

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ 1, NĂM HỌC 2023-2024

ĐỀ TÀI:

SỬ DỤNG DJANGO ĐỂ PHÁT TRIỂN ỨNG DỤNG
WEBSITE BÁN QUẦN ÁO THỂ THAO

Giảng viên hướng dẫn
ĐOÀN PHƯỚC MIỀN

Sinh viên thực hiện
TRẦN PHÚC VĨ
Mã Lớp DA20TTA
MSSV 110120084
Khóa 2020-2024

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ 1, NĂM HỌC 2023-2024**

ĐỀ TÀI:

**SỬ DỤNG DJANGO ĐỂ PHÁT TRIỂN ỨNG DỤNG
WEBSITE BÁN QUẦN ÁO THỂ THAO**

**Giảng viên hướng dẫn
ĐOÀN PHƯỚC MIỀN**

**Sinh viên thực hiện
TRẦN PHÚC VĨ
Mã Lớp DA20TTA
MSSV 110120084
Khóa 2020-2024**

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Trà Vinh, ngày tháng năm

Giáo viên hướng dẫn
(Ký tên và ghi rõ họ tên)

Đoàn Phước Miền

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

[illegible]

Trà Vinh, ngày tháng năm

Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Tôi xin chân thành cảm ơn quý thầy cô đã giúp đỡ tôi thực hiện đề tài này. Đặc biệt thầy **Đoàn Phước Miên** đã tận tình hướng dẫn, giúp đỡ, chỉ bảo tôi trong suốt thời gian thực hiện thực tập đồ án chuyên ngành.

Đồng thời tôi cũng xin trân trọng cảm ơn những tình cảm quý báu mà các thầy cô trong trường Đại Học Trà Vinh đã truyền đạt cho tôi, những kinh nghiệm, kỹ thuật và cách thức trong việc xây dựng đề tài này.

Tuy nhiên, do khả năng đọc hiểu cũng như vận dụng kiến thức vào bài làm chưa được chắc chắn nên tôi không thể phát huy hết những ý tưởng, khả năng hỗ trợ của ngôn ngữ và kỹ thuật lập trình vào đề tài. Trong quá trình xây dựng website không thể tránh khỏi những sai sót, mong quý thầy cô đóng góp và cảm thông.

Tôi xin chân thành cảm ơn.

***Trà Vinh**, ngày....tháng....năm....*

Sinh viên thực hiện

Trần Phúc Vĩ

MỤC LỤC

| | |
|---|----|
| MỞ ĐẦU | 1 |
| CHƯƠNG 1: TỔNG QUAN | 2 |
| 1.1 Đặt vấn đề | 2 |
| 1.2 Mục tiêu hướng, giải quyết và kế hoạch thực hiện..... | 2 |
| 1.2.1 Mục tiêu cần đạt được | 2 |
| 1.2.2 Hướng giải quyết và kế hoạch thực hiện | 3 |
| CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT | 4 |
| 2.1 Khái niệm Python | 4 |
| 2.1.1 Cách cài đặt Python | 5 |
| 2.2 Khái niệm về Django | 6 |
| 2.2.1 Tính chất của Django..... | 6 |
| 2.2.2 Các tính năng cơ bản | 8 |
| 2.2.3 Cài đặt Django | 8 |
| 2.2.4 Các thành phần chính có trong Django | 9 |
| 2.2.4.1. Views | 9 |
| 2.2.4.2. Templates | 10 |
| 2.2.4.3. Models | 12 |
| 2.2.4.4. Hệ thống admin | 16 |
| 2.3 Mô hình MVT..... | 20 |
| 2.3.1 Giới thiệu..... | 20 |
| 2.3.2 Sự hoạt động của mô hình MVT | 21 |
| 2.3.3 Ưu nhược điểm của mô hình | 21 |
| CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU | 22 |
| 3.1 Mô tả bài toán..... | 22 |
| 3.2 Mô tả bài toán..... | 24 |
| 3.2.1 Mô hình dữ liệu: | 24 |
| 3.2.2 Bảng thực thể:..... | 24 |
| 3.2.3 Mô hình Use Case tổng quát: | 27 |
| 3.2.4 Mô hình Use Case tác nhân khách hàng:..... | 27 |
| 3.2.5 Mô hình Use Case tác nhân quản trị:..... | 28 |
| 3.3 Xây dựng website | 28 |
| 3.3.1 Xây dựng dữ liệu bên trong Django | 28 |
| 3.3.2 Xây dựng chức năng xử lý..... | 31 |
| 3.3.2.1. Xây dựng chức năng đăng ký: | 31 |
| 3.3.2.2. Xây dựng chức năng đăng nhập: | 31 |
| 3.3.2.3. Xây dựng chức năng trang Dashboard | 32 |
| 3.3.2.4. Xây dựng chức năng trang Home:..... | 33 |
| 3.3.2.5. Xây dựng chức năng trang Search:..... | 33 |

| | |
|---|----|
| 3.3.2.6. Xây dựng chức năng trang Product_detail: | 34 |
| 3.3.2.7. Xây dựng chức năng trang Category: | 35 |
| 3.3.2.8. Xây dựng chức năng trang Cart:..... | 36 |
| 3.3.2.9. Xây dựng chức năng trang Checkout:..... | 37 |
| CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU | 39 |
| 4.1 Giao diện người dùng | 39 |
| 4.1.1 Giao diện trang chủ..... | 39 |
| 4.1.2 Giao diện trang đăng nhập | 39 |
| 4.1.3 Giao diện trang sản phẩm | 40 |
| 4.1.4 Giao diện trang tìm kiếm | 40 |
| 4.1.5 Giao diện chi tiết sản phẩm | 41 |
| 4.1.6 Giao diện trang giỏ hàng | 41 |
| 4.1.7 Giao diện trang thanh toán..... | 42 |
| 4.2 Giao diện quản trị | 43 |
| 4.2.1 Giao diện trang đăng nhập quản trị | 43 |
| 4.2.1 Giao diện trang chủ quản trị | 43 |
| CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN | 44 |
| 5.1 Kết luận | 44 |
| 5.1.1 Kết quả đạt được..... | 44 |
| 5.1.2 Hạn chế | 44 |
| 5.2 Hướng phát triển..... | 44 |
| DANH MỤC TÀI LIỆU THAM KHẢO | 45 |

DANH MỤC TỪ VIẾT TẮT

| Từ viết tắt | Giải thích |
|-------------|---------------------------|
| HTML | HyperText Markup Language |
| MVT | Model – View – Templates |
| MVC | Model – View – Controller |
| URL | Uniform Resource Locator |
| CSS | Cascading Style Sheets |
| JS | JavaScript |

DANH MỤC HÌNH ẢNH

| | |
|---|----|
| Hình 2. 1 Trang chủ Python | 5 |
| Hình 2. 2 Cài đặt Python trong Visual Studio Code..... | 5 |
| Hình 2. 3 Kiểm tra Python..... | 6 |
| Hình 2. 4 Kiểm tra Pip..... | 8 |
| Hình 2. 5 Cài đặt Django | 9 |
| Hình 2. 6 Ví dụ về Views | 9 |
| Hình 2. 7 Ví dụ về Urls | 10 |
| Hình 2. 8 Ví dụ về phần nội dung được hiển thị | 10 |
| Hình 2. 9 Hàm được urls gọi đến | 11 |
| Hình 2. 10 Urls cấu hình gọi đến View | 11 |
| Hình 2. 11 Giao diện hiển thị lên trang base.html | 11 |
| Hình 2. 12 Kết quả hiển thị | 12 |
| Hình 2. 13 Cấu trúc của Django | 12 |
| Hình 2. 14 Giao diện input | 13 |
| Hình 2. 15 Giao diện output | 13 |
| Hình 2. 16 Tạo trường data lưu dữ liệu | 13 |
| Hình 2. 17 Form nhập dữ liệu..... | 14 |
| Hình 2. 18 Xây dựng chức năng trong Views | 14 |
| Hình 2. 19 Cấu hình urls..... | 15 |
| Hình 2. 20 Lệnh tạo cơ sở dữ liệu | 15 |
| Hình 2. 21 Lệnh cấu hình, cập nhật các thay đổi..... | 15 |
| Hình 2. 22 Giao diện input | 16 |
| Hình 2. 23 Giao diện output | 16 |
| Hình 2. 24 Đăng nhập Admin..... | 17 |
| Hình 2. 25 Tạo Superuser..... | 17 |
| Hình 2. 26 Giao diện trang Admin | 18 |
| Hình 2. 27 Chỉnh sửa Users..... | 18 |
| Hình 2. 28 Phân quyền users | 19 |
| Hình 2. 29 Cấu hình để hiển thị lên Admin..... | 19 |
| Hình 2. 30 Giao diện sau khi đã cấu hình ở Admin | 19 |
| Hình 2. 31 Mô hình MVT..... | 20 |
| | |
| Hình 3. 1 Mô hình dữ liệu | 24 |
| Hình 3. 2 Mô hình Use Case tổng quát | 27 |
| Hình 3. 3 Mô hình Use Case tác nhân khách hàng..... | 27 |
| Hình 3. 4 Mô hình User Case tác nhân quản trị | 28 |
| Hình 3. 5 Bảng Category | 28 |
| Hình 3. 6 Bảng Product | 29 |
| Hình 3. 7 Bảng Order | 29 |
| Hình 3. 8 Bảng OrderProduct..... | 30 |
| Hình 3. 9 Bảng CustomerPurchase..... | 30 |
| Hình 3. 10 Bảng PurchaseItem | 30 |
| Hình 3. 11 Xây dựng form CreatorUserForm..... | 31 |
| Hình 3. 12 Hàm register xây dựng chức năng đăng ký | 31 |
| Hình 3. 13 Hàm loginuser xây dựng chức năng đăng nhập..... | 32 |
| Hình 3. 14 Hàm dashboard xây dựng chức năng trang Dashboard | 32 |
| Hình 3. 15 Trang hiển thị lên dashboard.html..... | 32 |
| Hình 3. 16 Hàm home xây dựng chức năng trang Home | 33 |
| Hình 3. 17 Hàm search xây dựng chức năng trang Search..... | 33 |

| | | |
|------------|--|----|
| Hình 3. 18 | Hiển thị lên trang search.html..... | 34 |
| Hình 3. 19 | Hàm product_detail xây dựng chức năng trang Product_detail | 34 |
| Hình 3. 20 | Hiển thị lên trang product_detail.html..... | 35 |
| Hình 3. 21 | Hàm category xây dựng chức năng trang Category | 35 |
| Hình 3. 22 | Hiển thị lên trang category.html | 35 |
| Hình 3. 23 | Hàm cart xây dựng chức năng trang Cart..... | 36 |
| Hình 3. 24 | Hiển thị lên trang cart.html..... | 36 |
| Hình 3. 25 | Tạo CheckoutForm..... | 37 |
| Hình 3. 26 | Hàm checkout xây dựng chức năng trang Checkout | 37 |
| Hình 3. 27 | Hàm checkout xây dựng chức năng trang Checkout tiếp theo | 38 |
| | | |
| Hình 4. 1 | Giao diện trang chủ..... | 39 |
| Hình 4. 2 | Giao diện trang đăng nhập..... | 40 |
| Hình 4. 3 | Giao diện trang sản phẩm | 40 |
| Hình 4. 4 | Giao diện trang tìm kiếm | 41 |
| Hình 4. 5 | Giao diện chi tiết sản phẩm | 41 |
| Hình 4. 6 | Giao diện trang giỏ hàng | 42 |
| Hình 4. 7 | Giao diện trang thanh toán..... | 42 |
| Hình 4. 8 | Giao diện đăng nhập quản trị..... | 43 |
| Hình 4. 9 | Giao diện trang chủ quản trị | 43 |

DANH MỤC BẢNG BIỂU

| | |
|------------------------------------|----|
| Bảng 1 So sánh MVT và MVC | 21 |
| Bảng 2 Bảng User..... | 24 |
| Bảng 3 Bảng Category..... | 24 |
| Bảng 4 Bảng Customer..... | 25 |
| Bảng 5 Bảng Order | 25 |
| Bảng 6 Bảng Product..... | 25 |
| Bảng 7 Bảng Purchase Item..... | 26 |
| Bảng 8 Bảng Order Product..... | 26 |
| Bảng 9 Bảng product_category | 26 |

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Xã hội ngày càng phát triển, con người ngày càng bận rộn với công việc, nhưng mặc dù bận rộn đến đâu trong mỗi người vẫn có niềm đam mê với thể thao, khi đó việc chọn các loại quần áo thể thao phù hợp cũng là một trong những vấn đề mà ai cũng phải đau đầu suy nghĩ. Vì vậy, một website quần áo thể thao sẽ giúp khách hàng tiết kiệm thời gian và công sức hơn là việc ra đến cửa hàng, chỉ cần có kết nối mạng, khách hàng có thể thoải mái lựa chọn những bộ quần áo thể thao mà mình yêu thích ở bất cứ đâu.

Để giải quyết vấn đề trên, trước hết ta cần xem xét về yêu cầu của khách hàng đối với một website

- Về mặt giao diện: giao diện tối ưu, dễ nhìn, phù hợp với mọi lứa tuổi.
- Về mặt nội dung: trình bày các sản phẩm phải dễ nhìn, phối màu phù hợp, tránh các màu quá sáng hoặc quá tối làm ảnh hưởng đến khách hàng.
- Về mặt sử dụng: mượt mà, tiện lợi, dễ sử dụng cho mọi lứa tuổi.

Nhờ đưa ra các giải pháp phù hợp để giải quyết vấn đề, tôi đã thiết kế được một giao diện website với giao diện đơn giản, dễ dàng tiếp cận và sử dụng với mọi người, nội dung gọn gàng, dễ nhìn không làm ảnh hưởng đến khả năng nhìn của khách hàng, các chức năng dễ dàng sử dụng.

MỞ ĐẦU

1. Lý do chọn đề tài

Trong thời đại công nghệ hiện nay, việc mua bán trên Internet đã trở nên phổ biến, trở thành một phần không thể thiếu của đời sống, việc nắm bắt tìm hiểu, phân tích nhu cầu tiêu dùng là một phần quan trọng, nó giúp người tiêu dùng dễ dàng chọn lựa sản phẩm phù hợp, giúp các nhà kinh doanh nắm bắt được thị hiếu các sản phẩm có trên thị trường mà khách hàng mong muốn và đáp ứng.

Do đó tôi chọn thực hiện đề tài “Sử dụng DJANGO để phát triển ứng dụng website bán quần áo thể thao” với mục đích cung cấp một website tiện ích giúp cho người dùng có thể mua bán hàng hóa một cách thuận tiện nhất.

2. Mục tiêu nghiên cứu

Nắm vững kiến thức cơ bản của Django, thành thạo một vài công nghệ hỗ trợ xây dựng giao diện như HTML, CSS, JS,...

Nghiên cứu thị trường, tìm hiểu thị hiếu của khách hàng trên thị trường về mua bán quần áo trên các website.

3. Đối tượng nghiên cứu

Các khách hàng bận rộn, không có nhiều thời gian ra ngoài, có nhu cầu cần và mua quần áo trực tuyến trên các website.

Khảo sát thị trường, tìm hiểu thông tin trên các web bán quần áo khác, tìm hiểu trên các diễn đàn bán hàng trực tuyến.

4. Phạm vi nghiên cứu

Nhu cầu thị hiếu của khách hàng trên thị trường về việc mua hàng trực tuyến.

Các công nghệ liên quan đến Django

Các tính năng và chức năng cơ bản của website bán quần áo thể thao, như trang chủ, danh mục sản phẩm, chi tiết sản phẩm, giỏ hàng, đăng nhập, đăng ký, quản lý tài khoản, quản lý đơn hàng, quản lý sản phẩm, quản lý danh mục, etc.

