**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**⬩⬩🙠🕮🙢⬩⬩**



**CÔNG NGHỆ PHẦN MỀM MỚI**

**XÂY DỰNG WEBSITE BÁN DỤNG CỤ THỂ THAO VỚI CÔNG NGHỆ MEAN STACK**

**GVHD: TS. Lê Vĩnh Thịnh**

**Sinh viên thực hiện MSSV**

**Trần Tấn Quý 16110196**

**Phạm Hoàng Việt 16110547**

**Thành phố Hồ Chí Minh, tháng 12 năm 2019**

**NHẬN XÉT CỦA GIẢNG VIÊN**

Chữ ký

**TS. Lê Vĩnh Thịnh**

**MỤC LỤC**

**DANH SÁCH TỪ VIẾT TẮT**

**DANH SÁCH HÌNH ẢNH**

**DANH SÁCH BẢNG**

# **NỘI DUNG**

# ***1. Mô tả đồ án***

## **1.1. Giới thiệu đề tài**

SPORTSTORE là một trang web thương mại điện tử được xây dựng với mục đích chuyên buôn bán các mặt hàng, dụng cụ thể thao cho những người yêu thích vận động, có niềm đam mê với thể dục thể thao. Trang web này sẽ tạo sự tiện lợi cho người dùng bằng việc cung cấp các thông tin về các mặt hàng dụng cụ thể thao, hỗ trợ người dùng mua và đặt hàng trực tuyến các sản phẩm sẵn có của cửa hàng.

## **1.2. Mục tiêu đề tài**

Trang web SPORTSTORE được xây dựng với các tính năng chính phục vụ cho các actor bao gồm khách, người dùng và người quản trị như sau:

- Đăng ký

- Đăng nhập

- Đăng xuất

- Xem sản phẩm

- Xem chi tiết sản phẩm

- Mua hàng và đặt hàng trực tuyến

- Quản lý sản phẩm

- Quản lý đơn hàng

- Quản lý tài khoản người dùng

- Xác nhận đơn hàng và gửi email đến người dùng (hệ thống)

**1.3. Chức năng cho các actors**

Bảng 1. 1. Chức năng của các actors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Chức năng** | **Khách** | **Người dùng** | **Người quản trị** |
| 1 | Đăng ký | x |  |  |
| 2 | Đăng nhập | x | x | x |
| 3 | Đăng xuất |  | x | x |
| 4 | Xem sản phẩm | x | x |  |
| 5 | Xem chi tiết sản phẩm | x | x |  |
| 6 | Mua hàng (quản lý giỏ hàng) |  | x |  |
| 7 | Đặt hàng |  | x |  |
| 8 | Quản lý sản phẩm |  |  | x |
| 9 | Quản lý đơn hàng |  |  | x |
| 10 | Quản lý tài khoản người dùng |  |  | x |

## **1.4. Lược đồ usecase**



## **1.5. Mô tả usecase**

Bảng 1. 2. Mô tả usecase 'Đăng ký'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Đăng ký | | |
| Mô tả | Cho phép khách đăng ký tài khoản mới | | |
| Actor | Khách | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào web app |  |
|  | 2 | Nhấn nút ‘Sign up’ |  |
|  | 3 |  | Hiện ra giao diện đăng ký |
|  | 4 | Nhập đầy đủ thông tin yêu cầu |  |
|  | 5 | Nhấn nút ‘Sign up’ để tạo tài khoản mới |  |
|  | 6 |  | Lưu tài khoản vừa tạo xuống database |
| Điều kiện cần | - Kết nối internet | | |
| Kết quả | - Khi khách đăng ký thành công, hiện ra giao diện đăng nhập  - Khi khách đăng ký thất bại, hiển thị lỗi | | |

Bảng 1. 3. Mô tả usecase 'đăng nhập với tài khoản hệ thống'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Đăng nhập với tài khoản hệ thống | | |
| Mô tả | Cho phép khách đăng nhập vào website | | |
| Actor | Khách | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Nhấn nút ‘Sign in’ |  |
|  | 3 |  | Hiện ra giao diện đăng nhập |
|  | 4 | Nhập tài khoản & mật khẩu sau đó click nút ‘Login’ |  |
|  | 5 |  | Xử lý tài khoản và mật khẩu |
| Điều kiện cần | - Đã có tài khoản trong hệ thống  - Kết nối internet | | |
| Kết quả | - Khi khách đăng nhập thành công, chuyển đến trang chủ  - Khi khách đăng nhập thất bại, hiển thị lỗi | | |

Bảng 1. 4. Mô tả usecase 'Đăng nhập với tài khoản Google'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Đăng nhập với tài khoản Google | | |
| Mô tả | Cho phép khách đăng nhập vào website | | |
| Actor | Khách | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Nhấn nút ‘Sign in’ |  |
|  | 3 |  | Hiện ra giao diện đăng nhập |
|  | 4 | Nhấn nút ‘Google’ |  |
|  | 5 |  | Chuyển tới giao diện đăng nhập Google |
|  | 6 | Nhập tài khoản Google và mật khẩu |  |
|  | 7 | Chọn tài khoản Google để đăng nhập (nếu đã có sẵn) |  |
|  | 8 |  | Xử lý đăng nhập với OAuth |
| Điều kiện cần | - Kết nối internet | | |
| Kết quả | - Khi khách đăng nhập thành công, chuyển đến trang chủ  - Khi khách đăng nhập thất bại, hiển thị lỗi | | |

Bảng 1. 5. Mô tả usecase 'Đăng nhập với tài khoản Facebook'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Đăng nhập với tài khoản Facebook | | |
| Mô tả | Cho phép khách đăng nhập vào website | | |
| Actor | Khách | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Nhấn nút ‘Sign in’ |  |
|  | 3 |  | Hiện ra giao diện đăng nhập |
|  | 4 | Nhấn nút ‘Facebook’ |  |
|  | 5 |  | Chuyển tới giao diện đăng nhập Facebook |
|  | 6 | Nhập tài khoản Facebook và mật khẩu |  |
|  | 7 | Chọn tài khoản Facebook để đăng nhập (nếu đã có sẵn) |  |
|  | 8 |  | Xử lý đăng nhập với OAuth |
| Điều kiện cần | - Kết nối internet | | |
| Kết quả | - Khi khách đăng nhập thành công, chuyển đến trang chủ  - Khi khách đăng nhập thất bại, hiển thị lỗi | | |

Bảng 1. 6. Mô tả usecase 'Xem sản phẩm'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Xem sản phẩm | | |
| Mô tả | Cho phép khách và người dùng xem thông tin về sản phẩm | | |
| Actor | Khách, người dùng | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào web app |  |
|  | 2 | Nhấn nút ‘Product’ |  |
|  | 3 |  | Hiện ra giao diện với các sản phẩm |
|  | 4 | Chọn một sản phẩm |  |
|  | 5 |  | Hiện ra thông tin của sản phẩm đó |
| Điều kiện cần | - Kết nối internet | | |
| Kết quả | Khi sản phẩm không thể tải lên, hiển thị lỗi | | |

Bảng 1. 7. Mô tả usecase 'Đăng xuất'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Đăng xuất | | |
| Mô tả | Người dùng có đăng xuất khỏi hệ thống | | |
| Actor | Người dùng, người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Nhấn nút ‘Sign out’ |  |
|  | 2 |  | Disconnect người dùng ra khỏi website |
|  | 3 |  | Điều hướng người dùng về trang chính |
| Điều kiện cần | - User đã đăng nhập  - Kết nối internet | | |
| Kết quả | - Nếu website không load thành công, hiện ra thông báo lỗi | | |

Bảng 1. 8. Mô tả usecase ‘Quản lý giỏ hàng’

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Quản lý giỏ hàng | | |
| Mô tả | Người dùng có thể quản lý giỏ hàng của bản thân | | |
| Actor | Người dùng | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Cart’ |  |
|  | 4 |  | Hiện giao diện giỏ hàng |
| Điều kiện cần | - User đã đăng nhập  - Kết nối internet | | |
| Kết quả | - Nếu giỏ hàng load không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 9. Mô tả usecase ‘Thêm sản phẩm vào giỏ hàng’

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Thêm sản phẩm vào giỏ hàng | | |
| Mô tả | Người dùng có thêm sản phẩm vào giỏ hàng | | |
| Actor | Người dùng | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Add to cart’ của sản phẩm mong muốn |  |
|  | 4 |  | Thêm sản phẩm vào giỏ hàng và chuyển tới giao diện giỏ hàng |
| Điều kiện cần | - User đã đăng nhập  - Kết nối internet | | |
| Kết quả | - Nếu giỏ hàng load không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 10. Mô tả usecase 'Sửa số lượng sản phẩm’

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Sửa số lượng sản phẩm | | |
| Mô tả | Người dùng có thể sửa số lượng sản phẩm trong giỏ hàng | | |
| Actor | Người dùng | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Cart’ |  |
|  | 4 |  | Hiện giao diện giỏ hàng |
|  | 5 | Người dùng tăng/giảm số lượng của từng sản phẩm |  |
|  | 6 |  | Số tiền tăng/giảm dựa trên số lượng thay đổi |
| Điều kiện cần | - User đã đăng nhập  - Kết nối internet | | |
| Kết quả | - Nếu giỏ hàng load không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 11. Mô tả usecase 'Xóa sản phẩm khỏi giỏ hàng'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Xóa sản phẩm khỏi giỏ hàng | | |
| Mô tả | Người dùng có thể xóa sản phẩm khỏi giỏ hàng | | |
| Actor | Người dùng | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Cart’ |  |
|  | 4 |  | Hiện giao diện giỏ hàng |
|  | 5 | Người dùng xóa sản phẩm bằng cách nhất nút ‘Remove’ |  |
|  | 6 |  | Số tiền giảm dựa trên các sản phẩm đã được xóa khỏi giỏ hàng |
| Điều kiện cần | - User đã đăng nhập  - Kết nối internet | | |
| Kết quả | - Nếu giỏ hàng load không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 12. Mô tả usecase 'Đặt hàng'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Đặt hàng | | |
| Mô tả | Người dùng có thể đặt hàng | | |
| Actor | Người dùng | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Cart’ |  |
|  | 4 |  | Hiện giao diện giỏ hàng |
|  | 5 | Người dùng nhấn nút ‘Proceed to checkout’ để tiến hành đặt hàng |  |
|  | 6 |  | Hiện giao diện đặt hàng |
|  | 7 | Người dùng nhập thông tin cá nhân và nhấn nút ‘Checkout’ |  |
|  | 8 |  | Redirect người dùng tới trang ‘Thank you’ và gửi email cho người dùng |
| Điều kiện cần | - User đã đăng nhập  - Kết nối internet | | |
| Kết quả | - Nếu giỏ hàng load không thành công, hiện ra thông báo lỗi  - Nếu đặt hàng không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 13. Mô tả useacse 'Quản lý sản phẩm'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Quản lý sản phẩm | | |
| Mô tả | Người quản trị có thể quản lý sản phẩm của cửa hàng | | |
| Actor | Người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Product management’ |  |
|  | 4 |  | Hiện giao diện quản lý sản phẩm |
| Điều kiện cần | - User đã đăng nhập dưới vai trò quản trị  - Kết nối internet | | |
| Kết quả | - Nếu sản phẩm load không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 14. Mô tả usecase 'Thêm sản phẩm'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Thêm sản phẩm | | |
| Mô tả | Người quản trị có thể thêm sản phẩm mới | | |
| Actor | Người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Product management’ |  |
|  | 4 |  | Hiện giao diện quản lý sản phẩm |
|  | 5 | Nhấn nút ‘Add a new product’ |  |
|  |  |  | Hiện giao diện thêm sản phẩm |
|  | 6 | Nhập thông tin sản phẩm và nhấn nút ‘Add’ |  |
|  | 7 |  | Lưu sản phẩm xuống database và load lại trang |
| Điều kiện cần | - User đã đăng nhập dưới vai trò quản trị  - Kết nối internet | | |
| Kết quả | - Nếu sản phẩm load không thành công, hiện ra thông báo lỗi  - Nếu thêm sản phẩm thành công, hiện thông báo  - Nếu thêm sản phẩm không thành công, hiện thông báo lỗi | | |

Bảng 1. 15. Mô tả usecase 'Sửa sản phẩm'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Sửa sản phẩm | | |
| Mô tả | Người quản trị có thể sửa thông tin sản phẩm | | |
| Actor | Người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Product management’ |  |
|  | 4 |  | Hiện giao diện quản lý sản phẩm |
|  | 5 | Nhấn biểu tượng sửa trên một sản phẩm |  |
|  |  |  | Hiện giao diện sửa sản phẩm |
|  | 6 | Nhập thông tin sản phẩm cần sửa và nhấn nút ‘Save’ |  |
|  | 7 |  | Lưu sản phẩm xuống database và load lại trang |
| Điều kiện cần | - User đã đăng nhập dưới vai trò quản trị  - Kết nối internet | | |
| Kết quả | - Nếu sản phẩm load không thành công, hiện ra thông báo lỗi  - Nếu lưu sản phẩm thành công, hiện thông báo  - Nếu lưu sản phẩm không thành công, hiện thông báo lỗi | | |

