TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC ASP.NET

ĐỀ TÀI:

“XÂY DỰNG WEBSITE QUẢN LÝ CỬA HÀNG BÁN LINH KIỆN ĐIỆN TỬ”

Sinh viên thực hiện:

3120410514 – Trực Gia Minh Thuận

3120410491 - Huỳnh Công Minh Thiện

Giảng viên hướng dẫn:

Từ Lãng Phiêu

TP. HCM, tháng 12/2024

MỤC LỤC

1. Phạm vi của đề tài

Phạm vi của đề tài "Website quản lý cửa hàng LINH KIỆN ĐIỆN TỬ" có thể được xác định như sau:

1. Quản lý sản phẩm: Website sẽ cung cấp một giao diện quản lý sản phẩm, cho phép bạn thêm, sửa đổi và xóa các mặt hàng trong cửa hàng. Bạn có thể nhập thông tin chi tiết về sản phẩm, bao gồm các thông tin cơ bản như: mô tả, giá cả và số lượng tồn kho.
2. Quản lý đơn hàng: Website sẽ cho phép bạn theo dõi và quản lý các đơn hàng được đặt từ khách hàng. Bạn có thể xem thông tin chi tiết về đơn hàng, tình trạng thanh toán và giao hàng, và cập nhật trạng thái đơn hàng để thông báo cho khách hàng.
3. Quản lý khách hàng: Website sẽ lưu trữ thông tin về khách hàng, bao gồm tên, địa chỉ, số điện thoại và lịch sử mua hàng. Điều này giúp bạn theo dõi khách hàng, tạo ra các chương trình khuyến mãi và chăm sóc khách hàng tốt hơn.
4. Quản lý kho hàng: Website cung cấp chức năng quản lý kho hàng, cho phép bạn theo dõi số lượng tồn kho của từng loại sản phẩm, cập nhật số lượng tồn sau mỗi đơn hàng.
5. Quản lý giao hàng: Website giúp bạn quản lý lịch trình giao hàng và theo dõi quá trình giao hàng. Bạn có thể nhập thông tin về địa chỉ giao hàng, lựa chọn dịch vụ vận chuyển và theo dõi trạng thái giao hàng.
6. Trang thương mại điện tử: Để tăng tính tiện lợi cho khách hàng, website có thể cung cấp chức năng mua hàng trực tuyến. Khách hàng có thể xem và chọn mua sản phẩm, thêm vào giỏ hàng và thanh toán qua các phương thức thanh toán trực tuyến.

7. Quảng bá thương hiệu: Website sẽ cung cấp một giao diện hấp dẫn và chuyên nghiệp, phản ánh thương hiệu của cửa hàng LINH KIỆN ĐIỆN TỬ. Bạn có thể trình bày thông tin về cửa hàng, các dịch vụ, chia sẻ các bài viết liên quan đến hoa và thiết kế hoa để gây ấn tượng và thu hút khách hàng.

2. Mô hình MVC

MVC (Model – View - Controller) là một design partern đã tồn tại rất lâu trong ngành công nghệ phần mềm. Một ứng dụng viết theo mô hình MVC sẽ bao gồm 3 thành phần tách biệt nhau đó là Model, View, Controller. Giống như trong cấu trúc Three – Tier, mô hình MVC giúp tách biệt 3 tầng trong mô hình lập trình web, vì vậy giúp tối ưu ứng dụng, dễ dàng thêm mới và chỉnh sửa code hoặc giao diện

- Model: ở phần trước mình đã nhắc lại cho các bạn về 3 tầng trong mô hình
    - Three – Tier thì trong đó gồm có 2 tầng Data Access Layer và tầng Business Logic Layer. Hai tầng này là hai tầng tương đương với tầng model trong mô hình MVC.
- View: là tầng giao diện, hiển thị dữ liệu được truy xuất từ tầng model. Tầng này tương đương với tầng Presentation Layer trong cấu trúc Three – Tier.
- Controller: đây là tầng giúp kết nối giữa tầng model và tầng view trong mô hình MVC, có nghĩa là nếu phía client yêu cầu hiển thị dữ liệu thì controller gọi giữ liệu từ model và trả về cho view và tương tác trực tiếp với client

2.1. Model

- Phần Model của kiến trúc MVC là thành phần chính và nó chỉ chứa nghiệp vụ logic, các phương thức xử lý dữ liệu, truy xuất dữ liệu từ database và gửi đến views.
- Model độc lập với giao diện người dùng.

2.2. View

- Phần View giúp người dùng có thể xem được thông tin của trang web, ứng dụng một cách trực quan.
- Bạn có thể hiểu là View là phần bạn nhìn thấy trên trang Web.

2.3. Controller

- Controller dịch ra là điều khiển.
- Đúng như vậy, chức năng của Controller chính là điều khiển, điều hướng các yêu cầu / request từ người dùng và chỉ định phương thức này, phương thức kia trong Model sẽ xử lý.

2.4. Cách làm việc trong MVC

- MVC thường được sử dụng trong các ứng dụng web. View trong các ứng dụng này là các tệp HTML hoặc XHTML do ứng dụng tạo ra.
- Controller nhận đầu vào (bằng form hoặc bất kỳ thứ gì) và sau đó nó quản lý và xử lý đầu vào cho Model.
- Model chứa dữ liệu và các quy tắc về quá trình thực hiện một nhiệm vụ cụ thể.
- Vì dữ liệu được theo dõi bởi chế độ View và nó được kiểm soát cách trình bày cho người dùng, nên các Lập trình viên có thể sử dụng chế độ View tương tự với các dữ liệu khác nhau cho các ứng dụng khác nhau.

- Hoặc có thể sử dụng chung phần Model, Controller chỉ thay đổi phần View.
- Note: Đây chính là cách người ta làm các web giá rẻ. Xây một phần Base chung và chỉ thay đổi phần View khi lập trình web cho các khách hàng khác nhau.

2.5. Ưu điểm của Mô hình MVC

- Nhiều chế độ View có thể được thực hiện cho các Model
- Phân vùng nhiệm vụ giúp Lập trình viên chuyên sâu trong việc phát triển và nâng cấp trong tương lai.
- Lý thuyết MVC hoạt động có hành vi ghép thấp giữa các mô hình, khung nhìn và bộ điều khiển.
- Nhiều Lập trình viên có thể cùng làm việc trên Model, View, Controller cùng một lúc. Điều này giúp việc gia tăng nhân lực để tăng tốc độ dự án là khả thi.
- Các View cho một mô hình cần thiết được nhóm lại với nhau.