CHƯƠNG 1: TỔNG QUAN

1.1 Đặt vấn đề

Hiện nay, sự phát triển của công nghệ tiên tiến ngày càng mạnh mẽ và ứng dụng của nó ngày càng rộng rãi hơn. Trước đó khi chưa đạt đến trình độ công nghệ như bây giờ, người dân thật sự phải tốn rất nhiều thời gian cũng như công sức để có thể đi đến các nơi mua sắm. Tầm quan trọng của công nghệ hiện nay là không thể thay thế được.

Trong cuộc sống ngày càng bận rộn như hiện nay, để giảm bớt các bất cập trong việc phải đến các cửa hàng, các shop quần áo thể thao để mua quần áo mà mình yêu thích, các website bán quần áo thể thao ngày càng nhiều và trở nên phổ biến, nắm bắt cơ hội này việc xây dựng một website bán quần áo thể thao trực tuyến sẽ giúp khách hàng giảm bớt thời gian và công sức đến cửa hàng mua hàng.

Để thiết kế website bán quần áo thể thao ta cần có những công nghệ cũng như các phần mềm để viết ra trang web. Tôi sử dụng Visual Studio Code với Django để thiết kế ra một website bán quần áo thể thao. Vì Django là một framework mạnh nên tôi kết hợp giữa HTML, CSS làm giao diện và Django xử lý các chức năng như: đăng ký, đăng nhập, thêm thông tin sản phẩm vào giao diện, thêm sản phẩm vào giỏ hàng, sửa, xóa sản phẩm,...

1.2 Mục tiêu hướng, giải quyết và kế hoạch thực hiện

1.2.1 Mục tiêu cần đạt được

Am hiểu Django framework căn bản.

Hoàn thiện trang website giúp khách hàng có thể tự do mua các mặt hàng mà mình yêu thích và phù hợp với mọi lứa tuổi, với những chức năng cơ bản như:

1. Hiện thị các sản phẩm theo từng danh mục.
2. Đăng nhập vào hệ thống: nhập thông tin tài khoản, mật khẩu để vào hệ thống. Phân quyền theo nhu cầu của người dùng.
3. Quản lý hàng hóa: thêm, xóa, sửa thông tin các loại hàng hóa và các mặt hàng, tìm kiếm các thông tin hàng hóa.
4. Cung cấp cho khách hàng những cập nhật mới nhất về sản phẩm và giá cả.
5. Giao diện tiện lợi, dễ dàng sử dụng.

6. Quản lý giỏ hàng: xóa, sửa số lượng các loại hàng hóa

1.2.2 Hướng giải quyết và kế hoạch thực hiện

Hướng giải quyết

- Tham khảo các diễn đàn các website về bán quần áo thể thao khác.
- Liên tục cập nhật dữ liệu mới, mặt hàng mới.
- Học hỏi và đổi mới website bằng cách tham khảo những website bán điện thoại hiện có trên thị trường.

- Thiết kế các giao diện của website.
- Cài đặt chương trình ứng dụng, nhập liệu, chạy thử và kiểm tra lỗi.
- Viết một bài báo cáo về công việc đã thực hiện theo mẫu quy định

Kế hoạch thực hiện

- Tìm hiểu về Django, các chức năng, lợi ích của việc sử dụng Django trong việc xây dựng một website bán hàng.

- Khảo sát yêu cầu của khách hàng.
- Tiến hành tìm hiểu, nghiên cứu cách hoạt động của một website bán hàng.
- Tham khảo một số trang web bán điện thoại hiện có trên thị trường như: Coolmate, Sporter.vn, ...

- Tiến hành các bước cơ bản, thiết kế giao diện.
- Làm các chức năng cho trang web.
- Chạy thử và kiểm lỗi.
- Hoàn thiện trang web bán quần áo thể thao.

Môi trường cài đặt:

Để thực hiện được yêu cầu phải cần có một ứng dụng lập trình đáp ứng được các chức năng thiết kế, cùng với các thư viện cơ bản hỗ trợ.

- Ứng dụng: Visual Studio Code.
- Sử dụng HTML, CSS, JavaScript, thư viện: Django framework.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 Khái niệm Python

Python là một ngôn ngữ lập trình mạnh mẽ, linh hoạt và dễ học. Guido van Rossum phát triển Python vào những năm 1990, và từ đó, ngôn ngữ này đã trở thành một trong những ngôn ngữ phổ biến nhất trên toàn cầu.

Tính linh hoạt và đa dụng là những điểm mạnh của Python. Nó có khả năng thực hiện nhiều loại công việc khác nhau, từ viết các script đơn giản cho đến xây dựng các ứng dụng web phức tạp.

Cú pháp của Python rất dễ đọc và gần gũi với ngôn ngữ tự nhiên. Điều này làm cho việc viết và đọc mã trở nên dễ dàng hơn, giúp tăng tốc quá trình phát triển và hiểu mã nguồn của người khác.

Cộng đồng Python rất lớn, đa dạng và sôi động. Cộng đồng này không chỉ đóng góp các thư viện và framework hữu ích mà còn cung cấp sự hỗ trợ cho nhau, từ người mới học đến những chuyên gia hàng đầu.

Một số điểm nổi bật của Python:

- Python có cú pháp rõ ràng và gần gũi với ngôn ngữ tự nhiên, giúp người dùng dễ dàng tiếp cận và hiểu code.
- Nó được sử dụng cho nhiều loại dự án, từ phát triển web, xử lý dữ liệu, machine learning cho đến hacking và tự động hóa công việc.
- Python có một cộng đồng lớn và đa dạng, cung cấp tài liệu phong phú, thư viện và sự hỗ trợ từ cộng đồng người dùng.
- Nó cung cấp một loạt các công cụ và frameworks mạnh mẽ giúp giải quyết nhiều vấn đề khác nhau.
- Python có khả năng di động và có thể chạy trên nhiều hệ điều hành và nền tảng khác nhau.

2.1.1 Cách cài đặt Python

Đầu tiên, truy cập vào địa chỉ <https://www.python.org/downloads/> và tải bản Python về máy



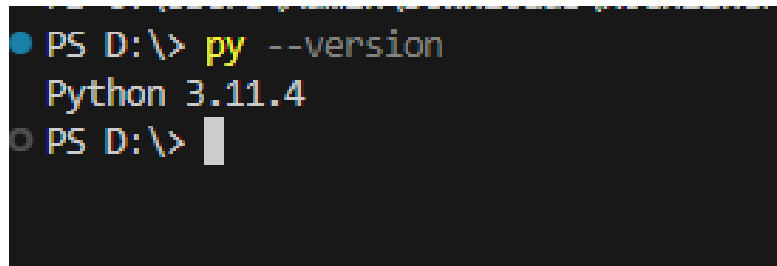
Hình 2. 1 Trang chủ Python

Sau đó ta tiếp tục cài đặt Python vào Visual Studio Code để tiến hành viết dự án Django, khởi động Visual Studio Code vào phần **Extensions** sau đó gõ vào Python và ấn **install**



Hình 2. 2 Cài đặt Python trong Visual Studio Code

Để kiểm tra xem Python đã được cài đặt thành công hay chưa, ta vào terminal của Visual Studio Code và gõ `py --version`



```
PS D:\> py --version
Python 3.11.4
PS D:\> 
```

Hình 2. 3 Kiểm tra Python

2.2 Khái niệm về Django

Django là framework giúp cho người dùng có thể sử dụng để phát triển các ứng dụng web một cách nhanh chóng và hiệu quả. Hầu hết các ứng dụng web có một số chức năng phổ biến, như xác thực, truy xuất dữ liệu từ cơ sở dữ liệu và quản lý cookie. Người dùng phải viết mã cho chức năng tương tự vào mọi ứng dụng web mà họ viết. Django giúp họ làm việc dễ dàng và nhanh chóng hơn bằng cách nhóm các chức năng khác nhau thành một tập hợp lớn các mô-đun có thể tái sử dụng, được gọi là một khung ứng dụng web. Người dùng sử dụng khung web Django để sắp xếp và viết mã của họ hiệu quả hơn và giảm đáng kể thời gian phát triển web.

Django tập trung vào hai khía cạnh quan trọng là tính "tái sử dụng" và "tự chạy", đồng thời cung cấp khả năng phát triển nhanh và tránh việc làm lại công việc đã có sẵn. Framework này được thiết kế với mục tiêu chuyển ý tưởng thành sản phẩm một cách nhanh nhất có thể. Bằng cách kết hợp hoàn hảo các tính năng này, Django cho phép chúng ta xây dựng các ứng dụng như website bán hàng hoặc hệ thống quản lý hàng hóa với độ chi tiết và độ chính xác cao một cách hiệu quả..

2.2.1 Tính chất của Django

Độ hoàn thành cao:

- Django cung cấp hầu hết mọi thứ mà các developer có thể muốn để phát triển web theo hướng mình muốn.

- Tất cả các phần trong framework hoạt động liên mạch với nhau, tuân theo một nguyên tắc thiết kế nhất quán và có tài liệu để bạn tham khảo. Nhờ đó, bạn có thể vừa phát triển web theo phong cách riêng, vừa tiết kiệm được thời gian

Linh hoạt:

- Django có thể sử dụng để xây dựng hầu hết mọi loại trang web- từ hệ thống quản lý nội dung (như wiki), cho đến các trang mạng xã hội, tin tức.
- Nó có thể hoạt động cùng với các framework bên ngoài và cũng có thể cung cấp nội dung ở hầu hết mọi định dạng (bao gồm HTML, RSS feeds, JSON, XML,...v.v..).
- Framework này còn cung cấp các tùy chọn khác nhau cho hầu hết các chức năng (như công cụ tạo template, cơ sở dữ liệu phổ biến,...).

Bảo mật:

- Django giúp developer tránh được nhiều lỗi bảo mật phổ biến bằng cách cung cấp framework có khả năng tự bảo vệ trang.
- Django còn bảo vệ website khỏi những lỗ hổng khỏi những loại tấn công mạng như: tấn công SQL injection, Cross-site Scripting, cross-site request forgery và clickjacking.

Khả năng mở rộng:

- Django sử dụng kiến trúc thành phần riêng nên bạn có thể mở rộng quy mô bằng cách thêm phần cứng vào các cấp độ (máy chủ bộ nhớ đệm, máy chủ cơ sở dữ liệu hoặc máy chủ ứng dụng).

Khả năng duy trì:

- Code của Django được viết bằng cách sử dụng các nguyên tắc và mẫu thiết kế khuyến khích việc tạo mã có thể bảo trì và tái sử dụng.
- Nó còn thúc đẩy việc nhóm các chức năng liên quan thành các “ứng dụng” có thể tái sử dụng, từ đó giúp website có khả năng duy trì cao hơn.

2.2.2 Các tính năng cơ bản

Dưới đây là một số tính năng cơ bản và thường sử dụng:

- **Admin Interface tự động:** Django cung cấp một giao diện quản trị tự động để quản lý dữ liệu của ứng dụng mà không cần phải viết code. Bằng cách định nghĩa các models, Django tạo ra một giao diện quản trị sẵn có để thêm, sửa, xóa dữ liệu.
- **URL Routing:** Django sử dụng **URLconf (URL configuration)** để ánh xạ các URL của ứng dụng đến các views tương ứng.
- **Templates:** Django cung cấp hệ thống template để xây dựng giao diện người dùng.
- **Authentication và Authorization:** Django cung cấp các công cụ mạnh mẽ để xác thực người dùng và kiểm soát quyền truy cập thông qua hệ thống user và groups.
- **ORM (Object-Relational Mapping):** Django cung cấp một ORM mạnh mẽ cho phép tương tác với cơ sở dữ liệu thông qua việc sử dụng Python thay vì việc viết các truy vấn SQL trực tiếp.

2.2.3 Cài đặt Django

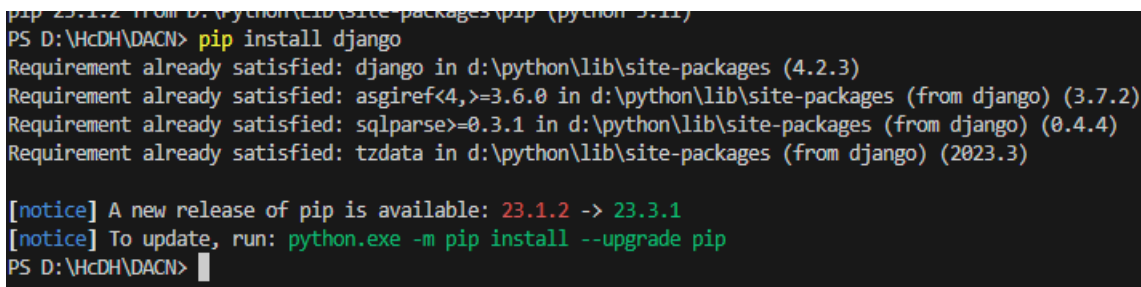
Django là Framework mạnh mẽ, giúp việc xây dựng website dễ dàng và tiện lợi hơn, để cài đặt Django, trước hết ta cần cài đặt pip để có thể chạy câu lệnh cài đặt Django. Để cài đặt pip ta vào <https://pip.pypa.io/> làm theo hướng dẫn và cài đặt pip

Để kiểm tra xem pip đã cài đặt thành công chưa, ta chạy lại pip --version

```
PS D:\HCDH\DACN> pip --version
pip 23.1.2 from D:\Python\Lib\site-packages\pip (python 3.11)
PS D:\HCDH\DACN>
```

Hình 2. 4 Kiểm tra Pip

Sau khi đã cài đặt pip thành công ta tiến hành cài đặt Django bằng câu lệnh “**pip install Django**”



```
PS D:\HCDH\DACN> pip install django
Requirement already satisfied: django in d:\python\lib\site-packages (4.2.3)
Requirement already satisfied: asgiref<4,>=3.6.0 in d:\python\lib\site-packages (from django) (3.7.2)
Requirement already satisfied: sqlparse>=0.3.1 in d:\python\lib\site-packages (from django) (0.4.4)
Requirement already satisfied: tzdata in d:\python\lib\site-packages (from django) (2023.3)

[notice] A new release of pip is available: 23.1.2 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS D:\HCDH\DACN>
```

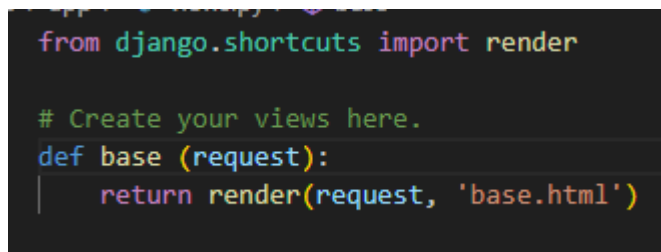
Hình 2. 5 Cài đặt Django

2.2.4 Các thành phần chính có trong Django

Trong phần trên, các thành phần của Django chỉ được nói sơ lược qua chứ không cụ thể từng phần, ở trong phần này, ta sẽ tìm hiểu kỹ hơn về các thành phần chính của Django.