Bảng 1. 16. Mô tả usecase 'Xóa sản phẩm'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Xóa sản phẩm | | |
| Mô tả | Người quản trị có thể xóa sản phẩm | | |
| Actor | Người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Product management’ |  |
|  | 4 |  | Hiện giao diện quản lý sản phẩm |
|  | 5 | Nhấn biểu tượng xóa trên một sản phẩm |  |
|  |  |  | Hiện giao diện confirm delete |
|  | 6 | Chọn ‘yes’ |  |
|  | 7 |  | Xóa sản phẩm khỏi database và load lại trang |
| Điều kiện cần | - User đã đăng nhập dưới vai trò quản trị  - Kết nối internet | | |
| Kết quả | - Nếu sản phẩm load không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 17. Mô tả usecase 'Quản lý tài khoản người dùng'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Quản lý tài khoản người dùng | | |
| Mô tả | Người quản trị có thể quản lý tài khoản của người dùng | | |
| Actor | Người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Account management’ |  |
|  | 4 |  | Hiện giao diện quản lý tài khoản |
| Điều kiện cần | - User đã đăng nhập dưới vai trò quản trị  - Kết nối internet | | |
| Kết quả | - Nếu tài khoản load không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 18. Mô tả usecase 'Sửa tài khoản'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Sửa tài khoản | | |
| Mô tả | Người quản trị có thể sửa thông tin tài khoản người dùng | | |
| Actor | Người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Account management’ |  |
|  | 4 |  | Hiện giao diện quản lý tài khoản |
|  | 5 | Nhấn biểu tượng sửa trên một tài khoản bất kỳ |  |
|  | 6 |  | Hiện giao diện sửa tài khoản |
|  | 7 | Nhập thông tin cần sửa cho tài khoản và nhấn nút ‘Save’ |  |
|  | 8 |  | Lưu thông tin tài khoản xuống database |
| Điều kiện cần | - User đã đăng nhập dưới vai trò quản trị  - Kết nối internet | | |
| Kết quả | - Nếu tài khoản load không thành công, hiện ra thông báo lỗi  - Nếu sửa tài khoản thành công, hiện ra thông báo  - Nếu sửa tài khoản không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 19. Mô tả usecase 'Xóa tài khoản'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Xóa tài khoản | | |
| Mô tả | Người quản trị có thể xóa tài khoản người dùng | | |
| Actor | Người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Account management’ |  |
|  | 4 |  | Hiện giao diện quản lý tài khoản |
|  | 5 | Nhấn biểu tượng xóa trên một tài khoản bất kỳ |  |
|  | 6 |  | Hiện giao diện confirm delete |
|  | 7 | Chọn ‘yes’ |  |
|  | 8 |  | Xóa tài khoản khỏi cơ sở dữ liệu |
| Điều kiện cần | - User đã đăng nhập dưới vai trò quản trị  - Kết nối internet | | |
| Kết quả | - Nếu tài khoản load không thành công, hiện ra thông báo lỗi  - Nếu xóa tài khoản thành công, hiện ra thông báo  - Nếu xóa tài khoản không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 20. Mô tả usecase 'Quản lý đơn hàng'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Quản lý đơn hàng | | |
| Mô tả | Người quản trị có thể quản lý đơn hàng | | |
| Actor | Người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Orders management’ |  |
|  | 4 |  | Hiện giao diện quản lý đơn hàng |
| Điều kiện cần | - User đã đăng nhập dưới vai trò quản trị  - Kết nối internet | | |
| Kết quả | - Nếu đơn hàng load không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 21. Mô tả usecase 'Sửa đơn hàng'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Sửa đơn hàng | | |
| Mô tả | Người quản trị có thể sửa thông tin đơn hàng | | |
| Actor | Người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Orders management’ |  |
|  | 4 |  | Hiện giao diện quản lý đơn hàng |
|  | 5 | Nhấn biểu tượng sửa trên một đơn hàng bất kỳ |  |
|  | 6 |  | Hiện giao diện sửa đơn hàng |
|  | 7 | Nhập thông tin cần sửa cho đơn hàng và nhấn nút ‘Save’ |  |
|  | 8 |  | Lưu thông tin đơn hàng xuống database |
| Điều kiện cần | - User đã đăng nhập dưới vai trò quản trị  - Kết nối internet | | |
| Kết quả | - Nếu đơn hàng load không thành công, hiện ra thông báo lỗi  - Nếu sửa đơn hàng thành công, hiện ra thông báo  - Nếu sửa đơn hàng không thành công, hiện ra thông báo lỗi | | |

Bảng 1. 22. Mô tả usecase 'Xóa đơn hàng'

|  |  |  |  |
| --- | --- | --- | --- |
| Tên usecase | Xóa đơn hàng | | |
| Mô tả | Người quản trị có thể xóa đơn hàng | | |
| Actor | Người quản trị | | |
| Quy trình nghiệp vụ | STT | Actor | Phản hồi của hệ thống |
|  | 1 | Truy cập vào website |  |
|  | 2 | Đăng nhập vào website |  |
|  | 3 | Nhấn nút ‘Orders management’ |  |
|  | 4 |  | Hiện giao diện quản lý đơn hàng |
|  | 5 | Nhấn biểu tượng xóa trên một đơn hàng bất kỳ |  |
|  | 6 |  | Hiện giao diện confirm delete |
|  | 7 | Nhấn ‘Yes’ |  |
|  | 8 |  | Xóa đơn hàng khỏi database |
| Điều kiện cần | - User đã đăng nhập dưới vai trò quản trị  - Kết nối internet | | |
| Kết quả | - Nếu đơn hàng load không thành công, hiện ra thông báo lỗi  - Nếu xóa đơn hàng thành công, hiện ra thông báo  - Nếu xóa đơn hàng không thành công, hiện ra thông báo lỗi | | |

**2. Thiết kế hệ thống**

## **2.1. Công nghệ**

***2.1.1. OAuth***

**2.1.1.1. OAuth là gì?**

OAuth là một phương thức chứng thực các ứng dụng có thể chia sẻ tài nguyên với nhau mà không cần chia sẻ các thông tin username và mật khẩu. Trong OAuth, O tượng trưng cho ‘Open’ và ‘Auth’ tượng trưng cho Authentication (xác thực người dùng) & Authorization (ủy quyền cho ứng dụng truy cập tài nguyên của người dùng).

OAuth có 2 phiên bản là OAuth 1.0 và OAuth 2.0, trong đó OAuth 2.0 được sử dụng rất phổ biến trong các ứng dụng/website công nghệ thông tin ngày nay. Một số thuật ngữ trong OAuth 2.0:

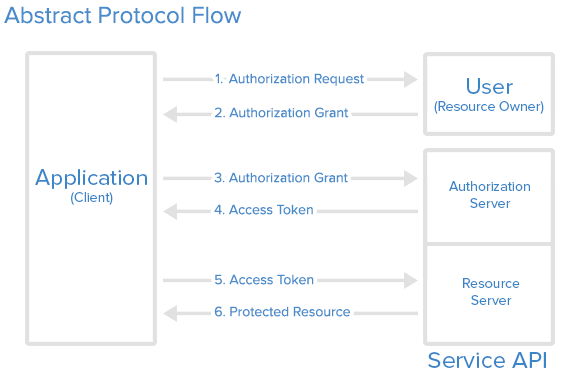
- **Resource owner:** có thể hiểu là người dùng.

**- Client:** ứng dụng của bên thứ 3. VD: một web app sử dụng Google API.

**- Authorization server:** là một đối tượng cấp quyền cho client để truy cập vào dữ liệu của server.

**- Resource server:** nơi chứa thông tin, dữ liệu cần chia sẻ.

**2.1.1.2. Cách hoạt động của OAuth 2.0**



Hình 2. 1. Luồng hoạt động của OAuth 2.0 [1]

Luồng hoạt động có thể được tóm gọn thành 3 luồng chính:

- Client gửi yêu cầu authorization đến cho user (resource owner), nếu user chấp nhận thì sẽ tiến hành cấp quyền cho client.

- Sau khi có authorization grant, client tiến hành gửi nó đến Authorization Server và sau khi xác nhận xong, server sẽ trả về một Access Token.

- Client sẽ sử dụng access token trong suốt quá trình làm việc với resource server. Resource server đóng vai trò kiểm tra tính hợp lệ của token mỗi khi nó được gửi lên và trả về dữ liệu cho client.

Trong OAuth, với các thông tin được gửi đi sẽ luôn đi kèm với 2 tham số quan trọng nhất:

- Client identifier (ClientID): là một chuỗi ký tự được sử dụng bởi service api để định danh client.

- Client secret: là chuỗi ký tự dùng để xác thực định danh ứng dụng khi ứng dụng yêu cầu truy cập vào thông tin tài khoản của user. Đây là chuỗi được giữ bí mật giữa client và authorization server.

**2.1.1.3. Ứng dụng của OAuth**

OAuth được ứng dụng nhiều nhất trong việc đăng nhập bằng cách sử dụng credential của các bên thứ ba. Nó giúp cho việc tiếp cận đến một trang web dễ dàng và nhanh chóng hơn mà không cần phải tiến hành đăng ký một tài khoản với nhiều bước rườm rà từ đó có quá nhiều tài khoản cần phải ghi nhớ bởi người dùng.

Những doanh nghiệp lớn như Facebook, Google, Twitter, Github,... là những doanh nghiệp cho phép người dùng sử dụng API để thiết lập OAuth cho riêng mình.

***2.1.2. PassportJS***

**2.1.2.1. PassportJS là gì?**

Passport.js là một module của NodeJS được ứng dụng trong việc authentication cho ứng dụng. Nó linh hoạt ở chỗ chúng ta có thể sử dụng để xác thực bằng nhiều trường hợp khác nhau như Facebook, Google, … hay dựa vào Token.



**2.1.2.2. Các hàm và middleware trong Passport.js**

- passport.initialize: kiểm tra session lấy ra passport.user nếu chưa có thì tạo rỗng.  
 - passport.session: sử dụng session lấy thông tin user rồi gắn vào req.user.  
 - passport.deserializeUser: giúp ta lấy dữ liệu user dựa vào thông tin lưu trên session và gắn vào req.user.  
 - passport.authenticate: middleware giúp ta gắn kịch bản local vào route.  
 - passport.serializeUser: hàm được gọi khi xác thực thành công để lưu thông tin user vào session

**2.1.2.3. Các bước xác thực tài khoản sử dụng Passport.js**

- Khi người dùng submit form đăng nhập, một request POST được tạo ra khởi chạy middleware passport.authenticate.

- Kịch bản Local đã được thiết lập cho passport.authenticate nên nó sẽ gọi đến cái kịch bản ta đã cài đặt.

- Nó lấy dữ liệu req.body.username và req.body.passport rồi gán cho hàm verify local.

- Truy vấn database rồi kiểm tra xem passport của người dùng đưa lên có đúng không.

- Trong trường hợp Error từ database ta gọi đến callback là done với param là err( done(err)). Khi mà nó không tìm thấy được người dùng hợp lý ta gọi đến done(null,false). Còn nếu thông tin đăng nhập đúng ta gọi done(null,user).

- Khi callback done được gọi, nó sẽ lấy dữ liệu err, user và dữ liệu bạn custom thêm nếu có trả lại cho passport.authenticate.

- Nếu dữ liệu trả về của callback là null, true, xác thực thành công passport tiếp tục gọi hàm req.login( cái này tự gắn vào từng request khi bạn cài đặt passport ở bước đầu tiên)

- Hàm req.login gọi đến thằng passport.serializeUser mà ta đã định nghĩa trước đó. Hàm này truy cập vào đối tượng user mà ta trả về cho middleware passport.authenticate và xác định xem thành phần nào của đối tượng sẽ lưu vào trong session. Kết quả của hàm này là ta sẽ có đối tượng req.session.passport.user = các thông tin ta truyền vào trong serializeUser. Trong ví dụ bên trên thì nó là [user.id](http://user.id/).

- Đồng thời với trên passport cũng có gắn thông tin user vào req.user.

- Việc xác thực kết thục, hàm requestHandler sẽ được gọi đưa chúng tra đến trang đã thiết lập.

**2.1.2.4. Một số hàm hỗ trợ cho request**

- req.login(): thiết lập phiên đăng nhập.

- req.logout(): đăng xuất khỏi phiên đăng nhập.

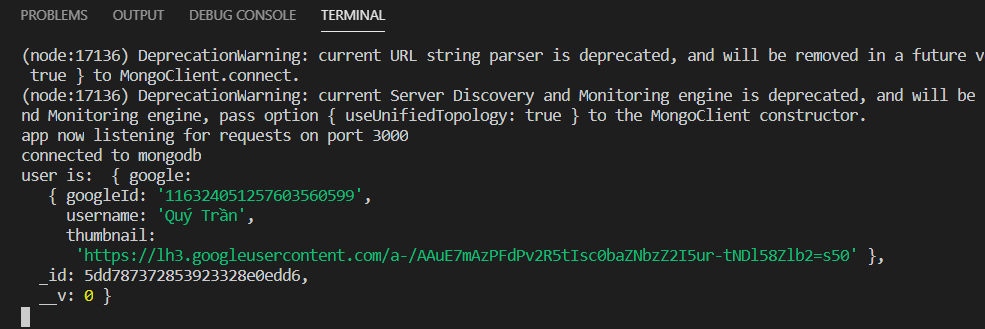
- req.isAuthenticated(): kiểm tra user đã được xác thực hay chưa. Hàm trả về true nếu user đã đăng nhập.

