

TRƯỜNG ĐẠI HỌC YERSIN ĐÀ LẠT
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN
MÔN : LẬP TRÌNH WED

ĐỀ TÀI: NỘI THẤT NHÀ XINH

Giáo viên hướng dẫn: Nguyễn Đức Tấn

Sinh viên thực hiện: Thuận Gia Phong-2301010022

Lời mở đầu

Trong thời đại công nghệ số phát triển mạnh mẽ, việc ứng dụng công nghệ thông tin vào kinh doanh ngày càng trở nên phổ biến và cần thiết. Đặc biệt, trong lĩnh vực nội thất – nơi yếu tố thẩm mỹ, trải nghiệm khách hàng và khả năng tiếp cận thông tin đóng vai trò quan trọng – một website hiện đại không chỉ giúp giới thiệu sản phẩm mà còn là cầu nối hiệu quả giữa doanh nghiệp và khách hàng.

Đề tài **“Lập trình web nội thất Nhà Xinh”** được thực hiện nhằm xây dựng một website bán hàng đơn giản nhưng đầy đủ chức năng cơ bản như: xem sản phẩm, giỏ hàng, đặt hàng, đăng nhập, liên hệ và quản lý đơn hàng. Qua đó, đề tài giúp người thực hiện hiểu rõ hơn về cách xây dựng một website thực tế bằng công nghệ ASP.NET Core MVC kết hợp với cơ sở dữ liệu SQL Server.

Thông qua quá trình thực hiện đề tài, sinh viên có cơ hội củng cố kiến thức lập trình, phát triển tư duy hệ thống và làm quen với các bước triển khai một ứng dụng web hoàn chỉnh. Hy vọng rằng, đề tài sẽ mang lại cái nhìn thực tiễn và là bước khởi đầu vững chắc cho những dự án chuyên sâu hơn trong tương lai.

Mục lục

Chương 1 Cơ sở lý thuyết.....	5
1.1 Giới thiệu mô hình MVC.....	5
1.2 Tổng quan về ASP.NET Core	8
1.3 Mô hình MVC trong ASP.NET Core	10
1.4 Kết luận chương.....	12
Chương 2 Xây dựng ứng dụng thực tế	12
2.1 Phát biểu bài toán, ý nghĩa thực tiễn.....	12
2.2 Phân tích yêu cầu ứng dụng.....	12
2.3 Thiết kế cơ sở dữ liệu.....	16
2.4. Thiết kế giao diện người dùng.....	16
2.5. Thiết kế thành phần MVC	17
2.6. Triển khai và cài đặt	25
Chương 3 Kết quả chương trình.....	26
3.1. Demo chức năng chính của hệ thống	26
3.2. Ảnh chụp màn hình chương trình hoạt động.....	26
3.3. Nhận xét và đánh giá kết quả.....	31
Kết luận.....	33
Tài liệu tham khảo.....	33

Danh mục hình ảnh

Hình 1 Model Product.....	10
Hình 2 View.....	11
Hình 3 Controller	11
Hình 4 Use Case.....	15
Hình 5 Database	16
Hình 6 MVC.....	25
Hình 7 Trang chủ.....	27
Hình 8 Shop	27
Hình 9 Giỏ hàng	28
Hình 10 Đặt hàng	28
Hình 11 Liên hệ.....	29
Hình 12 Đăng ký.....	29
Hình 13 Đăng nhập	30
Hình 14 Lịch sử đơn hàng.....	30

Chương 1 Cơ sở lý thuyết

1.1 Giới thiệu mô hình MVC

Khái niệm:

MVC là viết tắt của **Model - View - Controller**, là một mô hình kiến trúc phần mềm được sử dụng phổ biến trong phát triển ứng dụng web và phần mềm nói chung. Mô hình này chia ứng dụng thành ba thành phần riêng biệt, giúp dễ dàng quản lý mã nguồn, nâng cấp và bảo trì hệ thống.

- **Model (M)**: Là thành phần chịu trách nhiệm quản lý dữ liệu và logic nghiệp vụ của ứng dụng. Model tương tác trực tiếp với cơ sở dữ liệu và chịu trách nhiệm xử lý dữ liệu đầu vào từ người dùng, tính toán, và cập nhật dữ liệu. Ví dụ: Trong ứng dụng bán hàng, Model có thể là các lớp như Product, Order, User,... và tương ứng với các bảng trong cơ sở dữ liệu.
- **View (V)**: Là thành phần hiển thị thông tin cho người dùng. View chỉ nhận dữ liệu từ Controller và hiển thị chúng theo định dạng mong muốn. View không xử lý logic mà chỉ tập trung vào việc hiển thị nội dung. Trong ASP.NET Core MVC, View thường được viết bằng Razor (.cshtml).
- **Controller (C)**: Là cầu nối giữa Model và View. Controller nhận các yêu cầu từ người dùng (thường là các hành động như nhấp chuột, gửi form...), xử lý logic (nếu có), gọi Model để lấy dữ liệu cần thiết và truyền dữ liệu đó sang View để hiển thị. Controller cũng là nơi quyết định luồng xử lý của chương trình.

Đặc điểm:

Kiến trúc MVC không chỉ là một mô hình thiết kế mà còn là một triết lý giúp tổ chức mã nguồn ứng dụng một cách hiệu quả, mang lại nhiều lợi ích đáng kể cho quá trình phát triển và bảo trì. Dưới đây là các đặc điểm nổi bật của MVC

1. Phân chia rõ ràng trách nhiệm từng phần.

- **Mô tả**: Đây là nguyên tắc cốt lõi và là nền tảng của MVC. Nó đảm bảo mỗi thành phần (Model, View, Controller) chỉ tập trung vào một nhiệm vụ cụ thể và không can thiệp vào trách nhiệm của các thành phần khác.

Giải thích chi tiết:

- **Model**: Chịu trách nhiệm hoàn toàn về **dữ liệu** và **logic nghiệp vụ** (Business Logic). Nó xử lý việc truy xuất, lưu trữ, cập nhật dữ liệu từ cơ sở dữ liệu (ví dụ: thông tin sản phẩm nội thất, đơn hàng, khách hàng), thực hiện các phép tính, kiểm tra ràng buộc kinh doanh (ví dụ: kiểm tra số lượng tồn kho trước khi bán). Model không quan tâm đến việc dữ liệu sẽ được hiển thị như thế nào hay người dùng tương tác ra sao.

- **View:** Chỉ có trách nhiệm về **giao diện người dùng (User Interface)**. Nó hiển thị dữ liệu nhận được từ Controller (mà Controller lấy từ Model) dưới dạng mà người dùng có thể nhìn thấy và tương tác (HTML, CSS, JavaScript). View không chứa logic nghiệp vụ phức tạp và không trực tiếp thao tác với cơ sở dữ liệu.
- **Controller:** Đóng vai trò là **trung gian điều phối (Mediator)**. Nó nhận các yêu cầu từ người dùng (thông qua trình duyệt), phân tích yêu cầu đó, gọi các phương thức tương ứng trong Model để xử lý nghiệp vụ, và sau đó chọn View phù hợp để hiển thị kết quả cho người dùng. Controller không chứa logic nghiệp vụ nặng nề hay định dạng giao diện.

2. Dễ mở rộng, bảo trì và phát triển.

- **Mô tả:** Nhờ vào sự phân chia trách nhiệm rõ ràng, ứng dụng MVC trở nên linh hoạt hơn, dễ dàng thêm tính năng mới, sửa lỗi hoặc cải tiến mà không làm ảnh hưởng lớn đến cấu trúc hiện có.

Giải thích chi tiết:

Dễ mở rộng (Extend):

- **Thêm tính năng mới:** Khi bạn muốn thêm một tính năng mới (ví dụ: quản lý đánh giá sản phẩm, tích hợp cổng thanh toán mới, hệ thống điểm thưởng), bạn có thể thêm các Model, Controller và View mới cho tính năng đó mà không cần phải thay đổi toàn bộ ứng dụng
- **Hỗ trợ đa nền tảng/đa giao diện:** Cùng một Model (logic nghiệp vụ) có thể được sử dụng để xây dựng nhiều loại giao diện khác nhau (web app, mobile app, desktop app) hoặc các API. Ví dụ, logic xử lý giỏ hàng của website bán hàng nội thất có thể được tái sử dụng nếu sau này bạn phát triển ứng dụng di động

Dễ bảo trì (Maintain):

- **Sửa lỗi nhanh chóng:** Khi có lỗi, việc xác định vị trí lỗi trở nên dễ dàng hơn rất nhiều. Nếu lỗi liên quan đến việc hiển thị, bạn biết phải tìm trong View. Nếu lỗi liên quan đến tính toán giá hay truy xuất dữ liệu, bạn sẽ tìm trong Model
- **Cập nhật đơn giản:** Nếu có thay đổi về yêu cầu (ví dụ: thêm trường "chất liệu" cho sản phẩm nội thất, thay đổi thuật toán gợi ý sản phẩm), bạn chỉ cần sửa đổi ở thành phần liên quan (Model cho dữ liệu, View cho hiển thị, Controller cho logic điều phối) mà không cần phải chạm vào các phần không liên quan.

