo lỗi sẽ được hiển thị.
* Trong phần giao diện, FormContainer được sử dụng để tạo khung giao diện cho form đăng nhập. Form là một thành phần giao diện để tạo form và các trường nhập liệu. Button là một nút nhấn. Loader sẽ được hiển thị khi isLoading có giá trị true. Link là một thành phần để tạo liên kết đến trang khác.

1. **OrderScreen**

* Chức năng: Hiển thị chi tiết về một đơn hàng.
* Trong phần render của thành phần OrderScreen, nếu isLoading là true, sẽ được hiển thị thành phần Loader. Nếu có error (có lỗi), sẽ hiển thị một thông báo lỗi (Message) với thuộc tính variant được đặt là 'danger' và nội dung lỗi được truyền vào.
* Nếu không có lỗi:
* Tiêu đề đơn hàng (<h1>Order {order.\_id}</h1>).
* Một hàng ngang (<Row>) chứa hai cột (<Col>).
* Cột trái (<Col md={8}>) chứa danh sách (<ListGroup>) các mục sau:
* Thông tin vận chuyển (<ListGroup.Item>): Hiển thị tên, email và địa chỉ giao hàng từ order.user và hiển thị trạng thái giao hàng.
* Phương thức thanh toán (<ListGroup.Item>): Hiển thị phương thức thanh toán từ order.paymentMethod và hiển thị trạng thái thanh toán.
* Các mặt hàng trong đơn hàng (<ListGroup.Item>): Hiển thị danh sách các mặt hàng từ order.orderItems, bao gồm hình ảnh, tên mặt hàng và giá cả.
* Cột phải (<Col md={4}>) chứa một thẻ Card và danh sách (<ListGroup>) các mục:
* Tổng kết đơn hàng (<ListGroup.Item>): Hiển thị tổng số tiền từ order.
* Nếu đơn hàng chưa được thanh toán, hiển thị:
* Nếu đang tải, hiển thị Loader.
* Nếu đang chờ, hiển thị Loader.
* Nếu không tải, hiển thị nút thanh toán PayPal.
* Nếu đang tải giao hàng (loadingDeliver), hiển thị thành phần Loader.

1. **PlaceOrderScreen**

* Chức năng: Hiển thị trang xác nhận đặt hàng, hiển thị thông tin về địa chỉ giao hàng, phương thức thanh toán và các sản phẩm trong giỏ hàng.
* Import các module và component cần thiết từ các thư viện React và thư mục components.
* Sử dụng hooks như useNavigate, useSelector, useDispatch để quản lý điều hướng, lấy thông tin từ Redux và gửi các action.
* Sử dụng useCreateOrderMutation hook để gửi yêu cầu đặt hàng lên server.
* Sử dụng useEffect hook để kiểm tra xem người dùng đã cung cấp đủ thông tin địa chỉ giao hàng và phương thức thanh toán hay chưa, nếu chưa thì điều hướng đến các trang tương ứng.
* Xử lý sự kiện placeOrderHandler để gửi yêu cầu đặt hàng lên server, xóa các mục trong giỏ hàng sau khi đặt hàng thành công và điều hướng đến trang xác nhận đặt hàng.
* Render giao diện trang xác nhận đặt hàng với các phần tử giao diện như CheckoutSteps, danh sách thông tin địa chỉ giao hàng, phương thức thanh toán và các sản phẩm trong giỏ hàng.
* Hiển thị các thông báo lỗi, hiển thị thông báo loading trong quá trình đặt hàng.