2.2.4.1. Views

Là các hàm, hay các lớp xử lý logic của ứng dụng, nhận yêu cầu URL từ người dùng, khi người dùng vào các trang khác nhau, **views** sẽ trở đến các hàm tương ứng để phản hồi lại người dùng. Trong Django khi một trang web được tạo ra trong file **views.py** nó sẽ được gọi đến tùy thuộc vào URL mà ta đã chọn khi khai báo trong file **urls.py**, khái niệm này được gọi là URLConf, đây là một module Python của Django làm nhiệm vụ phân tích và dẫn truyền đến một hàm View nhất định, đây là một hàm cơ bản trong **Views**



```
from django.shortcuts import render

# Create your views here.
def base(request):
    return render(request, 'base.html')
```

Hình 2. 6 Ví dụ về Views

Cấu trúc của file **urls.py** gồm đoạn đầu để **import** các thư viện cần thiết, tiếp theo là phần **urlpatterns** là nơi chứa có liên kết dẫn đến các trang trong web. Cấu trúc của khai báo đường dẫn cơ bản gồm: path(‘đây là tên của địa chỉ khi hiển thị trên web’, đây

là nơi khai báo các **templates** đã được khai báo ở file **views.py** , name='đây là tên của liên kết').

```
1
2 from django.urls import path
3 from . import views
4
5 urlpatterns = [
6     path('', views.base, name="base"),
7 ]
8
```

Hình 2. 7 Ví dụ về Urls

2.2.4.2. Templates

Templates trong Django là các layout web sẵn có, giúp tiết kiệm thời gian phát triển trang web bằng cách chỉ cần thêm nội dung chính vào chúng. Trình duyệt chỉ hiểu và hiển thị code HTML, không hiểu code Python. Để sử dụng code Python trong trang web, Django cung cấp các thẻ template, được đánh dấu bắt đầu và kết thúc bằng cặp kí tự `{% %}` hoặc `{{ }}`. Các câu lệnh Python nằm trong cặp `{% %}`, trong khi các biến được đặt trong cặp `{{ }}`. Ngôn ngữ Template của Django được thiết kế để hỗ trợ những người đã có kinh nghiệm làm việc với HTML, vì vậy nếu bạn đã quen với HTML, việc làm quen với Template trong Django sẽ không quá khó khăn.

Ở đây trang **base.html** là phần giao diện tĩnh, vào trang **test_templates.html** là phần nội dung được hiển thị khi ta gọi đến.

```
myweb > app > templates > <> test_templates.html
1  {% extends "base.html" %}
2
3  {% block test_templates %}
4  Test templates
5  {% endblock test_templates %}
6
7
```

Hình 2. 8 Ví dụ về phần nội dung được hiển thị

Tại file **views.py** ta tạo một hàm để gọi đến trang, ở đây hàm có tên là **test**

```
myweb > app > views.py
from django.shortcuts import render

# Create your views here.
def base(request):
    return render(request, 'base.html')

def test(request):
    return render(request, 'test_templates.html')
```

Hình 2. 9 Hàm được urls gọi đến

Ta vào **urls.py** để cấu hình đường dẫn đến hàm test trong View

```
myweb > app > urls.py
1
2 from django.urls import path
3 from . import views
4
5 urlpatterns = [
6     path('', views.base, name="base"),
7     path('test/', views.test),
8 ]
9
```

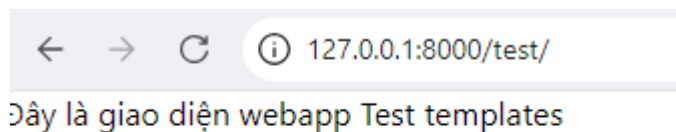
Hình 2. 10 Urls cấu hình gọi đến View

Để có thể hiển thị nội dung của trang **test_templates.html** lên trang **base.html** ta sử dụng cặp dấu ngoặc **{% %}** như sau:

```
myweb > app > templates > base.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7 </head>
8 <body>
9     Đây là giao diện webapp
10
11     {% block test_templates %}{% endblock test_templates %}
12 </body>
13 </html>
```

Hình 2. 11 Giao diện hiển thị lên trang base.html

Đây là kết quả, như ta thấy khi trở đến đường dẫn **test/** giao diện của web sẽ có thêm nội dung của trang **test_templates.html**



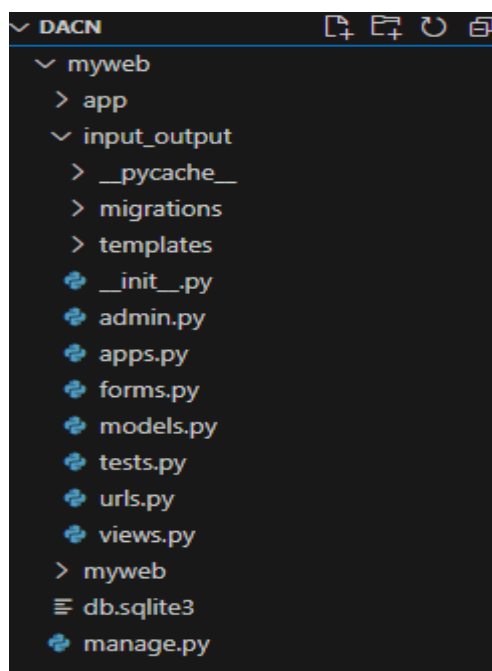
Hình 2. 12 Kết quả hiển thị

2.2.4.3. Models

Ở phần **Models** này sẽ có một ví dụ khác so với các ví dụ đơn giản ở trên, nhưng trước hết hãy tìm hiểu **Models** trong **Django** là gì:

Trong **Django**, **Models** được dùng để định nghĩa và cấu trúc nên cơ sở dữ liệu bằng cách sử dụng các lớp trong Python. Mỗi lớp tương ứng là một bảng trong cơ sở dữ liệu, còn các trường bên trong tương ứng với các cột trong cơ sở dữ liệu. Giúp cho cơ sở dữ liệu có thể linh hoạt hơn khi chỉ cần chạy lệnh sẽ tự tạo ra cơ sở dữ liệu.

Trong ví dụ này, ta sẽ tạo mới một webapp có tên là **input_output** và tạo ra thư mục **templates** chứa giao diện của webapp.



Hình 2. 13 Cấu trúc của Django

Trong thư mục **templates** tạo ra 2 file giao diện với tên gọi là **input.html** và **output.html**

Đây là phần code của 2 giao diện để có thể nắm rõ các tương tác cũng như các dữ liệu nhập vào đâu và xuất ra ở đâu.

```
myweb > input_output > templates > <> input_form.html
1  <form method="post">
2      {% csrf_token %}
3      {{ form.as_p }}
4      <button type="submit">Submit</button>
5  </form>
6
```

Hình 2. 14 Giao diện input

```
yweb > input_output > templates > <> output.html
1  <ul>
2      {% for item in data %}
3          <li>{{ item.data }}</li>
4      {% endfor %}
5  </ul>
```

Hình 2. 15 Giao diện output

Sau đó ta vào **Models** để định nghĩa một model dùng để lưu trữ dữ liệu. Tạo ra một trường **data** với kiểu dữ liệu Char và độ dài 100

```
nyweb > input_output > models.py
1  from django.db import models
2
3  # Create your models here.
4
5  class InputData(models.Model):
6      data = models.CharField(max_length=100)
7
```

Hình 2. 16 Tạo trường data lưu dữ liệu

Tạo một form để nhập dữ liệu trong **form.py**

```
web > input_output > forms.py
from django import forms
from .models import InputData

class InputDataForm(forms.ModelForm):
    class Meta:
        model = InputData
        fields = ['data']
```

Hình 2. 17 Form nhập dữ liệu

Tiếp theo ta cấu hình trong **Views**. Như chúng ta thấy, hàm **input_data** sẽ gọi đến phương thức **POST** để xác nhận dữ liệu nhập vào và lưu vào **form**, xét điều kiện nếu tồn tại dữ liệu sẽ gọi ra biến **save()** để lưu vào cơ sở dữ liệu và return về **output_data**. Ngược lại nếu request không phải là phương thức **POST** thì sẽ gọi ra **InputDataForm()** để có thể nhập giá trị vào. Còn ở hàm **output_data** sẽ lấy tất cả dữ liệu của hàm **InputData** và gọi đến trang **output.html** với **'data' = data** đã lấy ở trên.

```
web > input_output > views.py
1 from django.shortcuts import render, redirect
2 from .forms import InputDataForm
3 from .models import InputData
4
5 def input_data(request):
6     if request.method == 'POST':
7         form = InputDataForm(request.POST)
8         if form.is_valid():
9             form.save()
10            return redirect('output_data')
11     else:
12         form = InputDataForm()
13     return render(request, 'input_form.html', {'form': form})
14
15 def output_data(request):
16     data = InputData.objects.all()
17     return render(request, 'output.html', {'data': data})
18
```

Hình 2. 18 Xây dựng chức năng trong Views

Cấu hình đường dẫn trong **urls.py**

```
5 > input_output > urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('input/', views.input_data, name='input_data'),
    path('output/', views.output_data, name='output_data'),
]
```

Hình 2. 19 Cấu hình urls

Để có thể chạy được web có sử dụng cơ sở dữ liệu trong **Models** ta cần chạy lệnh **Python manage.py makemigrations** để Django tạo ra cơ sở dữ liệu.

```
PS D:\HcDH\DACN\myweb> python manage.py makemigrations
Migrations for 'input_output':
  input_output\migrations\0001_initial.py
    - Create model InputData
PS D:\HcDH\DACN\myweb> python manage.py migrate
```

Hình 2. 20 Lệnh tạo cơ sở dữ liệu

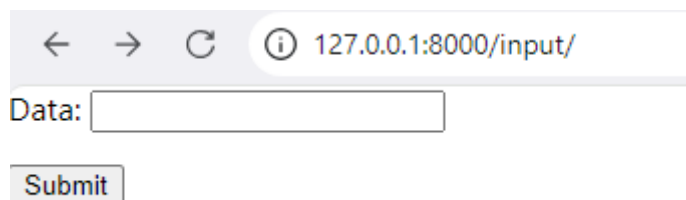
Tiếp theo đó, ta chạy lệnh **Python manage.py migrate** để cấu hình lại trang web cập nhật các thay đổi.

```
PS D:\HcDH\DACN\myweb> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, input_output, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
```

Hình 2. 21 Lệnh cấu hình, cập nhật các thay đổi

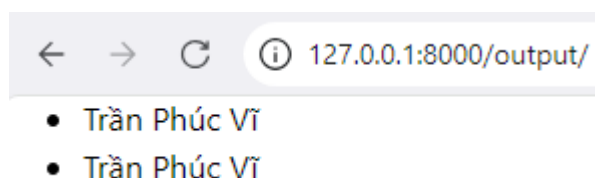
Sau khi đã cập nhật các thay đổi, ta chạy lệnh **Python manage.py runserver** để khởi động trang web, truy cập vào đường dẫn <http://127.0.0.1:8000/> để truy cập.

Đây là trang **input** khi nhập và ấn **Submit** dữ liệu sẽ được lưu vào cơ sở dữ liệu.



Hình 2. 22 Giao diện input

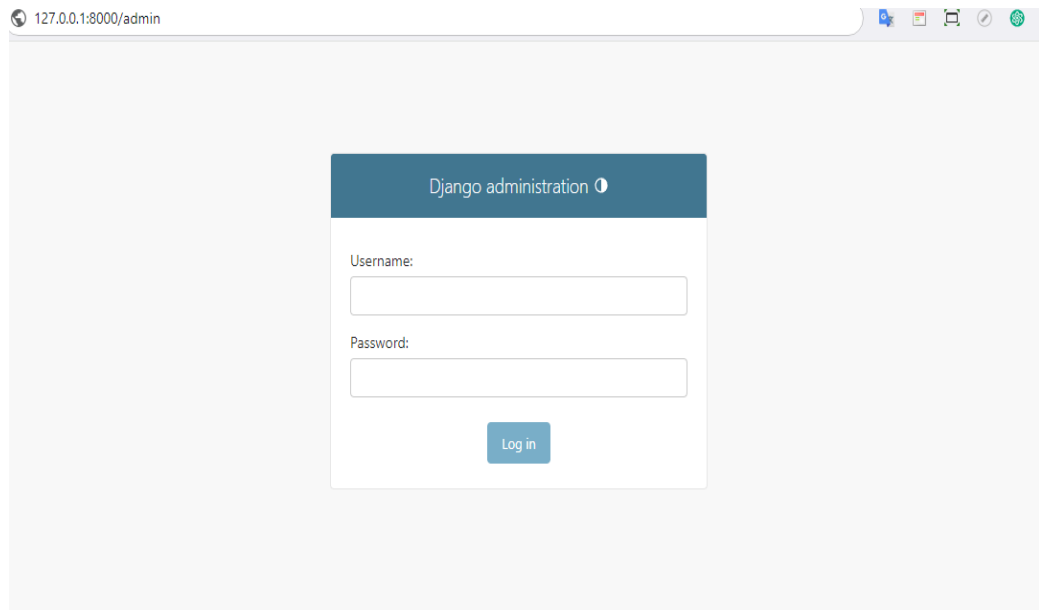
Đây là trang **output**, sau khi dữ liệu được nhập và ấn **Submit** sẽ tự động truy cập đến trang **output**, lấy tất cả dữ liệu và hiển thị lên thẻ ``.



Hình 2. 23 Giao diện output

2.2.4.4. Hệ thống admin

Khi xây dựng một ứng dụng web như blog, cửa hàng trực tuyến, hay diễn đàn, không chỉ cần các trang hiển thị thông tin mà còn cần một trang admin để quản lý nội dung. Django cung cấp sẵn một trang quản trị mạnh mẽ, giúp người dùng quản lý nội dung dễ dàng bằng cách truy cập vào đường dẫn `"/admin"`. Trang admin cung cấp nhiều tính năng quan trọng như thêm, sửa, xóa bài viết và cấu hình cho trang web. Điều này giúp người phát triển tiết kiệm thời gian và công sức trong việc quản lý nội dung của ứng dụng, để có thể truy cập vào trang admin, ta truy cập vào đường dẫn <http://127.0.0.1:8000/admin>.



Hình 2. 24 Đăng nhập Admin

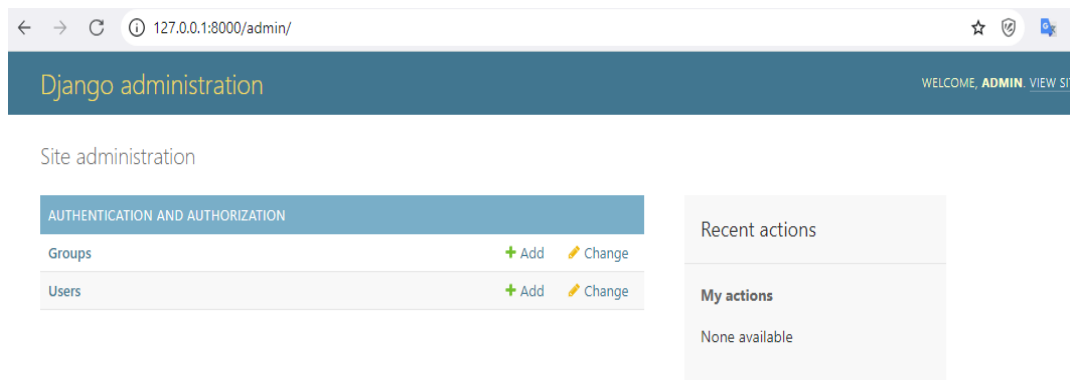
Nhưng để truy cập vào được bên trong của trang admin, ta cần phải có tài khoản admin. Để có thể tạo được tài khoản admin, ta sử dụng câu lệnh

Python manage.py createsuperuser

```
PS D:\HCDH\DACN\myweb> python manage.py createsuperuser
Username (leave blank to use 'admin'): admin
Email address: admin@gmail.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
PS D:\HCDH\DACN\myweb>
```