3. Cấu trúc chung của hệ thống:

Sử dụng cấu trúc MVC (Model – View - Controller) kèm theo các Repositoy để thực hiện các thao tác CRUD (Create, Read, Update, Delete). Có thêm tầng mô hình dữ liệu Entities/Models Chứa các lớp mô hình (Model) đại diện cho dữ liệu trong cơ sở dữ liệu

Chức năng và vai trò của Repository :

- Repository là một lớp hoặc thành phần chịu trách nhiệm truy xuất, thêm mới, cập nhật và xóa dữ liệu từ nguồn lưu trữ (database, file, API, v.v.). Nó đóng vai trò là một "kho lưu trữ trung gian" giữa Controller và Database.

 Tách biệt logic dữ liệu: Đảm bảo rằng các thao tác với database không trộn lẫn vào logic nghiệp vụ. Điều này giúp mã dễ đọc và dễ bảo trì.
 Đóng gói logic truy vấn: Tất cả các câu truy vấn SQL hoặc các logic liên quan đến dữ liệu (EF, Dapper, hoặc SQL thuần) sẽ được đặt trong Repository.
 Đơn giản hóa việc thay đổi database: Nếu bạn thay đổi hệ thống lưu trữ dữ liệu (chuyển từ SQL Server sang MySQL chẳng hạn), bạn chỉ cần thay đổi Repository mà không ảnh hưởng đến các lớp khác.
 Dễ dàng kiểm thử: Vì repository là một lớp riêng biệt, bạn có thể dễ dàng mock hoặc kiểm thử các thao tác với dữ liệu mà không cần thực sự kết nối đến database.

Chức năng và vai trò của Interface:

Interface định nghĩa các phương thức mà một Repository phải thực hiện, nhưng không chứa bất kỳ logic triển khai nào. Nó cung cấp một hợp đồng (contract) để các lớp triển khai theo.

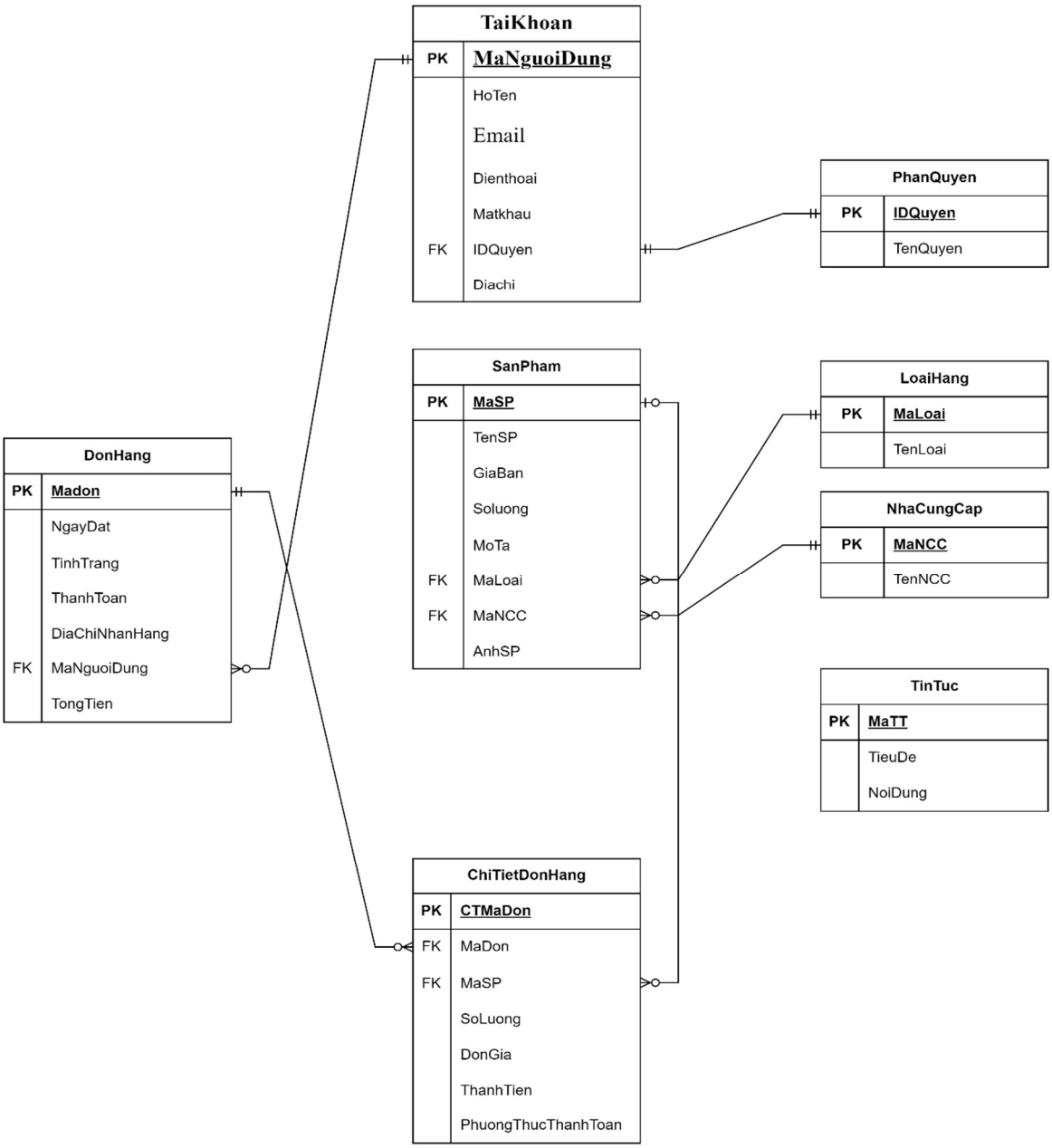
 Tăng tính trừu tượng (Abstraction): Giúp bạn không phụ thuộc vào các triển khai cụ thể của Repository. Bạn chỉ làm việc với Interface và có thể dễ dàng thay thế hoặc thay đổi các lớp triển khai.
 Dễ dàng mở rộng: Nếu muốn thêm các loại Repository khác (ví dụ: InMemoryRepository hay NoSQLRepository), bạn chỉ cần đảm bảo chúng tuân thủ Interface.
 Hỗ trợ Dependency Injection (DI): Interface cho phép Dependency Injection hoạt động hiệu quả hơn. Các lớp nghiệp vụ (BLL) không cần quan tâm đến cách Repository được triển khai, mà chỉ cần làm việc với Interface.