1. **ProductScreen**

* Chức năng: Hiển thị trang chi tiết sản phẩm, bao gồm hình ảnh, thông tin, giá cả và đánh giá sản phẩm. Cho phép người dùng thêm sản phẩm vào giỏ hàng và viết đánh giá cho sản phẩm.
* Import các module và component cần thiết từ các thư viện React và thư mục components.
* Sử dụng hooks như useParams, useDispatch, useNavigate, useSelector để lấy thông tin từ URL, gửi các action và quản lý trạng thái.
* Sử dụng useState hook để lưu trữ các giá trị qty, rating, comment.
* Xử lý sự kiện addToCartHandler để thêm sản phẩm vào giỏ hàng bằng cách gửi action addToCart và điều hướng đến trang giỏ hàng.
* Sử dụng useGetProductDetailsQuery hook từ productsApiSlice để lấy thông tin chi tiết sản phẩm từ server dựa trên id.
* Sử dụng Redux useSelector để lấy thông tin người dùng đã đăng nhập.
* Sử dụng useCreateReviewMutation hook để gửi yêu cầu tạo đánh giá sản phẩm lên server.
* Xử lý sự kiện submitHandler để gửi yêu cầu tạo đánh giá và cập nhật lại thông tin sản phẩm sau khi tạo đánh giá thành công.
* Render giao diện trang chi tiết sản phẩm với các phần tử giao diện như hình ảnh, thông tin, giá cả, số lượng, đánh giá sản phẩm và form viết đánh giá.
* Hiển thị thông báo lỗi, hiển thị thông báo loading trong quá trình tải dữ liệu hoặc tạo đánh giá.

1. **RegisterScreen**

* Chức năng: Hiển thị trang đăng ký người dùng.
* Import các module và component cần thiết từ các thư viện React và thư mục components.
* Sử dụng hooks như useState, useEffect, Link, useLocation, useNavigate, Form, Button, Row, Col, useDispatch, useSelector để quản lý state, xử lý định tuyến, giao diện và tương tác với Redux store.
* Định nghĩa các biến state như name, email, password, confirmPassword để lưu trữ giá trị nhập liệu từ người dùng.
* Sử dụng hook useEffect để kiểm tra khi userInfo thay đổi trong Redux store và thực hiện chuyển hướng đến đường dẫn redirect.
* Xử lý sự kiện submitHandler để gửi yêu cầu đăng ký người dùng đến API và cập nhật thông tin người dùng trong Redux store sau khi đăng ký thành công.
* Render giao diện trang đăng ký với các thành phần như Form, Button, Row, Col, Link để tạo biểu mẫu đăng ký và liên kết đến màn hình đăng nhập nếu người dùng đã có tài khoản.
* Hiển thị loader trong quá trình tải dữ liệu.

1. **ShippingScreen**

* Chức năng: Hiển thị trang chọn địa chỉ giao hàng.
* Định nghĩa cấu trúc giao diện bằng JSX với các thành phần như FormContainer, CheckoutSteps, Form, Form.Group, Form.Label, và Form.Control để tạo giao diện người dùng.
* Truyền và sử dụng dữ liệu động từ state và props để điền thông tin mặc định và cập nhật giá trị khi người dùng thay đổi thông qua hàm onChange.
* Xử lý sự kiện submitHandler để lưu địa chỉ giao hàng và chuyển hướng người dùng đến trang thanh toán.
* Sử dụng các thư viện và hook như useSelector, useDispatch, useNavigate để truy cập và tương tác với Redux store và điều hướng người dùng.

1. **ProfileScreen**

* Chức năng: Hiển thị trang thông tin người dùng và đơn hàng của người dùng.
* Hiển thị thông tin người dùng từ userInfo trong Redux store và cho phép người dùng cập nhật thông tin.
* Hiển thị danh sách đơn hàng của người dùng và hiển thị loader trong quá trình tải dữ liệu.
* Sử dụng các thư viện và hook như useSelector, useDispatch, useEffect để truy cập và tương tác với Redux store và gọi API.
* Sử dụng các thành phần JSX như Table, Button, Loader, Message để tạo giao diện người dùng và hiển thị thông báo.