Dễ phát triển (Develop):

- **Phát triển song song:** Các nhóm hoặc cá nhân có thể làm việc đồng thời trên các thành phần khác nhau của ứng dụng mà không gây cản trở lẫn nhau. Ví dụ, một lập trình viên có thể xây dựng các Model cho sản phẩm và đơn hàng, trong khi một lập trình viên khác thiết kế giao diện View và một người thứ ba viết Controller để kết nối chúng.
 - **Kiểm thử đơn vị (Unit Testing) dễ dàng:** Việc tách biệt giúp dễ dàng viết các bài kiểm thử tự động cho từng thành phần (đặc biệt là Model và Controller), đảm bảo từng phần hoạt động đúng đắn trước khi tích hợp, giảm thiểu lỗi khi kết hợp.
3. tính tái sử dụng mã nguồn.
- **Mô tả:** Các thành phần trong MVC được thiết kế để có thể được sử dụng lại ở nhiều nơi khác nhau trong cùng một ứng dụng hoặc thậm chí trong các dự án khác, giảm thiểu việc viết lại mã.

Giải thích chi tiết:

- **Model:** Là thành phần có tính tái sử dụng cao nhất. Logic nghiệp vụ và cấu trúc dữ liệu được định nghĩa trong Model có thể được sử dụng bởi nhiều Controller và nhiều View khác nhau.
- **Controller:** Một Controller có thể phục vụ nhiều View khác nhau tùy thuộc vào ngữ cảnh hoặc kết quả xử lý. Ví dụ, ProductController có thể có action Index() trả về View danh sách sản phẩm, và một action Search() cũng có thể trả về cùng một View danh sách đó nhưng với kết quả đã lọc.
- **View (partial views, layout):** Các phần giao diện dùng chung (ví dụ: header, footer, menu điều hướng, thành phần giỏ hàng nhỏ) có thể được tách ra thành các Partial View hoặc Layout và được nhúng vào nhiều View khác nhau, giúp duy trì sự nhất quán về giao diện.

Nguyên lý hoạt động:

Nguyên lý hoạt động của kiến trúc MVC có thể được hình dung như một vòng lặp luân phiên giữa ba thành phần chính (Model, View, Controller) để xử lý các yêu cầu từ người dùng và trả về phản hồi. Dưới đây là các bước chi tiết của quy trình này

1. Người dùng thiết kế giao diện trên View.
 - Đây là điểm khởi đầu của mọi hoạt động. Người dùng truy cập vào website bán hàng nội thất của bạn thông qua trình duyệt web.
 - Họ thực hiện một hành động nào đó trên giao diện người dùng, ví dụ:
 - Nhấp vào một liên kết (ví dụ: "Xem sản phẩm" hoặc "Ghé Sofa").

- Gửi một form (ví dụ: "Thêm vào giỏ hàng", "Tìm kiếm sản phẩm", "Đăng nhập").
 - Sử dụng nút bấm (ví dụ: "Thanh toán").
2. Controller chuyển dữ liệu đã xử lý từ model đến view để view có thể hiển thị: Sau khi Controller đã tương tác với Model và nhận được dữ liệu (hoặc kết quả xử lý nghiệp vụ) từ Model, nhiệm vụ tiếp theo của Controller là chuẩn bị dữ liệu này và chuyển nó tới View phù hợp để hiển thị cho người dùng. Đây là một khâu then chốt, đảm bảo View có đầy đủ thông tin để tạo ra giao diện động.

1.2 Tổng quan về ASP.NET Core

Khái niệm:

ASP.NET Core là một nền tảng (framework) mã nguồn mở và đa nền tảng được Microsoft phát triển, dùng để xây dựng các ứng dụng dựa trên nền tảng đám mây, các ứng dụng kết nối internet (Internet-connected applications) như ứng dụng web, API, microservices, và các ứng dụng di động back-end. Nó là phiên bản kế nhiệm và được thiết kế lại hoàn toàn từ đầu của ASP.NET truyền thống.

Đặc điểm nổi bật:

1. Hiệu năng cao, nhẹ, linh hoạt.

- **Hiệu năng cao:** ASP.NET Core được xây dựng lại từ đầu để đạt được hiệu suất vượt trội. Nó sử dụng một kiến trúc tối ưu, ít tài nguyên hơn và có khả năng xử lý nhiều yêu cầu đồng thời hơn so với các phiên bản trước. Điều này rất quan trọng đối với một website bán hàng.

Áp dụng vào website nội thất: Khi hàng ngàn khách hàng cùng truy cập và duyệt sản phẩm, thêm vào giỏ hàng hoặc đặt mua, website của bạn sẽ phản hồi nhanh chóng, giảm thiểu thời gian chờ đợi và mang lại trải nghiệm mượt mà. Tốc độ tải trang nhanh cũng giúp cải thiện SEO và giữ chân khách hàng.

Nhẹ: ASP.NET Core có kiến trúc mô-đun (modular architecture). Điều này có nghĩa là bạn chỉ cần bao gồm các thư viện và thành phần mà ứng dụng của bạn thực sự cần, thay vì phải tải toàn bộ framework công kênh.

Áp dụng vào website nội thất: Ứng dụng của bạn sẽ có kích thước nhỏ hơn, khởi động nhanh hơn, tiêu thụ ít tài nguyên máy chủ hơn. Điều này giúp giảm chi phí triển khai và tối ưu hóa hiệu suất trên các môi trường đám mây hoặc máy chủ có tài nguyên hạn chế.

2. Hỗ trợ Dependency Injection.

- **Mô tả:** Dependency Injection là một kỹ thuật thiết kế phần mềm giúp quản lý các "phụ thuộc" (dependencies) giữa các thành phần khác nhau của ứng dụng. Thay vì một đối tượng tự tạo ra các đối tượng mà nó cần, các đối tượng đó sẽ được "tiêm" (inject) vào nó từ bên ngoài. ASP.NET Core có một bộ chứa DI (Dependency Injection Container) mạnh mẽ được tích hợp sẵn ngay từ lõi của framework.

Giải thích:

Khi Controller cần tương tác với Model để lấy dữ liệu sản phẩm, thay vì Controller phải tự tạo ra một ProductService (dịch vụ xử lý logic sản phẩm), ASP.NET Core sẽ tự động cung cấp (tiêm vào) một thể hiện của ProductService cho Controller.

Áp dụng vào website nội thất:

- **Dễ kiểm thử:** Bạn có thể dễ dàng kiểm thử từng phần riêng biệt. Ví dụ, khi kiểm thử ProductController, bạn có thể "giả lập" (mock) ProductService thay vì phải kết nối thật với cơ sở dữ liệu, giúp việc kiểm thử nhanh chóng và đáng tin cậy hơn.
- **Mã nguồn sạch và dễ bảo trì:** Các thành phần ít phụ thuộc vào nhau một cách cứng nhắc, làm cho mã nguồn dễ đọc, dễ hiểu và dễ thay đổi khi có yêu cầu mới hoặc khi cần sửa lỗi. Ví dụ, nếu sau này bạn muốn thay đổi cách lưu trữ sản phẩm (từ SQL sang NoSQL), bạn chỉ cần thay đổi phần implementation của ProductService mà không ảnh hưởng đến Controller.
- **Khả năng mở rộng:** Dễ dàng thay thế các implementation của dịch vụ mà không ảnh hưởng đến các thành phần sử dụng chúng, tạo điều kiện cho việc mở rộng tính năng linh hoạt.

3. Hỗ trợ Razor Pages, MVC, API.

Mô tả: ASP.NET Core không chỉ bó hẹp trong một kiến trúc duy nhất mà cung cấp nhiều mô hình lập trình để phù hợp với các loại ứng dụng và phong cách phát triển khác nhau.

Giải thích và Áp dụng vào website nội thất:

- **MVC (Model-View-Controller):** Đây là mô hình chính mà bạn đang sử dụng cho website bán hàng nội thất. Nó rất phù hợp cho các ứng dụng web phức tạp với nhiều logic nghiệp vụ và yêu cầu về tính tổ chức cao. MVC giúp tách biệt dữ liệu (Model), giao diện (View) và logic điều khiển (Controller), đảm bảo dự án của bạn có cấu trúc rõ ràng, dễ bảo trì và mở rộng.
- **Razor Pages:** Là một cách tiếp cận đơn giản hóa để xây dựng các trang web tập trung vào trang (page-focused). Mỗi trang Razor Pages kết hợp mã C# và

HTML trong một tệp duy nhất. Nó thích hợp cho các trang web đơn giản, ít logic phức tạp hơn hoặc các trang quản trị cục bộ.

- **Áp dụng:** Bạn có thể sử dụng Razor Pages cho các trang như "Liên hệ", "Giới thiệu", hoặc các trang quản lý dữ liệu đơn giản trong khu vực quản trị của website nội thất (ví dụ: quản lý danh mục sản phẩm đơn giản) để giảm bớt sự phức tạp của Controller nếu không cần thiết.
- **Web API:** ASP.NET Core là một lựa chọn tuyệt vời để xây dựng các dịch vụ Web API (Application Programming Interface) theo phong cách RESTful. Các API này cung cấp dữ liệu (thường dưới dạng JSON hoặc XML) cho các ứng dụng khác.
- **Áp dụng:** Nếu sau này bạn muốn phát triển một ứng dụng di động cho website nội thất, hoặc tích hợp với các hệ thống bên thứ ba (ví dụ: đối tác vận chuyển, hệ thống CRM), bạn có thể xây dựng các Web API trong cùng dự án ASP.NET Core để cung cấp dữ liệu sản phẩm, đơn hàng, thông tin khách hàng một cách an toàn và hiệu quả.