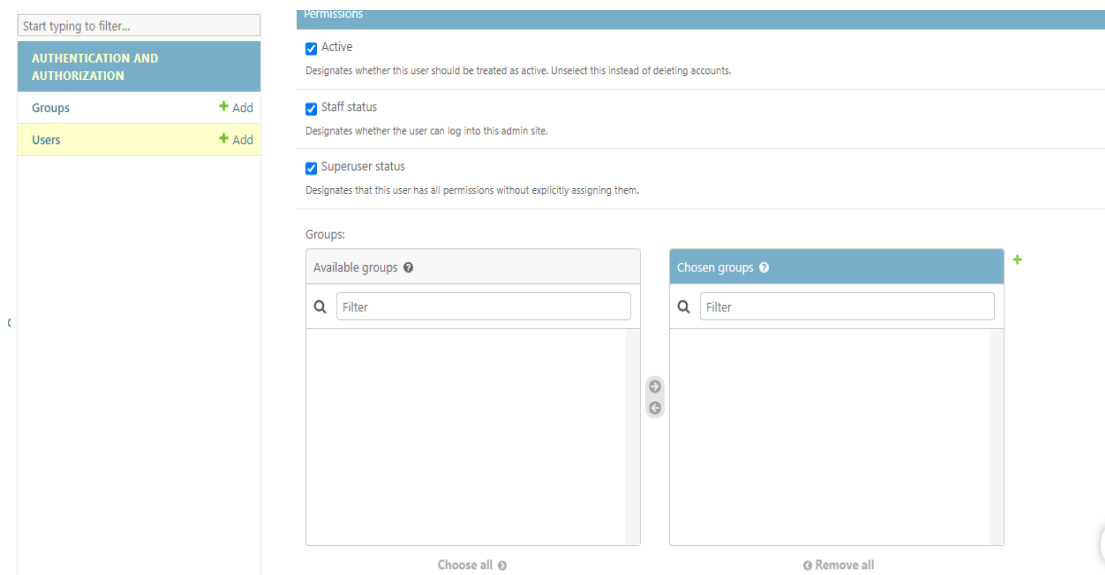
Hình 2. 25 Tạo Superuser

Khi đã tạo được tài khoản admin, ta tiến hành truy cập vào trang admin, theo đường dẫn trên, giao diện trang admin gồm: **Group** và **User**



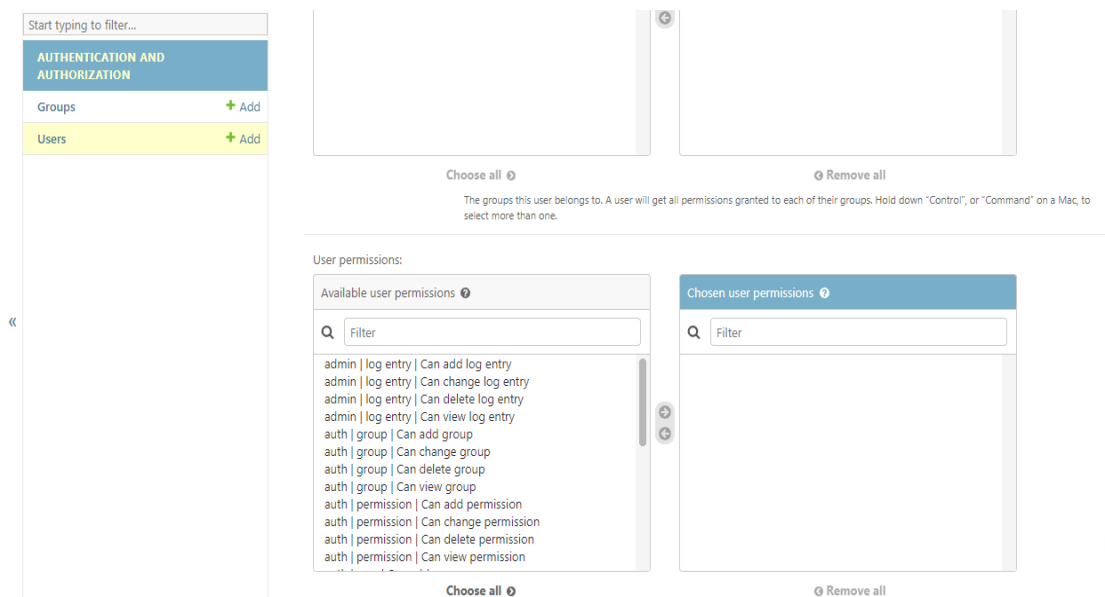
Hình 2. 26 Giao diện trang Admin

Trong trang admin, ta có thể thay thêm, sửa , xoá, phân quyền cho User có các quyền khác nhau gồm các quyền: có được phép truy cập vào trang admin hay không.



Hình 2. 27 Chỉnh sửa Users

Còn ở đây dùng để cấp quyền cho User đó có các quyền gì đối với cơ sở dữ liệu trong hệ thống, như có thể xem, thêm, sửa, xoá Group,....



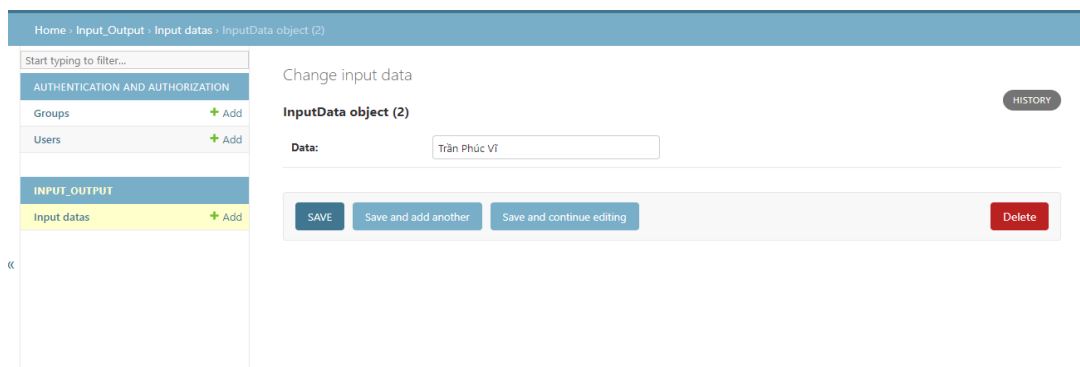
Hình 2. 28 Phân quyền users

Và tại sao khi này có định nghĩa các lớp trong **Models** là các bảng trong cơ sở dữ liệu, vậy bản đó đang ở đâu. Để có thể truy cập được vào các bảng đã tạo trong **Models** ta cần phải **register** nó trong **admin.py**, đây là cấu trúc của phương thức **register**.

```
nyweb > input_output > admin.py
1  from django.contrib import admin
2  from .models import InputData
3
4  admin.site.register(InputData)
5
```

Hình 2. 29 Cấu hình để hiển thị lên Admin

Sau khi đã đăng kí xong, thì bảng **Input datas** sẽ xuất hiện trên giao diện của admin, có thể truy cập vào để xem, thêm, xoá, sửa dữ liệu của bảng.

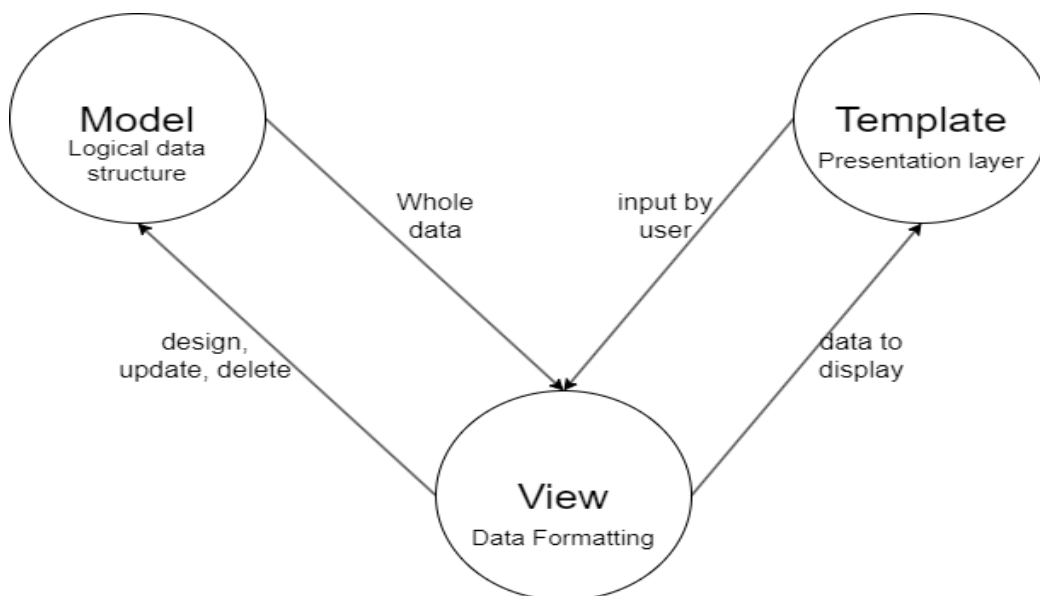


Hình 2. 30 Giao diện sau khi đã cấu hình ở Admin

2.3 Mô hình MVT

2.3.1 Giới thiệu

MVT và MVC là hai mô hình phần mềm đều chia ứng dụng thành các phần: logic xử lý, hiển thị và tương tác cơ sở dữ liệu. MVT (Model-View-Template) là biến thể của MVC, nhưng chúng về cơ bản giống nhau.



Hình 2. 31 Mô hình MVT

Model (Mô hình): Đây là phần chịu trách nhiệm xử lý dữ liệu và logic của ứng dụng. Mô hình biểu diễn cấu trúc dữ liệu và các chức năng xử lý dữ liệu, bao gồm truy xuất, cập nhật và xử lý logic.

View (Giao diện): Phần này là nơi hiển thị thông tin cho người dùng. Giao diện là thành phần tương tác với người dùng và hiển thị dữ liệu từ mô hình. Trong MVT, Template (mẫu giao diện) giúp tạo ra giao diện từ dữ liệu được cung cấp bởi mô hình.

Template (Mẫu): Đây là thành phần tạo ra giao diện dựa trên dữ liệu từ mô hình. Nó có vai trò tương tự như View trong MVC, nhưng thường chịu trách nhiệm nhiều hơn về việc tạo ra cấu trúc của giao diện từ dữ liệu.

Bảng 1 So sánh MVT và MVC

| Mô hình MVC | Mô hình MVT | Chức năng |
|-------------|-------------|----------------------------|
| Model | Model | Làm việc với cơ sở dữ liệu |
| View | Template | Hiển thị giao diện |
| Controller | View | Xử lý logic |

2.3.2 Sự hoạt động của mô hình MVT

- Khi người dùng tương tác các hoạt động với hệ thống, yêu cầu sẽ được gửi đến Model.
- Model sẽ xử lý yêu cầu này và trả về dữ liệu cần thiết với yêu cầu
- Tiếp đó kết quả sẽ được Model gửi cho Templates, tại đây là nơi dữ liệu sẽ được chuyển thành giao diện.
- Cuối cùng, giao diện sẽ được hiển thị thông qua các trình duyệt theo yêu cầu của người dùng

2.3.3 Ưu nhược điểm của mô hình

Ưu điểm:

Nói về ưu điểm của mô hình MVT trước hết sẽ nói về khả năng quản lý và phát triển dự án, nhờ vào sự tách biệt của 3 phần (Model-View-Template), ta có thể dễ dàng quản lý cũng như kiểm lỗi và phát triển hệ thống.

Tiếp theo là khả năng linh hoạt, nhờ vào sự tách biệt mà mô hình MVT rất linh hoạt, khi việc có thể thay đổi một thành phần mà không ảnh hưởng nhiều đến các phần khác. Với khả năng tách biệt 3 phần cũng góp phần giúp cho hiệu suất được nâng cao.

Nhược điểm:

Khả năng tách biệt tuy điểm lại nhiều ưu điểm, nhưng đó cũng tồn tại nhược điểm, việc tách biệt các thành phần yêu cầu khả năng hiểu biết chuyên sâu về cả 3 lĩnh vực (Model, View, Template) để có thể làm việc hiệu quả.

Việc đồng bộ và xử lý các logic cũng trở nên phức tạp và khó khăn hơn trong quá trình phát triển

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Mô tả bài toán

Để có thể xây dựng nên một web bán quần áo thể thao, trước hết ta cần nắm rõ tình hình của thị trường cũng như thị hiếu của khách hàng, xem xét các đối tượng nào sẽ là đối tượng tiềm năng trong tương lai.

Trước tiên ta cần phải khảo sát khắp các mạng xã hội cũng như các trang web khác để có thể cập nhật tình hình chung, sau đó tiến hành thiết kế các cơ sở dữ liệu và tiến hành vào thực hiện dự án.

Phân quyền cho hệ thống

Bộ phận quản lý: là một trong những bộ phận có quyền hạn trong hệ thống, chịu trách nhiệm quản lý tài khoản của khách hàng, cũng như xem xét cập nhật các mặt hàng mới trên thị trường, khảo sát tình hình, cập nhật lại thông tin, có khả năng thêm mới các sản phẩm, cũng như tạo mới tài khoản cho người dùng, có thể xem thông tin của các sản phẩm, có quyền sửa sản phẩm, nhưng sẽ không có quyền xóa sản phẩm.

Bộ phận quản trị: Có vai trò cao nhất trong hệ thống, đảm nhiệm chức năng tạo mới các quản lý, thêm xóa các quản lý và cũng có toàn quyền đối với hệ thống, có khả năng thêm, sửa, xóa các loại sản phẩm.

Quy trình làm việc

Khách hàng sẽ truy cập vào website thông qua địa chỉ URL, hoặc vào các công cụ tìm kiếm của trình duyệt để tìm kiếm.

Sau khi đã truy cập được vào website, khách hàng có thể xem xét các sản phẩm mà mình yêu thích, có thể xem các sản phẩm theo từng loại khác nhau được phân chia cụ thể rõ ràng.

Nếu khách hàng đã có sản phẩm muốn mua từ trước thì có thể tìm kiếm sản phẩm đó trên thanh tìm kiếm của website, nếu hệ thống có tồn tại sản phẩm thì sẽ trả về cho khách hàng các thông tin của sản phẩm cũng như các mặt hàng có tên tương tự.

Sau khi đã tìm thấy sản phẩm, khách hàng có thể vào xem các thông tin chi tiết của sản phẩm đã được cung cấp bên trong, khi đã ưng ý khách hàng có thể ấn mua để

thêm vào giỏ hàng, sau đó khách hàng có thể mua các sản phẩm khác hoặc có thể tiến hành tới bước xác nhận.

Khi đã chọn xong các sản phẩm ưng ý, khách hàng sẽ vào giỏ hàng để chỉnh sửa lại số lượng các sản phẩm mà khách hàng đã chọn, thêm các thông tin chi tiết như: họ tên, địa chỉ, số điện thoại,... và tiến hành xác nhận để hoàn tất.

Sau khi khách hàng đã xác nhận, trang quản trị sẽ được cập nhật đơn hàng, và tiến hành chuẩn bị hàng giao cho khách.

