**Food Application Project in Asp.net core**

Đề tài cuối kỳ môn CN.NET



**Giảng viên hướng dẫn: Lương Trần Hy Hiến**

**Sinh viên thực hiện:**

* Đinh Nguyễn Phương Vy 4501104282
* Lê Thị An Bình 4501104018
* Đỗ Hoàng Phúc 4501104178
* Lê Chí Hải 4501104066
* Nguyễn Võ Hiếu Trung 4501104263



[**Chapter I. Mô hình MVC** 6](#_Toc122381907)

[**I.i. Khái quát về MVC** 6](#_Toc122381908)

[**i)\_Khái niệm MVC** 6](#_Toc122381909)

[**ii)\_Thành phần của MVC** 6](#_Toc122381910)

[**I.ii. Sự tương tác giữa các thành phần** 8](#_Toc122381911)

[**I.iii. Luồng xử lý trong MVC** 8](#_Toc122381912)

[**I.iv. Ưu điểm và nhược điểm của MVC** 8](#_Toc122381913)

[**I.v. Vì sao nên sử dụng mô hình MVC** 9](#_Toc122381914)

[**I.vi. Sử Dụng framework Laravel Trong Mô Hình MVC** 11](#_Toc122381915)

[**Chapter II. ASP.NET Core API** 13](#_Toc122381916)

[**II.i. Sơ lượt về lịch sử của ASP.NET Core** 13](#_Toc122381917)

[**II.ii.** ASP.NET Core là gì? 13](#_Toc122381918)

[**II.iii. Thành phần của ASP.NET** 15](#_Toc122381919)

[**II.iv. Ưu điểm của ASP.NET Core** 15](#_Toc122381920)

[**II.v. Sự khác nhau giữa ASP.NET và ASP.NET Core** 16](#_Toc122381921)

[**Chapter III. ASP.NET Core MVC** 18](#_Toc122381922)

[**Mô hình MVC là gì?** 18](#_Toc122381923)

[**SP.NET Core MVC** 18](#_Toc122381924)

[**Model binding** 20](#_Toc122381925)

[**Model validation** 20](#_Toc122381926)

[**Dependency injection** 21](#_Toc122381927)

[**Filters** 22](#_Toc122381928)

[**Areas** 22](#_Toc122381929)

[**Web APIs** 22](#_Toc122381930)

[**Testability** 23](#_Toc122381931)

[**Razor view engine** 23](#_Toc122381932)

[**Tag Helpers** 23](#_Toc122381933)

[**Chapter VI. Food Application project** 24](#_Toc122381934)

[**Tài liệu tham khảo** 27](#_Toc122381935)

**Bảng phân công công viêc**

|  |  |  |
| --- | --- | --- |
| **Họ và tên** | **Main task** | **Detail** |
| Đinh Nguyễn Phương Vy | home page  css | Menubar  carousel on home page  footer  recipe details  css page |
| Lê Thị An Bình | search page  fix | add search page  add search link in slide menu  add order button on recipe page  add order page  add order button on home page  fix show all cart list  fix save order  fix remove cart from list  fix add order model |
| Đỗ Hoàng Phúc | login and registration  Save order | add account controller  add login model  add register model  add login action and view  add register action and view  install packages  add querystring  add dbcontext  create user model  registration functionlity  login functionlity  customize menu  create logout action  customize bill details  add random recipe price  quantity increse and decrese  add order model  show user details  add order action  add authentication  save order |
| Lê Chí Hải | add to cart  fix | add cart icon in menu  add cart controller and view  add cart model  add cart icon in recipe  add repository pattern  save cart  show added cart icon  remove cart from list  create add to cart list  show all cart list  fix connect database |
| Nguyễn Võ Hiếu Trung | word  ppt  Recipe page | add recipes controller and view  add search bar  add slide menubar  auto suggestion  recipes |