1. **UseListScreen**

* Chức năng: Hiển thị giao diện danh sách người dùng trong ứng dụng.
* Import các module và component cần thiết từ các thư viện React và thư mục components.
* Sử dụng hook useGetUsersQuery để lấy danh sách người dùng và các trạng thái liên quan như isLoading, error, và data.
* Sử dụng hook useDeleteUserMutation để lấy hàm deleteUser để xóa người dùng.
* Xử lý sự kiện deleteHandler để xóa người dùng khi nhấn nút "Delete". Nếu người dùng xác nhận xóa, gọi API để xóa người dùng bằng cách gọi hàm deleteUser, sau đó gọi hàm refetch để tải lại danh sách người dùng. Nếu có lỗi, hiển thị thông báo lỗi sử dụng toast.error.
* Render giao diện danh sách người dùng. Nếu đang tải dữ liệu (isLoading), hiển thị Loader. Nếu có lỗi (error), hiển thị Message với thông báo lỗi. Nếu không có lỗi và dữ liệu đã tải xong, hiển thị một bảng (Table) để hiển thị danh sách người dùng. Mỗi người dùng được hiển thị trong một hàng (<tr>) và thông tin của người dùng được hiển thị trong các ô (<td>). Các biểu tượng FaCheck và FaTimes được hiển thị tùy thuộc vào trạng thái quản trị viên (user.isAdmin). Các nút "Edit" và "Delete" chỉ hiển thị khi người dùng không phải là quản trị viên (!user.isAdmin). Chúng được xử lý bằng cách sử dụng LinkContainer để tạo liên kết đến trang chỉnh sửa người dùng và sử dụng Button để thực hiện hành động xóa người dùng.
* Cuối cùng, export UserListScreen như một default export để có thể sử dụng nó trong các file khác.

1. **OrderListScreen**

* Chức năng: Hiển thị danh sách các đơn hàng trên màn hình.
* Import các module và component cần thiết từ các thư viện React và thư mục components.
* Sử dụng hook useGetOrdersQuery để gọi API lấy danh sách các đơn hàng và quản lý trạng thái liên quan.
* Render giao diện danh sách đơn hàng. Nếu đang tải dữ liệu (isLoading), hiển thị Loader. Nếu có lỗi (error), hiển thị Message với thông báo lỗi. Nếu không có lỗi và dữ liệu đã được tải (!isLoading && !error), hiển thị một bảng (Table) chứa danh sách các đơn hàng. Mỗi đơn hàng được hiển thị trong một hàng (<tr>) và thông tin về mỗi đơn hàng được hiển thị trong các ô (<td>). Nếu đơn hàng đã được thanh toán (order.isPaid), hiển thị ngày thanh toán (order.paidAt.substring(0, 10)). Nếu đơn hàng đã được giao hàng (order.isDelivered), hiển thị ngày giao hàng (order.deliveredAt.substring(0, 10)). Mỗi hàng của bảng cũng bao gồm một nút "Details" (<Button variant='light' className='btn-sm'>Details</Button>) dùng để xem chi tiết đơn hàng. Sử dụng LinkContainer để tạo liên kết đến trang chi tiết đơn hàng dựa trên ID của đơn hàng (<LinkContainer to={'/order/${order.\_id}'}>).
* Cuối cùng, export OrderListScreen như một default export để có thể sử dụng nó trong các file khác.

1. **ProductEditScreen**

* Chức năng: Hiển thị giao diện chỉnh sửa sản phẩm trong ứng dụng.
* Import các module và component cần thiết từ các thư viện React và thư mục components.
* Sử dụng hook useState và useEffect để quản lý trạng thái và xử lý hiệu ứng phụ.
* Sử dụng các hooks custom useParams, useNavigate, useGetProductDetailsQuery, useUpdateProductMutation, useUploadProductImageMutation từ các slice và hook tương ứng để lấy thông tin chi tiết sản phẩm, cập nhật sản phẩm và tải lên hình ảnh sản phẩm.
* Xử lý sự kiện submitHandler để gửi yêu cầu cập nhật thông tin sản phẩm và xử lý kết quả trả về. Nếu thành công, hiển thị thông báo thành công và chuyển hướng đến danh sách sản phẩm. Nếu không thành công, hiển thị thông báo lỗi.
* Render giao diện chỉnh sửa sản phẩm. Sử dụng các thành phần JSX như Link, Form, Button, Message, Loader, FormContainer để tạo giao diện người dùng và hiển thị thông báo.
* Cuối cùng, export ProductEditScreen như một default export để có thể sử dụng nó trong các file khác.