1.3 Mô hình MVC trong ASP.NET Core

Trong ASP.NET Core, mô hình MVC được tích hợp đầy đủ với các thành phần:

- **Model:** Là các lớp POCO đại diện cho dữ liệu như Product, Order, OrderDetail.

```
namespace Duannoithat.Models
{
    30 references
    public class Product
    {
        19 references
        public int Id { get; set; }
        11 references
        public string? Name { get; set; }
        7 references
        public string? Detail { get; set; }
        10 references
        public string? ImageUrl { get; set; }
        12 references
        public decimal Price { get; set; }
        7 references
        public bool IsTrendingProduct { get; set; }
    }
}
```

Hình 1 Model Product

- View: Sử dụng Razor View Engine (.cshtml) để hiển thị giao diện HTML.

```
@model IEnumerable<Duannoithat.Models.Product>

<section data-bs-version="5.1" class="info3 cid-tsEY2vwh5a mbr-parallax-background"
id="info3-b">
    <div class="mbr-overlay"
style="opacity:0.9; background-image: url('https://ikay.vn/upload_images/images/Duyen/thi
</div>

    <div class="mbr-overlay" style="opacity: 0.7; background-color: #000000;"></div>
    <div class="container">
        <div class="row justify-content-center">
            <div class="card col-12 col-lg-10">
                <div class="card-wrapper">
                    <div class="card-box align-center">
                        <h4 class="card-title mbr-fonts-style align-center mb-4display-1">
                            <strong>Shop</strong>
                        </h4>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

<section data-bs-version="5.1" class="content2 cid-tsEZVFGbrL" id="content2-g">
    <div class="container">
        <div class="mbr-section-head">
            <h5 class="mbr-section-subtitle mbr-fonts-style align-center mb-0 mt-2display-5">
                The secret to a cozy home
            </h5>
        </div>
    </div>
</section>
```

Hình 2 View

- Controller: Là các lớp xử lý logic như ProductController, OrdersController.

```
using Duannoithat.Models.Interface;
using Microsoft.AspNetCore.Mvc;

namespace Duannoithat.Controllers
{
    1 reference
    public class ProductController : Controller
    {
        private IProductRepository productRepository;
        0 references
        public ProductController(IProductRepository productRepository)
        {
            this.productRepository = productRepository;
        }
        0 references
        public IActionResult Shop()
        {
            return View(productRepository.GetAllProducts());
        }
        0 references
        public IActionResult Index()
        {
            return View(productRepository.GetAllProducts());
        }
        0 references
        public IActionResult Detail(int id)
        {
            var product = productRepository.GetProductDetail(id);
            if (product != null)
            {
                return View(product);
            }
            return NotFound();
        }
    }
}
```

Hình 3 Controller

1.4 Kết luận chương

Mô hình MVC và công nghệ ASP.NET Core đóng vai trò nền tảng trong việc phát triển website nội thất. Việc áp dụng đúng nguyên lý MVC giúp website dễ bảo trì, tách biệt phần giao diện và logic. ASP.NET Core cung cấp môi trường mạnh mẽ để xây dựng ứng dụng web hiện đại, tối ưu trải nghiệm người dùng.

Chương 2 Xây dựng ứng dụng thực tế

2.1 Phát biểu bài toán, ý nghĩa thực tiễn

Mục tiêu hệ thống:

Website "Nội thất Nhà Xinh" là một ứng dụng bán hàng trực tuyến, nhằm cung cấp cho khách hàng một nền tảng để xem, chọn và đặt mua các sản phẩm nội thất chất lượng. Ý nghĩa thực tiễn của dự án là giúp doanh nghiệp:

- Tiếp cận khách hàng một cách nhanh chóng, mọi lúc, mọi nơi.
- Tăng doanh số thông qua nền tảng trực tuyến.
- Giảm chi phí vận hành cửa hàng truyền thống.

2.2 Phân tích yêu cầu ứng dụng

Chức năng chính:

1. Trang chủ (Homepage): Hiển thị thông tin giới thiệu về sản phẩm nổi bật.
 - Mục đích: Là điểm khởi đầu cho người dùng khi truy cập website, thu hút sự chú ý bằng các sản phẩm nổi bật và thông tin quan trọng.
 - Các yêu cầu chi tiết:
 - Banner/Carousel: Hiển thị các hình ảnh lớn, hấp dẫn về sản phẩm mới, chương trình khuyến mãi, hoặc bộ sưu tập nổi bật.
 - Sản phẩm nổi bật/Sản phẩm bán chạy: Liệt kê một số sản phẩm nội thất được chọn lọc hoặc đang có doanh số cao, kèm hình ảnh, tên, giá và nút "Xem chi tiết" hoặc "Thêm nhanh vào giỏ hàng".
 - Giới thiệu cửa hàng/thương hiệu: Một đoạn văn ngắn gọn về tầm nhìn, sứ mệnh, hoặc điểm mạnh của cửa hàng nội thất.
 - Danh mục phổ biến: Hiển thị các danh mục sản phẩm chính (ví dụ: Ghế, Bàn, Tủ, Đèn) với hình ảnh minh họa để người dùng dễ dàng điều hướng.
 - Review/Feedback khách hàng: Có thể có một mục nhỏ hiển thị các đánh giá tích cực từ khách hàng.
 - Footer: Chứa thông tin liên hệ, chính sách, bản quyền, liên kết mạng xã hội.
2. Trang Shop (Product Listing Page): Hiển thị tất cả sản phẩm.
 - Mục đích: Cung cấp cho người dùng cái nhìn tổng quan về tất cả các sản phẩm nội thất có sẵn và cho phép họ tìm kiếm, lọc, sắp xếp dễ dàng.
 - Các yêu cầu chi tiết:

- Hiển thị danh sách sản phẩm: Liệt kê tất cả các sản phẩm nội thất dưới dạng lưới (grid) hoặc danh sách, mỗi mục bao gồm: hình ảnh, tên, giá, mô tả ngắn.
 - Phân trang: Xử lý trường hợp có quá nhiều sản phẩm, chia thành nhiều trang để tải nhanh hơn và dễ duyệt.
 - Bộ lọc (Filter): Cho phép người dùng lọc sản phẩm theo các tiêu chí như:
 - Danh mục: Ghế sofa, bàn ăn, tủ quần áo, giường, v.v.
 - Giá: Theo khoảng giá (dưới X, từ X đến Y, trên Z).
 - Chất liệu: Gỗ, da, vải, kim loại, v.v.
 - Kích thước: Theo các tùy chọn kích thước phổ biến.
 - Màu sắc: Theo các lựa chọn màu sắc có sẵn.
 - Sắp xếp (Sort): Cho phép người dùng sắp xếp danh sách sản phẩm theo:
 - Giá (tăng dần/giảm dần).
 - Tên sản phẩm (A-Z, Z-A).
 - Sản phẩm mới nhất/cũ nhất.
 - Sản phẩm bán chạy nhất (nếu có dữ liệu).
 - Tìm kiếm (Search): Thanh tìm kiếm cho phép người dùng nhập từ khóa để tìm kiếm sản phẩm theo tên hoặc mô tả.
 - Thông tin chi tiết sản phẩm (Product Detail Page - *chức năng con*): Khi nhấp vào một sản phẩm trong trang Shop, người dùng sẽ được đưa đến một trang riêng hiển thị:
 - Hình ảnh lớn (có thể có gallery ảnh, zoom).
 - Tên sản phẩm, giá.
 - Mô tả chi tiết, thông số kỹ thuật (kích thước, chất liệu, bảo hành).
 - Số lượng tồn kho (hoặc trạng thái "Hết hàng").
 - Nút "Thêm vào giỏ hàng" và lựa chọn số lượng.
 - Các sản phẩm liên quan/gợi ý.
3. Giỏ hàng (Shopping Cart): Hiển thị sản phẩm đã chọn, cho phép cập nhật và xóa.
- Mục đích: Lưu trữ tạm thời các sản phẩm mà người dùng muốn mua trước khi tiến hành thanh toán.
 - Các yêu cầu chi tiết:
 - Danh sách sản phẩm trong giỏ: Hiển thị tên sản phẩm, hình ảnh thu nhỏ, đơn giá, số lượng, và tổng tiền cho mỗi mục.
 - Cập nhật số lượng: Cho phép người dùng tăng/giảm số lượng sản phẩm trong giỏ hàng.
 - Xóa sản phẩm: Cho phép người dùng loại bỏ một sản phẩm khỏi giỏ hàng.
 - Tổng số tiền giỏ hàng: Hiển thị tổng cộng số tiền cần thanh toán cho tất cả các sản phẩm trong giỏ.
 - Nút "Tiếp tục mua sắm" / "Tiến hành thanh toán": Điều hướng người dùng đến các trang tương ứng.
 - Lưu giỏ hàng: Giỏ hàng cần được lưu trữ (ví dụ: trong session hoặc cookie) để không bị mất khi người dùng điều hướng sang trang khác hoặc đóng trình duyệt (trong một khoảng thời gian nhất định).
4. Đặt hàng (Checkout): Cho phép người dùng nhập thông tin và đặt mua.