# **Chapter I. Mô hình MVC**

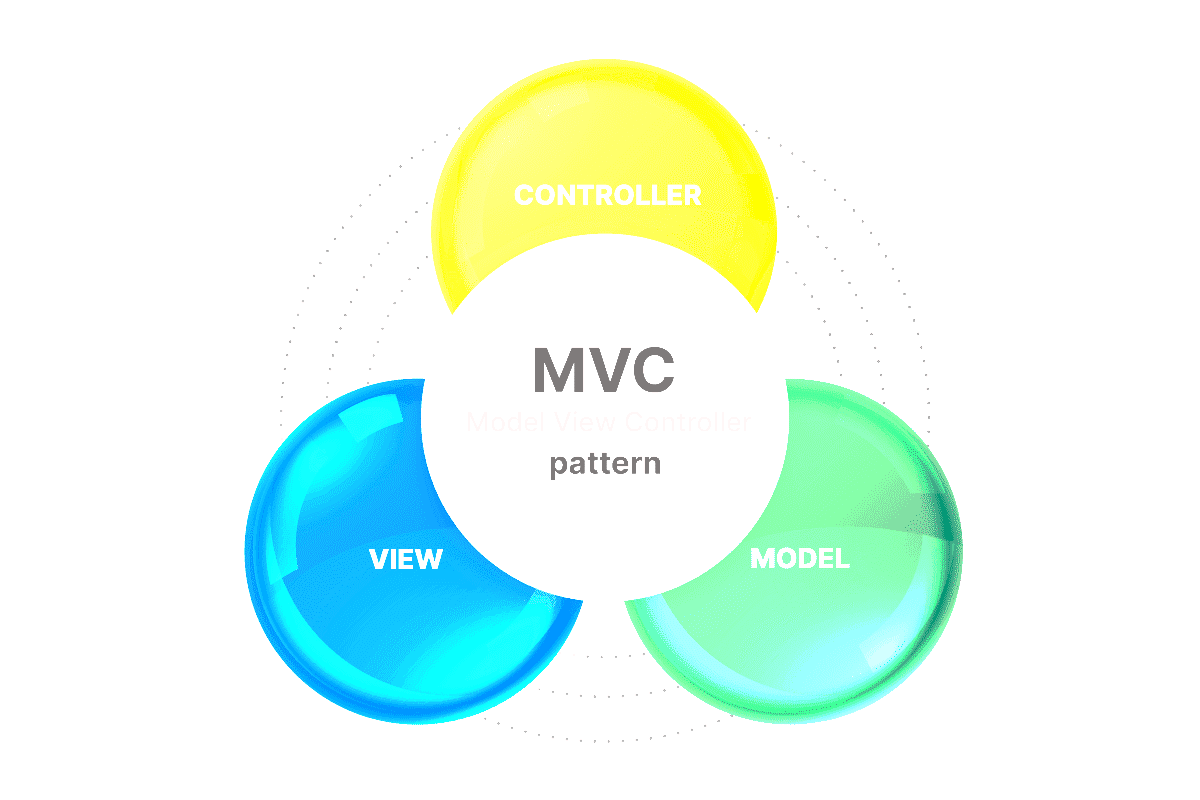