1. **ProductListScreen**

* Chức năng: Hiển thị danh sách sản phẩm dưới dạng bảng và các nút chức năng tương ứng.
* Import các module và component cần thiết từ các thư viện React và thư mục components.
* Sử dụng hook useParams để lấy tham số pageNumber từ URL.
* Sử dụng hook useGetProductsQuery để lấy danh sách sản phẩm và các trạng thái liên quan như data, isLoading, error, refetch.
* Sử dụng hook useDeleteProductMutation để lấy hàm deleteProduct để xóa sản phẩm.
* Xử lý sự kiện createProductHandler để điều hướng đến trang tạo sản phẩm khi nhấp vào nút "Create Product".
* Render giao diện danh sách sản phẩm. Hàng <Row> chứa tiêu đề "Products" và nút "Create Product". Khi nhấp vào nút "Create Product", gọi hàm createProductHandler. Nếu loadingCreate có giá trị true, hiển thị Loader để chỉ định rằng yêu cầu tạo sản phẩm đang được xử lý. Nếu loadingDelete có giá trị true, hiển thị Loader tương tự. Dùng câu điều kiện để kiểm tra trạng thái dữ liệu và hiển thị giao diện tương ứng. Nếu isLoading có giá trị true, hiển thị Loader để chỉ định rằng dữ liệu đang được tải. Nếu có lỗi (error), hiển thị Message với thông báo lỗi màu đỏ. Nếu không có lỗi và dữ liệu đã được tải, hiển thị bảng chứa thông tin sản phẩm. Các cột bao gồm ID, NAME, PRICE, CATEGORY, BRAND và một cột trống. Dữ liệu sản phẩm được hiển thị trong tbody với các dữ liệu tương ứng với từng cột. Trong cột trống, mỗi hàng dữ liệu hợp lệ sẽ hiển thị một LinkContainer để điều hướng đến trang chỉnh sửa sản phẩm và một nút "Delete" để gọi hàm deleteHandler với tham số là id của sản phẩm đó. Cuối cùng, hiển thị phân trang (Paginate) cho danh sách sản phẩm.
* Cuối cùng, export ProductListScreen như một default export để có thể sử dụng nó trong các file khác.

1. **UserEditScreen**

* Chức năng: Hiển thị một form để chỉnh sửa thông tin của người dùng.
* Import các module và component cần thiết từ các thư viện React và thư mục components.
* Sử dụng hook useParams để lấy thông tin id của người dùng từ URL.
* Sử dụng hook useState để lưu trữ các giá trị name, email và isAdmin của người dùng.
* Sử dụng hook useGetUserDetailsQuery để lấy chi tiết người dùng từ server dựa trên id.
* Sử dụng hook useUpdateUserMutation để gửi yêu cầu cập nhật thông tin người dùng lên server.
* Sử dụng hook useNavigate từ react-router-dom để thực hiện điều hướng sau khi cập nhật thành công.
* Sử dụng hook useEffect để cập nhật giá trị name, email và isAdmin khi người dùng thay đổi.
* Render một form sử dụng react-bootstrap và hiển thị các trường để chỉnh sửa thông tin người dùng như name, email và isAdmin.
* Xử lý sự kiện submit của form để gọi hàm updateUser, hiển thị thông báo thành công và thực hiện điều hướng trang sau khi cập nhật.
* Cuối cùng, export UserEditScreen như một default export để có thể sử dụng nó trong các file khác.

# **CHƯƠNG III: COMPONENT**

1. **Component AdminRoute**

* Component AdminRoute được sử dụng để xác định một tuyến đường bảo mật chỉ dành riêng cho người dùng quản trị viên (admin).
* Sử dụng hook useSelector từ thư viện react-redux để lấy thông tin người dùng từ state Redux. Điều này cho phép component re-render khi thông tin người dùng thay đổi.
* Kiểm tra nếu userInfo tồn tại và userInfo.isAdmin là true, tức là người dùng đã đăng nhập và là admin, thì component AdminRoute sẽ render ra các thành phần con bên trong (Outlet).
* Nếu không thỏa mãn điều kiện trên, component AdminRoute sẽ thực hiện điều hướng đến trang "/login" bằng cách sử dụng component Navigate từ thư viện react-router-dom, và thêm thuộc tính replace để thay thế địa chỉ hiện tại trong lịch sử điều hướng.
* Component AdminRoute được sử dụng trong cấu trúc định tuyến (routing) của ứng dụng để bảo vệ các tuyến đường chỉ dành riêng cho người dùng quản trị viên.