- Mục đích: Hoàn tất quá trình mua sắm, thu thập thông tin cần thiết để xử lý và giao hàng.
- Các yêu cầu chi tiết:
 - Thông tin người nhận: Form cho phép người dùng nhập tên, địa chỉ giao hàng, số điện thoại, email.
 - Phương thức thanh toán: Lựa chọn các phương thức thanh toán (ví dụ: Thanh toán khi nhận hàng - COD, Chuyển khoản ngân hàng, Thẻ tín dụng/Cổng thanh toán điện tử - nếu tích hợp).
 - Tóm tắt đơn hàng: Hiện thị lại danh sách sản phẩm, số lượng, giá và tổng tiền của đơn hàng để người dùng kiểm tra lại.
 - Xác nhận đặt hàng: Nút hoặc quy trình xác nhận để gửi đơn hàng.
 - Giảm số lượng tồn kho: Sau khi đặt hàng thành công, số lượng tồn kho của sản phẩm phải được cập nhật giảm đi trong hệ thống.
 - Trang xác nhận đơn hàng: Sau khi đặt hàng thành công, hiện thị trang xác nhận với mã đơn hàng và thông tin tóm tắt.

5. Quản lý đơn hàng (Order Management for Users): Người dùng xem lại đơn hàng đã đặt.

- Mục đích: Cho phép người dùng theo dõi lịch sử mua sắm và tình trạng các đơn hàng của họ.
- Các yêu cầu chi tiết:
 - Danh sách đơn hàng: Hiện thị tất cả các đơn hàng mà người dùng đã đặt, bao gồm: Mã đơn hàng, ngày đặt, tổng tiền, trạng thái hiện tại (ví dụ: "Đang xử lý", "Đang giao hàng", "Đã hoàn thành", "Đã hủy").
 - Xem chi tiết đơn hàng: Khi nhấp vào một đơn hàng, hiện thị chi tiết các sản phẩm trong đơn hàng (tên, số lượng, giá), địa chỉ giao hàng, phương thức thanh toán.
 - Lịch sử trạng thái: (Nâng cao) Có thể hiện thị lịch sử thay đổi trạng thái của đơn hàng.
 - Chỉ dành cho người dùng đã đăng nhập: Chức năng này yêu cầu người dùng phải đăng nhập vào tài khoản của họ.

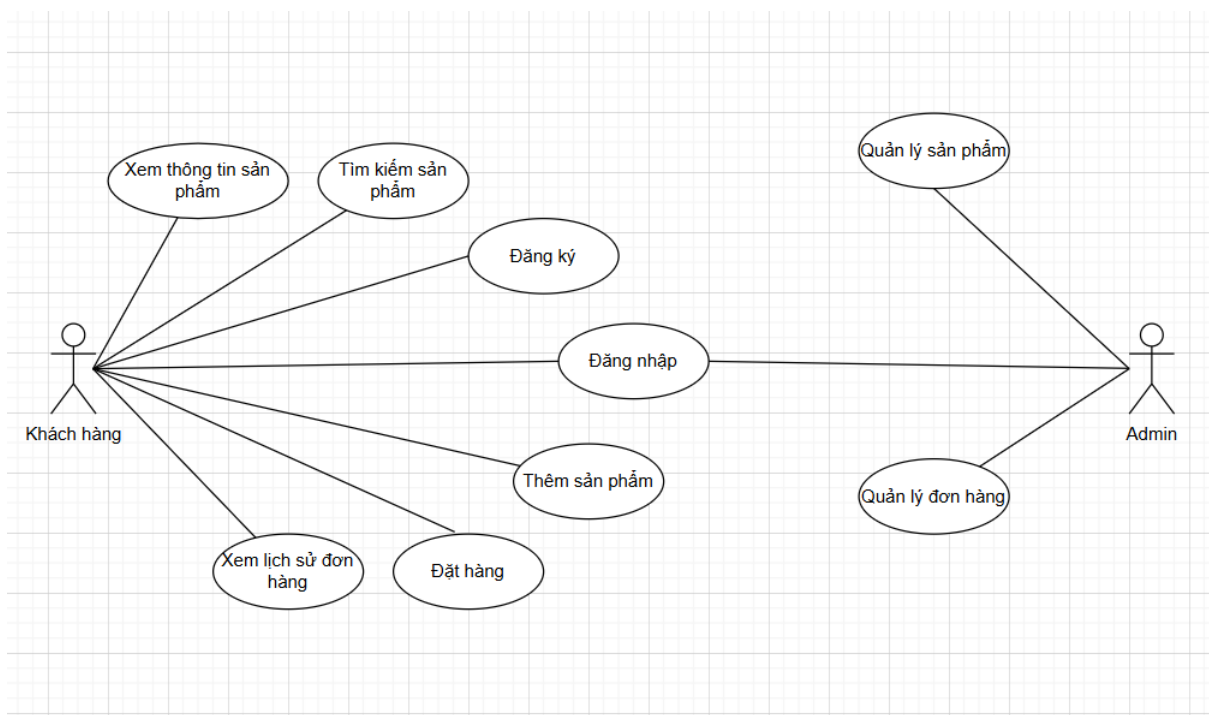
Đăng ký, Đăng nhập (Registration, Login): Quản lý tài khoản người dùng.

- Mục đích: Cho phép người dùng tạo tài khoản, đăng nhập để truy cập các tính năng cá nhân hóa và quản lý thông tin của họ.
- Các yêu cầu chi tiết:
 - Đăng ký: Form đăng ký tài khoản mới (email, mật khẩu, xác nhận mật khẩu, tên, số điện thoại). Kiểm tra tính hợp lệ của dữ liệu nhập (email đúng định dạng, mật khẩu đủ mạnh).
 - Đăng nhập: Form đăng nhập (email/tên đăng nhập, mật khẩu).
 - Quên mật khẩu: Chức năng phục hồi mật khẩu (thường qua email).
 - Đăng xuất: Thoát khỏi phiên làm việc.
 - Thông tin cá nhân: (Nâng cao) Người dùng có thể xem và chỉnh sửa thông tin cá nhân của mình (địa chỉ, số điện thoại).
 - Quản lý địa chỉ: (Nâng cao) Lưu trữ nhiều địa chỉ giao hàng.

Quản lý sản phẩm (Product Management - Admin Panel): Admin được quyền thêm, sửa, xóa sản phẩm nội thất (sắp tới).

- Mục đích: Cung cấp cho quản trị viên (Admin) quyền kiểm soát hoàn toàn đối với danh mục sản phẩm của cửa hàng. Đây là một chức năng quan trọng cho phép doanh nghiệp vận hành website.
- Các yêu cầu chi tiết:
 - Đăng nhập Admin: Khu vực riêng biệt yêu cầu quyền truy cập đặc biệt (phân quyền).
 - Danh sách sản phẩm (Admin): Hiển thị tất cả sản phẩm với các thông tin chi tiết hơn (ví dụ: tồn kho thực tế, trạng thái hoạt động).
 - Thêm sản phẩm mới: Form cho phép Admin nhập đầy đủ thông tin của một sản phẩm mới (tên, mô tả, giá, số lượng tồn kho, danh mục, hình ảnh, v.v.).
 - Sửa sản phẩm: Cho phép Admin chỉnh sửa thông tin của một sản phẩm hiện có.
 - Xóa sản phẩm: Cho phép Admin xóa một sản phẩm khỏi hệ thống (có thể kèm xác nhận hoặc ẩn sản phẩm).
 - Quản lý hình ảnh sản phẩm: (Nâng cao) Cho phép thêm/xóa nhiều hình ảnh cho một sản phẩm.
 - Quản lý danh mục: (Nâng cao) Thêm, sửa, xóa các danh mục sản phẩm.

Sơ đồ use case



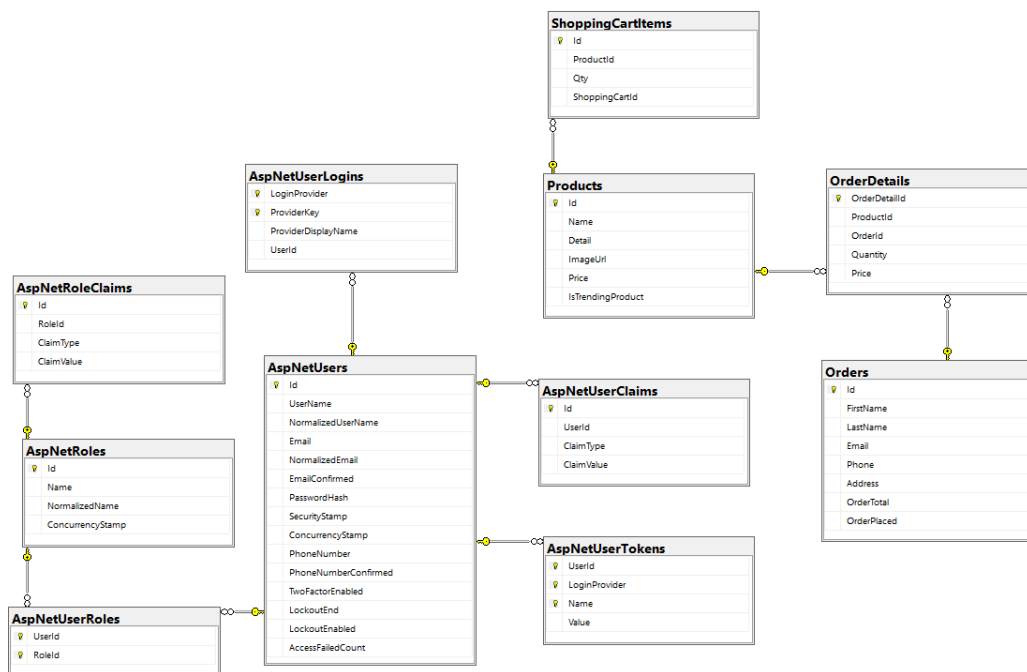
Hình 4 Use Case

2.3 Thiết kế cơ sở dữ liệu

Các bảng chính:

- Products: chứa thông tin sản phẩm.
- Orders: lưu đơn hàng.
- OrderDetails: chi tiết từng đơn hàng.
- ShoppingCartItems: tạm lưu sản phẩm trong giỏ hàng.
- AspNetUsers: bảng mặc định của ASP.NET Identity.