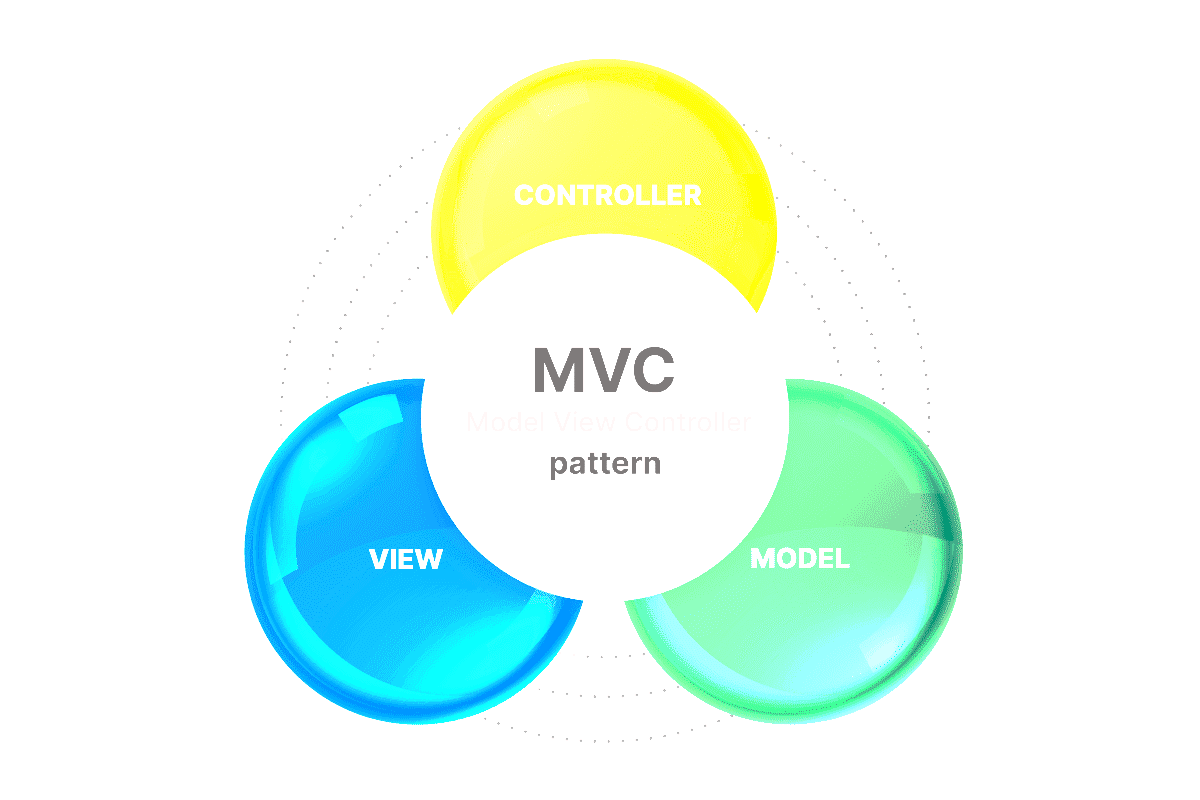
## **I.i. Khái quát về MVC**