1. **Component CheckoutSteps**

* Component CheckoutSteps được sử dụng để hiển thị thanh điều hướng trong quy trình thanh toán.
* JSX được sử dụng để tạo các thành phần UI như <Nav>, <Nav.Item>, và <Nav.Link> từ thư viện 'react-bootstrap'.
* Để ràng buộc dữ liệu động, JSX sử dụng các biểu thức JavaScript như {step1 ? (...) : (...)} để hiển thị các thành phần tương ứng dựa trên giá trị của step1.
* JSX cũng sử dụng các hàm xử lý sự kiện như to='/login' trong <LinkContainer> và onSubmit={submitHandler} trong <Form> để định nghĩa các hành động xử lý khi người dùng tương tác với thành phần.
* Đồng thời, JSX tích hợp với các thư viện và module như 'react-router-bootstrap' để tạo liên kết dẫn đến các trang khác trong ứng dụng React.

1. **Component Footer**

* Component Footer được sử dụng để hiển thị phần chân trang cho trang web.
* Import các component cần thiết từ thư viện React Bootstrap như Container, Row, và Col.
* Trong phần giao diện, component sử dụng thẻ <footer> để định dạng phần chân trang và tạo các thành phần Container, Row, và Col để tạo bố cục chân trang.
* Component Footer trả về giao diện chân trang với nội dung là văn bản "ProShop ©" và năm hiện tại (currentYear) được lấy từ biến currentYear.

1. **Component FormContainer**

* Component FormContainer được sử dụng để hiển thị khung chứa trung tâm cho form hoặc các thành phần con khác.
* Import các thành phần và thư viện như 'react-bootstrap' để tạo cấu trúc của form.
* Sử dụng các thành phần Container, Row, và Col để tạo cấu trúc giao diện của khung chứa form.
* Trong cột của FormContainer, hiển thị nội dung con được truyền vào thông qua thuộc tính 'children'. Điều này cho phép nhúng bất kỳ thành phần con nào vào trong khung chứa form.

1. **Component Header**

* Component Header được sử dụng để hiển thị phần đầu trang (header) của trang web.
* Import các thành phần và thư viện từ React Bootstrap, react-icons/fa, react-router-bootstrap, react-redux, và react-router-dom.
* Sử dụng useSelector để lấy trạng thái state.cart và state.auth từ Redux Store và gán giá trị cho các biến cartItems và userInfo.
* Sử dụng useDispatch để gửi các hành động Redux thông qua đối tượng dispatch.
* Sử dụng useNavigate để điều hướng đến các đường dẫn khác trong ứng dụng.
* Định nghĩa hàm bất đồng bộ logoutHandler để xử lý đăng xuất người dùng.
* Trong phần render, sử dụng các thành phần của React Bootstrap để tạo thanh điều hướng.
* Sử dụng LinkContainer để tạo liên kết đến trang chủ và các trang khác.
* Hiển thị logo của trang web bên trong Navbar.Brand.
* Hiển thị nút tìm kiếm (SearchBox).
* Hiển thị biểu tượng giỏ hàng (FaShoppingCart) và số lượng hàng trong giỏ hàng (nếu có) bên trong Nav.Link.
* Hiển thị tên người dùng và dropdown menu chứa các liên kết (Profile và Logout) khi người dùng đã đăng nhập.
* Hiển thị nút đăng nhập (FaUser) khi người dùng chưa đăng nhập.
* Hiển thị menu quản trị khi người dùng đã đăng nhập và là quản trị viên.
* Cuối cùng, component Header được xuất khẩu bằng default export để có thể sử dụng trong các file khác.

1. **Component Loader**

* Component Loader trả về một Spinner từ thư viện React Bootstrap.
* Spinner này được sử dụng để hiển thị một biểu tượng quay tròn để chỉ thị việc tải dữ liệu hoặc xử lý.
* Spinner được cấu hình với thuộc tính animation là 'border' và role là 'status'.

1. **Component Message**

* Component Message nhận hai props là "variant" và "children".
* Nó trả về một Alert từ thư viện React Bootstrap để hiển thị thông báo với kiểu dáng tương ứng.
* Giá trị của prop "variant" xác định kiểu dáng của thông báo (ví dụ: success, danger, warning, info).
* Nội dung của thông báo được chứa trong prop "children".