Sơ đồ database:



Hình 5 Database

2.4. Thiết kế giao diện người dùng

Công cụ thiết kế: Sử dụng ASP.NET Core Web App (Model-View-Controller) làm công cụ chính để phát triển và thiết kế giao diện.

Các trang giao diện chính:

- Trang chủ: hiển thị sản phẩm nổi bật, điều hướng các trang chức năng như Shop, Đăng nhập.
- Trang Shop: danh sách sản phẩm với ảnh, tên, giá và nút “Thêm vào giỏ hàng”.

- Trang giỏ hàng: hiển thị các sản phẩm đã chọn, có thể cập nhật số lượng hoặc xoá.
- Trang đặt hàng: form nhập thông tin người mua (họ tên, email, địa chỉ...).
- Trang đăng ký/đăng nhập: sử dụng giao diện mặc định của ASP.NET Identity, có thể tùy chỉnh CSS theo phong cách website nội thất.

2.5. Thiết kế thành phần MVC

Kiến trúc Model-View-Controller (MVC) là một mô hình thiết kế phần mềm mạnh mẽ giúp tách biệt ứng dụng thành ba thành phần chính, mỗi thành phần có một vai trò và trách nhiệm riêng biệt. Trong dự án website bán hàng nội thất này, chúng tôi áp dụng MVC để đảm bảo tính module hóa, dễ bảo trì và khả năng mở rộng.

Model:

Trong kiến trúc MVC, Model là trái tim của dữ liệu ứng dụng. Nó bao gồm các lớp C# đơn giản, được gọi là POCO (Plain Old CLR Objects), dùng để đại diện cho các thực thể (entities) trong cơ sở dữ liệu và các cấu trúc dữ liệu cần thiết cho logic nghiệp vụ. Các lớp này chủ yếu chứa các thuộc tính (properties) để lưu trữ thông tin.

- **Product (Sản phẩm):**
 - **Mô tả:** Lớp này định nghĩa cấu trúc dữ liệu cho mỗi mặt hàng nội thất có sẵn trên website. Đây là nơi lưu trữ tất cả thông tin liên quan đến một sản phẩm.

```
namespace Duannoithat.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string? Name { get; set; }
        public string? Detail { get; set; }
        public string? ImageUrl { get; set; }
        public decimal Price { get; set; }
        public bool IsTrendingProduct { get; set; }
    }
}
```

- **Order (Đơn hàng):**
 - **Mô tả:** Lớp này biểu diễn một giao dịch mua hàng hoàn chỉnh của một khách hàng trên website.

```
namespace Duannoithat.Models
{
    public class Order
    {
        public int Id { get; set; }
        public string? FirstName { get; set; }
        public string? LastName { get; set; }
        public string? Email { get; set; }
    }
}
```

```

        public string? Phone { get; set; }
        public string? Address { get; set; }
        public decimal OrderTotal { get; set; }
        public DateTime OrderPlaced { get; set; }
        public List<OrderDetail>? OrderDetails { get; set; }
    }
}

```

- **OrderDetail (Chi tiết Đơn hàng):**

- **Mô tả:** Lớp này đại diện cho từng mặt hàng riêng lẻ được mua trong một đơn hàng cụ thể. Một đơn hàng (Order) có thể chứa nhiều OrderDetail.

```

namespace Duannoithat.Models
{
    public class OrderDetail
    {
        public int OrderDetailId { get; set; }
        public int ProductId { get; set; }
        public Product? Product { get; set; }
        public int OrderId { get; set; }
        public Order? Order { get; set; }
        public int Quantity { get; set; }
        public decimal Price { get; set; }
    }
}

```

- **ShoppingCartItem (Mục Giỏ hàng):**

- **Mô tả:** Lớp này biểu diễn một sản phẩm đang nằm trong giỏ hàng tạm thời của người dùng trước khi họ quyết định đặt mua.

```

namespace Duannoithat.Models
{
    public class ShoppingCartItem
    {
        public int Id { get; set; }
        public Product? Product { get; set; }
        public int Qty { get; set; }
        public string? ShoppingCartId { get; set; }
    }
}

```

View:

View là thành phần chịu trách nhiệm về giao diện người dùng. Nó hiển thị dữ liệu đã được Controller chuẩn bị (lấy từ Model) và là nơi người dùng tương tác trực tiếp với ứng dụng. Trong ASP.NET Core, **Razor View Engine** là công nghệ mặc định để tạo các View, sử dụng cú pháp .cshtml để kết hợp HTML với mã C# một cách linh hoạt.

1. Views/Product/Shop.cshtml:

- **Mô tả:** Đây là trang chính để người dùng duyệt xem và khám phá các sản phẩm nội thất. Nó đóng vai trò như "trang cửa hàng" hoặc "trang danh mục sản phẩm".

```
@model IEnumerable<Duannoithat.Models.Product>

<section data-bs-version="5.1" class="info3 cid-tsEY2vwh5a mbr-parallax-
background"
    id="info3-b">

    <div class="mbr-overlay"
        style="opacity:0.9; background-image:
url('https://ikay.vn/upload_images/images/Duyen/thiet-ke-showroom-noi-that-
cao-cap-tai-vung-tau-2.jpeg');">
    </div>

    <div class="mbr-overlay" style="opacity: 0.7; background-color: rgb(0,
0,0);"></div>
    <div class="container">
        <div class="row justify-content-center">
            <div class="card col-12 col-lg-10">
                <div class="card-wrapper">
                    <div class="card-box align-center">
                        <h4 class="card-title mbr-fonts-style align-center
mb-4display-1">
                            <strong>Shop</strong>
                        </h4>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

<section data-bs-version="5.1" class="content2 cid-tsEZVFGbrL" id="content2-
g">
    <div class="container">
        <div class="mbr-section-head">

            <h5 class="mbr-section-subtitle mbr-fonts-style align-center mb-
0 mt-2display-5">
                The secret to a cozy home
            </h5>
        </div>
        <partial name="_Product" model="@Model" />
    </div>
</section>
```

2. Views/ShoppingCart/Index.cshtml:

- **Mô tả:** Trang này là nơi khách hàng xem lại các sản phẩm đã được thêm vào giỏ hàng của họ.

```
@using System.Globalization
@model IEnumerable<Duannoithat.Models.ShoppingCartItem>

@if (Model.Count() != 0)
```

```

{
<section class="h-100 h-custom">
    <div class="container py-5 h-100">
        <div class="row d-flex justify-content-center align-items-center h-100">
            <div class="col-12">
                <div class="card card-registration card-registration-2"
                    style="border-radius: 15px; background-color: #fafafa">
                    <div class="card-body p-0">
                        <div class="row g-0">
                            <div class="col-lg-7">
                                <div class="p-5">
                                    <div class="d-flex justify-content-between align-items-center mb-5">
                                        <h2 class="fw-bold mb-0 text-black">Shopping Cart</h2>
                                        <h6 class="mb-0 text-muted">3 items</h6>
                                    </div>
                                    <hr class="my-4">
                                    @foreach (var item in Model)
                                    {
                                        <div class="row mb-4 d-flex justify-content-between align-items-center">
                                            <div class="col-md-2 col-lg-2 col-xl-2">
                                                
                                            </div>
                                            <div class="col-md-3 col-lg-3 col-xl-3">
                                                <h6 class="text-black mb-0">@item.Qty x @item.Product.Name</h6>
                                            </div>
                                            <div class="col-md-3 col-lg-2 col-xl-2 offset-lg-1">
                                                <h6 class="mb-0">@((item.Qty * item.Product.Price).ToString("N0", new CultureInfo("vi-VN"))) + "$"</h6>
                                            </div>
                                            <div class="col-md-1 col-lg-1 col-xl-1 text-end">
                                                <a asp-controller="ShoppingCart" asp-action="RemoveFromShoppingCart"
                                                    asp-route-pId="@item.Product.Id"
                                                    class="text-decoration-none"><h3>#128465;</h3></a>
                                            </div>
                                        </div>
                                    }
                                <hr class="my-4">
                                <div class="pt-5">
                                    <h6 class="mb-0">
                                        <a asp-controller="Product" asp-action="Shop"
                                            class="text-body">
                                            <i class="fas fa-long-arrow-altleft me-2"></i>Back to shop
                                        </a>
                                    </h6>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
}