Lợi ích của Repository Pattern với Interface

- **Tách biệt các tầng:** Dễ dàng bảo trì và thay đổi tầng truy cập dữ liệu mà không ảnh hưởng tới logic nghiệp vụ.

- **Tính mở rộng:** Dễ dàng thêm các tính năng mới bằng cách thêm các phương thức vào Interface hoặc Repository.
- **Khả năng kiểm thử:** Interface cho phép viết các unit test dễ dàng bằng cách mock tầng Data Access.
- **Đảm bảo tính nhất quán:** Các lớp thực thi phải tuân theo hợp đồng của Interface, đảm bảo logic đồng bộ.

4. Cấu hình Database:



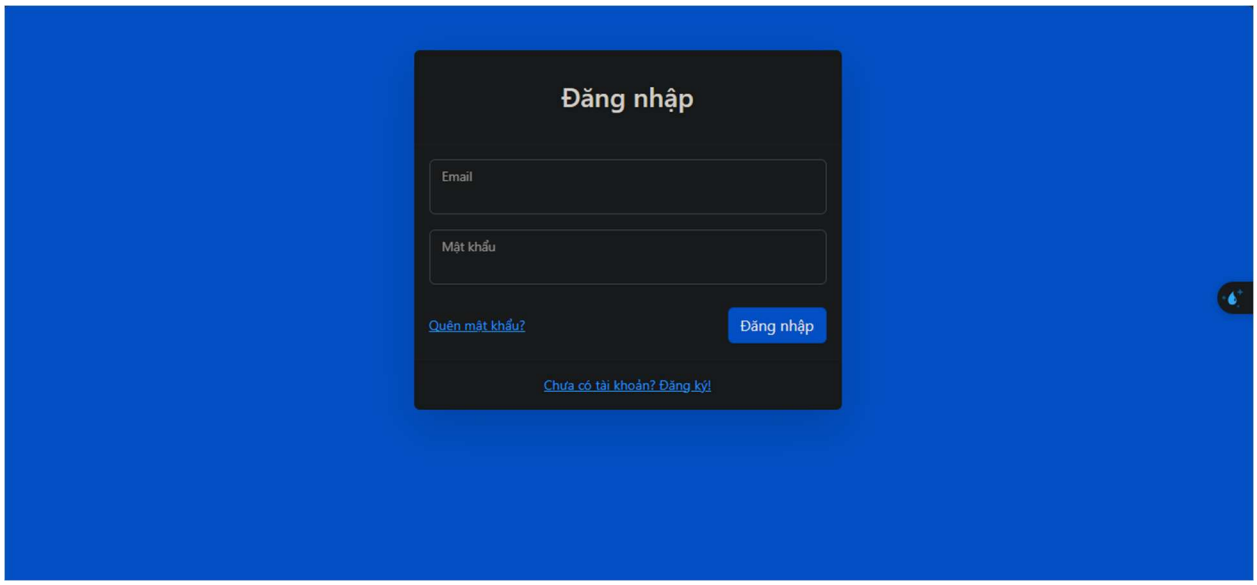
Hình 1: RelationalSchemas của hệ thống

5. Giao diện:

Trang Chủ:

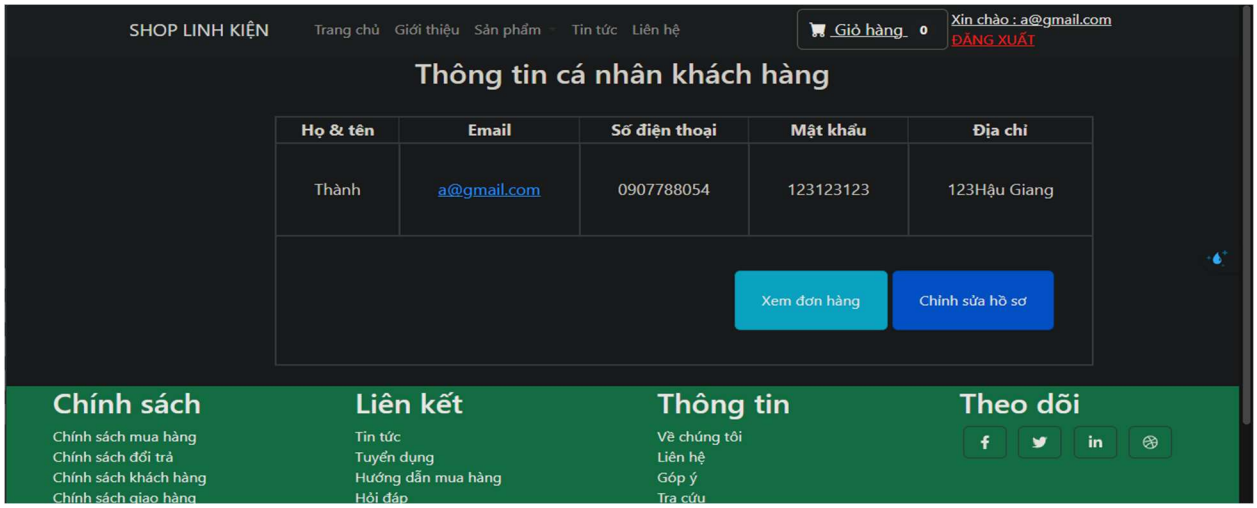
Hình 2: Trang Chủ

Trang Đăng Nhập:

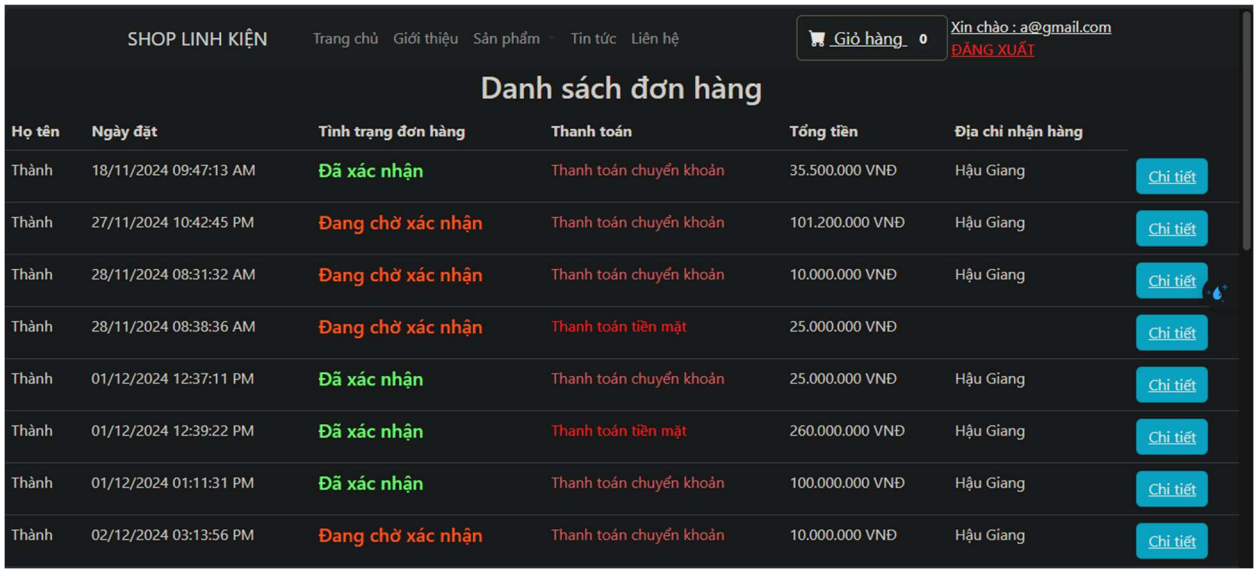


Hình 3: Trang Đăng Nhập

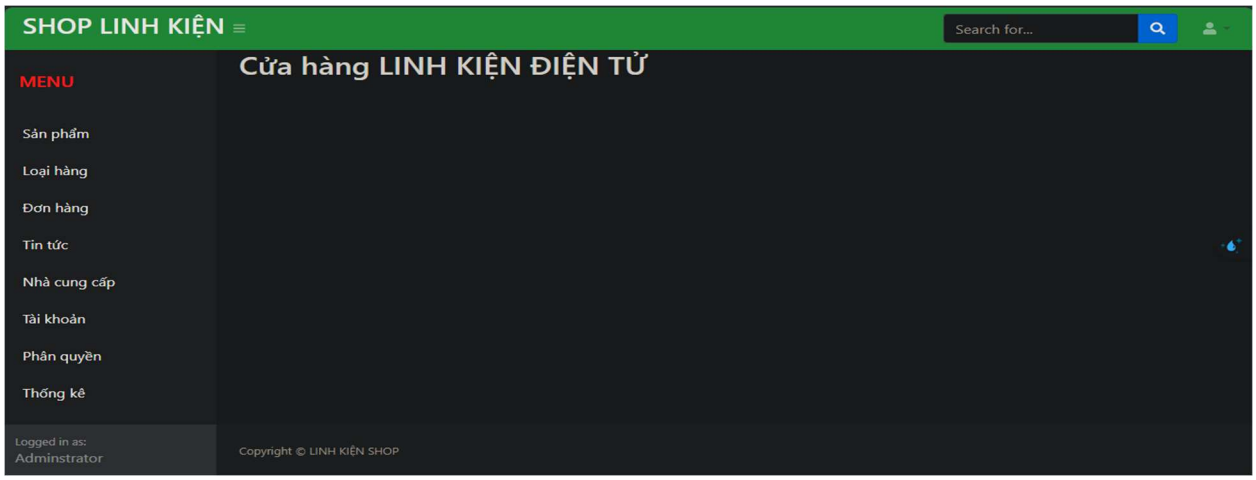
Trang Thông Tin:



Trang Xem Đơn Hàng : *Hình 4: Trang Xem Thông Tin Cá Nhân*



Trang Admin: *Hình 5: Trang Xem Đơn Hàng*



Hình 7: Trang Admin

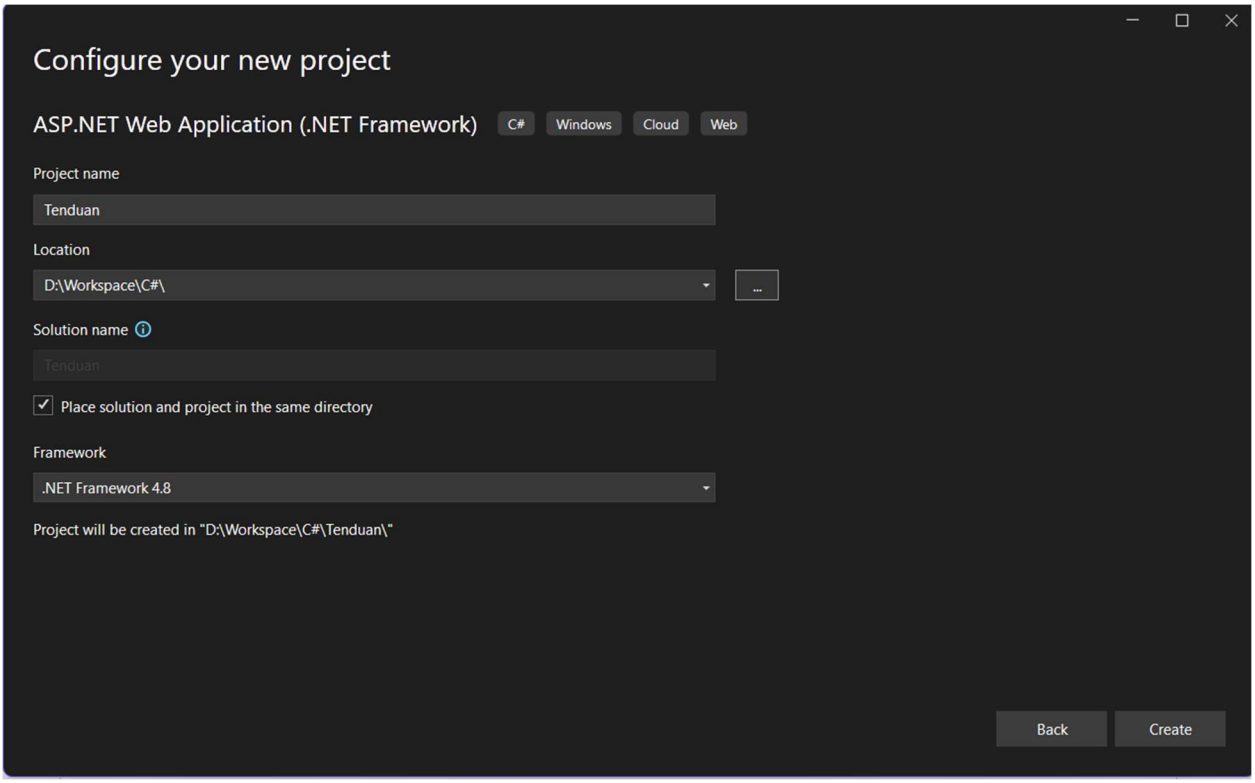
6. Qui trình phát triển hệ thống

A. Phần User :

1. Tạo dự án ASP.NET Framework MVC

1.1. Mở Visual Studio

- Chọn **File** > **New** > **Project**.
- Trong cửa sổ **New Project**, chọn:
  - **Visual C#** > **Web** > **ASP.NET Web Application (.NET Framework)**.
- Đặt tên cho dự án (ví dụ: MvcRepositoryDemo) và chọn **OK**.