- req.isUnauthenticated()

**2.1.2.4. Ứng dụng PassportJS trong xác thực tài khoản sử dụng Facebook, Google trong dự án**

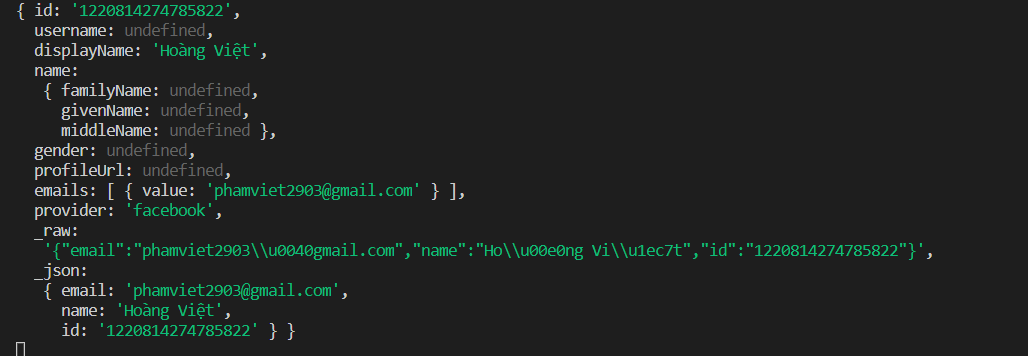
**Google:**

****

** Kết quả trả về:**

**Facebook**

****

** Kết quả trả về**

**MEAN STACK**

***2.1.3. MongoDB***

**2.1.3.1. MongoDB là gì?**



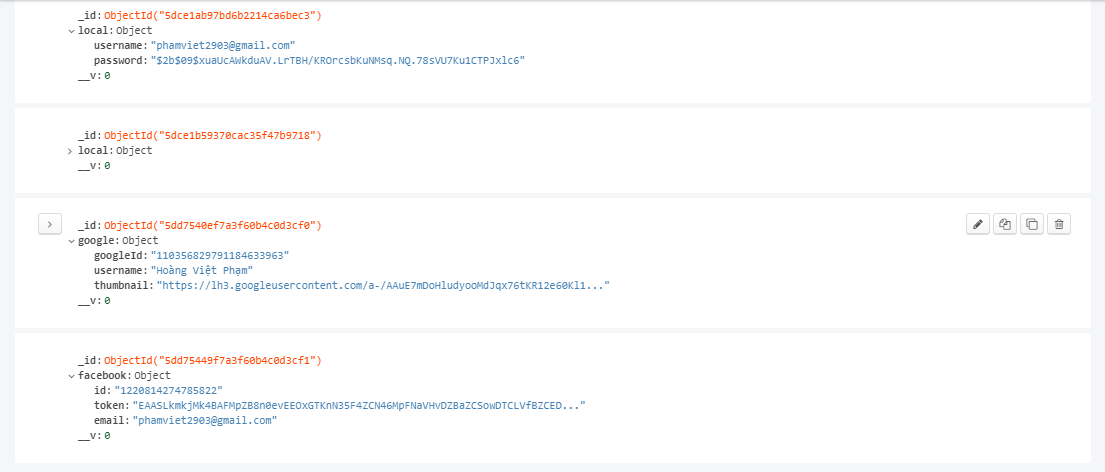
MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở thuộc học NoSQL. Nó được thiết kế theo kiểu hướng đối tượng, các bảng trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ trên bảng không cần tuân theo một cấu trúc nhất định nào cả (điều này rất thích hợp để làm big data). MongoDB lưu trữ dữ liệu theo hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON nên truy vấn sẽ rất nhanh.

**2.1.3.2. Vai trò của MongoDB trong dự án**

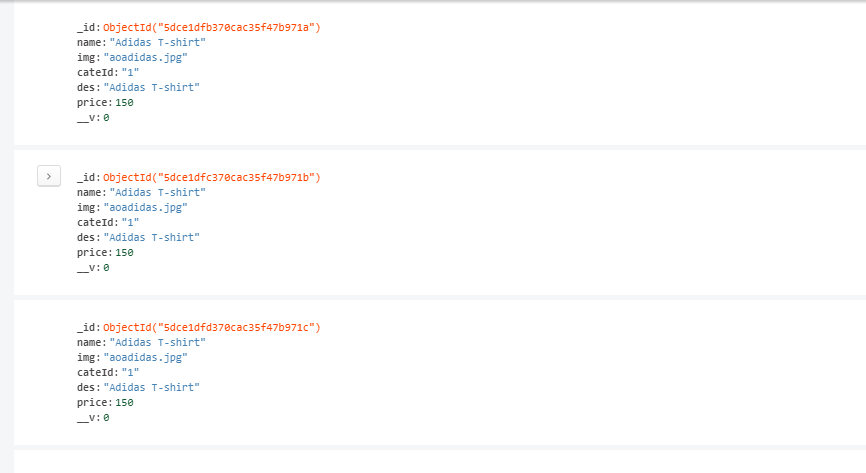
**-** MongoDB đóng vai trò như một database trong dự án, lưu trữ dữ liệu cho toàn bộ hệ thống .

- Cấu trúc Schema:

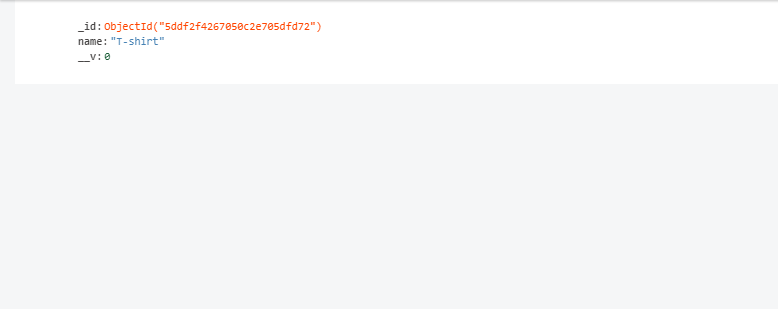
+ Users: Lưu thông tin người dùng, thông tin đăng nhập local, đăng nhập bằng Facebook, Google.

****

**+** Products: Lưu thông tin của sản phẩm được bán



+ Categories: Lưu thông tin loại sản phẩm



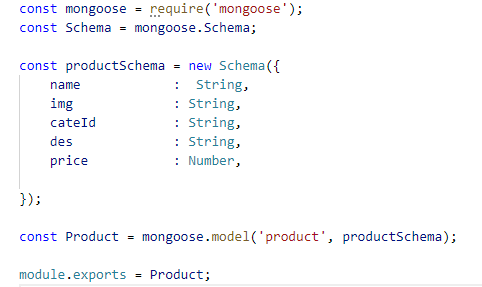
Thư viện mongoose đóng vai trò kết nối ứng dụng NodeJS với MongoDB database:



config đường dẫn tới MongoDB như sau:



Để tương tác giữa NodeJS với MongoDB chúng ta cần xây dựng các model để mapping, cấu trúc model tương tự như cấu trúc các collection trên MongoDB.



***2.1.4. ExpressJS***

**2.1.4.1. ExpressJS là gì?**



ExpressJS là một framework được xây dựng trên nền tảng của NodeJS. Đây là một framework cung cấp nhiều tính năng mạnh mẽ trên nền tảng web cũng như các ứng dụng di động, nó hỗ trợ các phương thức HTTP và middleware để tạo ra một API vô cùng mạnh mẽ và dễ sử dụng.

**2.1.4.2. Vai trò của ExpressJS trong dự án**

**-** ExpressJS giúp tạo ra các RESTful API tương tác với cơ sở dữ liệu

- Bao gồm nhiều middleware modules cho phép người dùng sử dụng để thực thi các task trên request và response.

Module Express:



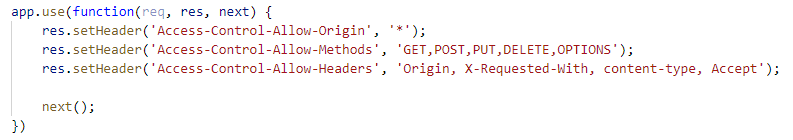
Import module để sử dụng:



Khai báo express trong file cấu hình chính:



Áp dụng middleware vào việc định nghĩa những thông tin cần thiết cho header khi gửi một request hợp lệ:

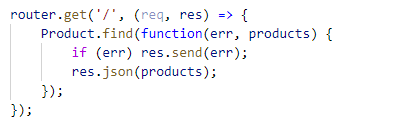


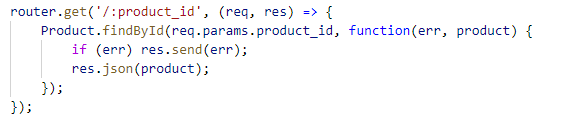
RESTful API:

Trong dự án, nhóm đã xây dựng 4 phương thức chính của RESTful API để làm việc với cơ sở dữ liệu bao gồm: GET, POST, PUT, DELETE.

Dưới đây là ví dụ trong việc xây dựng các Route để tương tác với collection Product:

GET: Phương thức GET được sử dụng để lấy ra danh sách các sản phẩm trong cơ sở dữ liệu cũng như một sản phẩm cụ thể.

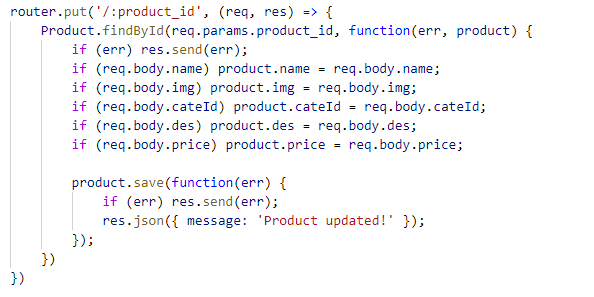




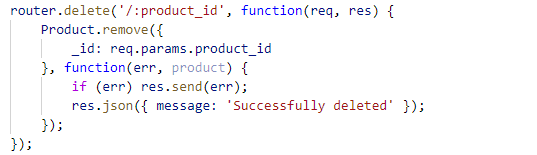
POST: Phương thức POST được dùng khi muốn thêm mới một sản phẩm vào collection.



PUT: Phương thức PUT được dùng để chỉnh sửa các thuộc tính trong sản phẩm.



DELETE: Phương thức DELETE được sử dụng khi muốn xoá đi một sản phẩm trong collection.



***2.1.5. AngularJS***

**2.1.5.1. Angular JS là gì?**



AngularJS là một framework có cấu trúc cho các ứng dụng web động. Nó cho phép người dùng sử dụng HTML như một ngôn ngữ mẫu và cho phép người dùng mở rộng cú pháp của HTML để diễn đạt các thành phần ứng dụng một cách rõ ràng và súc tích. Hai tính năng cốt lõi: Data binding và Dependency injection của AngularJS loại bỏ phần lớn code mà người dùng thường phải viết. Nó xảy ra trong tất cả các trình duyệt, làm cho nó trở thành đối tác lý tưởng của bất kỳ công nghệ Server nào [2].

**Tính năng chính của Angular JS**

**- Data-binding**: Tự động đồng bộ hóa dữ liệu giữa Model và View.

**- Module**: dùng để định nghĩa một ứng dụng AngularJS. Module giống như một chiếc hộp để chứa các thành phần khác của ứng dụng bên trong nó. Module còn được gọi là một container của ứng dụng.

**- Scope**: Là những đối tượng hướng đến model. Nó là cầu nối giữa Controller và View.

**- Controller**: Là những hàm JavaScrip xử lí kết hợp với Scope.

**- Service**: AngularJS sử dụng các API được xây dựng từ các web service để thao tác với Database.

**- Directives**: Đánh dấu vào các yếu tố của DOM, nghĩa là sẽ tạo ra các thẻ HTML tùy chỉnh.

**- Template**: Các rendered view với các thông tin từ controller và model có thể sử dụng trong các file riêng biệt hoặc với view với trang sử dụng Partials.

**- Routing**: Chuyển đổi giữa các action trong Controller.

**- Model View Controller**: Mô hình chia thành phần riêng biệt thành Model, View, Controller. Đây là một mô hình khá hay nhưng trong Angular thì nó được chế biến lại một chút thành Model View View Model.

**- Dependency Injection**: Giúp các nhà phát triển tạo ứng dụng dễ dàng hơn.

**2.1.5.2. Vai trò của Angular JS trong dự án**

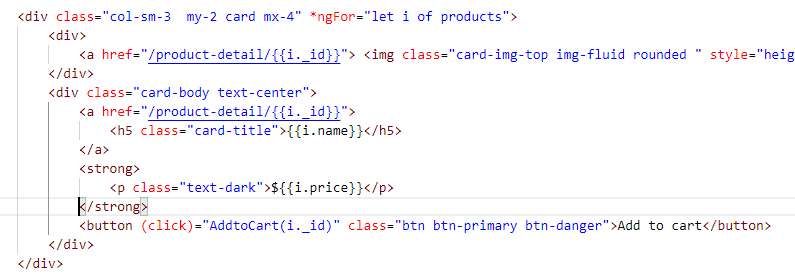
**-** AngularJS đóng vai trò tạo ra giao diện cho dự án, gọi API từ ứng dụng node để tương tác giữa giao diện với phần back-end giúp hệ thống hoạt động.