```

```
</div>
<div class="col-lg-5 bg-grey">
<div class="p-5">
<h4 class="fw-bold mb-5 mt-2 pt1">Summary</h4>
<hr class="my-4">
<div class="d-flex justify-content-between mb5">
<h5 class="text-uppercase">Total price</h5>
<h5>@(ViewBag.TotalCart.ToString("N0", new CultureInfo("vi-VN"))) + "$")</h5>
</div>
<a asp-controller="Orders" asp-action="Checkout">
<button type="button" class="btn btn-primary w-100 text-center" data-mdb-ripple-color="dark">Proceed</button>
</a>
</div>
</div>
</div>
</div>
</div>
</section>
}
else
{
<br />
<br />
<div class="container">
<br><br><br><br>
<div class="text-center">

<br>
<h1><strong>Your cart is empty</strong></h1>
<br>
<p>Before proceed to checkout you must add some products to your shopping cart. You will find a lot of interesting products on our "Shop" page.</p>
<a asp-controller="Product" asp-action="Shop" class="btn btn-primary w-20">Start Shopping</a>
</div>
<br />
<br />
```

```

<br />
<br />
<div class="col-lg-5 bg-grey">
    <div class="p-5">
        <h4 class="fw-bold mb-5 mt-2 pt-1">Summary</h4>
        <hr class="my-4">
        <div class="d-flex justify-content-between mb-5">
            <h5 class="text-uppercase">Total price</h5>
            <h5>@(ViewBag.TotalCart.ToString("N0", new
                CultureInfo("vi-VN"))) + "$"</h5>
        </div>
        <a asp-controller="Orders" asp-action="Checkout">
            <button type="button"
                class="btn btn-primary w-100 text-
                center"
                data-mdb-ripple-color="dark">
                Proceed
            </button>
        </a>
    </div>
</div>
}

```

3. Views/Orders/List.cshtml:

- **Mô tả:** Trang này cung cấp cho người dùng một cái nhìn tổng quan về lịch sử các đơn hàng mà họ đã đặt.

```

@model IEnumerable<Duannoithat.Models.Order>

<h2 class="mb-4">Danh sách đơn hàng của bạn</h2>

@if (!Model.Any())
{
    <div class="alert alert-info">Bạn chưa có đơn hàng nào.</div>
}
else
{
    @foreach (var order in Model)
    {
        <div class="border rounded p-4 mb-4 shadow-sm">
            <h5 class="text-primary">📄 Đơn hàng #@order.Id - <small>Ngày
đặt: @order.OrderPlaced.ToString("dd/MM/yyyy")</small></h5>
            <p><strong>👤 Tên khách:</strong> @order.FirstName
@order.LastName</p>
            <p><strong>✉ Email:</strong> @order.Email</p>
            <p><strong>☎ Số điện thoại:</strong> @order.Phone</p>
            <p><strong>🏠 Địa chỉ:</strong> @order.Address</p>
            <p><strong>💰 Tổng tiền:</strong>
@order.OrderTotal.ToString("#,0$ ", new System.Globalization.CultureInfo("en-
US"))</p>

            @if (order.OrderDetails != null && order.OrderDetails.Any())
            {
                <table class="table table-striped mt-3">
                    <thead>
                        <tr>
                            <th>Sản phẩm</th>

```

```

                <th>Số lượng</th>
                <th>Giá</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in order.OrderDetails)
            {
                <tr>
                    <td>@item.Product?.Name</td>
                    <td>@item.Quantity</td>
                    <td>@item.Price.ToString("#,0$", new
System.Globalization.CultureInfo("en-US"))</td>
                </tr>
            }
        </tbody>
    </table>
}
else
{
    <div class="text-muted">Không có sản phẩm nào trong đơn hàng
    này.</div>
}
</div>
}
}

```

Controller:

Controller là trung tâm điều khiển trong kiến trúc MVC. Nó nhận các yêu cầu HTTP từ trình duyệt (do người dùng tương tác với View), xử lý các yêu cầu đó bằng cách tương tác với Model (để lấy hoặc cập nhật dữ liệu), và cuối cùng quyết định View nào sẽ được hiển thị để trả lời người dùng.

1. ProductController:

- **Trách nhiệm chính:** Xử lý tất cả các yêu cầu liên quan đến việc hiển thị và tương tác với sản phẩm trên website.
- **Các Action Method tiêu biểu:**
 - Shop(): Action này sẽ được gọi khi người dùng truy cập trang sản phẩm chính (ví dụ: /Product/Shop). Nó sẽ làm việc với một lớp dịch vụ (ví dụ: ProductService) để lấy danh sách các đối tượng Product từ cơ sở dữ liệu, sau đó truyền danh sách này đến View Views/Product/Shop.cshtml để hiển thị.
 - Detail(int id): Action này xử lý yêu cầu xem chi tiết một sản phẩm cụ thể (ví dụ: /Product/Detail/1). Nó sẽ lấy id của sản phẩm, yêu cầu ProductService tìm sản phẩm tương ứng và trả về đối tượng Product, sau đó truyền đối tượng này đến View chi tiết sản phẩm (Views/Product/Detail.cshtml - không nằm trong danh sách bạn đưa nhưng là một View phụ trợ quan trọng).
 - Search(string query): Xử lý yêu cầu tìm kiếm sản phẩm dựa trên từ khóa.

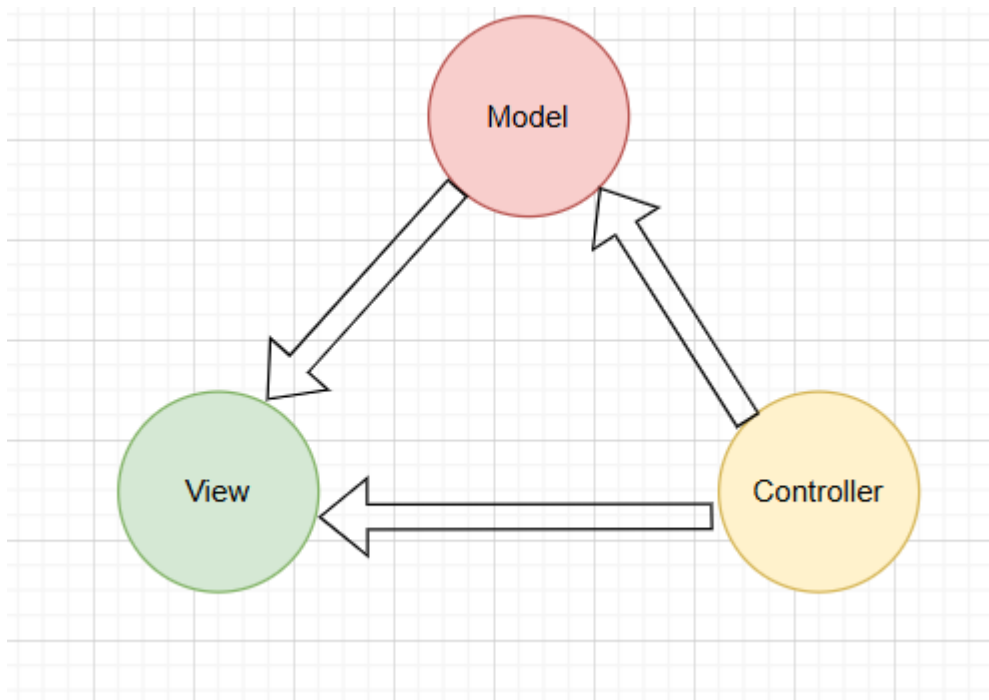
2. ShoppingCartController:

- **Trách nhiệm chính:** Quản lý các thao tác liên quan đến giỏ hàng của người dùng.
- **Các Action Method tiêu biểu:**
 - Index(): Action này hiển thị trang giỏ hàng (ví dụ: /ShoppingCart/Index). Nó sẽ lấy các ShoppingCartItem của người dùng hiện tại (thường từ session hoặc database tạm thời) và truyền chúng đến View Views/ShoppingCart/Index.cshtml.
 - AddToCart(int productId, int quantity): Xử lý yêu cầu khi người dùng nhấp "Thêm vào giỏ hàng". Action này sẽ nhận productId và quantity, sau đó sử dụng một dịch vụ quản lý giỏ hàng để thêm hoặc cập nhật số lượng của ShoppingCartItem trong giỏ. Nó có thể chuyển hướng người dùng hoặc trả về kết quả JSON cho AJAX.
 - UpdateQuantity(int itemId, int newQuantity): Xử lý việc cập nhật số lượng của một mục trong giỏ.
 - RemoveFromCart(int itemId): Xử lý việc xóa một mục khỏi giỏ hàng.

3. OrdersController:

- **Trách nhiệm chính:** Quản lý quy trình đặt hàng và cho phép người dùng xem lại các đơn hàng đã đặt.
- **Các Action Method tiêu biểu:**
 - Checkout(): Action này hiển thị trang thanh toán, nơi người dùng điền thông tin giao hàng và chọn phương thức thanh toán.
 - PlaceOrder(OrderViewModel model) (thường là POST): Xử lý việc tạo đơn hàng thực tế sau khi người dùng gửi form thanh toán. Nó sẽ lấy dữ liệu từ giỏ hàng, tạo đối tượng Order và các OrderDetail tương ứng, lưu vào cơ sở dữ liệu, và sau đó chuyển hướng người dùng đến một trang xác nhận.
 - List(): Action này hiển thị danh sách các đơn hàng mà người dùng đã đặt (ví dụ: /Orders/List). Nó sẽ gọi dịch vụ quản lý đơn hàng để lấy các đối tượng Order của người dùng hiện tại và truyền đến View Views/Orders/List.cshtml.
 - Detail(int orderId): (Nếu có) Hiển thị thông tin chi tiết của một đơn hàng cụ thể.