Hình 8: Tạo dự án mới

1.2. Chọn mẫu dự án

- Trong cửa sổ **New ASP.NET Project**, chọn **MVC**.
- Bỏ chọn **Host in the cloud** (nếu có).
- Nhấn **OK** để tạo dự án.

2. Cài đặt các gói cần thiết

2.1. Cài đặt Entity Framework

Mở Package Manager Console từ Tools > NuGet Package Manager > Package Manager Console.

Chạy lệnh:

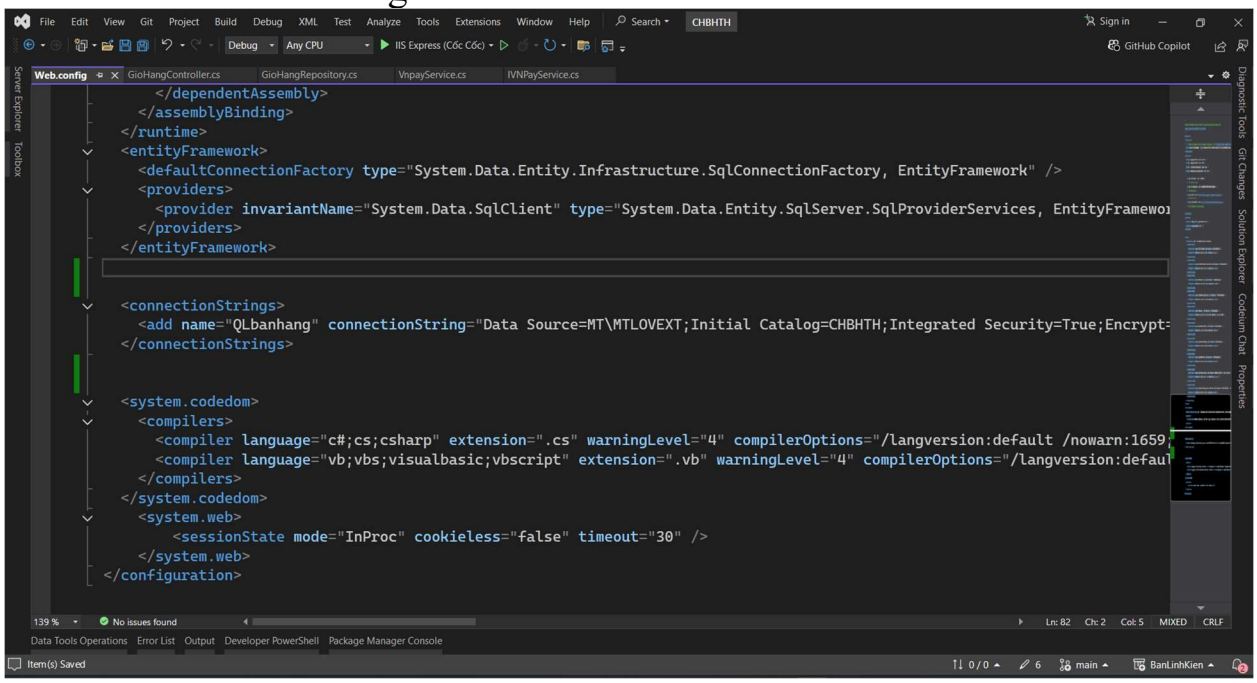
Install-Package EntityFramework

2.2. Cấu hình chuỗi kết nối

- Mở file Web.config.
- Thêm chuỗi kết nối:

xml
Sao chép mã

```
<connectionStrings>
    <add name="DefaultConnection" connectionString="Data
Source=.;Initial Catalog=MvcDemoDB;Integrated Security=True"
providerName="System.Data.SqlClient" />
</connectionStrings>
```

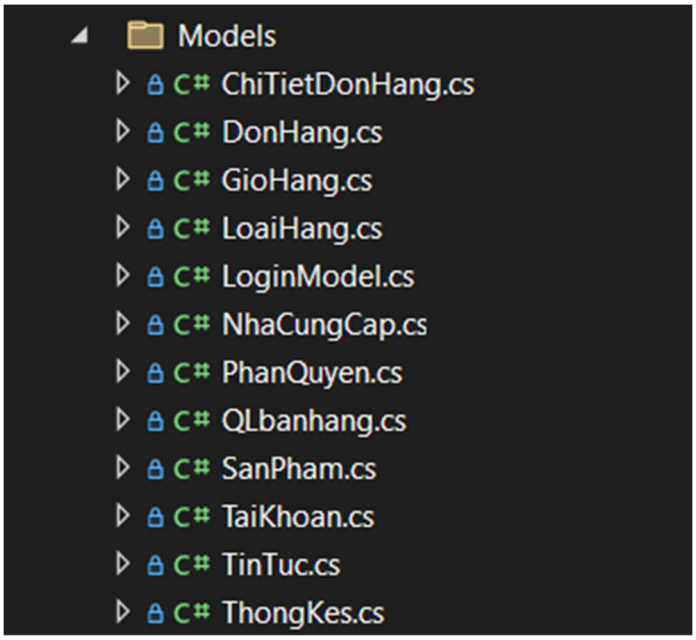


Hình 9: Cấu hình Config

3. Tạo Models

3.1. Thêm thư mục Models (nếu chưa có)

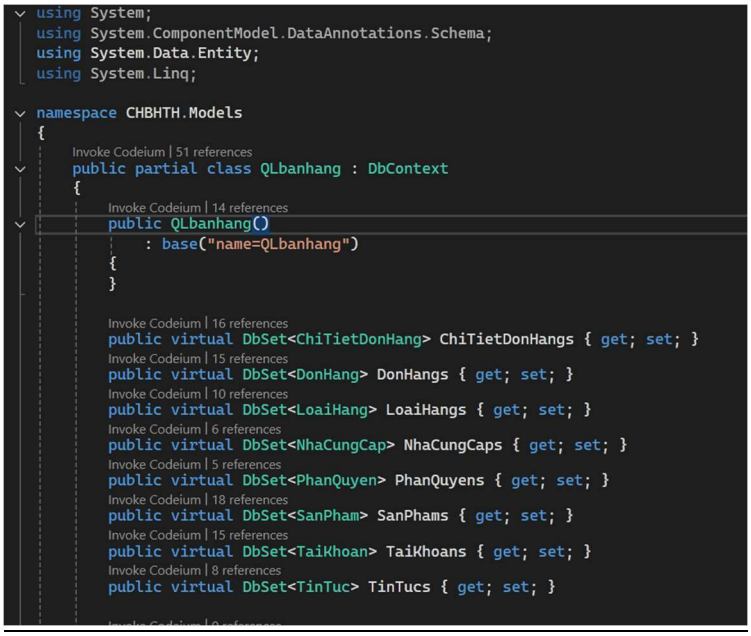
Nhấp chuột phải vào **Solution Explorer**, chọn **Add** > **New Folder**. Đặt tên là Models. Và thêm các Models vào