**-** Một số khái niệm và trong AngularJS được áp dụng trong dự án:

**ngIf:** được sử dựng khi muốn tạo ra một điều kiện hiển thị nào đó trên giao diện.



**ngFor:** được sử dụng khi muốn render một list các phần tử



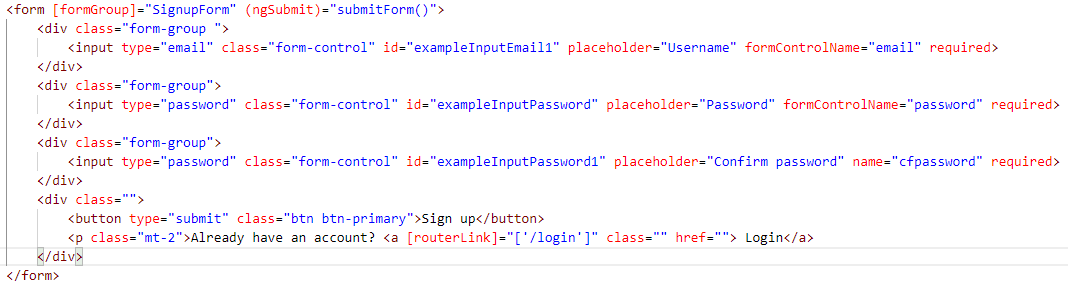
**Hiển thị dữ liệu khi render lên giao diện:**

****

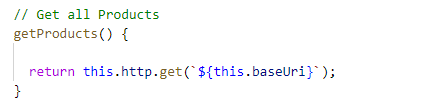
**Event:** gọi phương thức được định nghĩa trong file typescript từ giao diện



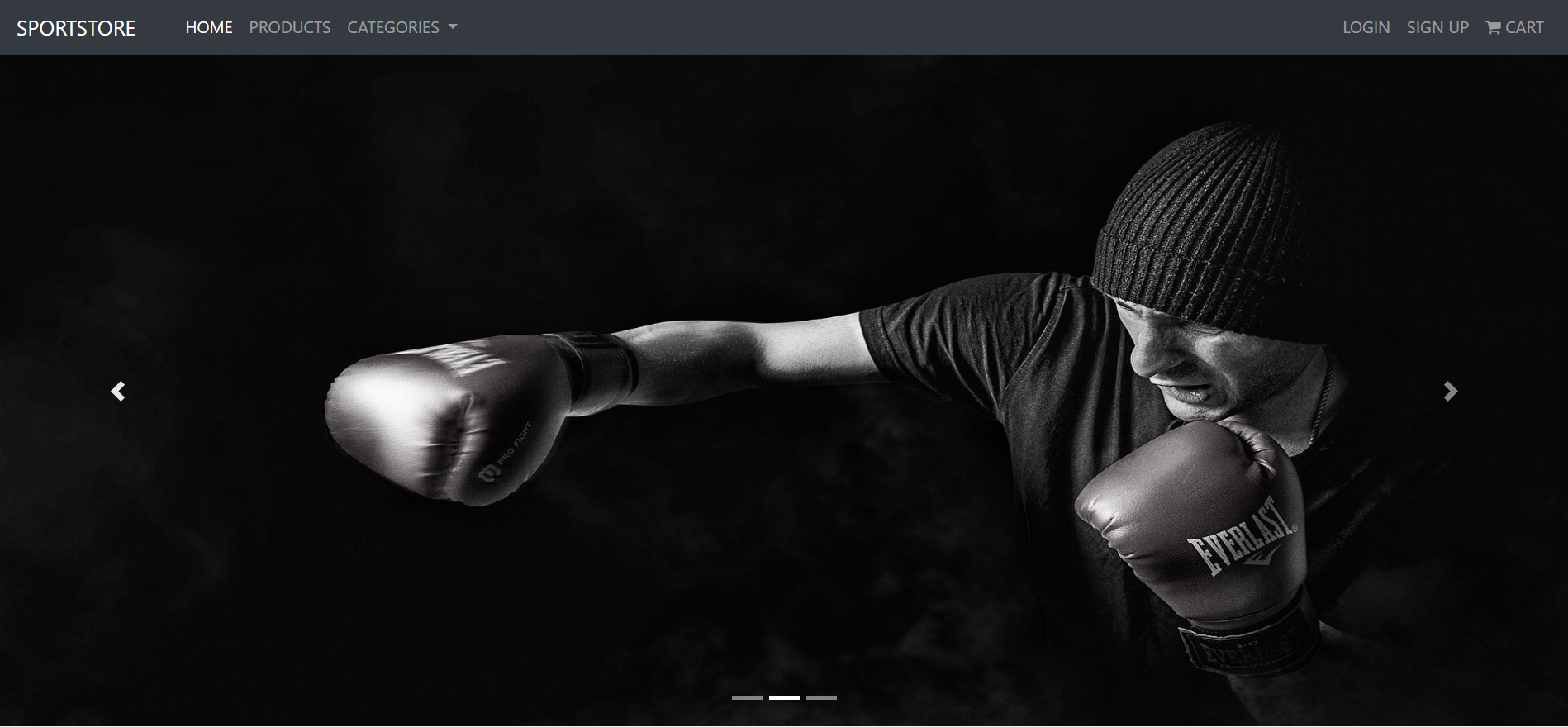
**FormGroup:** chứa các giá trị được nhập vào một form bởi người dùng trong thuộc tính formControlName và event ngSubmit để gọi phương thức xử lý tương tứng

****

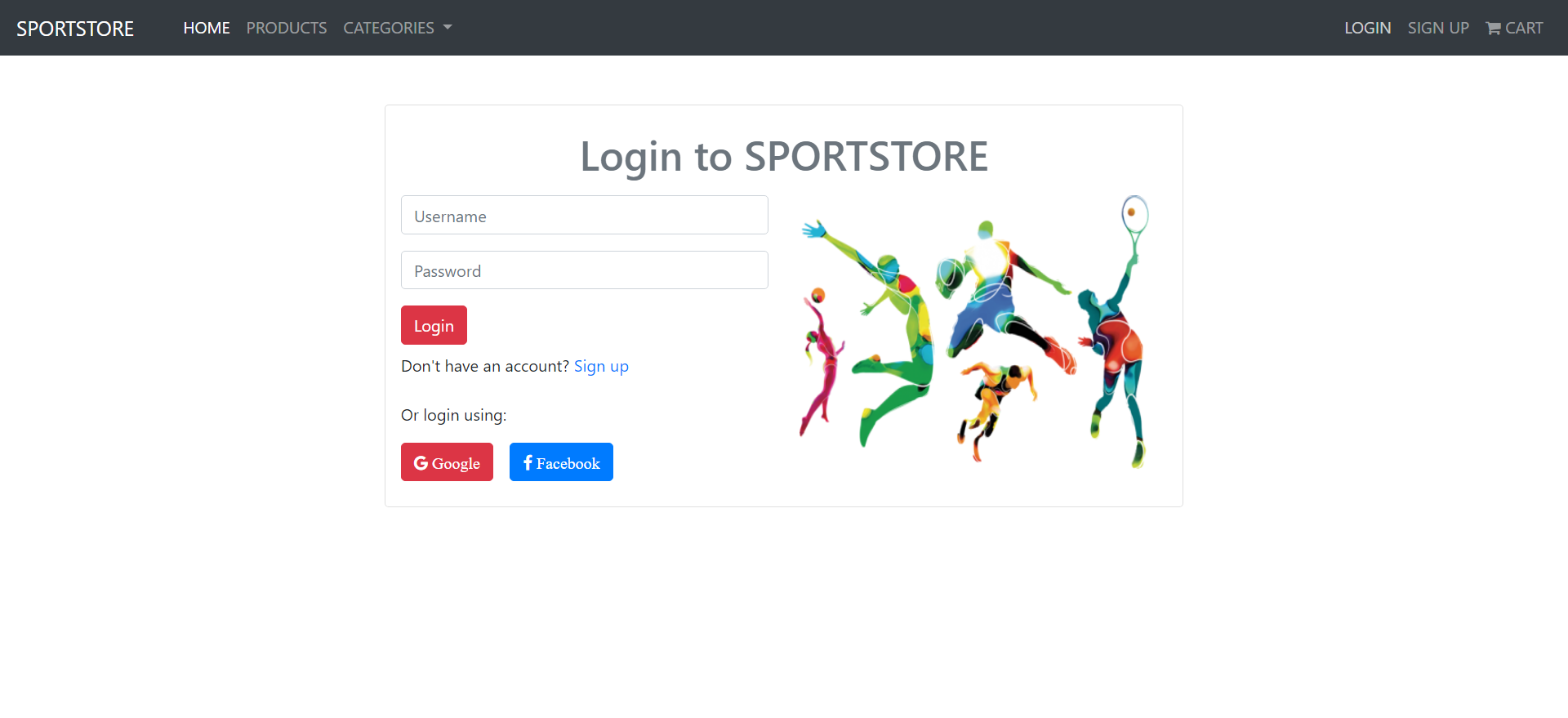
**Gọi API tới ứng dụng NodeJS:**

****

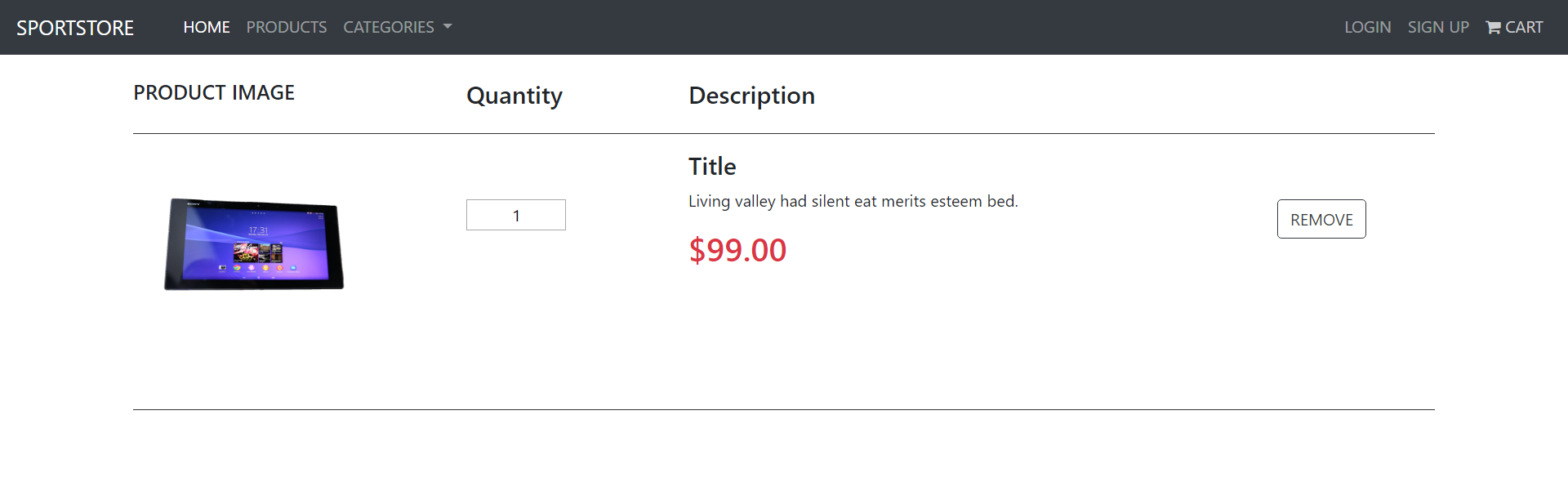
**-** Một số giao diện được thế kế sử dụng Angular:



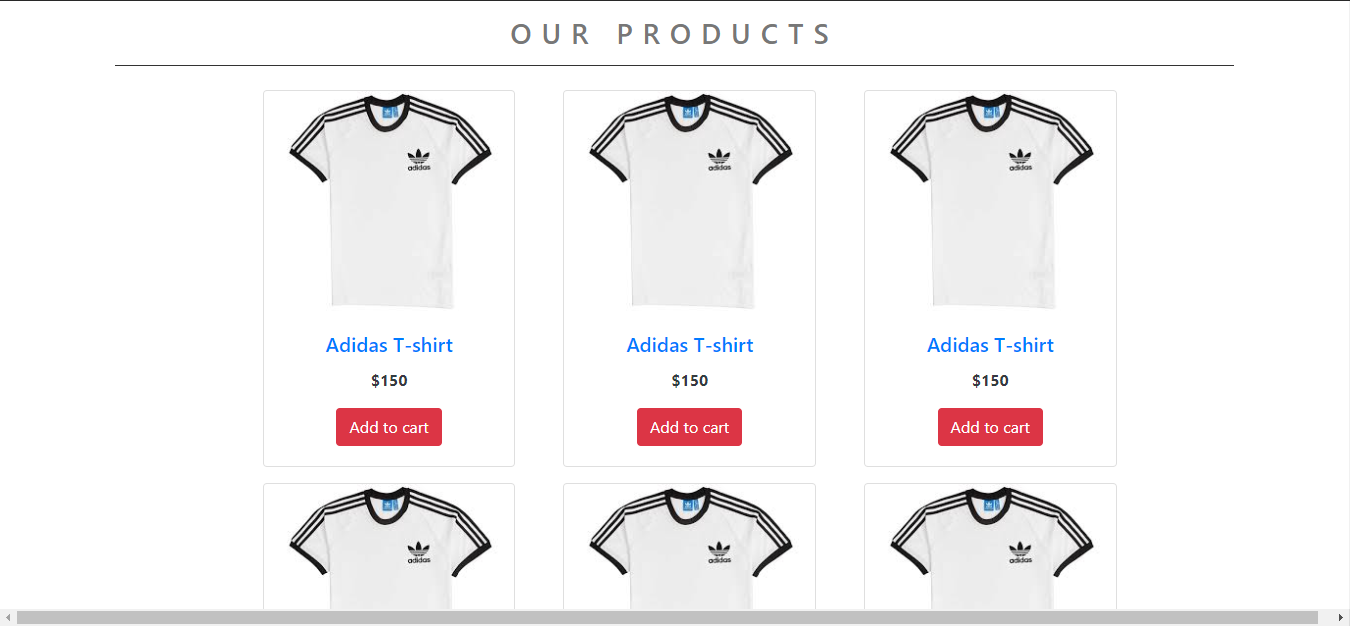
*Giao diện trang chủ*



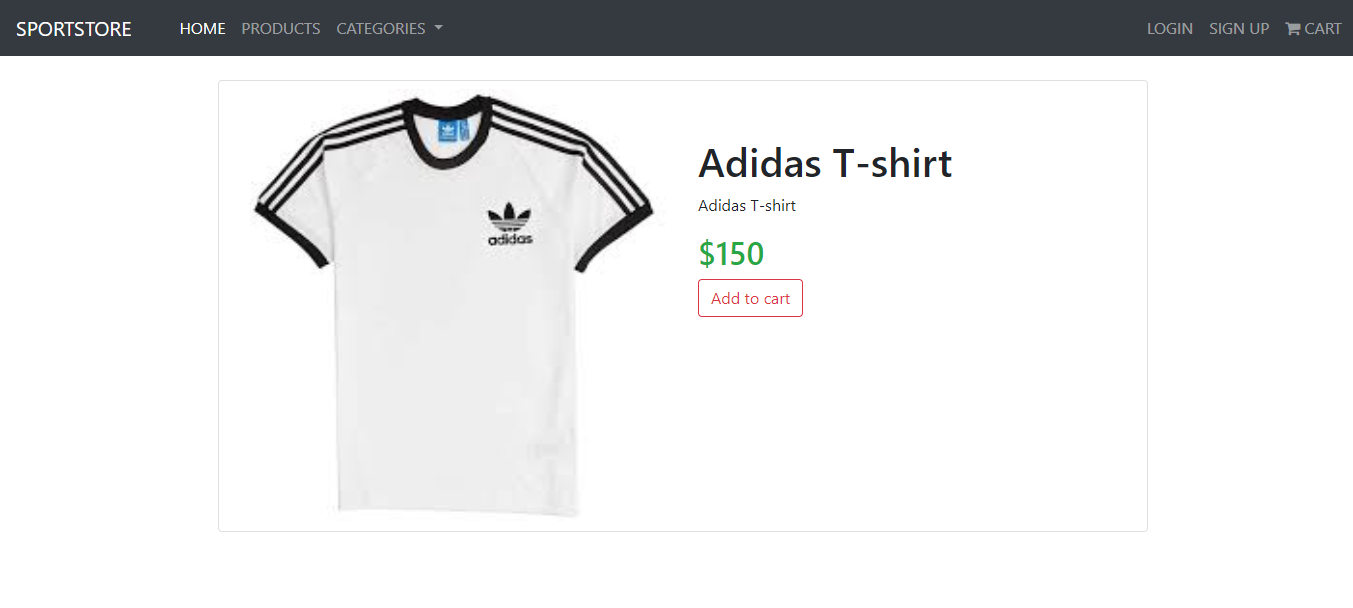
*Giao diện đăng nhập*



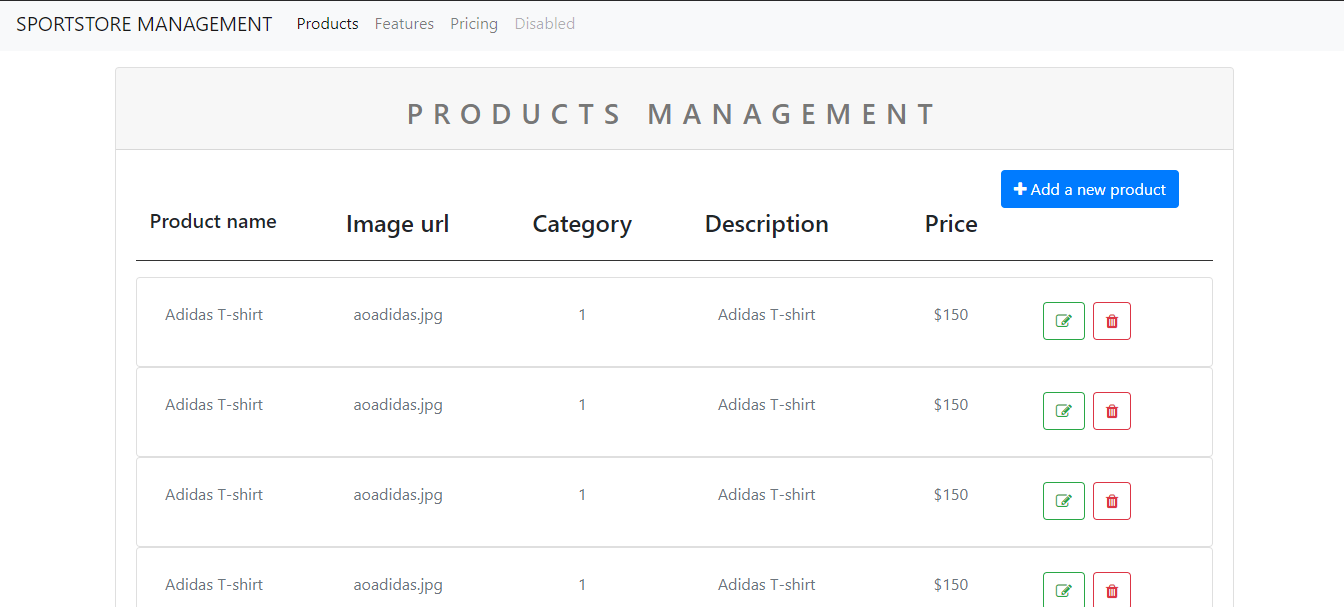
*Giao diện giỏ hàng*

******

*Giao diện sản phẩm*

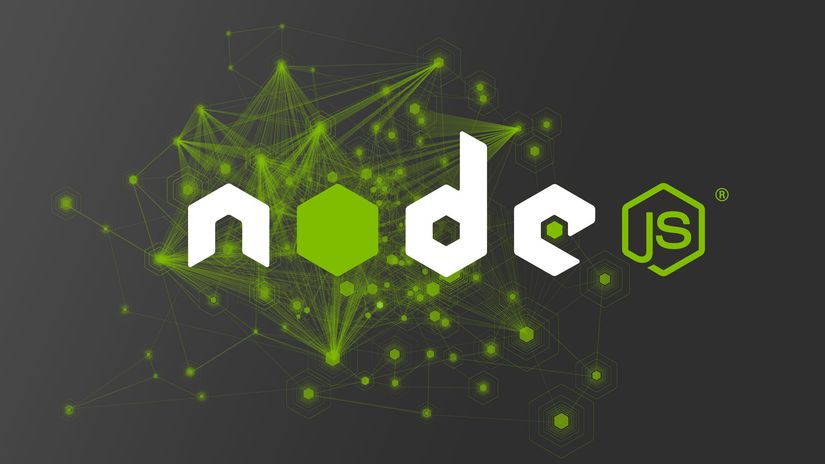
******

*Giao diện chi tiết sản phẩm*

******

*Giao diện quản lý*

***2.1.6. NodeJS***

**2.1.6.1. NodeJS là gì?**

- Nodejs là một nền tảng (Platform) phát triển độc lập được xây dựng ở trên Javascript Runtime của Chrome mà chúng ta có thể xây dựng được các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.

- Nodejs được xây dựng và phát triển từ năm 2009, bảo trợ bởi công ty Joyent, trụ sở tại California, Hoa Kỳ.

- Phần Core bên dưới của Nodejs được viết hầu hết bằng C++ nên cho tốc độ xử lý và hiệu năng khá cao.

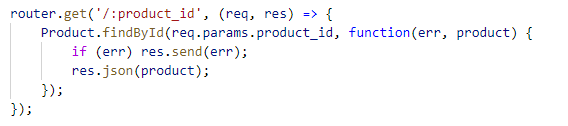
- Nodejs tạo ra được các ứng dụng có tốc độ xử lý nhanh, realtime thời gian thực.

- Nodejs áp dụng cho các sản phẩm có lượng truy cập lớn, cần mở rộng nhanh, cần đổi mới công nghệ, hoặc tạo ra các dự án Startup nhanh nhất có thể.

**2.1.6.2. Vai trò của NodeJS trong dự án**

- Node.js được xem là linh hồn của hệ thống khi đảm nhận vai trò back-end, cung cấp các module để chúng ta có thể dễ dàng tương tác với các thành phần khác như database và giao diện.

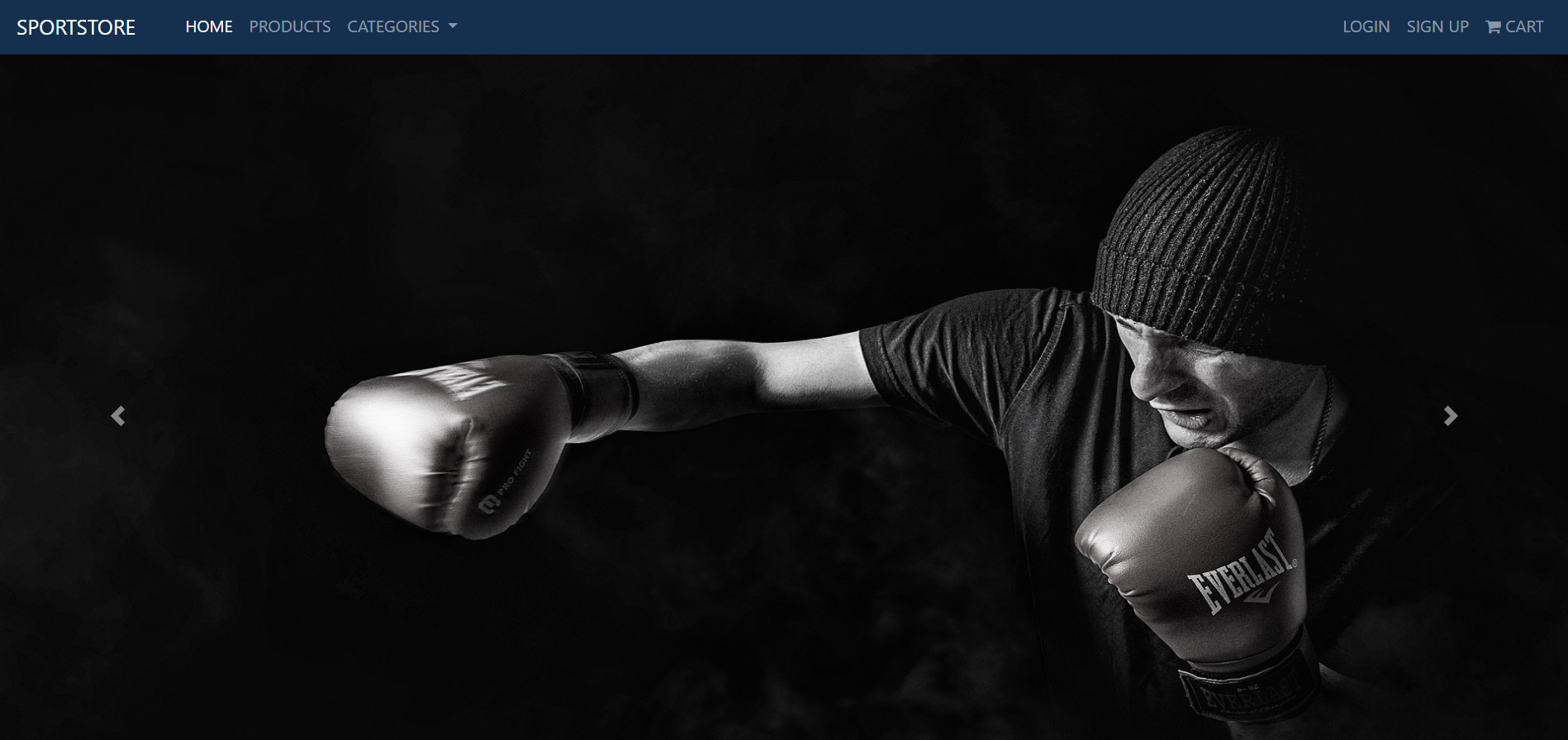
Ví dụ về việc tương tác giữa ExpressJS với AngularJS sử dụng NodeJS:



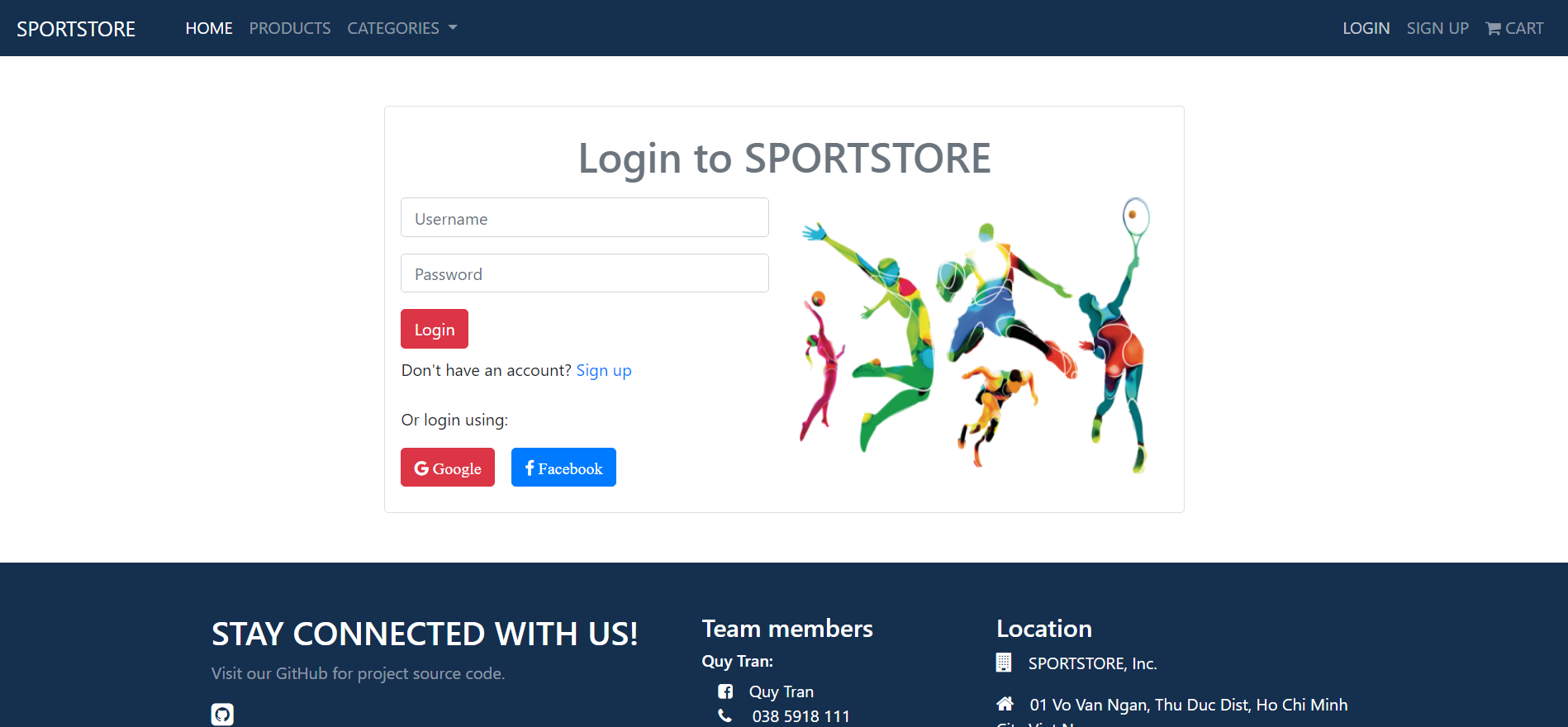
API trên được xây dựng để lấy ra một sản phẩm cụ thể từ database, sau khi thiết lập đường dẫn, Angular truyền tham số product\_id thông qua URL, NodeJS sử dụng req.params.product\_id để lấy tham số đó và thực hiện phương thức findById được hỗ trợ bởi ExpressJS để tìm ra sản phẩm có id trùng với tham số product\_id vừa nhận được. Nếu có lỗi trong quá trình gọi API, chi tiết lỗi sẽ được trả về thông qua res.send(err). Nếu thành công, res.json(product) sẽ trả về cho chúng ta thông tin sản phẩm vừa lấy được dưới dạng JSON, việc cuối cùng của AngularJS là hiển thị thông tin đó lên giao diện.