Sơ đồ:



Hình 6 MVC

2.6. Triển khai và cài đặt

Môi trường phát triển:

- Hệ điều hành: Windows 10
- IDE: Visual Studio 2022
- Framework: ASP.NET Core 6.0
- Cơ sở dữ liệu: SQL Server Management

Các thư viện/framework sử dụng:

- ASP.NET Core MVC
- Entity Framework Core
- ASP.NET Core Identity
- Bootstrap, Mobirise CSS

Các bước triển khai:

1. Khởi tạo project ASP.NET Core MVC.
2. Thiết lập DbContext và kết nối chuỗi cấu hình trong appsettings.json.
3. Tạo các model, repository và controller.
4. Cài đặt Razor View cho giao diện.
5. Chạy Migration và cập nhật CSDL.
6. Khởi tạo tài khoản Admin qua Program.cs.

7. Chạy ứng dụng, test tính năng.

Chương 3 Kết quả chương trình

3.1. Demo chức năng chính của hệ thống

Hệ thống “Website nội thất Nhà Xinh” được xây dựng với các chức năng chính sau:

1. Trang chủ – giới thiệu

- Giao diện thân thiện, hiển thị banner, sản phẩm nổi bật.
- Người dùng có thể điều hướng đến các phần khác như cửa hàng, liên hệ, giỏ hàng.

2. Trang cửa hàng (Shop)

- Hiển thị danh sách sản phẩm từ cơ sở dữ liệu.
- Sản phẩm gồm tên, ảnh, mô tả ngắn, giá và nút “Thêm vào giỏ hàng”.

3. Giỏ hàng (Shopping Cart)

- Cho phép người dùng xem các sản phẩm đã thêm.
- Có thể cập nhật số lượng hoặc xóa sản phẩm.

4. Đặt hàng (Checkout)

- Người dùng đăng nhập → điền thông tin đặt hàng → xác nhận đơn hàng.
- Thông tin gồm: họ tên, email, địa chỉ, số điện thoại và sản phẩm đã chọn.

5. Đăng nhập/Đăng ký

- Tích hợp ASP.NET Core Identity.
- Tự động tạo tài khoản Admin khi chạy ứng dụng lần đầu.

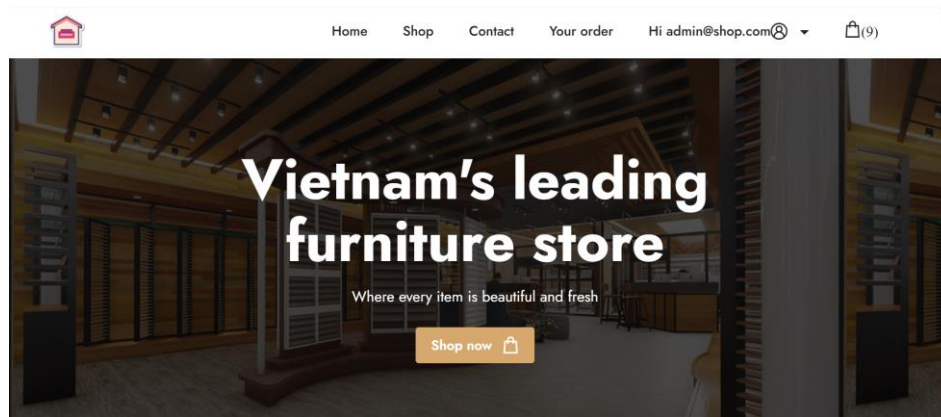
6. Quản lý đơn hàng (Your Order)

- Cho phép người dùng xem lại các đơn hàng đã đặt.
- Hiển thị chi tiết từng đơn gồm thông tin khách hàng, sản phẩm và tổng tiền.

3.2. Ảnh chụp màn hình chương trình hoạt động

1. Giao diện trang chủ

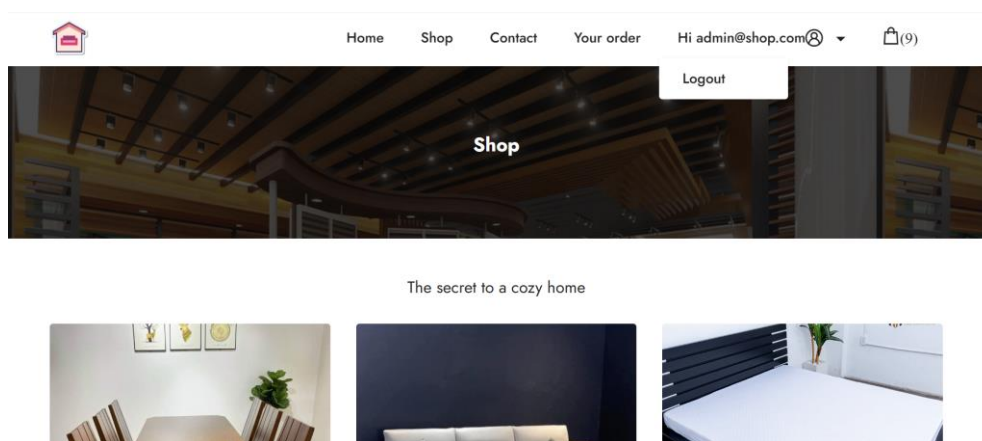
Hiển thị logo, điều hướng menu, banner sản phẩm nổi bật



Hình 7 Trang chủ

2. Giao diện Shop

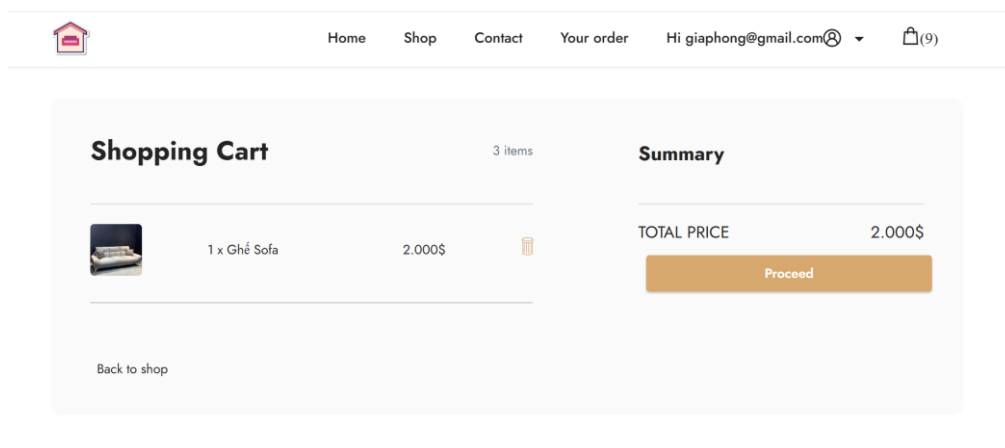
Các sản phẩm có ảnh, giá, nút thêm giỏ hàng



Hình 8 Shop

3. Giao diện giỏ hàng

Danh sách sản phẩm đã chọn, cập nhật/xóa số lượng



Hình 9 Giỏ hàng

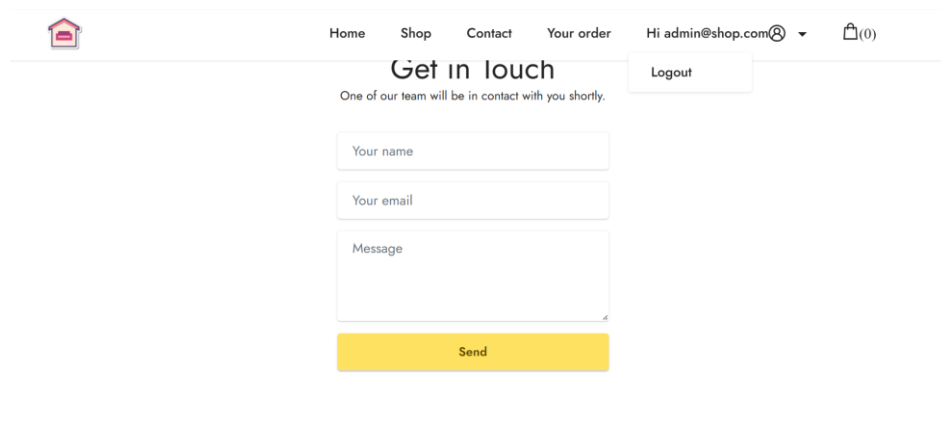
4. đặt hàng

Nhập thông tin giao hàng và xác nhận

Hình 10 Đặt hàng

5. Giao diện liên hệ

Nhập thông tin liên hệ

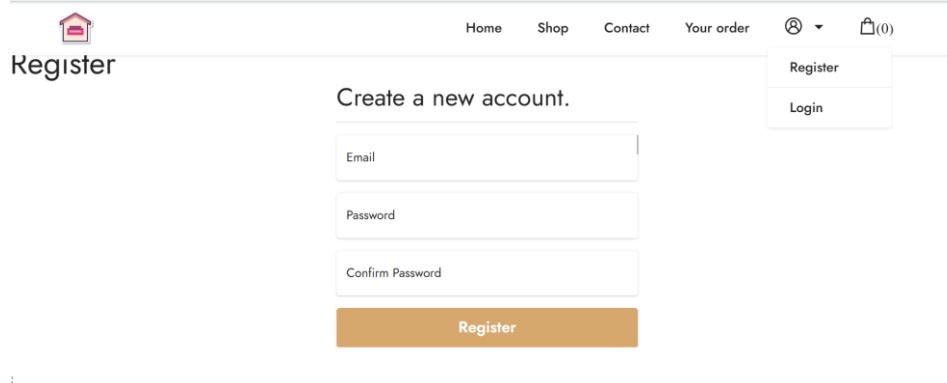


The screenshot shows a contact form on a website. At the top, there is a navigation bar with links: Home, Shop, Contact, Your order, and a user profile section showing 'Hi admin@shop.com' with a dropdown arrow and a shopping cart icon with '(0)'. Below the navigation bar, the form is titled 'Get in touch' with a subtext 'One of our team will be in contact with you shortly.' To the right of the title is a 'Logout' button. The form contains three input fields: 'Your name', 'Your email', and 'Message'. Below these fields is a yellow 'Send' button.