1. **Component Meta**

* Component Meta được sử dụng để đặt các thông tin về tiêu đề, mô tả và từ khóa cho trang web.
* Nó nhận các props là title, description và keywords.
* Component Meta sử dụng các phần tử <title> và <meta> để đặt các thông tin trên trang web.
* Phần tử <title> đại diện cho tiêu đề trang web.
* Phần tử <meta name='description'> đại diện cho mô tả trang web.
* Phần tử <meta name='keywords'> đại diện cho từ khóa trang web.
* Component Meta cũng cung cấp giá trị mặc định cho các props nếu không có giá trị được truyền vào.

1. **Component Paginate**

* Component Paginate được sử dụng để hiển thị các liên kết phân trang.
* Component nhận các props là pages (tổng số trang), page (trang hiện tại), isAdmin (giá trị mặc định là false) và keyword.
* Sử dụng câu điều kiện để kiểm tra xem có cần hiển thị phân trang hay không (nếu pages > 1).
* Sử dụng thành phần Pagination của React Bootstrap để tạo danh sách các liên kết phân trang.
* Sử dụng map để tạo các liên kết phân trang từ 1 đến số trang (pages) bằng cách sử dụng phương thức keys() và Array(pages).
* Nếu không phải là admin, các liên kết sẽ chuyển hướng đến đường dẫn /search/${keyword}/page/${x + 1} nếu keyword tồn tại, ngược lại sẽ chuyển hướng đến đường dẫn /page/${x + 1}.
* Nếu là admin, các liên kết sẽ chuyển hướng đến đường dẫn /admin/productlist/${x + 1}.
* Cuối cùng, component Paginate được xuất khẩu bằng default export để có thể sử dụng trong các file khác.

1. **Component PrivateRoute**

* Component PrivateRoute được sử dụng để bảo vệ các tuyến đường riêng tư.
* Import các thành phần Navigate và Outlet từ thư viện react-router-dom.
* Sử dụng hook useSelector để lấy thông tin người dùng (userInfo) từ Redux store.
* Kiểm tra nếu userInfo không phải là null hoặc undefined (người dùng đã đăng nhập), component sẽ hiển thị nội dung của Outlet (các thành phần con của PrivateRoute).
* Nếu người dùng chưa đăng nhập, component sẽ sử dụng thành phần Navigate để chuyển hướng đến trang đăng nhập với đường dẫn "/login".
* Thuộc tính replace được sử dụng để thay thế lịch sử điều hướng, để người dùng không thể quay lại trang trước đó sau khi đăng nhập thành công.
* Component PrivateRoute được xuất khẩu để có thể sử dụng trong cây định tuyến của ứng dụng.

1. **Component Product**

* Component Product được sử dụng để hiển thị thông tin về một sản phẩm.
* Import các thành phần Card, Link và Rating từ thư viện react-bootstrap và react-router-dom.
* Hàm thành phần React Product nhận một đối số product, chứa thông tin về sản phẩm.
* Sử dụng thành phần Card để tạo một thẻ Card hiển thị thông tin sản phẩm.
* Sử dụng thành phần Link để tạo một liên kết đến trang chi tiết sản phẩm với đường dẫn /product/${product.\_id}.
* Sử dụng thành phần Rating để hiển thị số sao dựa trên giá trị product.rating.
* Sử dụng các thành phần khác để hiển thị tiêu đề, đánh giá, giá sản phẩm, và hình ảnh.
* Cuối cùng, component Product được xuất khẩu để có thể sử dụng trong các file khác.

1. **Component ProductCarousel**

* Component ProductCarousel được sử dụng để hiển thị carousel các sản phẩm.
* Sử dụng JSX để tạo các thành phần UI như Carousel, Carousel.Item, Link, Image, và Carousel.Caption từ thư viện react-bootstrap và react-router-dom.
* Ràng buộc dữ liệu động bằng cách sử dụng các biểu thức JavaScript để truyền dữ liệu từ biến products (có được từ useGetTopProductsQuery) vào giao diện.
* Tích hợp với các thư viện và module khác như 'react-router-dom' để tạo liên kết đến trang chi tiết sản phẩm và 'react-bootstrap' để hiển thị carousel và hình ảnh sản phẩm.
* Xử lý điều kiện bằng cách sử dụng cú pháp điều kiện để xử lý trạng thái tải dữ liệu (isLoading, error) và hiển thị thông báo lỗi (Message) hoặc carousel và thông tin sản phẩm.
* Component ProductCarousel được xuất khẩu để có thể sử dụng trong các file khác.