Hình 10 : Thư mục Models

4. Tạo DbContext

- Tạo một lớp mới trong Models tên là AppDbContext.cs



```
using System;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;
using System.Linq;

namespace CHBHTH.Models
{
    Invoke Codeium | 51 references
    public partial class QLbanhang : DbContext
    {
        Invoke Codeium | 14 references
        public QLbanhang()
            : base("name=QLbanhang")
        {
        }

        Invoke Codeium | 16 references
        public virtual DbSet<ChiTietDonHang> ChiTietDonHangs { get; set; }
        Invoke Codeium | 15 references
        public virtual DbSet<DonHang> DonHangs { get; set; }
        Invoke Codeium | 10 references
        public virtual DbSet<LoaiHang> LoaiHangs { get; set; }
        Invoke Codeium | 6 references
        public virtual DbSet<NhaCungCap> NhaCungCaps { get; set; }
        Invoke Codeium | 5 references
        public virtual DbSet<PhanQuyen> PhanQuyens { get; set; }
        Invoke Codeium | 18 references
        public virtual DbSet<SanPham> SanPhams { get; set; }
        Invoke Codeium | 15 references
        public virtual DbSet<TaiKhoan> TaiKhoans { get; set; }
        Invoke Codeium | 8 references
        public virtual DbSet<TinTuc> TinTucs { get; set; }
```

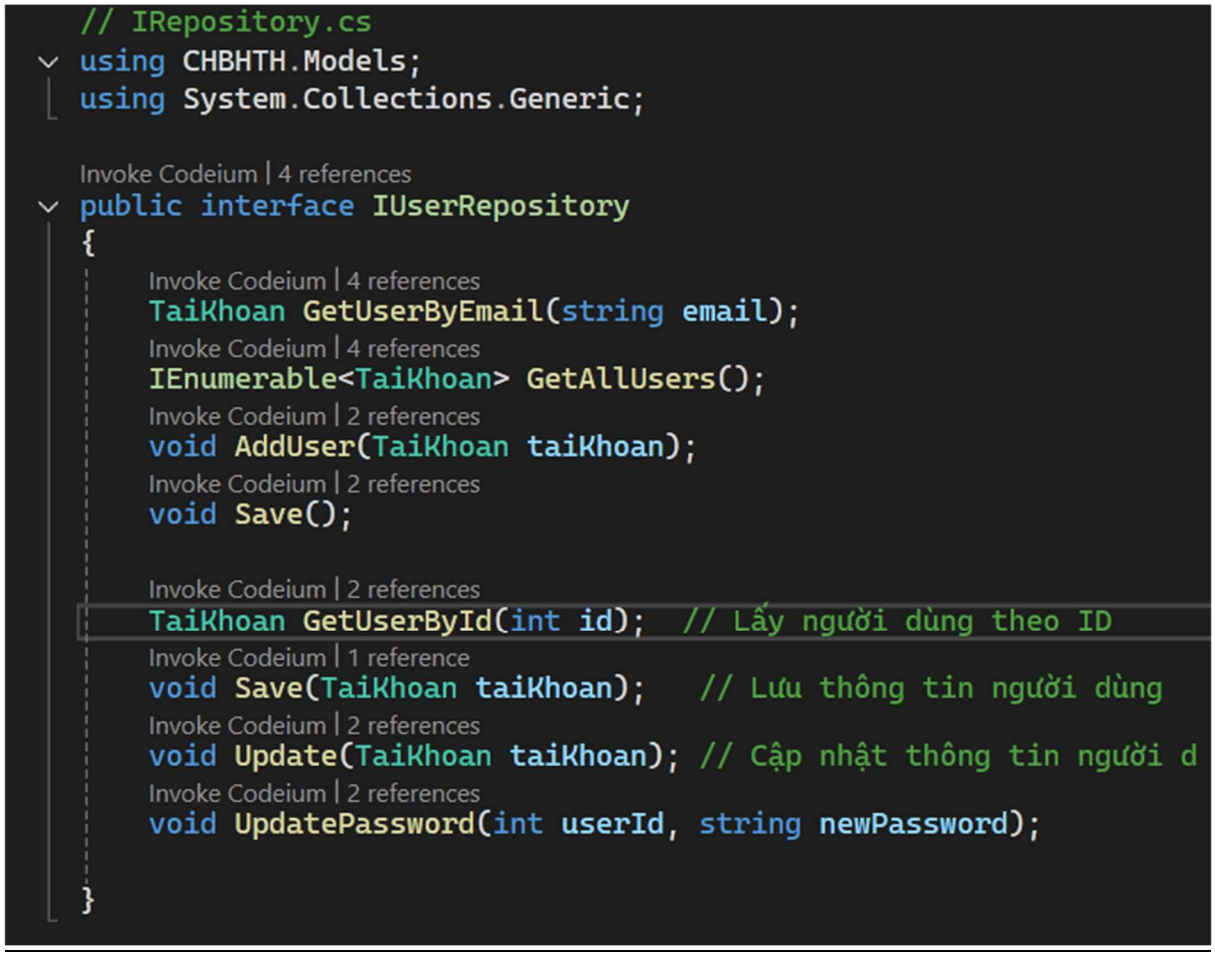
Hình 11: Lớp AppDbContext thay bằng QLbanhang

5. Tạo Repository và Interface

- Thêm thư mục Repository

- Tạo thư mục mới trong dự án, đặt tên là Repository để lưu các file Repository và Interface vào chung