### **i)\_Khái niệm MVC**

**MVC** là viết tắt của cụm từ “**Model-View-Controller**“. Đây là mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. MVC là một **mẫu kiến trúc phần mềm** để tạo lập giao diện người dùng trên máy tính. Tuy là một mô hình thiết kế phần mềm nhưng nó cũng được sử dụng rộng rãi trong web, sự khác biệt được tùy chỉnh liên quan đến sự có mặt của server – client. Mô hình MVC (MVC pattern) thường được dùng để phát triển giao diện người dùng. Nó cung cấp các thành phần cơ bản để thiết kế một chương trình cho máy tính hoặc điện thoại di động, cũng như là các ứng dụng web.

MVC được tiến sĩ Trygve Reenskaug đưa vào ngôn ngữ lập trình Smalltalk-76 khi ông đến trung tâm Nghiên cứu Xerox Palo Alto (PARC) vào giữa năm 1970. Sau đó, việc triển khai trở nên phổ biến trong các phiên bản khác của Small- Talk. Năm 1988, các bài báo “The Journal of Object Technology” – JOT mang lại bước tranh toàn cảnh về MVC mang liệu sự hiệu quả tốt nhất.

### **ii)\_Thành phần của MVC**



MVC chia thành ba phần kết nối với nhau và mỗi thành phần đều có một nhiệm vụ riêng của nó và độc lập với các thành phần khác. Tên gọi 3 thành phần:

* Model (dữ liệu): Quản lí xử lí các dữ liệu.
* View (giao diện): Nới hiển thị dữ liệu cho người dùng.
* Controller (bộ điều khiển): Điều khiển sự tương tác của hai thành phần Model và View

*Là một dạng mẫu dữ liệu, có nhiệm vụ thao tác với cơ sở dữ liệu, nghĩa là nó sẽ chứa tất cả các hàm, các phương thức truy vấn trực tiếp với dữ liệu và controller sẽ thông qua các hàm, phương thức đó để lấy dữ liệu rồi gửi qua View.*

Model

*Là các giao diện người dùng, có nhiệm vụ tiếp nhận dữ liệu từ controller là nơi chứa những giao diện như một nút bấm, khung nhập, menu, hình ảnh… nó đảm nhiệm nhiệm vụ hiển thị dữ liệu và giúp người dùng tương tác với hệ thống.*

View

*Là các hành vi, hành động, xử lý của hệ thống đóng vai trò trung gian giữa Model và View. Nó có nhiệm vụ tiếp nhận yêu cầu từ client sau đó xử lý request, load model tương ứng và gửi data qua view tương ứng rồi trả kết quả về cho client.*

Controller

## **I.ii. Sự tương tác giữa các thành phần**

Ngoài việc chia ứng dụng thành các thành phần này, thiết kế model – view – controller xác định các tương tác giữa chúng. Mô hình chịu trách nhiệm quản lý dữ liệu của ứng dụng. Nó nhận đầu vào của người dùng từ bộ điều khiển. Chế độ xem hiển thị bản trình bày của mô hình theo một định dạng cụ thể. Bộ điều khiển phản hồi đầu vào của người dùng và thực hiện các tương tác trên các đối tượng mô hình dữ liệu. Bộ điều khiển nhận đầu vào, xác nhận tùy chọn và sau đó chuyển đầu vào cho mô hình. Cũng như các mẫu phần mềm khác, MVC thể hiện "cốt lõi của giải pháp" cho một vấn đề đồng thời cho phép nó được điều chỉnh cho phù hợp với từng hệ thống. Các thiết kế MVC cụ thể có thể thay đổi đáng kể so với mô tả truyền thống ở đây.

## **I.iii. Luồng xử lý trong MVC**

*Luồng xử lý trong của mô hình MVC có thể được hình dung cụ thể và chi tiết qua từng bước dưới đây:*

* Khi một yêu cầu của từ máy khách (Client) gửi đến Server. Thì bị Controller trong MVC chặn lại để xem đó là URL request hay sự kiện.
* Sau đó, **Controller** xử lý **input** của user rồi giao tiếp với**Model** trong MVC.
* Model chuẩn bị data và gửi lại cho Controller.
* Cuối cùng, khi xử lý xong yêu cầu thì Controller gửi dữ liệu trở lại View và hiển thị cho người dùng trên trình duyệt.

## **I.iv. Ưu điểm và nhược điểm của MVC**

**Ưu điểm**

* Do được chia thành các thành phần độc lập nên Mô hình MVC giúp phát triển ứng dụng có code dễ đọc, dễ nâng cấp, bảo trì. Thể hiện tính chuyên nghiệp trong việc tạo ứng dụng. Đầu tiên, nhắc tới ưu điểm mô hình MVC thì đó là băng thông ([Bandwidth](https://vietnix.vn/bandwidth-la-gi/)) nhẹ vì không sử dụng viewstate nên khá tiết kiệm băng thông. Việc giảm băng thông giúp website hoạt động ổn định hơn.
* Kiểm tra đơn giản và dễ dàng, kiểm tra lỗi phần mềm trước khi bàn giao lại cho người dùng. Một lợi thế chính của MVC là nó tách biệt các phần Model, Controller và View với nhau.
* Sử dụng mô hình MVC chức năng Controller có vai trò quan trọng và tối ưu trên các nền tảng ngôn ngữ khác nhau.
* Ta có thể dễ dàng duy trì ứng dụng vì chúng được tách biệt với nhau.
* Có thể chia nhiều developer làm việc cùng một lúc. Công việc của các developer sẽ không ảnh hưởng đến nhau.
* Hỗ trợ [TTD](https://en.wikipedia.org/wiki/Test-driven_development) (test-driven development). Chúng ta có thể tạo một ứng dụng với unit test và viết các won [test case](https://vietnix.vn/test-case-la-gi/).
* Phiên bản mới nhất của MVC hỗ trợ trợ thiết kế responsive website mặc định và các mẫu cho mobile. Chúng ta có thể tạo công cụ View của riêng mình với cú pháp đơn giản hơn nhiều so với công cụ truyền thống.