1. **Component Rating**

* Component Rating được sử dụng để hiển thị đánh giá (rating) của sản phẩm.
* JSX được sử dụng để tạo các thành phần UI như <div> và <span> để định dạng và hiển thị đánh giá.
* Sử dụng cú pháp điều kiện để xác định xem nên hiển thị sao đầy (<FaStar />), nửa sao (<FaStarHalfAlt />), hay sao trống (<FaRegStar />) dựa trên giá trị của prop value.
* Sử dụng các thành phần từ thư viện 'react-icons/fa' như <FaStar />, <FaStarHalfAlt />, và <FaRegStar /> để hiển thị biểu tượng sao trong đánh giá.
* Ràng buộc dữ liệu động bằng cách sử dụng biến value để truyền giá trị đánh giá từ props của thành phần Rating.
* Component Rating cũng sử dụng biến text để hiển thị văn bản liên quan đến đánh giá (nếu có).
* Giá trị mặc định cho prop color được đặt trong Rating.defaultProps là '#f8e825' nếu không có giá trị color được truyền vào từ props.

1. **Component SearchBox**

* Component SearchBox được sử dụng để tạo hộp tìm kiếm.
* JSX được sử dụng để tạo các thành phần UI như <Form>, <Form.Control>, và <Button> từ thư viện 'react-bootstrap'.
* Sử dụng onSubmit trong thành phần <Form> để định nghĩa hành động xử lý khi người dùng gửi form. Hàm submitHandler sẽ được gọi để xử lý sự kiện gửi form.
* Ràng buộc dữ liệu động bằng cách sử dụng biến keyword và hàm setKeyword từ hook useState để ràng buộc và cập nhật giá trị của trường nhập liệu <Form.Control>. Giá trị này được đặt ban đầu là urlKeyword từ hook useParams (được truyền qua URL) hoặc rỗng nếu urlKeyword không tồn tại.
* Sử dụng hook useNavigate từ 'react-router-dom' để điều hướng người dùng đến các trang khác.
* Sử dụng các lớp CSS như 'd-flex', 'mr-sm-2', 'ml-sm-5', 'p-2', và 'mx-2' để định dạng các phần tử giao diện và tạo khoảng cách giữa chúng.

# **CHƯƠNG 5: TỔNG KẾT**

* Kiến trúc và chức năng đã phân tích trong bài báo cáo này cung cấp một hình dung tổng quan về cấu trúc và chức năng của một hệ thống phần mềm.
* Việc phân tích chi tiết các thành phần và chức năng trong kiến trúc giúp chúng ta hiểu rõ cách mà các thành phần tương tác với nhau và cung cấp những chức năng cụ thể cho hệ thống.
* Từ việc phân tích này, chúng ta có thể nhận thấy rằng kiến trúc được xây dựng dựa trên một mô hình phân tầng giữa phía client và phía server. Điều này giúp tách biệt quyền truy cập và xử lý dữ liệu giữa client và server, cung cấp tính bảo mật và mở rộng cho hệ thống.
* Chức năng chính của phía client bao gồm xem danh sách sản phẩm, xem chi tiết sản phẩm, quản lý giỏ hàng và xác nhận đặt hàng. Điều này cho phép người dùng tương tác với hệ thống và thực hiện các hoạt động như xem, thêm, chỉnh sửa và xóa sản phẩm.
* Tương tác với backend thông qua việc sử dụng API giúp trao đổi dữ liệu giữa phía client và phía server. Việc sử dụng các hooks và query trong React giúp thực hiện các yêu cầu API một cách dễ dàng và hiệu quả.
* Tổng kết lại, việc phân tích kiến trúc và chức năng trong bài báo cáo này đã cung cấp một cái nhìn tổng quan về cấu trúc và hoạt động của một hệ thống phần mềm. Điều này giúp chúng ta hiểu rõ hơn về các thành phần, tương tác và chức năng của hệ thống.