Interface.cs



```
// IRepository.cs
using CHBHTH.Models;
using System.Collections.Generic;

Invoke Codeium | 4 references
public interface IUserRepository
{
    Invoke Codeium | 4 references
    TaiKhoan GetUserByEmail(string email);
    Invoke Codeium | 4 references
    IEnumerable<TaiKhoan> GetAllUsers();
    Invoke Codeium | 2 references
    void AddUser(TaiKhoan taiKhoan);
    Invoke Codeium | 2 references
    void Save();

    Invoke Codeium | 2 references
    TaiKhoan GetUserById(int id);  // Lấy người dùng theo ID
    Invoke Codeium | 1 reference
    void Save(TaiKhoan taiKhoan);   // Lưu thông tin người dùng
    Invoke Codeium | 2 references
    void Update(TaiKhoan taiKhoan); // Cập nhật thông tin người d
    Invoke Codeium | 2 references
    void UpdatePassword(int userId, string newPassword);

}
```

Hình 12: Lớp Interface Quản lý phương thức truy xuất

Repository.cs

```
// UserRepository.cs
using CHBHTH.Models;
using System.Collections.Generic;
using System.Linq;

Invoke Codeium | 2 references
public class UserRepository : IUserRepository
{
    private readonly QLbanhang _dbContext;

    Invoke Codeium | 0 references
    public UserRepository(QLbanhang dbContext)
    {
        _dbContext = dbContext;
    }

    Invoke Codeium | 4 references
    public TaiKhoan GetUserByEmail(string email)
    {
        return _dbContext.TaiKhoans.SingleOrDefault(x => x.Email.Equals(email));
    }

    Invoke Codeium | 4 references
    public IEnumerable<TaiKhoan> GetAllUsers()
    {
        return _dbContext.TaiKhoans.ToList();
    }
```

Hình 13: Lớp thực thi Repository

Cần khai báo trong file Global.asax để lớp thực thi Repository và Interface mới có thể chạy được như sau:

```
        Invoke Codeium | 0 references
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
            // Cấu hình Dependency Injection
            var container = ConfigureUnityContainer();
            DependencyResolver.SetResolver(new UnityDependencyResolver(container));
        }
        Invoke Codeium | 1 reference
        private IUnityContainer ConfigureUnityContainer()
        {
            var container = new UnityContainer();

            // Đăng ký interface với class implementation
            container.RegisterType<IUserRepository, UserRepository>();
            container.RegisterType<ITintucRepository, TinTucRepository>();
            container.RegisterType<ISanPhamRepository, SanPhamRepository>();
            container.RegisterType<IGioHangRepository, GioHangRepository>();
            container.RegisterType<IDonHangRepository, DonHangRepository>();
            container.RegisterType<IDanhMucRepository, DanhMucRepository>();

            //Admin
            container.RegisterType<ISanPhamRepositoryAdmin, SanPhamRepositoryAdmin>();
            container.RegisterType<ITaiKhoanRepositoryAdmin, TaiKhoanRepositoryAdmnin>();
            container.RegisterType<IPhanQuyenRepository, PhanQuyenRepository>();
            container.RegisterType<INhaCungCapRepository, NhaCungCapRepository>();
            container.RegisterType<ILoaiHangRepository, LoaiHangRepository>();
            container.RegisterType<ITinTucRepositoryAdmin, TinTucRepositoryAdmin>();
            container.RegisterType<IThongKeRepository, ThongKeRepository>();
            container.RegisterType<IChiTietDonHangRepositoryAdmin, ChiTietDonHangRepositoryAdmin>();
            container.RegisterType<IDonHangRepositoryAdmin, DonHangRepositoryAdmin>();

            // Đăng ký thêm các dịch vụ khác nếu cần
            container.RegisterType<IVNPayService, VnpayService>();

            return container;
        }
    }
}
```

Hình 14: Khai báo Repository để sử dụng

6. Tạo Controller

1. **Thêm thư mục Controllers (nếu chưa có)**
   - Thư mục này đã tồn tại mặc định trong dự án khi được khởi tạo bằng MVC.
2. Các tạo Controllers:

 Nhấp chuột phải vào thư mục **Controllers**, chọn **Add** > **Controller** > **MVC 5 Controller with views,using Entity Framework**.

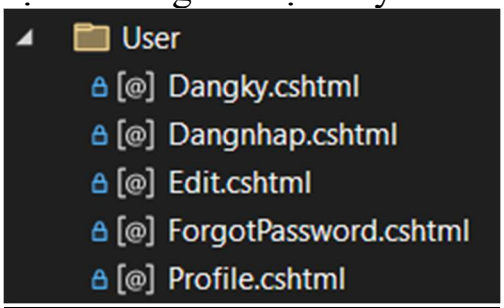
 Chọn Models class và Data Context class sau đó đặt tên cho Controller và nhấn **Add**.

```
using CHBHTH.Models;
using Microsoft.Ajax.Utilities;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;

namespace CHBHTH.Controllers
{
    Invoke Codeium | 1 reference
    public class UserController : Controller
    {
        QLbanhang db = new QLbanhang();
        private readonly IUserRepository _userRepository;

        Invoke Codeium | 0 references
        public UserController(IUserRepository userRepository)
        {
            _userRepository = userRepository;
        }

        // ĐĂNG KÝ
        Invoke Codeium | 0 references
        public ActionResult Dangky()
        {
            return View();
        }
```

Hình 15: Lớp Controller để giao tiếp với UI

7. Tạo View cho Controller

Khi tạo Controller **MVC 5 Controller with views,using Entity Framework**. Sẽ tự động tạo ra các giao diện truy xuất mặc định