**Nhược điểm**

* Đối với dự án nhỏ việc áp dụng mô hình MVC gây cồng kềnh, tốn thời gian trong quá trình phát triển và thời gian trung chuyển dữ liệu của các thành phần.
* MVC đa phần phù hợp với công ty chuyên về website hoặc các dự án lớn thì mô hình này phù hợp hơn so với với các dự án nhỏ, lẻ vì khá là cồng kềnh và mất thời gian.
* Không thể **Preview** các trang như [ASP.NET](https://vietnix.vn/asp-net-la-gi/).
* Khó triển khai.

## **I.v. Vì sao nên sử dụng mô hình MVC**

**Quy trình phát triển nhanh hơn**

MVC hỗ trợ phát việc phát triển nhanh chóng và song song. Nếu một mô hình MVC được dùng để phát triển bất kỳ ứng dụng web cụ thể nào, một lập trình viên có thể làm việc trên View và một developer khác có thể làm việc với Controller để tạo logic nghiệp vụ cho ứng dụng web đó. Do đó, ứng dụng mô hình MVC có thể được hoàn thành nhanh hơn ba lần so với các ứng dụng mô hình khác.

**Khả năng cung cấp nhiều chế độ view**

Trong mô hình MVC, bạn có thể tạo nhiều View cho chỉ một mô hình. Ngày nay, nhu cầu có thêm nhiều cách mới để truy cập ứng dụng và đang ngày càng tăng. Do đó, việc sử dụng MVC để phát triển chắc chắn là một giải pháp tuyệt vời. Hơn nữa, với phương pháp này, việc nhân bản code rất hạn chế. Vì nó tách biệt dữ liệu và logic nghiệp vụ khỏi màn hình.

**Các sửa đổi không ảnh hưởng đến toàn bộ mô hình**

Đối với bất kỳ ứng dụng web nào, người dùng có xu hướng thay đổi thường xuyên. Bạn có thể quan sát thông qua những thay đổi thường xuyên về màu sắc, font chữ, bố cục màn hình. Hay là thêm hỗ trợ thiết bị mới cho điện thoại hay máy tính bảng… Việc thêm một kiểu view mới trong MVC rất đơn giản. Vì phần Model không phụ thuộc vào phần View. Do đó, bất kỳ thay đổi nào trong Model sẽ không ảnh hưởng đến toàn bộ kiến trúc.

**MVC Model trả về dữ liệu mà không cần định dạng**

MVC pattern có thể trả về dữ liệu mà không cần áp dụng bất kỳ định dạng nào. Do đó, các thành phần giống nhau có thể được sử dụng với bất kỳ giao diện nào. Ví dụ: tất cả loại dữ liệu đều có thể được định dạng bằng HTML. Ngoài ra, nó cũng có thể được định dạng bằng Macromedia Flash hay Dream Viewer.

**Hỗ trợ kỹ thuật Asynchronous**

Kiến trúc MVC có thể được tích hợp với cả JavaScript Framework. Có nghĩa là, các ứng dụng MVC có thể hoạt động ngay cả với các file PDF, trình duyệt riêng cho web hay các widget trên desktop. Ngoài ra, MVC cũng hỗ trợ kỹ thuật Asynchronous, giúp các developer phát triển các ứng dụng có thể load rất nhanh.

**Nền tảng MVC thân thiện với SEO**

Nền tảng MVC hỗ trợ phát triển các trang web thân thiện với SEO. Bằng nền tảng này, bạn có thể dễ dàng phát triển các URL thân thiện với SEO để tạo ra nhiều lượt truy cập hơn.