## **2.2. Thiết kế giao diện**

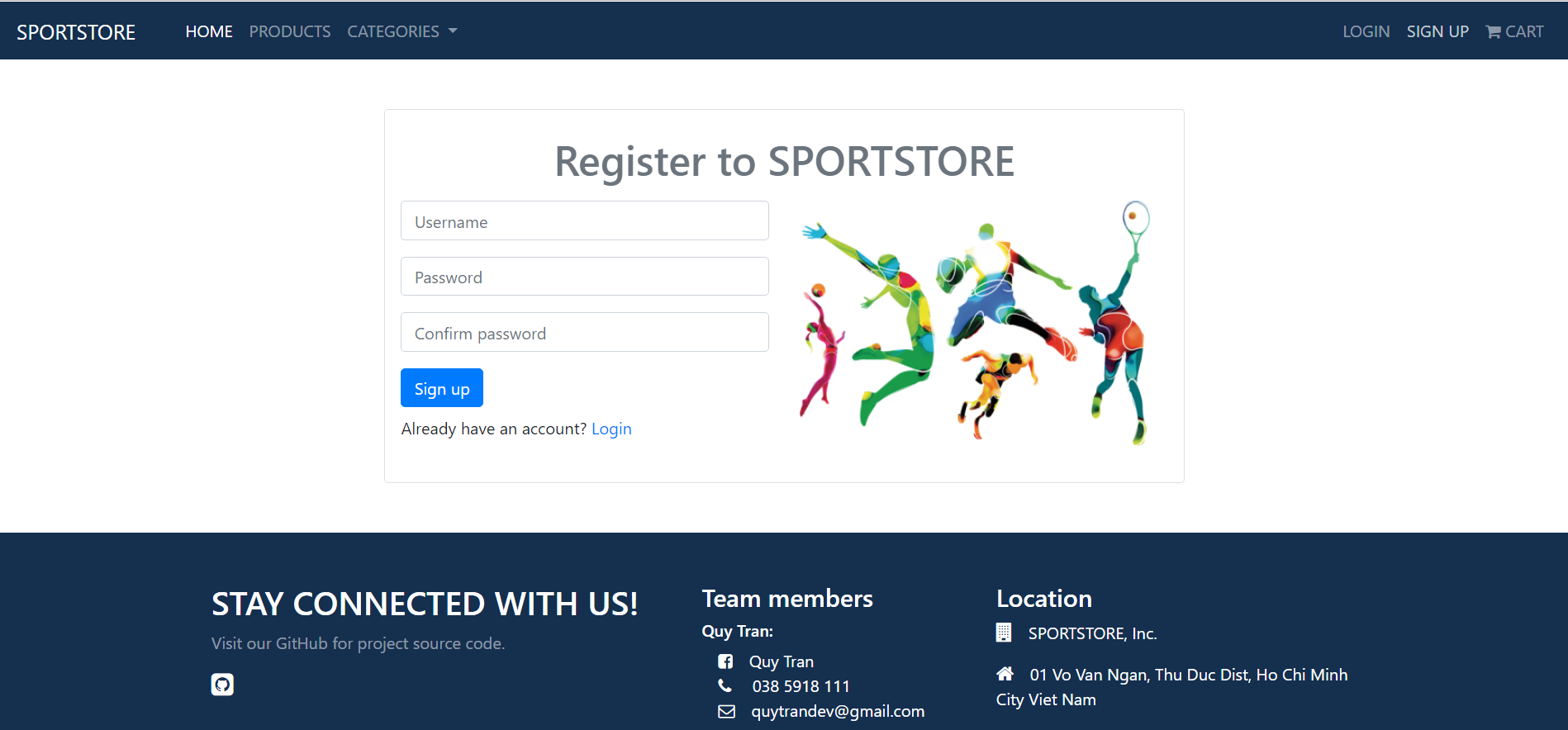
***2.2.1. Giao diện trang chủ***



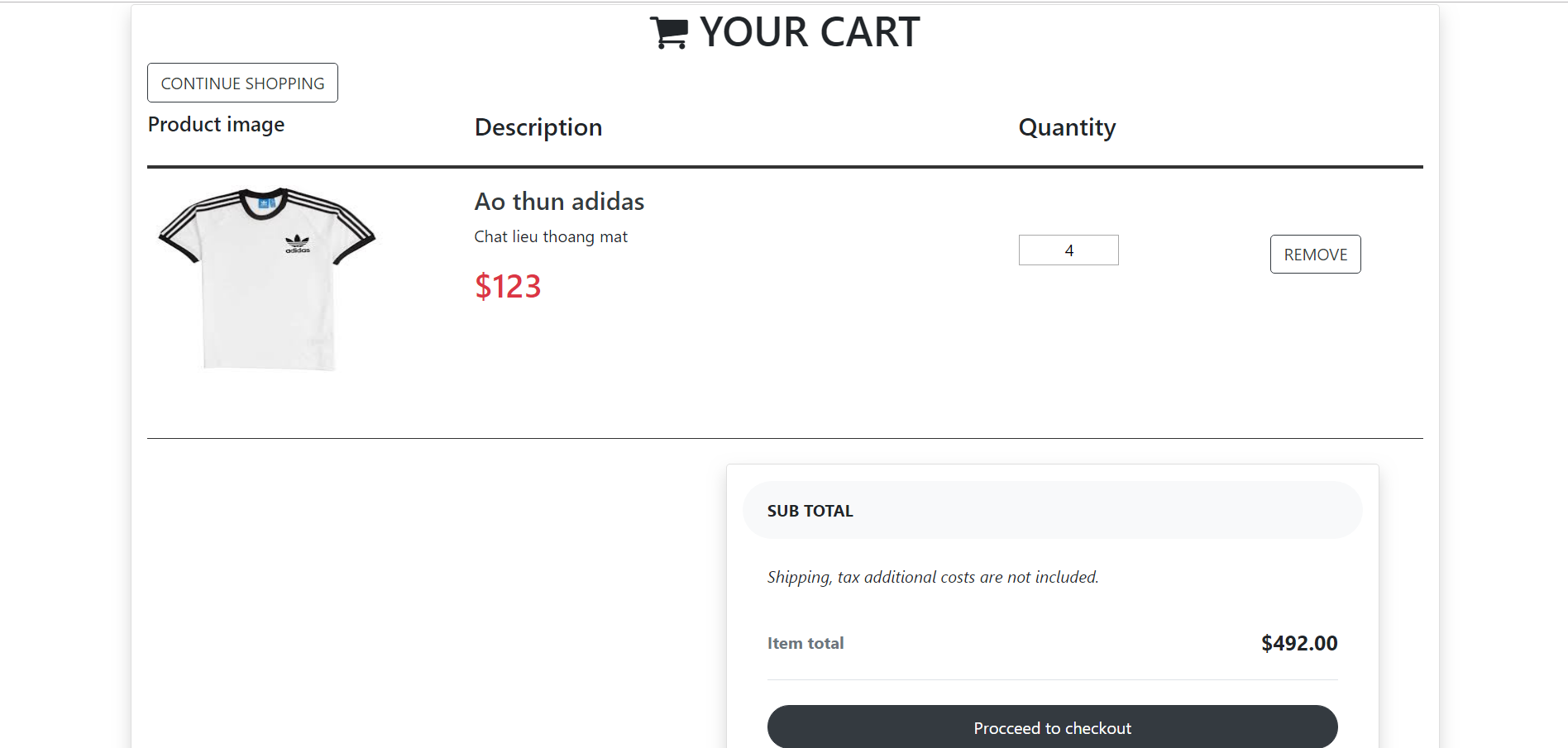
***2.2.2. Giao diện đăng nhập***



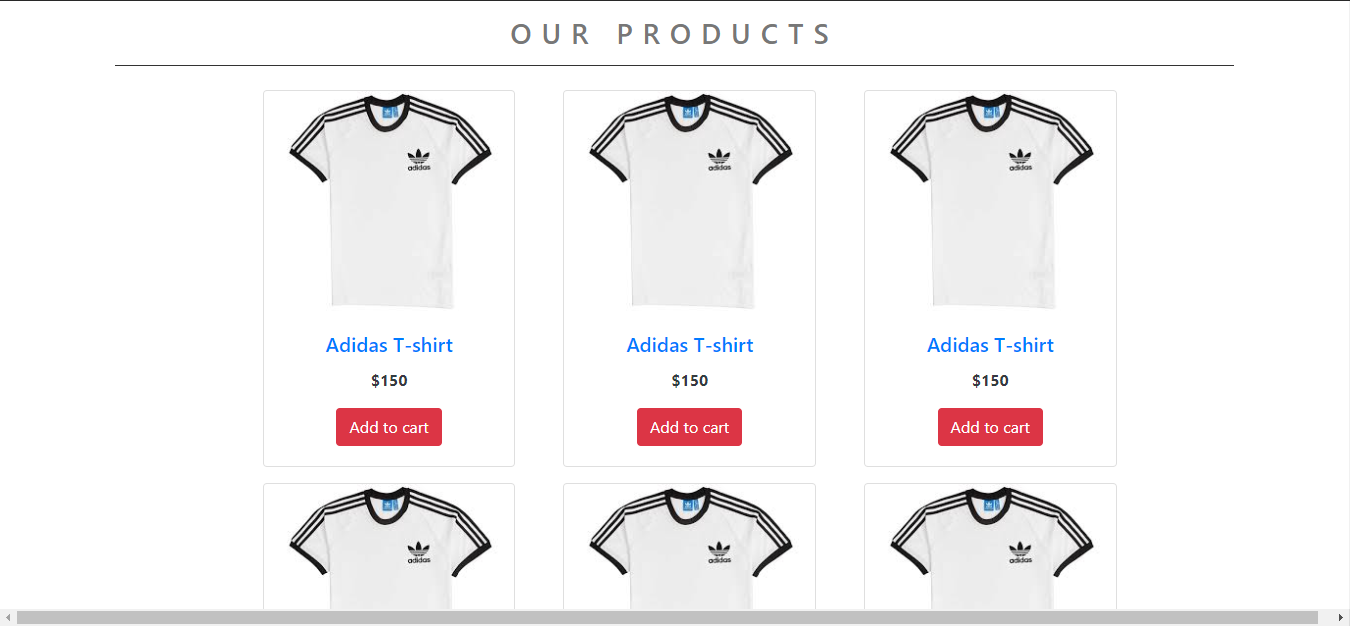
***2.2.3. Giao diện đăng ký***



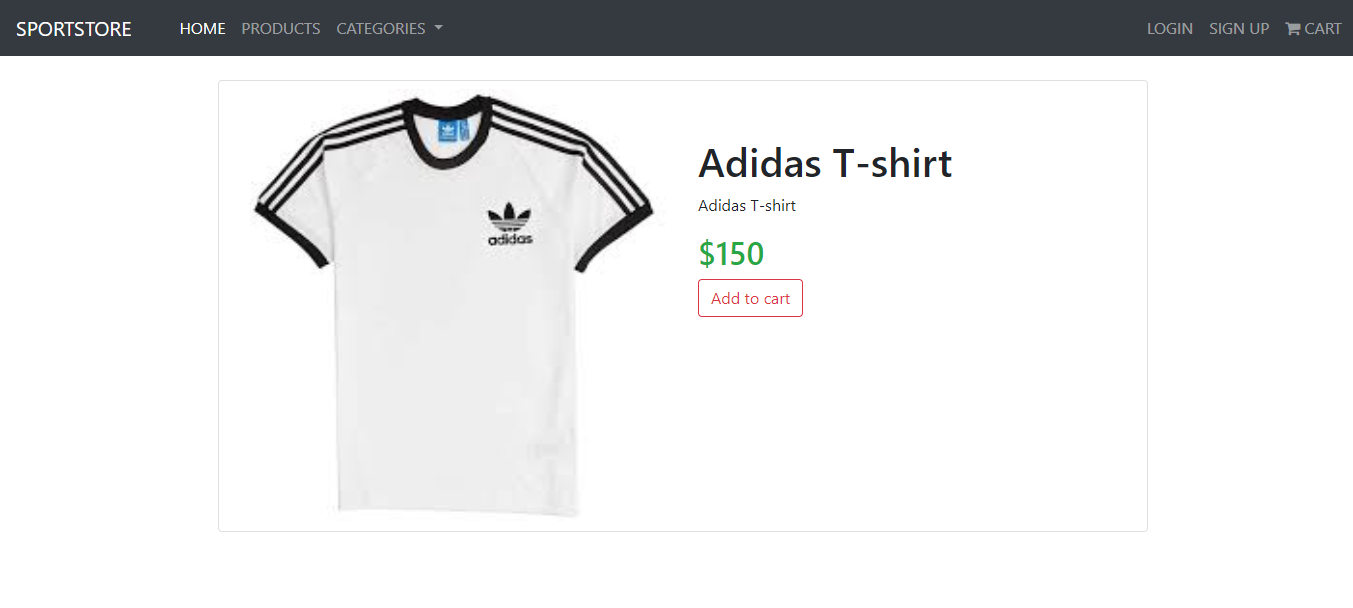