Yêu cầu về chức năng

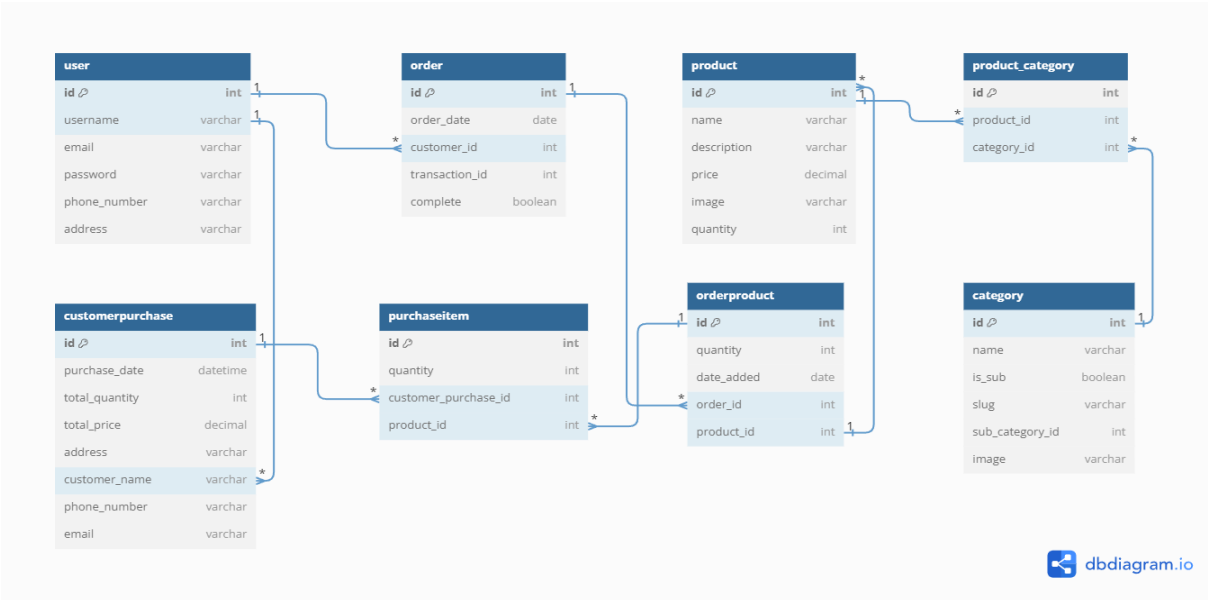
- Trang chủ: tổng quan về nội dung của website, thông tin liên lạc cũng như địa chỉ của website.
- Trang sản phẩm: thể hiện toàn bộ sản phẩm hiện có trong hệ thống, danh mục sản phẩm,...
- Trang thông tin sản phẩm: thể hiện cụ thể tên của sản phẩm, giá cả, cũng nhưng các thông tin chi tiết giúp khách hàng nắm rõ được chi tiết của sản phẩm, ngoài ra có thể có các hình từ nhiều góc khác nhau của sản phẩm để khách hàng tham khảo.
- Giỏ hàng: lưu trữ được các sản phẩm khách hàng đã từng đưa vào giỏ hàng, có khả năng thay đổi số lượng sản phẩm theo khách hàng mong muốn.
- Tìm kiếm: hiển thị ra được các sản phẩm mà khách hàng tìm kiếm theo tiêu chí là tên sản phẩm.

Yêu cầu phi vật lí:

- Để nói về một website bán hàng online, đầu tiên phải nói đến giao diện, giao diện phải ưu nhìn, trình bày rõ ràng, màu sắc hài hoà, không để mọi thứ quá lộn xộn, đảm bảo tính thẩm mỹ, tạo ấn tượng tốt với người dùng khi vừa truy cập vào website.
- Các chức năng phải dễ dàng sử dụng, phù hợp với hầu hết các độ tuổi của khách hàng, không nên quá phức tạp sẽ gây khó khăn cho một vài độ tuổi khách hàng.

3.2 Mô tả bài toán

3.2.1 Mô hình dữ liệu:



Hình 3. 1 Mô hình dữ liệu

3.2.2 Bảng thực thể:

Bảng 2 Bảng User

| STT | Tên thuộc tính | Mô tả | Kiểu dữ liệu |
|-----|----------------|--------------------|--------------|
| 1 | user_id | ID người dùng | int |
| 2 | username | Tên người dùng | varchar |
| 3 | password | Mật khẩu | varchar |
| 4 | email | Email người dùng | varchar |
| 5 | created_at | Thời gian tạo | datetime |
| 6 | updated_at | Thời gian cập nhật | datetime |

Bảng 3 Bảng Category

| STT | Tên thuộc tính | Mô tả | Kiểu dữ liệu |
|-----|-----------------|-----------------|--------------|
| 1 | category_id | ID danh mục | int |
| 2 | name | Tên danh mục | varchar |
| 3 | is_sub | Là danh mục phụ | boolean |
| 4 | slug | Slug | varchar |
| 5 | sub_category_id | ID danh mục con | int |
| 6 | image | Hình ảnh | varchar |

Bảng 4 Bảng Customer

| STT | Tên thuộc tính | Mô tả | Kiểu dữ liệu |
|-----|----------------------|-----------------------|--------------|
| 1 | customer_purchase_id | ID giao dịch mua hàng | int |
| 2 | purchase_date | Ngày mua hàng | datetime |
| 3 | total_quantity | Tổng số lượng | int |
| 4 | total_price | Tổng giá | decimal |
| 5 | address | Địa chỉ | varchar |
| 6 | customer_name | Tên khách hàng | varchar |
| 7 | phone_number | Số điện thoại | varchar |
| 8 | email | Email | varchar |

Bảng 5 Bảng Order

| STT | Tên thuộc tính | Mô tả | Kiểu dữ liệu |
|-----|----------------|---------------|--------------|
| 1 | order_id | ID đơn hàng | int |
| 2 | order_date | Ngày đặt hàng | date |
| 3 | customer_id | ID người dùng | int |
| 4 | transaction_id | ID giao dịch | int |
| 5 | complete | Hoàn thành | boolean |

Bảng 6 Bảng Product

| STT | Tên thuộc tính | Mô tả | Kiểu dữ liệu |
|-----|----------------|--------------|--------------|
| 1 | product_id | ID sản phẩm | int |
| 2 | name | Tên sản phẩm | varchar |
| 3 | description | Mô tả | varchar |
| 4 | price | Giá | decimal |
| 5 | image | Hình ảnh | varchar |
| 6 | quantity | Số lượng | int |

Bảng 7 Bảng Purchase Item

| STT | Tên thuộc tính | Mô tả | Kiểu dữ liệu |
|-----|----------------------|-----------------------|--------------|
| 1 | purchaseitem_id | ID mua hàng | int |
| 2 | quantity | Số lượng | int |
| 3 | customer_purchase_id | ID giao dịch mua hàng | int |
| 4 | product_id | ID sản phẩm | int |

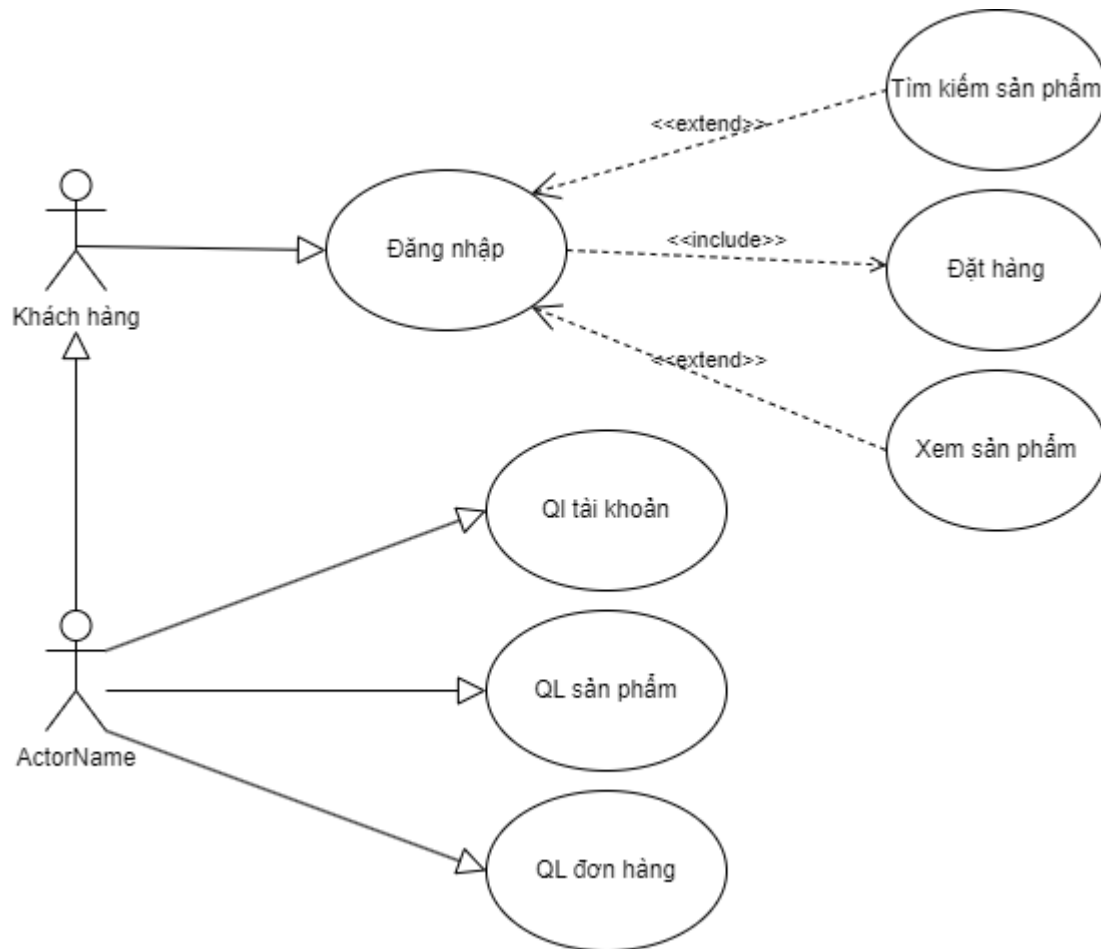
Bảng 8 Bảng Order Product

| STT | Tên thuộc tính | Mô tả | Kiểu dữ liệu |
|-----|-----------------|----------------------|--------------|
| 1 | orderproduct_id | ID sản phẩm đơn hàng | int |
| 2 | quantity | Số lượng | int |
| 3 | date_added | Ngày thêm | date |
| 4 | order_id | ID đơn hàng | int |
| 5 | product_id | ID sản phẩm | int |

Bảng 9 Bảng product_category

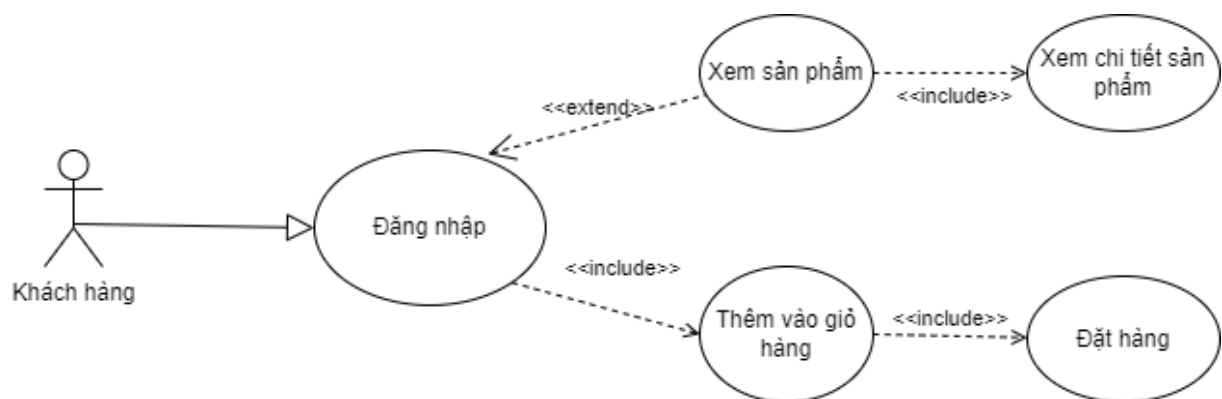
| STT | Tên thuộc tính | Mô tả | Kiểu dữ liệu |
|-----|---------------------|----------------------|--------------|
| 1 | product_category_id | ID sản phẩm danh mục | int |
| 2 | product_id | ID sản phẩm | int |
| 3 | category_id | ID danh mục | int |

3.2.3 Mô hình Use Case tổng quát:



Hình 3. 2 Mô hình Use Case tổng quát

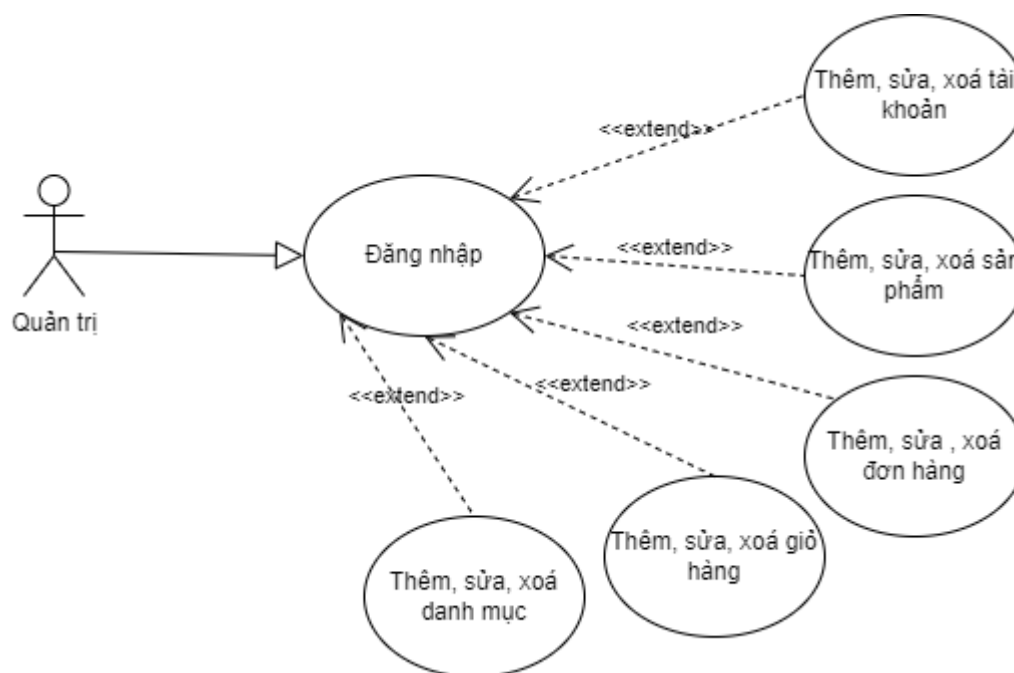
3.2.4 Mô hình Use Case tác nhân khách hàng:



Hình 3. 3 Mô hình Use Case tác nhân khách hàng

Khi khách hàng truy cập vào hệ thống, khách hàng có thể xem sản phẩm, xem chi tiết sản phẩm mà không cần phải đăng nhập, khi muốn thêm sản phẩm vào giỏ hàng và đặt hàng thì khách hàng bắt buộc phải đăng nhập.

3.2.5 Mô hình Use Case tác nhân quản trị:



Hình 3. 4 Mô hình User Case tác nhân quản trị

Quản trị bắt buộc phải đăng nhập mới được vào hệ thống, để có thể thực hiện các chức năng của một quản trị như: thêm, sửa, xoá,...