## **I.vi. Sử Dụng framework Laravel Trong Mô Hình MVC**

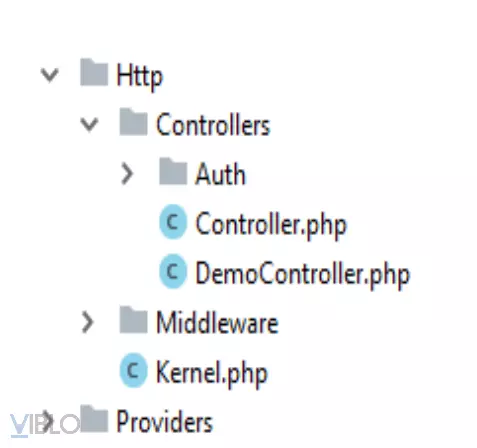
Laravel là một PHP Framework mã nguồn mở miễn phí, được phát triển bởi Taylor Otwell với phiên bản đầu tiên được ra mắt vào tháng 6 năm 2011. Laravel ra đời nhằm mục đích hỗ trợ phát triển các ứng dụng web, dựa trên mô hình MVC (Model – View – Controller).

**Controller**: Vị trí lưu Trong Laravel, controller được tạo ra trong folder app/Http/Controllers

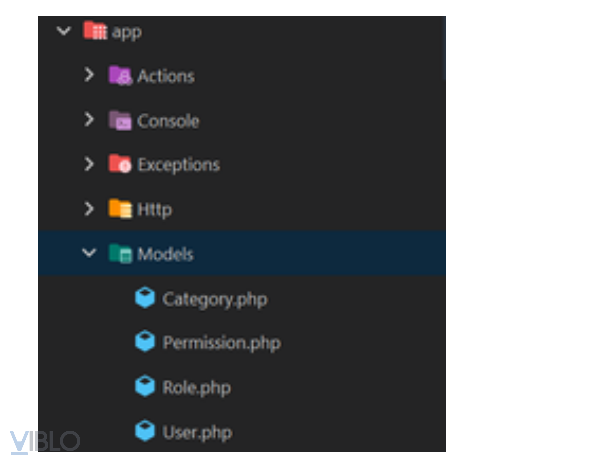
**View**: là các file trình bày dữ liệu (code html, css, js). Các file view được đặt trong folder ressources/views. Đặc biệt trong Laravel có 2 dạng view: view bình thường và view dạng blade.

**Route**: Ngoài các thành phần Model View Controller thì trong Laravel có 1 thứ rất quan trọng: route. Route trong Laravel được sử dụng để khai báo các đường dẫn, dẫn người dùng đến các chức năng trong Laravel. Các đường route trong Laravel được định nghĩa trong file routes/web.php, và thành routes/ api.php

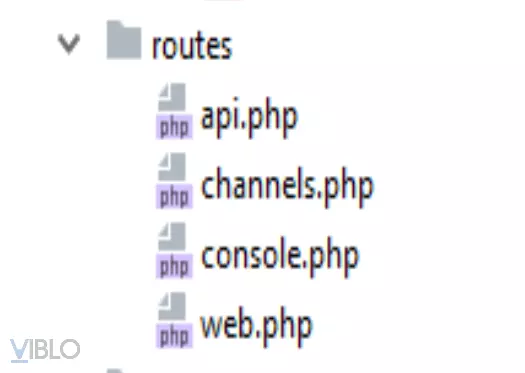
**Model**: Vị trí lưu: Trong Laravel, model được tạo ra trong folder app/Models