***2.2.4. Giao diện giỏ hàng***



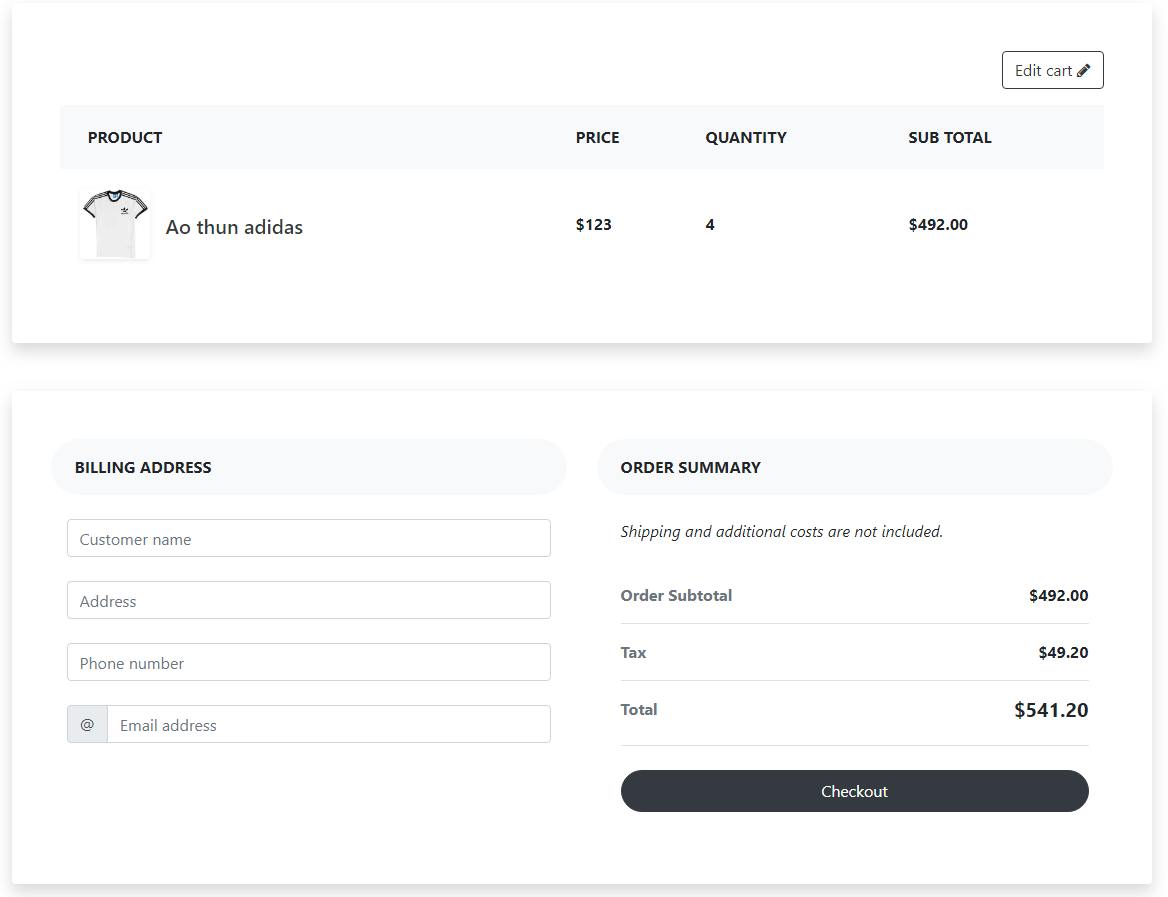
***2.2.5. Giao diện sản phẩm***

******

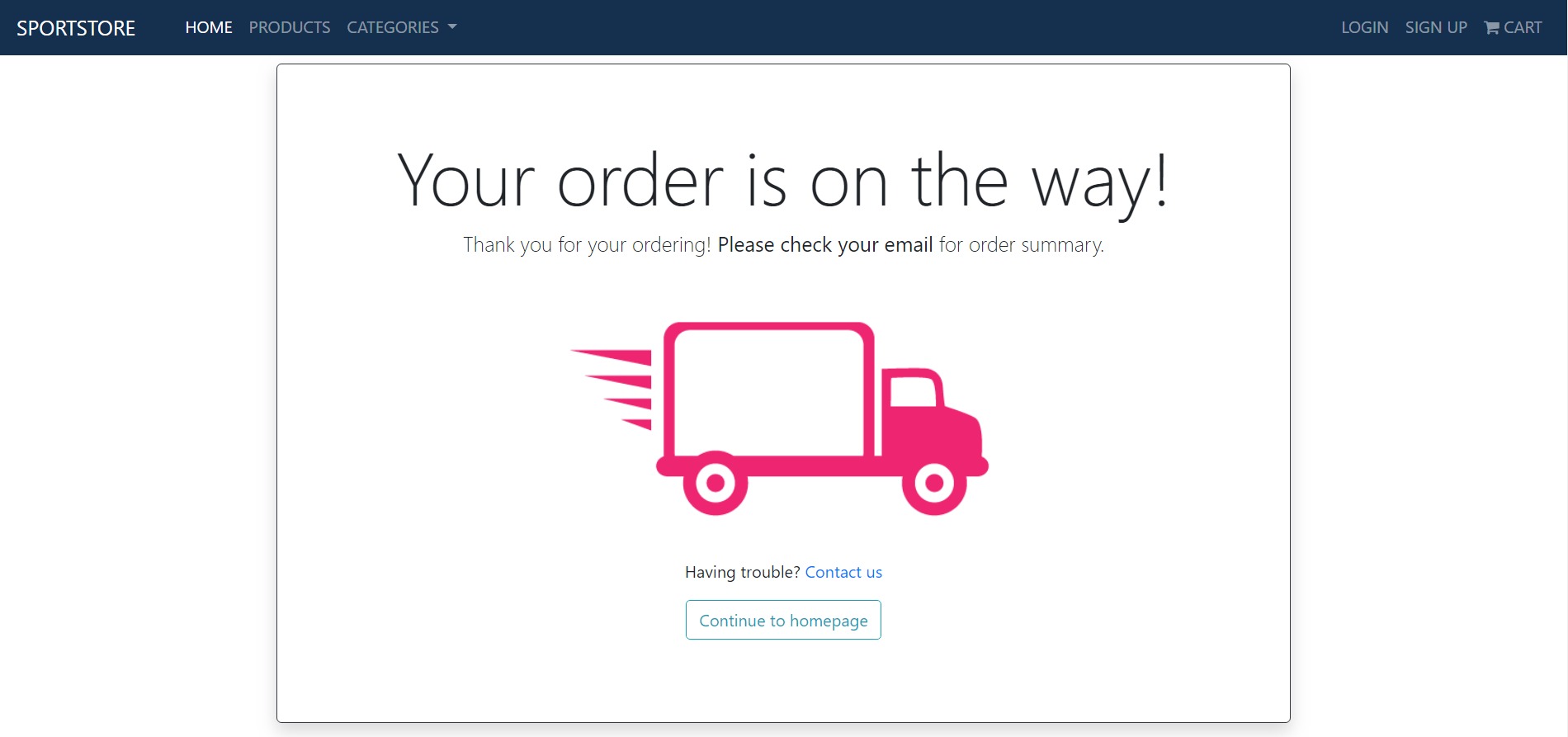
***2.2.6. Giao diện chi tiết sản phẩm***

******

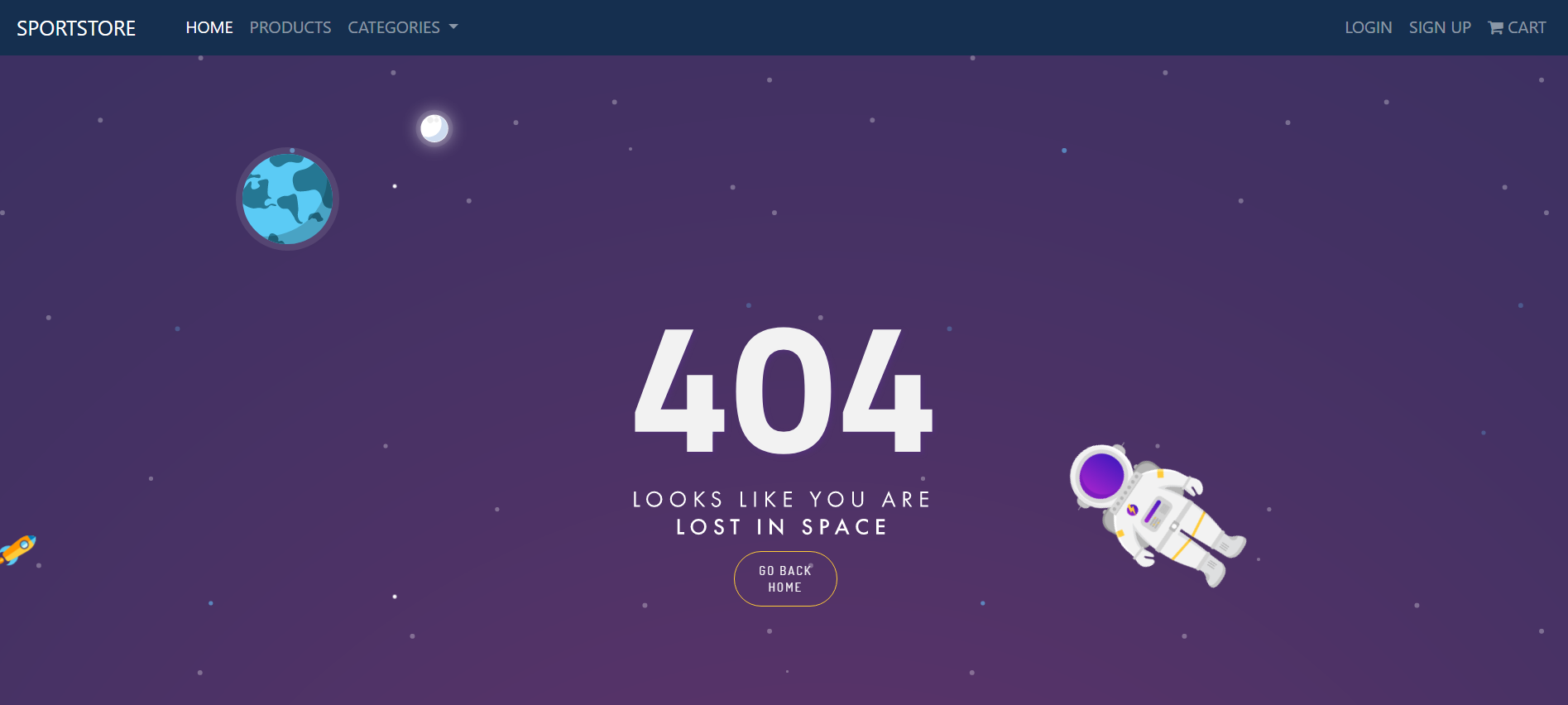
***2.2.7 Giao diện thanh toán***



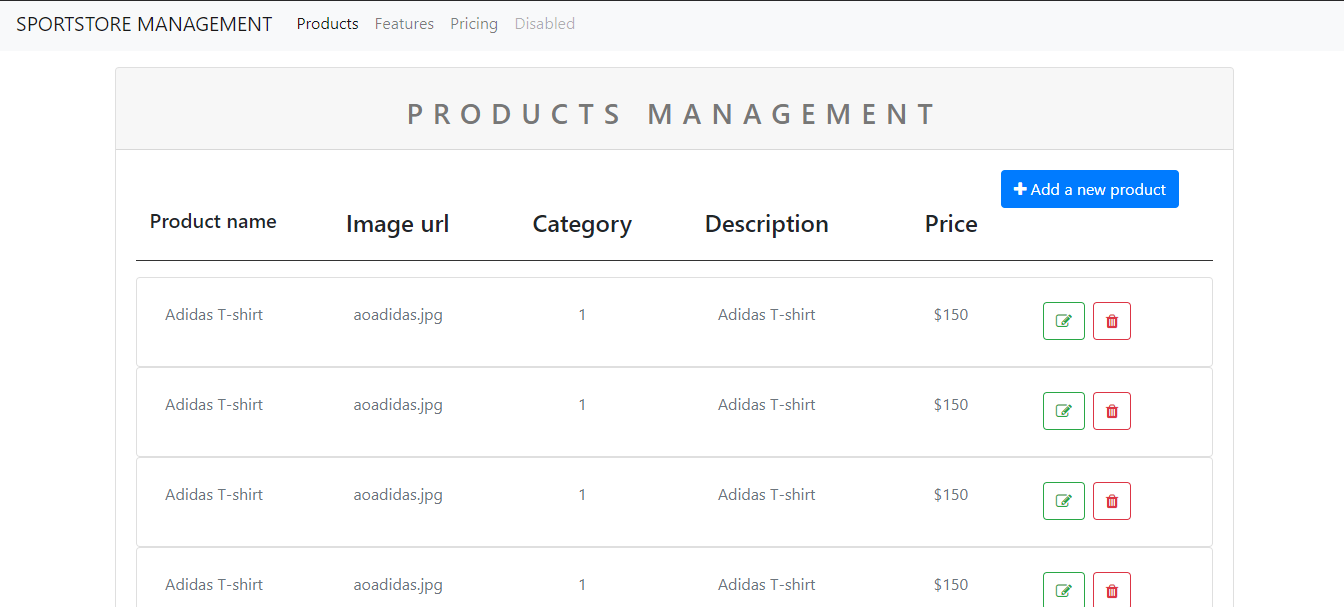
***2.2.8. Giao diện tri ân khách hàng***