3.3 Xây dựng website

3.3.1 Xây dựng dữ liệu bên trong Django

Trong Django, dữ liệu được tạo ra theo từng class bên trong tập tin Models.py, gồm các bảng sau:

Dữ liệu cho bảng Category

```

class Category(models.Model):
    name = models.CharField(max_length=200, null=True)
    image = models.ImageField(null=True, blank=True)
    sub_category = models.ForeignKey('self', on_delete=models.CASCADE, related_name='sub_categories', null=True)
    is_sub = models.BooleanField(default=True)
    slug = models.SlugField(max_length=200, null=True)
    def __str__(self):
        return self.name
    @property
    def ImgURL (self):
        try:
            url = self.image.url
        except:
            url = ''
        return url
    
```

Hình 3. 5 Bảng Category

Dữ liệu cho bảng Product


```
class Product(models.Model):
    name = models.CharField(max_length=200, null=True, blank=False)
    quantity = models.IntegerField(default=0, null=True, blank=False)
    description = RichTextField()
    price = models.DecimalField(max_digits=10, decimal_places=0)
    category = models.ManyToManyField(Category, related_name='Categories')
    image = models.ImageField(null=True, blank=True)

    def __str__(self):
        return self.name

    @property
    def ImgURL (self):
        try:
            url = self.image.url
        except:
            url = ''
        return url
```

Hình 3. 6 Bảng Product

Dữ liệu cho bảng Order

```
class Order(models.Model):
    customer = models.ForeignKey(CustomUser, on_delete=models.CASCADE, null=True, blank=False)
    order_date = models.DateTimeField(auto_now_add=True)
    complete = models.BooleanField(default=False, null=True, blank=False)
    transaction_id = models.CharField(max_length=200, null=True, blank=False)

    def __str__(self):
        return str(self.id)

    @property
    def get_cart_items(self):
        orderproduct = self.orderproduct_set.all()
        total = sum([item.quantity for item in orderproduct])
        return total

    @property
    def get_cart_total(self):
        orderproduct = self.orderproduct_set.all()
        total = sum([item.get_total for item in orderproduct])
        return total
```

Hình 3. 7 Bảng Order

Dữ liệu cho bảng OrderProduct

```
class OrderProduct(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE, null=True, blank=False)
    product = models.ForeignKey(Product, on_delete=models.CASCADE, null=True, blank=False)
    quantity = models.IntegerField(default=0, null=True, blank=False)
    date_added = models.DateTimeField(auto_now_add=True)

    @property
    def get_total(self):
        total = self.product.price * self.quantity
        return total
```

Hình 3. 8 Bảng OrderProduct

Dữ liệu cho bảng CustomerPurchase

```
class CustomerPurchase(models.Model):
    customer_name = models.CharField(max_length=100, null=True, blank=False)
    address = models.CharField(max_length=200, null=True, blank=False)
    email = models.CharField(max_length=200, null=True, blank=False)
    phone_number = models.CharField(max_length=20, null=True, blank=False)
    products = models.ManyToManyField(Product, through='PurchaseItem')
    purchase_date = models.DateTimeField(default=timezone.now)
    total_quantity = models.PositiveIntegerField(default=0)
    total_price = models.DecimalField(max_digits=10, decimal_places=2, default=0.00)

    def __str__(self):
        local_purchase_date = timezone.localtime(self.purchase_date)
        return f"{self.customer_name} Mua hàng vào lúc {local_purchase_date.strftime('%Y-%m-%d %H:%M:%S')}
```

Hình 3. 9 Bảng CustomerPurchase

Dữ liệu cho bảng PurchaseItem

```
class PurchaseItem(models.Model):
    customer_purchase = models.ForeignKey(CustomerPurchase, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1) # Hoặc bất kỳ trường nào khác bạn muốn lưu thông tin sản phẩm

    # Các trường thông tin khác nếu cần

    def __str__(self):
        return f"Khách hàng: {self.customer_purchase}-Tên sản phẩm: {self.product.name} - Số lượng: {self.quantity}"

    def save(self, *args, **kwargs):
        super().save(*args, **kwargs)
        # Tính toán tổng giá và tổng số lượng của tất cả PurchaseItem liên quan
        purchase = self.customer_purchase
        purchase_items = PurchaseItem.objects.filter(customer_purchase=purchase)
        total_quantity = purchase_items.aggregate(models.Sum('quantity'))['quantity__sum'] or 0
        total_price = sum(item.product.price * item.quantity for item in purchase_items)

        # Cập nhật giá trị tổng giá và tổng số lượng vào CustomerPurchase
        purchase.total_quantity = total_quantity
        purchase.total_price = total_price
        purchase.save()
```

Hình 3. 10 Bảng PurchaseItem

3.3.2 Xây dựng chức năng xử lý

Trong Django, việc xử lý các chức năng của trang Web được xây dựng bên trong tập tin Views.py

3.3.2.1. Xây dựng chức năng đăng ký:

Trước hết, để xây dựng chức năng đăng ký, ta cần tạo một forms để có thể dễ dàng chỉnh sửa cũng như sắp xếp bố cục.

```
class CreatorUserForm(UserCreationForm):
    class Meta:
        model = CustomUser
        fields = ['username', 'email', 'first_name', 'last_name', 'password1', 'password2', 'address', 'phone_n
```

Hình 3. 11 Xây dựng form CreatorUserForm

Tiếp theo, tạo ra hàm **register**, gọi **CreatorUserForm** từ forms bên trên gán vào **form**, sau đó kiểm tra nếu có dữ liệu được nhập vào và phương thức **POST** được gọi đến, sẽ lưu dữ liệu và đi đến trang **Login**, Nếu phương thức không phải là **POST** đi đến địa chỉ **register.html**.

```
def register(request):
    form = CreatorUserForm()
    if request.method == "POST":
        form = CreatorUserForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    context={'form': form}
    return render(request, 'app/register.html', context)
```

Hình 3. 12 Hàm register xây dựng chức năng đăng ký

3.3.2.2. Xây dựng chức năng đăng nhập:

Sau khi đã tạo tài khoản khách hàng bằng trang **Register** khách hàng sẽ được đưa đến trang **Login** để tiến hành đăng nhập.

Tạo một hàm **loginuser**, nếu phương thức **POST** được gọi đến sẽ lấy giá trị username, password từ **request.POST** và sử dụng để xác thực thông tin người dùng bằng cách gọi hàm **authenticate** được Django hỗ trợ sẵn, kiểm tra nếu **user** tồn tại sẽ gọi đến user đó và đưa đến trang **home**, ngược lại sẽ đưa ra thông báo lỗi. Nếu phương thức không phải là **POST** thì hàm sẽ trả về trang **login.html**

```
def loginuser(request):
    if request.method == "POST":
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password = password)

        if user is not None:
            login(request, user)
            return redirect('home')
        else: messages.info(request, 'Tài khoản hoặc mật khẩu chưa đúng')

    context={}
    return render(request, 'app/login.html', context)
```

Hình 3. 13 Hàm loginuser xây dựng chức năng đăng nhập

3.3.2.3. Xây dựng chức năng trang Dashboard

Tạo hàm **dashboard**, gọi **Category** lấy tất cả dữ liệu có trong đó với **is_sub = False**, lấy ra tất cả sản phẩm trong **Product**, sau đó sẽ đi đến trang **dashboard.html**.

```
def dashboard(request):
    if request.user.is_authenticated:
        customer = request.user
        order, created = Order.objects.get_or_create(customer=customer, complete = False)
        cartItems = order.get_cart_items
    else:
        order={'get_cart_items':0, 'get_cart_total':0}
        cartItems = order['get_cart_items']
        categories = Category.objects.filter(is_sub = False)
        Products = Product.objects.all()
        context={'categories': categories, 'Products': Products, 'CartItems': cartItems}
    return render(request, 'app/dashboard.html', context)
```

Hình 3. 14 Hàm dashboard xây dựng chức năng trang Dashboard

Tại trang **dashboard.html** để hiển thị thông tin đã lấy ra, ta sử dụng vòng lặp **for** và hiển thị thông qua cặp ngoặc **{{ }}**

```
{% for category in categories %}
<div class="cate-hover" style="width:20%;height:80%; border: 1px ridge #c0c1c2 ">
    <a style="color:black;" href="{% url "category" %}?category={{category.slug}}">
    
    <hr>
    <p style="text-align:center;">{{category.name}}</p>
</a></div>
{% endfor %}
```

Hình 3. 15 Trang hiển thị lên dashboard.html

3.3.2.4. Xây dựng chức năng trang Home:

Tạo hàm **home**, gọi **Category** lấy tất cả dữ liệu có trong đó với **is_sub = False**, lấy ra tất cả sản phẩm trong **Product**, sau đó sẽ đi đến trang **home.html**.

```
def home(request):
    if request.user.is_authenticated:
        customer = request.user
        order, created = Order.objects.get_or_create(customer=customer, complete = False)
        cartItems = order.get_cart_items
    else:
        order={ 'get_cart_items':0, 'get_cart_total':0}
        cartItems = order['get_cart_items']
    categories = Category.objects.filter(is_sub = False)
    Products = Product.objects.all()
    context={'categories': categories, 'Products': Products, 'CartItems': cartItems}
    return render(request, 'app/home.html', context)
```

Hình 3. 16 Hàm home xây dựng chức năng trang Home

3.3.2.5. Xây dựng chức năng trang Search:

Tạo hàm **search**, nếu phương thức được gọi đến là **POST** lấy dữ liệu từ “searched” và gán vào biến **searched**, tạo biến **keys** lọc dữ liệu từ **Product** bằng hàm **filter** theo biến **searched**, nếu phương thức khác **POST** sẽ trả về trang **search.html** và lấy ra dữ liệu từ các biến đã được xử lý.

```
def search(request):
    searched = ''
    keys = []
    cartItems = 0
    if request.method == "POST":
        searched = request.POST['searched']
        keys = Product.objects.filter(name__icontains=searched)

    if request.user.is_authenticated:
        customer = request.user
        order, created = Order.objects.get_or_create(customer=customer, complete = False)
        items = order.orderproduct_set.all()
        cartItems = order.get_cart_items
    else:
        items=[]
        order={ 'get_cart_items':0, 'get_cart_total':0}
        cartItems = order['get_cart_items']

    categories = Category.objects.filter(is_sub = False)
    Products = Product.objects.all()
    return render(request, 'app/search.html', {'searched':searched, 'keys':keys, 'cartItems':cartItems, 'Products':
```

Hình 3. 17 Hàm search xây dựng chức năng trang Search

Tiếp theo, để trang **search.html** có thể hiển thị dữ liệu từ hàm **search** sẽ sử dụng vòng lặp **for** và hiển thị dữ liệu thông qua cặp ngoặc **{{ }}**.

```

<h2>Bạn muốn tìm: {{searched}}</h2>
{% for product in keys %}

<div style="background-color:white; width:21.5%; height:300px; margin-left:20px; margin-top:10px;" clas

    <a href="{% url 'product_detail' id=product.id%}" class="a" style="width:100%;" >
        
        <div style="width: 94%;margin-left: 3%;" >
            <p style="margin:0;">{{product.name}}</p>

            <p style="color:red;">đ{{product.price}} VNĐ</p>
        </div>
    </a>
</div>

{% endfor %}

```

Hình 3. 18 Hiển thị lên trang search.html

3.3.2.6. Xây dựng chức năng trang Product_detail:

Tạo hàm **product_detail**, gọi **Category** lấy tất cả dữ liệu có trong đó với **is_sub = False**, lấy ra tất cả sản phẩm trong **Product**, sau đó sẽ đi đến trang **product_detail.html**.

```

def product_detail(request, id):
    if request.user.is_authenticated:
        customer = request.user
        order, created = Order.objects.get_or_create(customer=customer, complete = False)
        items = order.orderproduct_set.all()
        cartItems = order.get_cart_items
    else:
        items=[]
        order={'get_cart_items':0, 'get_cart_total':0}
        cartItems = order['get_cart_items']
    products = Product.objects.filter(id=id)
    categories = Category.objects.filter(is_sub=False)
    context = {'products': products, 'categories': categories, 'items': items, 'order': order, 'CartItems': car
    return render(request, 'app/product_detail.html', context)

```

Hình 3. 19 Hàm product_detail xây dựng chức năng trang Product_detail

Tiếp theo, để hiển thị dữ liệu đã lấy từ hàm **product_detail** ta sử dụng vòng lặp **for** và dữ liệu sẽ được hiển thị thông qua cặp ngoặc **{{ }}**, trong vòng lặp **for** ta sẽ kiểm tra khách hàng đã đăng nhập chưa bằng cách sử dụng **if request.user.is_authenticated**, nếu khách hàng đã đăng nhập, sẽ kiểm tra số lượng của sản phẩm **if product.quantity > 0** có lớn hơn 0, nếu có sẽ hiển thị nút cho phép khách hàng thêm vào giỏ, ngược lại sẽ hiển thị “sản phẩm đã hết hàng”. Nếu khách hàng chưa đăng nhập sẽ hiển thị “Vui lòng đăng nhập để thêm vào giỏ hàng”.

```
{% for product in products %}

<div class="infor_product" style=" width:80%; margin-left:10%;" >
<div style="background-color:white; width: 100%;height: 100%;padding:10px;box-shadow: 0px 0px 1px 0px;margin-top:
  
</div>
<div style=" box-shadow: 0px 0px 1px 0px;background-color:white; width:100%; height: 424px;display: flex;flex-
  <h2 style="text-align:center;">{{ product.name }}</h2>
  <p style="color:red;">₫{{product.price}} VNĐ</p>
  <p>Số lượng: {{product.quantity}}</p>
  {% if request.user.is_authenticated %}
    {% if product.quantity > 0 %}
      <div class="btn_css_cart">
        <a data-product="{{ product.id }}" data-action="add" class="update-cart" style="color: black;ma
      </div>
    {% else %}
      <p>Sản phẩm đã hết hàng</p>
    {% endif %}
  {% else %}

    <p>Vui lòng đăng nhập để thêm vào giỏ hàng<p>

  {% endif %}
```

Hình 3. 20 Hiển thị lên trang *product_detail.html*

3.3.2.7. Xây dựng chức năng trang Category:

Tạo hàm **category** nhận một đối tượng **Request** làm tham số đầu vào và trả về , trong hàm trước tiên ta sẽ lọc tất cả **Category** với **is_sub = False** và gán vào **categories** biến **active_category** được lấy từ yêu cầu ‘**category**’ của Http, sau đó kiểm tra nếu **active_category** có giá trị hàm sẽ truy vấn cơ sở dữ liệu để lấy danh sách sản phẩm thuộc danh mục tương ứng .

```
def category(request):
    categories = Category.objects.filter(is_sub = False)
    active_category = request.GET.get('category','')
    if active_category:
        products = Product.objects.filter(category__slug=active_category)
        context={'categories': categories, 'products': products, 'active_category':active_category}
        return render(request, 'app/category.html',context)
# Create your views here
```

Hình 3. 21 Hàm *category* xây dựng chức năng trang *Category*

Để hiển thị sản phẩm theo **category** ta sử dụng vòng lặp **for** với giá trị sẽ được hiển thị thông qua cặp ngoặc **{{}}**.

```
<h2 style="text-align:center;">Đây là danh mục: {{active_category}}</h2>
<hr>
{% for product in products %}

<div style="background-color:white; width:21.5%; height:300px; margin-left:20px; margin-top:10px;" clas
  <a href="{% url 'product_detail' id=product.id%}" class="a" style="width:100%;">
    
    <div style="width: 94%;margin-left: 3%;">
      <p style="margin:0;">{{product.name}}</p>
      <p style="color:red;">₫{{product.price}} VNĐ</p>
    </div>
  </a>
</div>

{% endfor %}
```

Hình 3. 22 Hiển thị lên trang *category.html*

3.3.2.8. Xây dựng chức năng trang Cart:

Tạo hàm **cart** nhận một đối tượng **Request** làm tham số đầu vào và trả về một trang web **cart.html**, trong hàm kiểm tra nếu khách hàng đã đăng nhập sẽ lấy thông tin khách hàng gán vào biến **customer**, sau đó gọi ra phương thức **get_or_create** (lấy hoặc tạo mới) để kiểm tra thông tin khách hàng vừa đăng nhập với **customer** được tạo của **Order**, sau đó lấy giá trị **get_cart_items** được tạo trong **Order** đưa vào **CartItem**s để hiển thị số lượng mà khách hàng đó đã thêm vào giỏ hàng. Nếu khách hàng chưa đăng nhập sẽ gán các giá trị trên bằng 0.

```
def cart(request):
    if not request.user.is_authenticated:
        return redirect('login')
    if request.user.is_authenticated:
        customer = request.user
        order, created = Order.objects.get_or_create(customer=customer, complete=False)
        items = order.orderproduct_set.all()
        cartItems = order.get_cart_items
    else:
        items=[]
        order={'get_cart_items':0, 'get_cart_total':0}
        cartItems = order['get_cart_items']
    categories = Category.objects.filter(is_sub=False)
    context={'categories': categories, 'items': items, 'order': order, 'CartItem': cartItems}
    return render(request, 'app/cart.html', context)
```