Hình 11 Liên hệ

6. Đăng nhập/đăng ký người dùng

Giao diện xác thực tài khoản



The screenshot shows a registration form on a website. At the top, there is a navigation bar with links: Home, Shop, Contact, Your order, and a user profile section showing a dropdown arrow and a shopping cart icon with '(0)'. Below the navigation bar, the form is titled 'Register'. To the right of the title is a dropdown menu with 'Register' and 'Login' options. The form contains three input fields: 'Email', 'Password', and 'Confirm Password'. Below these fields is an orange 'Register' button.

Hình 12 Đăng ký

Log in

Use a local account to log in.

Email
giaphong@gmail.com

Password

☐ Remember me?

Log in

[Forgot your password?](#)

[Register as a new user](#)

[Resend email confirmation](#)

Hình 13 Đăng nhập

7. Giao diện lịch sử đơn hàng

Xem danh sách các đơn hàng đã đặt, chi tiết từng đơn

Danh sách đơn hàng của bạn

Logout

Đơn hàng #6 - Ngày đặt: 30/06/2025

Tên khách: phong phong

Email: giaphong@gamil.com

Số điện thoại: 032666

Địa chỉ: fasfasfas

Tổng tiền: 2,000\$

Sản phẩm	Số lượng	Giá
Ghế Sofa	1	2,000\$

Đơn hàng #5 - Ngày đặt: 28/06/2025

Hình 14 Lịch sử đơn hàng

3.3. Nhận xét và đánh giá kết quả

Sau quá trình phát triển website bán hàng nội thất sử dụng nền tảng ASP.NET Core MVC, chúng tôi đã đạt được những kết quả nhất định, tuy nhiên cũng nhận thấy các điểm cần cải thiện và hướng phát triển tiếp theo.

1. Ưu điểm:

Giao diện rõ ràng, dễ sử dụng và thân thiện với người dùng:

- **Giải thích:** Website được thiết kế với bố cục trực quan, các thành phần được sắp xếp hợp lý, giúp người dùng dễ dàng tìm kiếm sản phẩm, xem thông tin và thực hiện các thao tác mua sắm. Màu sắc, font chữ và hình ảnh được lựa chọn cẩn thận để tạo ra một trải nghiệm thị giác dễ chịu, phù hợp với ngành nội thất. Quy trình từ việc xem sản phẩm đến thêm vào giỏ hàng và đặt mua được tối ưu hóa để đơn giản và nhanh chóng nhất có thể, giảm thiểu các bước phức tạp không cần thiết.
- **Ý nghĩa:** Điều này góp phần quan trọng vào việc nâng cao trải nghiệm khách hàng (UX), khuyến khích họ ở lại website lâu hơn, dễ dàng tìm thấy sản phẩm ưng ý và hoàn tất giao dịch, từ đó tăng tỷ lệ chuyển đổi mua hàng.

Hệ thống hoạt động ổn định trên nền ASP.NET Core:

- **Giải thích:** Nhờ việc xây dựng trên nền tảng ASP.NET Core - một framework hiện đại, hiệu suất cao và ổn định, website hoạt động mượt mà, ít gặp lỗi phát sinh trong quá trình vận hành. Các yêu cầu từ người dùng được xử lý nhanh chóng, đảm bảo tính liên tục của dịch vụ ngay cả khi có nhiều lượt truy cập cùng lúc.
- **Ý nghĩa:** Sự ổn định là yếu tố then chốt cho một website thương mại điện tử. Nó đảm bảo khách hàng luôn có thể truy cập và mua sắm mà không bị gián đoạn, xây dựng niềm tin vào dịch vụ và thương hiệu. Đồng thời, giúp giảm thiểu công sức và chi phí bảo trì hệ thống.

Tích hợp đầy đủ chức năng thương mại điện tử cơ bản: Sản phẩm – Giỏ hàng – Đặt hàng – Đơn hàng:

- **Giải thích:** Dự án đã thành công trong việc xây dựng các module cốt lõi của một hệ thống thương mại điện tử, bao gồm:
 - **Trang Shop:** Hiển thị danh sách sản phẩm, bộ lọc, tìm kiếm.
 - **Chi tiết sản phẩm:** Trình bày thông tin đầy đủ về từng món đồ nội thất.
 - **Giỏ hàng:** Cho phép người dùng quản lý các sản phẩm đã chọn (thêm, xóa, cập nhật số lượng).

- **Đặt hàng:** Cung cấp quy trình để người dùng nhập thông tin giao hàng và xác nhận mua.
- **Quản lý đơn hàng (phía người dùng):** Cho phép khách hàng xem lại lịch sử các đơn hàng đã đặt và theo dõi trạng thái cơ bản.
- **Ý nghĩa:** Các chức năng này tạo nên một chu trình mua sắm hoàn chỉnh và liền mạch cho người dùng, đáp ứng được nhu cầu cơ bản nhất của một website bán hàng trực tuyến, đặt nền móng vững chắc cho các tính năng nâng cao sau này.

Quản lý phiên đăng nhập hiệu quả nhờ ASP.NET Identity:

- **Giải thích:** Việc tích hợp ASP.NET Identity - một hệ thống quản lý người dùng mạnh mẽ và an toàn của Microsoft - đã giúp xử lý các nghiệp vụ liên quan đến tài khoản người dùng một cách hiệu quả. Điều này bao gồm các chức năng đăng ký, đăng nhập, phân quyền và quản lý phiên làm việc, đảm bảo thông tin cá nhân của người dùng được bảo mật và mỗi người dùng có thể truy cập các tính năng cá nhân hóa (như lịch sử đơn hàng) một cách an toàn.
- **Ý nghĩa:** Cung cấp trải nghiệm cá nhân hóa và bảo mật cho người dùng, khuyến khích họ tạo tài khoản và quay lại mua sắm. Việc sử dụng một framework có sẵn giúp tiết kiệm thời gian phát triển và đảm bảo tính bảo mật cao theo các tiêu chuẩn công nghiệp.
- **Hạn chế:**
 - Chưa có chức năng cập nhật trạng thái đơn hàng cho admin.
 - Chưa tích hợp thanh toán trực tuyến (chỉ đặt hàng thủ công).
 - Giao diện chưa được tối ưu cho thiết bị di động.
- **Hướng phát triển:**
 - Bổ sung phần quản trị Admin với dashboard quản lý sản phẩm, đơn hàng.
 - Tích hợp hệ thống thanh toán VNPay hoặc Momo.
 - Phát triển phiên bản mobile hoặc responsive hoàn chỉnh.

Kết luận

Qua quá trình thực hiện đề tài “*Thiết kế và xây dựng website nội thất Nhà Xinh bằng công nghệ ASP.NET Core MVC*”, em đã có cơ hội áp dụng kiến thức lý thuyết đã học vào thực tế. Từ bước phân tích yêu cầu, thiết kế cơ sở dữ liệu đến việc lập trình các chức năng chính và triển khai ứng dụng, đề tài đã giúp em củng cố kỹ năng lập trình web theo mô hình MVC một cách bài bản và hiệu quả.

Website đã hoàn thành các chức năng cơ bản như hiển thị danh sách sản phẩm, xem chi tiết, giỏ hàng, đặt hàng, cũng như đăng nhập, đăng ký người dùng. Giao diện được xây dựng thân thiện với người dùng, dễ sử dụng, phù hợp với đối tượng khách hàng phổ thông.

Tuy nhiên, do thời gian có hạn và kiến thức còn hạn chế, đề tài vẫn còn một số mặt chưa hoàn thiện. Chẳng hạn như chưa tích hợp chức năng thanh toán trực tuyến, chưa tối ưu giao diện cho các thiết bị di động và chưa xây dựng hệ thống quản lý nội dung riêng cho admin. Đây sẽ là những hướng phát triển tiếp theo nếu có điều kiện mở rộng đề tài trong tương lai. Em mong rằng sản phẩm sẽ là nền tảng để phát triển thêm nhiều tính năng nâng cao, phục vụ tốt hơn cho mục đích thương mại và học tập.

Tài liệu tham khảo

Thực hành Web Coffee Shop

Microsoft Learn. *ASP.NET Core MVC overview*.

Link: <https://learn.microsoft.com/en-us/aspnet/core/mvc/overview>

TutorialsTeacher. *MVC Architecture*.

Link: <https://www.tutorialsteacher.com/mvc/mvc-architecture>