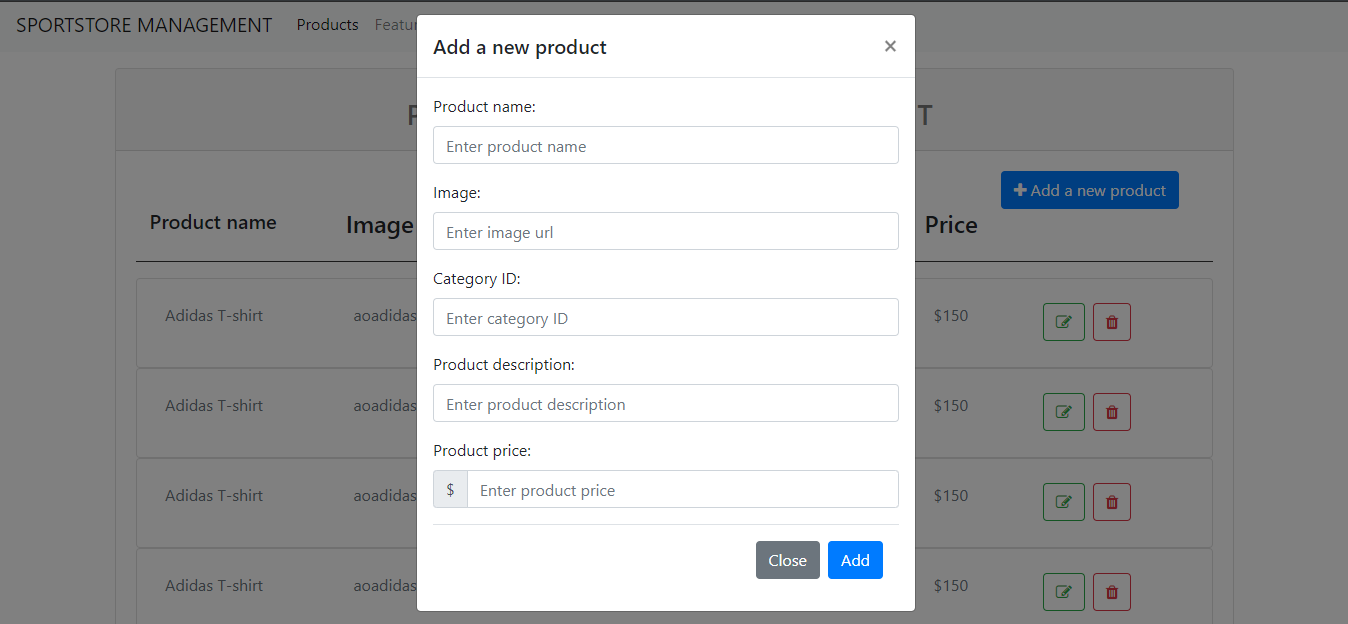
***2.2.9. Giao diện trang 404***



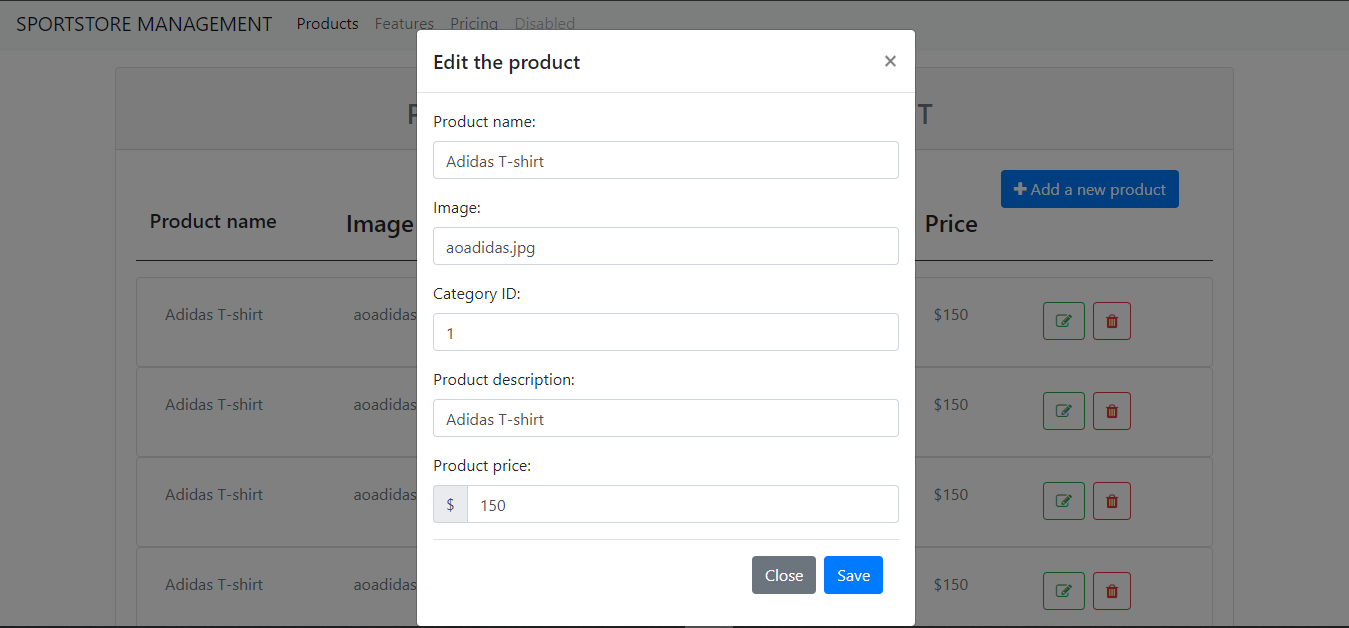
***2.2.10. Giao diện quản lý sản phẩm***

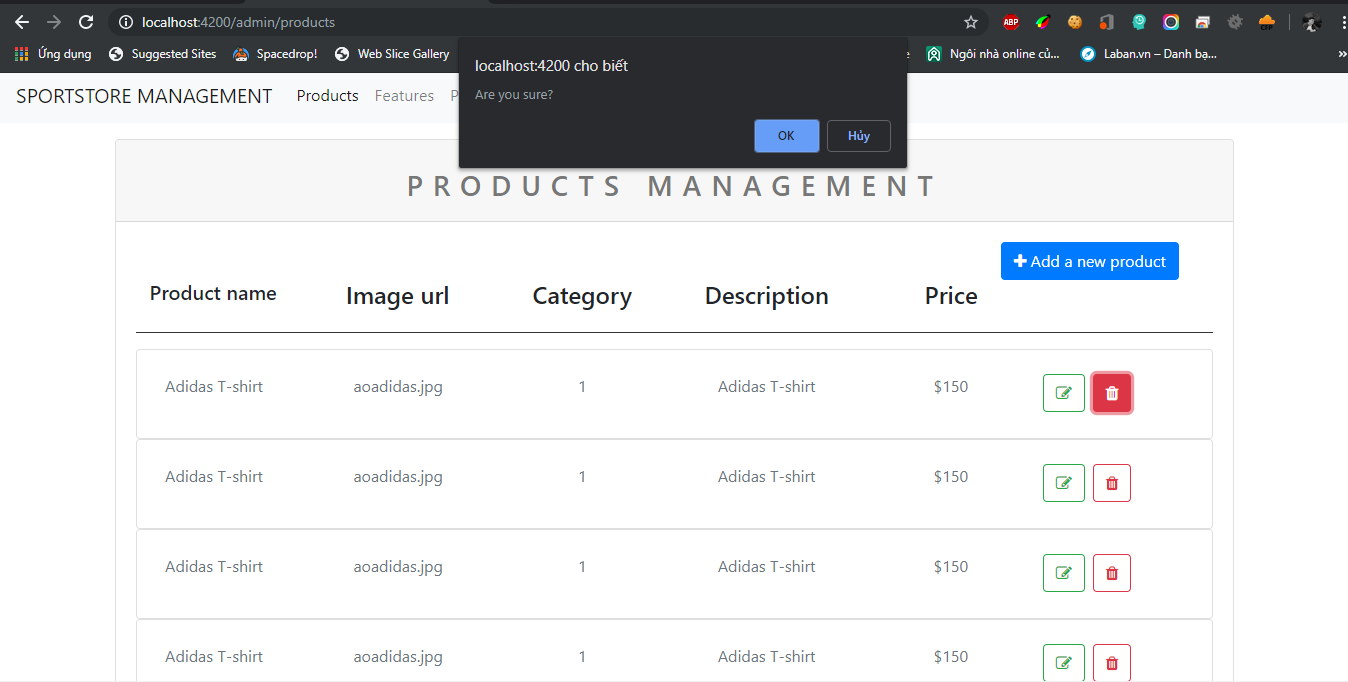
******

**2.2.10.1. Giao diện thêm sản phẩm**

****

**2.2.10.2. Giao diện sửa sản phẩm**

****

**2.2.10.3. Giao diện xoá sản phẩm**

.

# ***3. Cài đặt đồ án***

## **3.1. Môi trường**

- Ngôn ngữ lập trình: JavaScript (NodeJS).

- Thiết kế giao diện: HTML5, CSS, Bootstrap 4.

- Integrated Development Environment: Visual Studio Code.

- Database: MongoDB.

## **3.2. Phân công công việc**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Công việc** | **Trần Tấn Quý** | **Phạm Hoàng Việt** |
| 1 | Viết các Rest API (product, auth, user, category) |  | **x** |
| 2 | Chức năng đăng ký tài khoản Local |  | **x** |
| 3 | Chức năng đăng nhập (Facebook, Google, Local) |  | **x** |
| 4 | Chức năng gọi API và load sản phẩm lên trang chủ |  | **x** |
| **5** | Thiết kế các giao diện cho website sử dụng AngularJS | **x** |  |
| **6** | Chức năng quản lý sản phẩm (thêm, xoá, sửa) | **x** |  |
| **7** | Chức năng giỏ hàng | **x** |  |
| **8** | Chức năng gọi API và load lên trang chi tiết sản phẩm | **x** |  |
| **9** | Chức năng thanh toán/đặt hàng | **x** |  |
| **10** | Chức năng gửi email xác nhận đến người dùng | **x** |  |
| **11** | Viết báo cáo | **x** | **x** |

## **3.3. Đánh giá thành viên**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí đánh giá** | **Trần Tấn Quý** | **Phạm Hoàng Việt** |
| Tham gia họp nhóm đầy đủ | 100% | 100% |
| Có đóng góp ý kiến mang tính cải thiện cho đồ án | 100% | 100% |
| Hoàn thành tốt công việc được giao | 100% | 100% |
| Nghiêm túc trong quá trình thực hiện đồ án | 100% | 100% |
| Tích cực và chủ động trong công việc | 100% | 100% |
| Công việc được hoàn thành có chất lượng cao | 100% | 100% |
| **Tổng** | 100% | 100% |

# ***4. Tổng kết***

## **4.1. Kết quả**

Với công nghệ MEAN, nhóm đã xây dựng được những chắc năng cơ bản của một trang web thương mại điện tử cho người dùng và người quản trị sử dụng như:

- Đăng ký tài khoản.

- Đăng nhập vào hệ thống.

- Đăng xuất khỏi hệ thống.

- Xem sản phẩm và thông tin của sản phẩm.

- Quản lý giỏ hàng: thêm mặt hàng, sửa số lượng và xóa mặt hàng.

- Đặt hàng.

- Quản lý các sản phẩm của cửa hàng.

- Gửi email xác nhận cho khách hàng.

## **4.2. Ưu điểm**

- Giao diện thân thiện với người dùng.

- Trang web được xây dựng trên công nghệ MEAN Stack do đó dễ dàng triển khai hơn bất kỳ công nghệ nào khác.

- Hỗ trợ người dùng đăng nhập bằng nhiều tài khoản mạng xã hội khác nhau.

- Đã hỗ trợ được những chức năng cơ bản của một trang web thương mại điện tử.

## **4.3. Nhược điểm**

- Code chưa được tối ưu và độ bảo mật vẫn chưa cao.

- Các chức năng chưa được hoàn chỉnh đến mức tối đa để mang lại trải nghiệm người dùng tốt nhất.

- Chưa tích hợp các phương thức thanh toán từ những ngân hàng thực tế.

- Giao diện chưa hoàn toàn responsive.

## **4.4. Hướng phát triển**

- Cải thiện chức năng nhằm tăng trải nghiệm người dùng cao hơn.

- Áp dụng recommendation system vào hệ thống.

- Hoàn chỉnh giao diện trang web dưới dạng responsive nhằm hỗ trợ đa kích cỡ giao diện.

- Tối ưu lại code để tăng hiệu suất và tăng cường bảo mật cho hệ thống.

## **4.5. Source code**

Link Github: <https://github.com/hoangviet290398/CMPMM>

# **TÀI LIỆU THAM KHẢO**

[1] Angular, Angular Document, <https://angular.io/docs>, truy cập vào ngày 1 tháng 11 năm 2019.

[2] PositronX, Angular 7/8/9 MEAN Stack tutorial – CRUD App with Bootstrap, <https://www.positronx.io/mean-stack-tutorial-angular-7-crud-bootstrap/>, truy cập ngày 15 tháng 11 năm 2019.

[3] LearningProgramming, Building shopping cart in Angular 6, <http://learningprogramming.net/mean-stack/angular-6/build-shopping-cart-in-angular-6/>, truy cập ngày 20 tháng 11 năm 2019.

[4] Khoa Phạm, Lập trình Angular 4,   
<https://www.youtube.com/playlist?list=PLzrVYRai0riTA1m7Dasg8eraBr6R9nFgC>, truy cập ngày 1 tháng 11 năm 2019.