Hình 3. 23 Hàm cart xây dựng chức năng trang Cart

Để hiển thị sản phẩm trong **cart** ta sử dụng vòng lặp **for** với giá trị sẽ được hiển thị thông qua cặp ngoặc **{{ }}**.

```
{% for item in items %}
<tr>
  <td class="item_cart"></td>
  <td class="item_cart">{{ item.product.name }}</td>
  <td class="item_cart">{{ item.product.price }} VND</td>
  <td class="item_cart">
    {% if item.product.quantity > 0 %}
    <a class="quantity_cart" data-product="{{ item.product.id }}" data-action="add" style="cursor: pointer;">
    {% endif %}
    {{ item.quantity }}
    <a class="quantity_cart" data-product="{{ item.product.id }}" data-action="remove" style="cursor: pointer;">
    </td>
  <td class="item_cart">{{ order.get_total_item }}</td>
</tr>
{% endfor %}

</table>

</div>
<div class="all_item">
  <span><a href="{% url 'home' %}" style="text-decoration: none; color: black; border: 1px solid black; padding: 5px;">
  <span>Số lượng: {{order.get_cart_items}}</span>
  <span>Tổng: {{order.get_cart_total}} VND</span>
  <span><a type="button" class="btn_checkout" href="{% url 'checkout' %}">Checkout</a></span>
</div>
```

Hình 3. 24 Hiển thị lên trang cart.html

3.3.2.9. Xây dựng chức năng trang Checkout:

Trước khi xây dựng chức năng trang **checkout** ta cần tạo trước một lớp **CheckoutForm** để lấy dữ liệu của khách hàng mua hàng.

```
class CheckoutForm(forms.Form):
    customer_name = forms.CharField(max_length=100)
    email = forms.CharField(max_length=200)
    address = forms.CharField(max_length=200)
    phone_number = forms.CharField(max_length=20)
    class Meta:
        model = CustomUser
        fields = ['username', 'email', 'first_name', 'last_name', 'password1', 'password2', 'address', 'phone_n
```

Hình 3. 25 Tạo CheckoutForm

Hàm **checkout** xử lý quá trình thanh toán. Hàm này kiểm tra xem khách hàng đã đăng nhập hay chưa. Nếu khách hàng chưa đăng nhập, hàm sẽ chuyển hướng đến trang đăng nhập. Nếu khách hàng đã đăng nhập, hàm sẽ lấy đơn hàng chưa hoàn thành của người dùng từ cơ sở dữ liệu. Nếu không có đơn hàng chưa hoàn thành, hàm sẽ tạo một đơn hàng mới.

```
def checkout(request):
    order = None

    if request.method == 'POST':
        if request.user.is_authenticated:
            customer = request.user
            order = Order.objects.get(customer=customer, complete=False)
            cart_items = order.orderproduct_set.all()

            form = CheckoutForm(request.POST)
            if form.is_valid():
                # Lấy dữ liệu từ form
                customer_name = form.cleaned_data['customer_name']
                email = form.cleaned_data['email']
                address = form.cleaned_data['address']
                phone_number = form.cleaned_data['phone_number']

                # Tạo CustomerPurchase
                purchase = CustomerPurchase.objects.create(
                    customer_name=customer_name,
                    address=address,
                    phone_number=phone_number,
                    email=email
                )

                # Lưu thông tin sản phẩm từ giỏ hàng vào CustomerPurchase
                for item in cart_items:
                    PurchaseItem.objects.create(
                        customer_purchase=purchase,
                        product=item.product,
                        quantity=item.quantity
                    )

                # Đánh dấu đơn hàng đã hoàn thành
                order.complete = True
                order.save()

                # Xóa giỏ hàng sau khi thanh toán
                request.session['cart'] = {}
                # Chuyển hướng đến trang thành công sau khi nhận liệu
```

Hình 3. 26 Hàm checkout xây dựng chức năng trang Checkout

Sau đó, hàm sẽ lấy dữ liệu biểu mẫu thanh toán từ khách hàng. Nếu dữ liệu biểu mẫu hợp lệ, hàm sẽ tạo một đối tượng **CustomerPurchase** để lưu trữ thông tin khách hàng. Hàm cũng sẽ tạo các đối tượng **PurchaseItem** cho mỗi mặt hàng trong giỏ hàng. Cuối cùng, hàm sẽ đánh dấu đơn hàng là đã hoàn thành và xóa phiên giỏ hàng.

```
request.session['cart'] = {}
# Chuyển hướng đến trang thành công sau khi nhập liệu
return redirect('home')
else:
    return redirect('login')
else:
    if request.user.is_authenticated:
        customer = request.user
        order = Order.objects.get(customer=customer, complete=False)
        cartItems = order.get_cart_items
        form_data = {
            'customer_name': customer.first_name,
            'email': customer.email,
            'address': customer.address,
            'phone_number': customer.phone_number,
        }
        form = CheckoutForm(initial=form_data)
    else:
        return redirect('login')

total_quantity = 0
total_price = Decimal('0.00')
if order:
    items = order.orderproduct_set.all()
    total_quantity = order.get_cart_items
    total_price = order.get_cart_total
else:
    items = []

return render(request, 'app/checkout.html', {'form': form, 'items': items, 'total_quantity': total_quantity, 'total_price': total_price, 'c
```

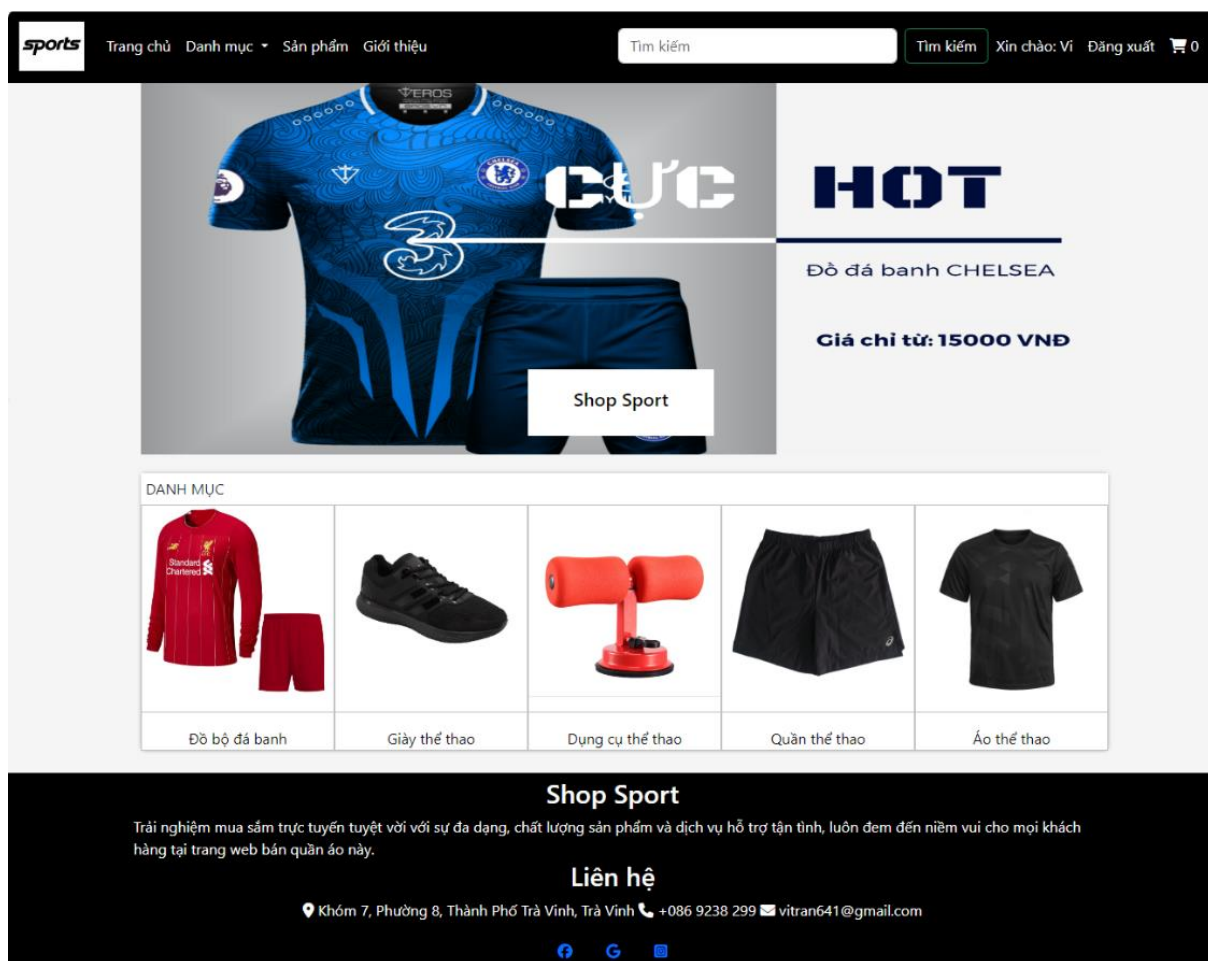
Hình 3. 27Hàm checkout xây dựng chắc chắn trang Checkout tiếp theo

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1 Giao diện người dùng

4.1.1 Giao diện trang chủ

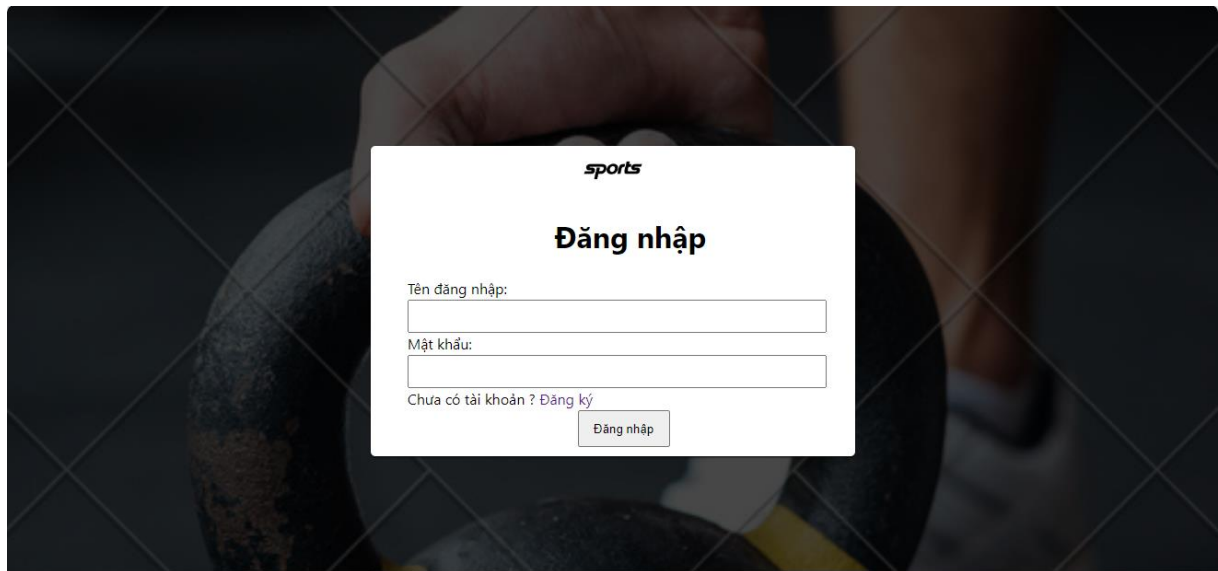
Trang chủ là trang đầu tiên hiển thị khi người dùng truy cập vào trang web, giao diện bao gồm banner và danh mục các loại sản phẩm hiện đang có trong shop. Giao diện tương đối dễ sử dụng và gọn gàng.



Hình 4. 1 Giao diện trang chủ

4.1.2 Giao diện trang đăng nhập

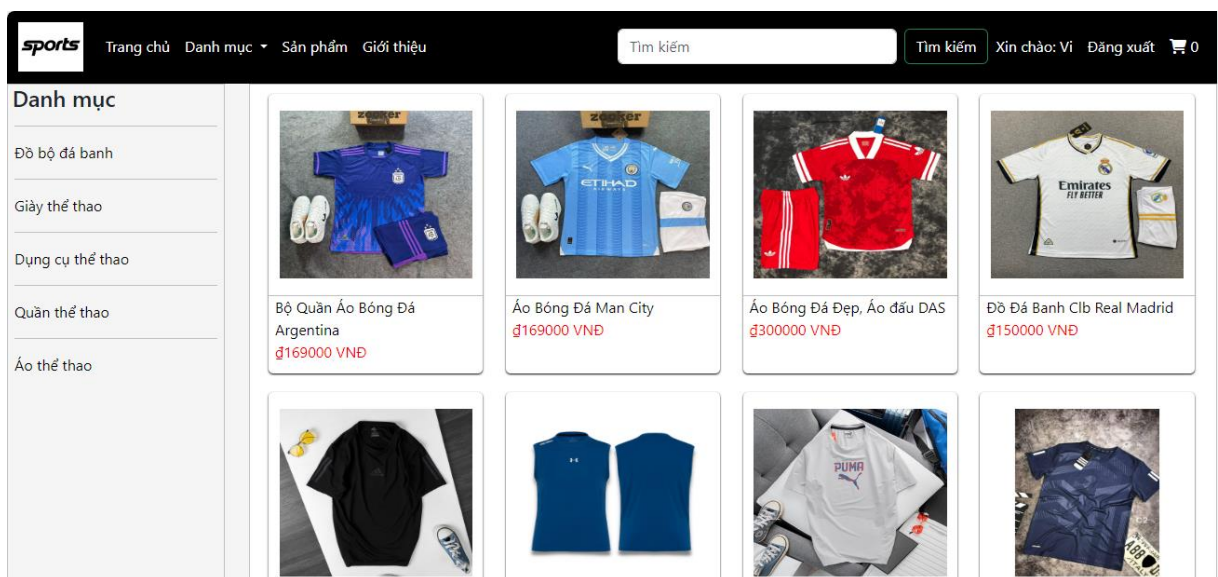
Khi ấn vào đăng nhập trên góc phải của trang web, người dùng được chuyển đến trang đăng nhập, tại đây người dùng có thể đăng nhập tài khoản mà mình có để có thể mua sản phẩm, nếu người dùng chưa có tài khoản ấn vào “Đăng ký” sẽ dẫn đến trang đăng ký cho người dùng đăng ký tài khoản mới.



Hình 4. 2 Giao diện trang đăng nhập

4.1.3 Giao diện trang sản phẩm

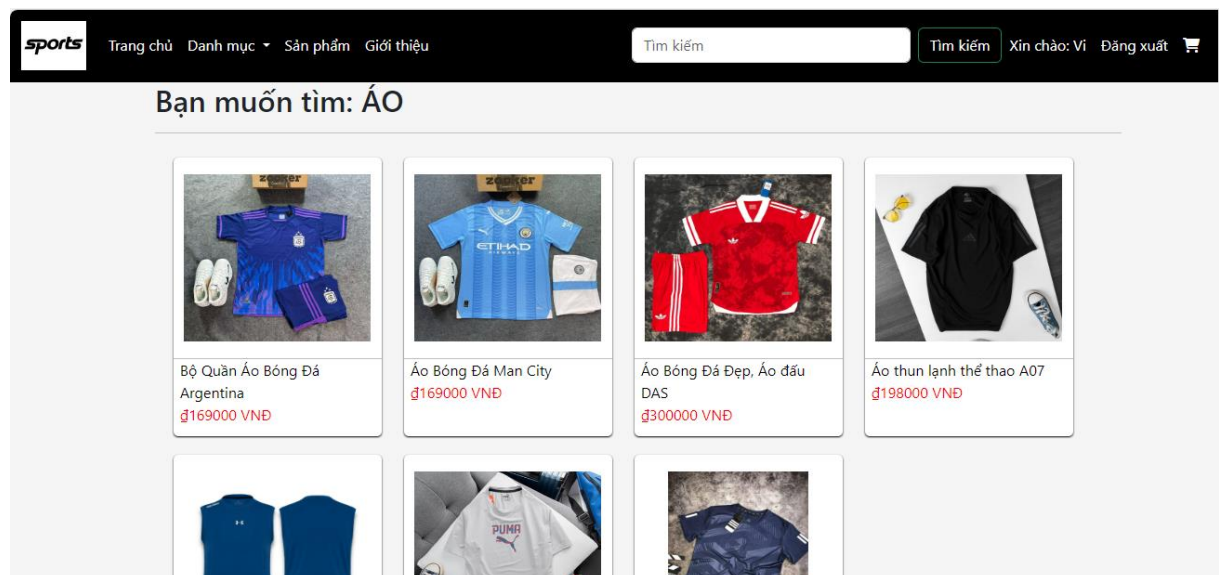
Đây là giao diện hiển thị toàn bộ sản phẩm hiện có trong Shop, tại đây người dùng có thể xem được tên, giá cũng như hình ảnh của các sản phẩm, bên trái là danh mục các loại sản phẩm để người dùng dễ dàng thao tác qua lại để có thể xem từng sản phẩm của từng loại sản phẩm.