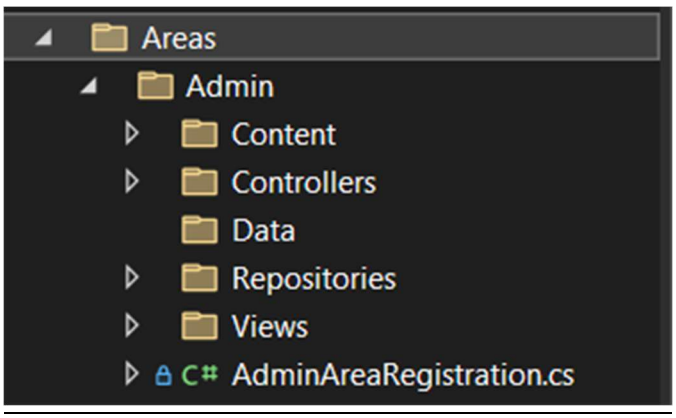


Hình 16: Views do Controller tạo

B. Phần Admin:

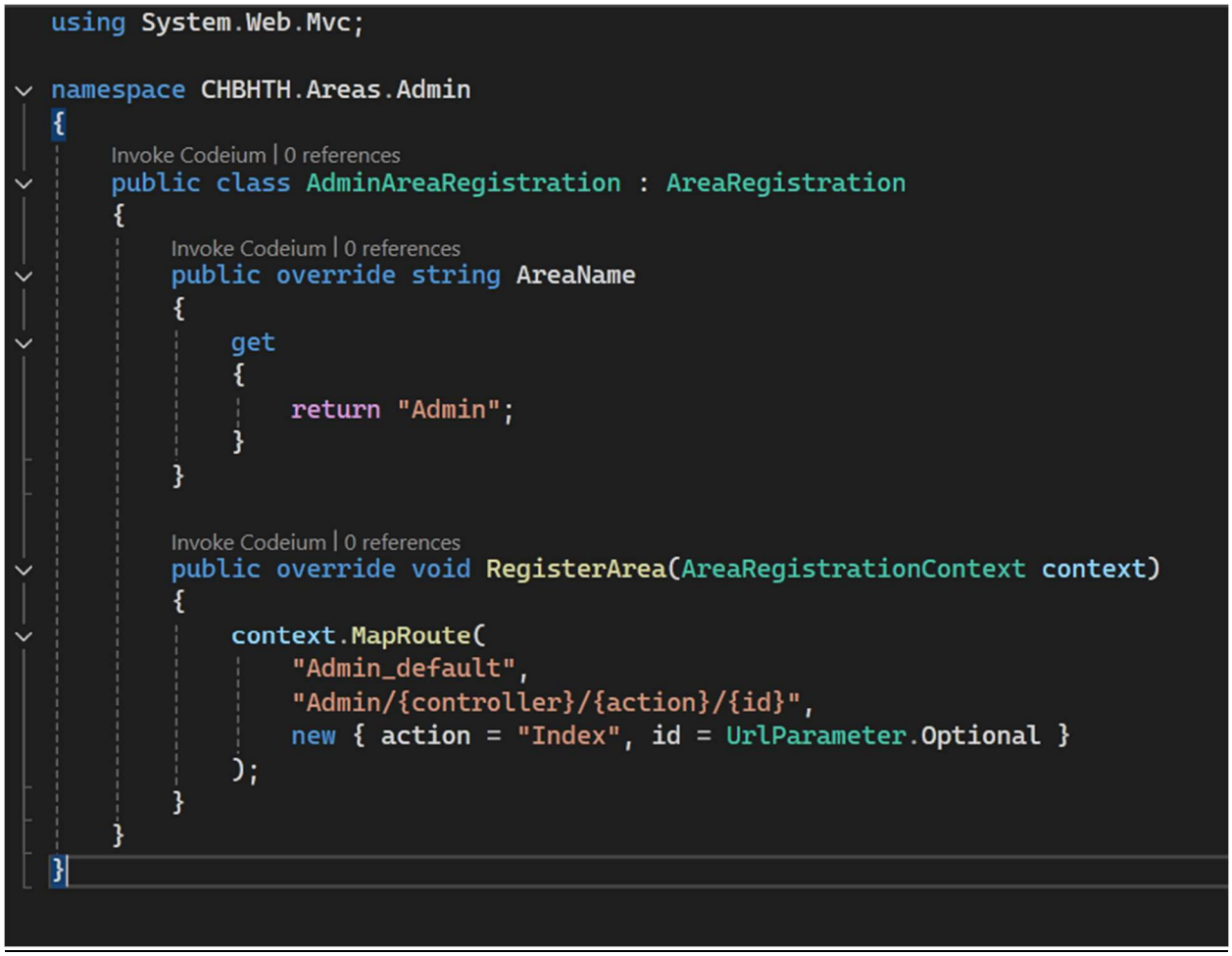
1. Tạo Area Admin

1. **Tạo Area**
   - Nhấp chuột phải vào dự án trong **Solution Explorer**.
   - Chọn **Add** > **Thư mục Tạo thư mục mới Với Tên** Area và tạo tiếp các thư mục Admin> sau đó tạo tiếp tục các thư mục Controller, Views, Repository.
2. **Tạo file AdminAreaRegistration.cs**
   - Đây là nơi cấu hình route cho **Admin Area**.



Hình 17: Tạo thư mục phân vùng Admin

File cấu cấu hình **AdminAreaRegistration**.cs



```
using System.Web.Mvc;

namespace CHBHTH.Areas.Admin
{
    Invoke Codeium | 0 references
    public class AdminAreaRegistration : AreaRegistration
    {
        Invoke Codeium | 0 references
        public override string AreaName
        {
            get
            {
                return "Admin";
            }
        }

        Invoke Codeium | 0 references
        public override void RegisterArea(AreaRegistrationContext context)
        {
            context.MapRoute(
                "Admin_default",
                "Admin/{controller}/{action}/{id}",
                new { action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

Hình 18: AdminAreaRegistration.cs

Rồi thực hiện tạo các file cần thiết như bên User.

7.Kết luận

7.1. Những tính năng đã làm được

- Có trang admin dùng để quản trị hệ thống
- Giao diện người dùng
- Giao diện giỏ hàng
- Theo dõi đơn hàng
- Đăng nhập, đăng ký tài khoản

7.2. Những tính năng chưa làm được

- Thanh toán Vnpay
- Chức năng tìm kiếm
- Chức năng đánh giá sản phẩm …

8. .Đánh giá

8.1. Thuận lợi và khó khăn

- Thuận lợi: có nhiều thông tin trên mạng giúp cho quá trình tìm hiểu để phát triển dự án, vô số nguồn để tìm hiểu và phát triển dự án
- Khó khăn: thời gian hạn chế do phải thực hiện một vài đề tài khác, còn nhiều sai sót cần chỉnh sửa trong đề tài và số lượng thành viên hạn chế nên việc dự án vẫn còn nhiều lỗi và sai sót, giao diện không được tối ưu cho người dùng.

8.2. Hướng phát triển

Trong tương lai, nhóm chúng em dự tính chỉnh sửa đề tài như sau:

- Chỉnh giao diện cho bắt mắt hơn.
- Update CSDL để hiển thị nhiều văn bản đa dạng hơn.
- Hoàn chỉnh chức năng thanh toán bằng vnpay.

TÀI LIỆU THAM KHẢO

1. Microsoft .NET Core Documentation
2. GitHub .NET Core
3. Youtube:
4. Youtube: freeCodeCamp.org
5. Bài học và bài tập trên lớp
6. Stack Overflow
7. Reddit