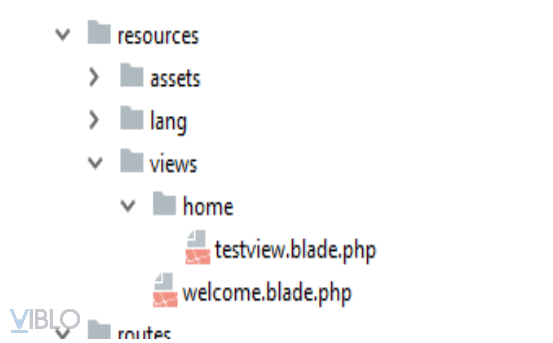
Controller trong Laravel



Model trong Laravel



Route trong Laravel



View trong Laravel

# **Chapter II. ASP.NET Core API**

*ASP.NET Core là một framework mới của Microsoft. Nó được thiết kế lại từ đầu để trở nên nhanh chóng, linh hoạt và chạy trên được nhiều nền tảng khác nhau. Trong bài viết này, ASP.NET Core là một framework có thể được sử dụng để phát triển web với .NET. Nếu bạn có kinh nghiệm làm việc với MVC hoặc Web API trong một vài năm thì bạn sẽ nhận thấy những tính năng quen thuộc. Kết thúc khóa học ASP.NET Core của NET CORE VN. Bạn sẽ có những thứ cần thiết để bắt đầu làm việc được với ASP.NET Core và viết 1 ứng dụng có thể thêm, sửa hay xem dữ liệu từ database như các môn lập trình khác bạn đã từng học.*

## **II.i. Sơ lượt về lịch sử của ASP.NET Core**

***ASP.NET được ra đời vào năm 2002 và đến nay đã được nâng cấp thành nhiều phiên bản khác nhau.***

ASP.NET đã được sử dụng từ nhiều năm để phát triển các ứng dụng web. Kể từ đó, framework này đã trải qua một sự thay đổi tiến hóa đều đặn và cuối cùng đã đưa chúng ta đến với ASP.NET Core 1.0 hậu duệ mới gần đây nhất của nó.

* ***ASP.NET Core*** *không phải là phiên bản tiếp theo của* ***ASP.NET 4.6***
* *Đó là một framework hoàn toàn mới, may mắn rằng nó một project side-by-side tương tự với mọi thứ mà chúng ta biết.*
* *Nó thực ra được viết lại trên framework* ***ASP.NET 4.6*** *hiện tại những kích thước nhỏ hơn và nhiều modular hơn.*

## **II.ii. ASP.NET Core là gì?**

ASP.NET Core là một web framework mã nguồn và được tối ưu hóa cho cloud để phát triển các ứng dụng web chạy trên nhiều nền tảng như Windows, Linux và Mac. Hiện tại, nó bao gồm MVC framework được kết hợp các tính năng của MVC và Web API thành một web framework duy nhất.

* *Các ứng dụng* ***ASP.NET Core*** *có thể chạy trên* ***.NET Core*** *hoặc trên****.NET Framework*** *hoàn chỉnh.*
* *Nó đã được thiết kế để cung cấp một framework tối ưu cho các ứng dụng để triển khai tới cloud hoặc chạy on-premises.*
* *Nó bao gồm những modular với các thành phần tối thiểu, do đó bạn giữ được tính linh hoạt trong quá trình xây dựng các giải pháp của mình.*
* *Bạn có thể phát triển và chạy các ứng dụng đa nền tảng từ****ASP.NET Core****trên* ***Windows****,* ***Mac*** *và* ***Linux****.*

ASP.NET là một nền tảng web được sử dụng để tạo các trang web, ứng dụng và dịch vụ web. Đây là sản phẩm tích hợp của các ngôn ngữ HTML, CSS và JavaScript. ASP.NET hoạt động trên HTTP, sử dụng các lệnh và chính sách của HTTP để trình duyệt có thể trở thành giao tiếp song phương của máy chủ.