Hình 4. 3 Giao diện trang sản phẩm

4.1.4 Giao diện trang tìm kiếm

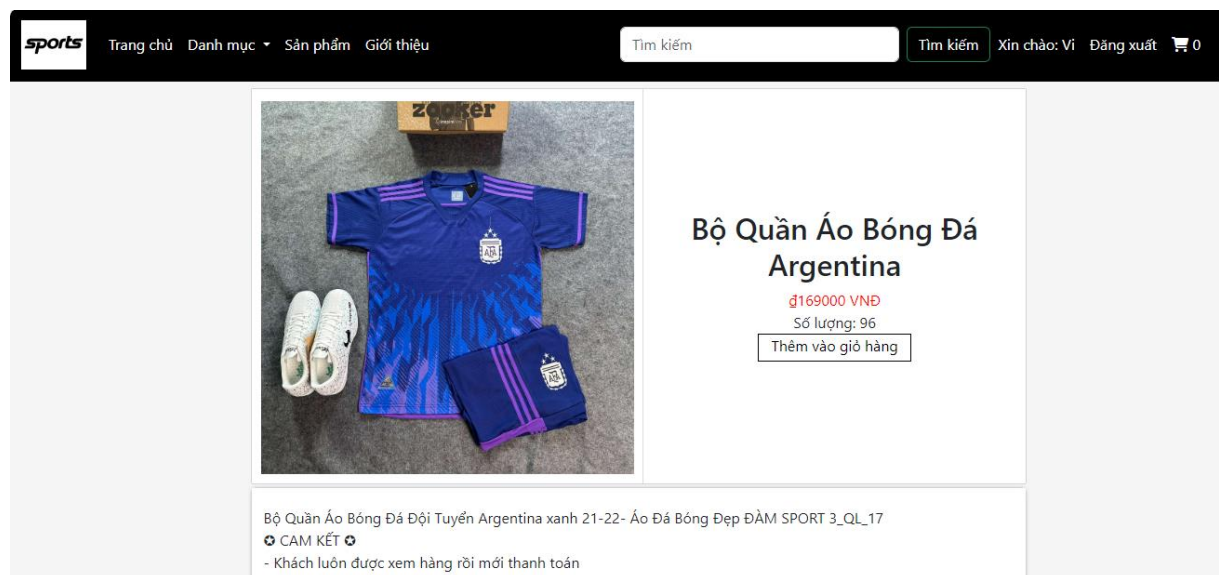
Người dùng có thể tìm kiếm sản phẩm theo kí tự có trong tên của sản phẩm thông qua nút tìm kiếm ở đầu trang



Hình 4. 4 Giao diện trang tìm kiếm

4.1.5 Giao diện chi tiết sản phẩm

Khi người dùng chọn vào sản phẩm, trang chi tiết sản phẩm sẽ xuất hiện, tại đây hiển thị gồm: tên sản phẩm, giá, số lượng, hình ảnh, thông tin chi tiết của sản phẩm. Khi muốn mua sản phẩm người dùng ấn vào “Thêm vào giỏ hàng” để thêm sản phẩm vào giỏ hàng.

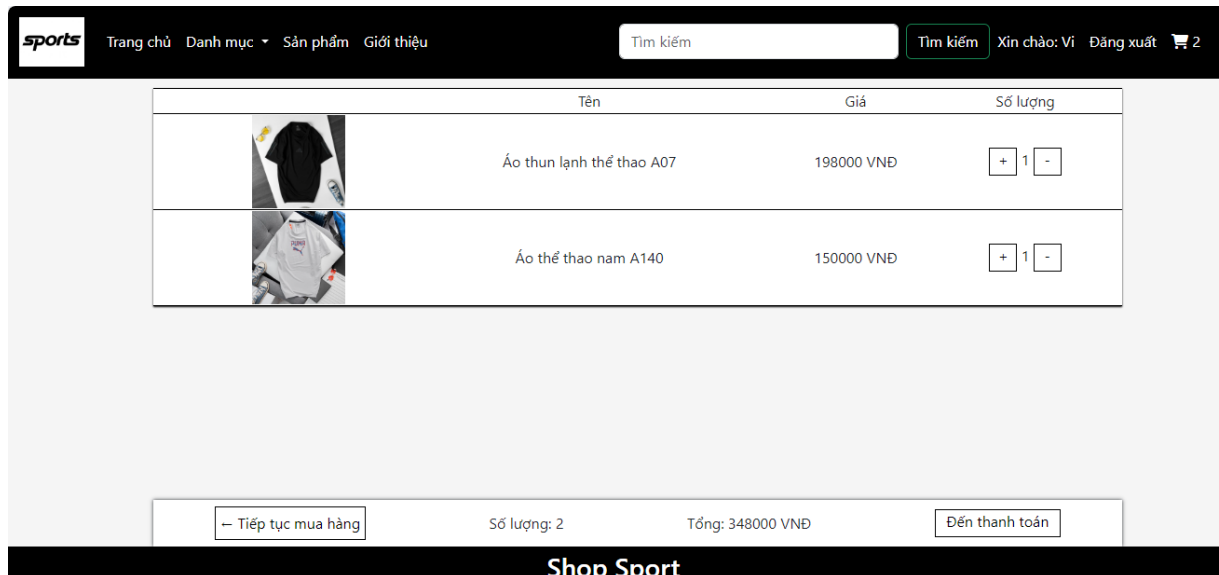


Hình 4. 5 Giao diện chi tiết sản phẩm

4.1.6 Giao diện trang giỏ hàng

Khi người dùng thêm sản phẩm vào giỏ hàng ở mục chi tiết sản phẩm, số lượng bên giỏ hàng sẽ tăng lên, và khi ấn vào giỏ hàng, đây là giao diện của trang giỏ hàng.

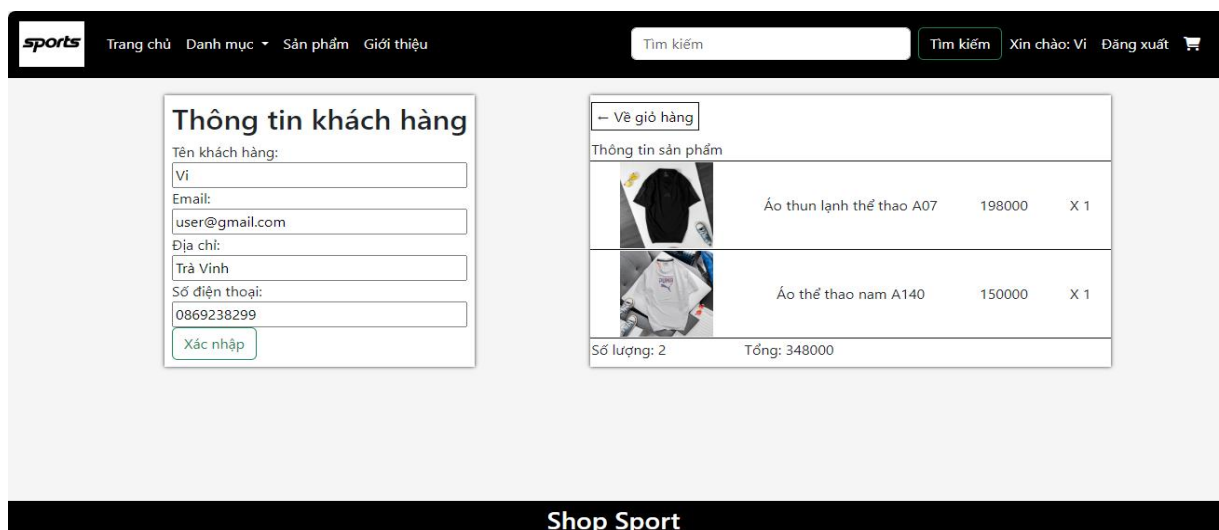
Tại đây người dùng sẽ xem được chi tiết các sản phẩm đã từng thêm vào giỏ hàng, hiển thị chi tiết bao gồm: tên sản phẩm, số lượng, giá, và tổng giá của sản phẩm. Nếu người dùng muốn tiếp tục mua hàng thì ấn vào “Tiếp tục mua hàng” sẽ chuyển người dùng về lại trang sản phẩm. Nếu muốn thanh toán thì ấn vào “Đến thanh toán” lúc này giao diện hiển thị thông tin để thanh toán sẽ xuất hiện.



Hình 4. 6 Giao diện trang giỏ hàng

4.1.7 Giao diện trang thanh toán

Khi ấn vào “Đến thanh toán” trong giỏ hàng, người dùng sẽ được chuyển đến trang xác nhận thanh toán. Tại đây thông tin của người dùng sẽ được truy xuất ra, nếu muốn thay đổi thông tin xác nhận, người dùng có thể chỉnh ở giao diện bên trái. Sau khi đã chỉnh sửa xong, người dùng sẽ ấn “Xác nhận”. Lúc này sản phẩm trong giỏ hàng sẽ mất đi và thông tin về khách hàng sẽ được gửi cho quản trị.

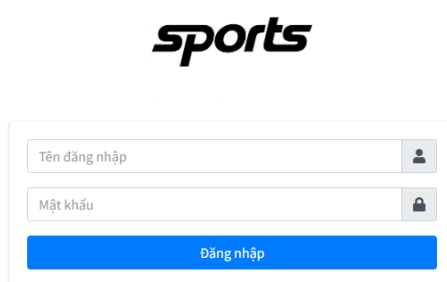


Hình 4. 7 Giao diện trang thanh toán

4.2 Giao diện quản trị

4.2.1 Giao diện trang đăng nhập quản trị

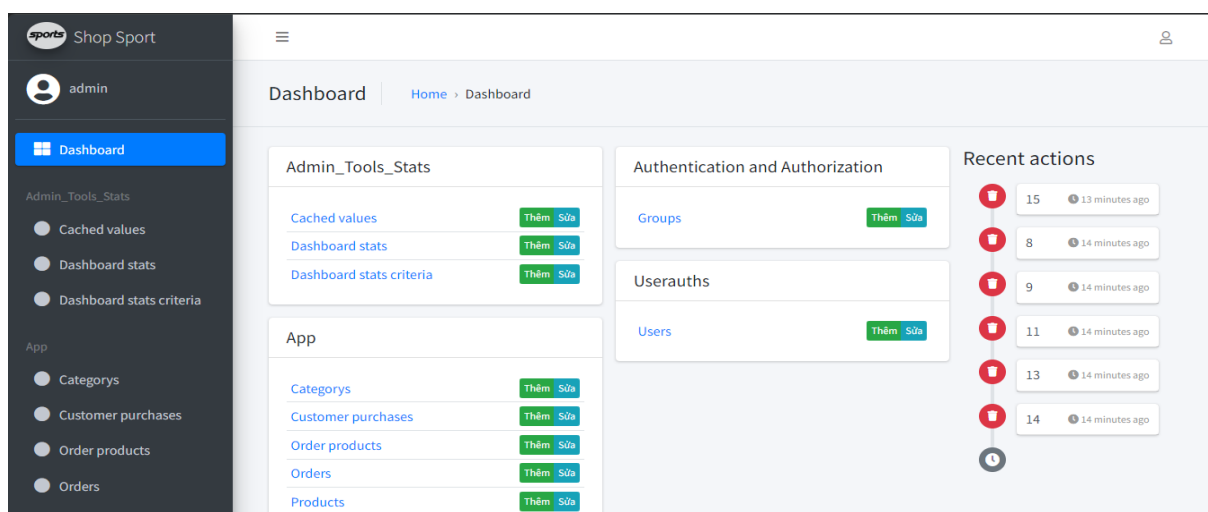
Để vào được trang Admin, quản trị cần phải truy cập vào đường dẫn <http://127.0.0.1:8000/admin/>. Sau đó tiến hành đăng nhập với tài khoản quản trị viên đã được cung cấp.



Hình 4. 8 Giao diện đăng nhập quản trị

4.2.1 Giao diện trang chủ quản trị

Đây là giao diện chính của trang quản trị, quản trị có thể xem chi tiết của từng loại sản phẩm, từng danh mục, cũng như thông tin khách hàng, đơn hàng và các sản phẩm đã được thêm vào giỏ hàng. Quản trị có quyền thêm mới, sửa hoặc xóa các thông tin cần thiết, ngoài ra có thể tạo tài khoản quản trị khác và phân quyền cụ thể cho từng tài khoản.



Hình 4. 9 Giao diện trang chủ quản trị

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

5.1.1 Kết quả đạt được

1. Về mặt lý thuyết

- Hiểu được cơ bản cách hoạt động của Django Framework
- Thiết kế được cơ sở dữ liệu tương đối, có thể áp dụng được trong phạm vi các chương trình nhỏ và vừa.

- Xây dựng được trang web với các chức năng của trang web bán hàng cơ bản.
- Ứng dụng kiến thức đã học vào xây dựng giao diện.

2. Về mặt phần mềm

- Hoàn thiện đầy đủ các chức năng cơ bản như: thêm, xóa, sửa, thêm thông tin vào giỏ hàng, lấy được dữ liệu của khách hàng,.....
- Có thể tìm kiếm được thông tin các sản phẩm mong muốn.
- Giao diện thân thiện, trực quan và linh hoạt.

5.1.2 Hạn chế

- Chức năng của chương trình vẫn còn nhiều sai sót, chưa thể gửi thông tin về sản phẩm mua hàng qua email cho khách hàng.
- Chưa tận dụng được hết tiềm năng của Framework, giao diện tương đối cơ bản.
- Chưa thể thống kê chi tiết có bao nhiêu sản phẩm, có bao nhiêu đơn đặt,....

5.2 Hướng phát triển

- Về giao diện: có thể cải thiện cho giao diện đẹp hơn, ưa nhìn hơn, hoạt động mượt hơn.
- Về các chức năng: bổ sung thêm thống kê, thêm vào khả năng gửi email khi khách hàng tạo tài khoản để xác nhận email, có thể gửi email cho về thông tin đơn hàng cho khách hàng,....

DANH MỤC TÀI LIỆU THAM KHẢO

DANH SÁCH TÀI LIỆU

- [1] **Phạm Minh Dương**, *Tài liệu giảng dạy môn Phân tích và thiết kế hệ thống thông tin (lưu hành nội bộ)*, trường ĐH Trà Vinh, 2014.
- [2] **Đoàn Phước Miên, Phạm Thị Trúc Mai**, *Tài liệu giảng dạy Thiết kế và lập trình web(lưu hành nội bộ)*, trường ĐH Trà Vinh, 2014.

DANH SÁCH WEBSITE

- [3] <https://docs.djangoproject.com/en/3.0/> (15/12/2023)
- [4] <https://www.w3schools.com/django/index.php> (13/12/2023)
- [5] <https://www.elib.vn/doc/2020/20200821/tim-hieu-lap-trinh-python-va-ung-dung-phat-trien-ung-dung-web-voi-django984.pdf> (13/12/2023)
- [6] <https://docs.python.org/> (14/12/2023)