*ASP.NET cung cấp những phát triển để tạo ứng dụng web như:*

* **Web Forms**: ASP.NET cung cấp tương tác hướng sự kiện cho các ứng dụng web, nó được sử dụng để phát triển ứng dụng có quyền truy cập dữ liệu. Đồng thời, kiểu phát triển web này cũng sẽ cung cấp dữ liệu cho máy chủ và sự kiện để tạo nên ứng dụng phù hợp. Web Forms được phát triển một cách nhanh chóng nhờ vào sử dụng thư viện điều khiển phong phú đánh dấu HTML.
* **ASP.NET MVC**: cung cấp các mẫu MVC (Model View Controller) để xây dựng lên trang web động. Những mẫu này được sử dụng cho dữ liệu (Model), giao diện người dùng (View) và logic ứng dụng (Controller). Mô hình này sẽ đảm nhận nhiệm vụ duy trì dữ liệu và logic của ứng dụng. Chế độ xem sẽ là giao diện người dùng và hiển thị dữ liệu. Bộ điều khiển sẽ xử lý yêu cầu của người dùng về phần View và Model này.
* **ASP.NET web Pages**: được sừ dụng để tạo ra các trang web động kết hợp máy chủ  với HTML một cách nhanh chóng. ASP.NET web Pages sẽ đánh dấu mã của bạn và HTML cùng nhau trong cùng một tệp.

## **II.iii. Thành phần của ASP.NET**

*Các thành phần chính của nền tảng ASP.NET cơ bản gồm:*

* **Ngôn ngữ**: ASP.NET sử dụng nhiều ngôn ngữ lập trình như **VB.NET** và **C#**.
* **Thư viện**: ASP.NET có bộ thư viện chuẩn bao gồm các giao diện, các lớp và kiểu giá trị. Bộ thư viện này có thể sử dụng lại cho quá trình phát triển ASP.NET và xây dựng các chức năng của hệ thống.
* **Thời gian chạy ngôn ngữ chung (CLR)**: CLR - Common Language Runtime được sử dụng để thực hiện các hoạt động mã. Các hoạt động này sẽ thực hiện xử lý các ngoại lệ và thu gom rác.

## **II.iv. Ưu điểm của ASP.NET Core**

**Xây dựng web UI và web API với ASP.NET Core MVC**

* Người dùng có thể tạo ra những ứng dụng web có thể thực hiện testing theo mô hình MVC.
* Xây dựng HTTP services hỗ trợ nhiều định dạng và đầy đủ những hỗ trợ cho nội dung của dữ liệu trả về.
* Razor cung cấp ngôn ngữ tạo Views hiệu quả.
* Tag Helper cho phép code server side tham gia vào quá trình tạo và render phần tử HTML.
* Model Binding có thể tự động ánh xạ dữ liệu từ HTTP request tới những tham số của method action.
* Model validation thực hiện validate client và server một cách tự động.

***Phát triển client-side: ASP.NET Core được thiết kế tích hợp với nhiều client side frameworks một cách liên tục bao gồm AngularJS, Bootstrap và KnockoutJS.***

**Ngoài ra, ASP.NET Core còn được đánh giá cao khi mang trong mình nhiều lợi ích khác như:**

* Là khung mẫu và gọn gàng hơn kiến trúc nhiều tầng.
* Là một khung công tác với mã nguồn mở.
* Dễ dàng tạo ứng dụng ASP.NET đa nền tảng trên Windows, Mac và Linux.
* Cấu hình là môi trường sẵn sàng cho đám mây.
* Khả năng lưu trữ trên: IIS, HTTP.sys, Kestrel, Nginx, Apache và Docker.
* Có một số thay đổi kiến trúc dẫn đến modular framework nhỏ hơn.
* ASP.NET Core không còn dựa trên System.Web.dll. Nó dựa trên một tập hợp nhiều yếu tố của Nuget packages.
* Điều này cho phép bạn tối ưu ứng dụng của mình chỉ cần những NuGet packages cần thiết.
* Lợi ích của diện tích bề mặt ứng dụng nhỏ hơn thì bảo mật chặt chẽ hơn, giảm dịch vụ, cải thiện hiệu suất và giảm chi phí.

**Với ASP.NET Core, bạn có thể nhận được các cải tiến sau:**

* Xây dựng và chạy các ứng dụng ASP.NET Core đa nền tảng trên Windows, Mac, Linux.
* Được xây dựng trên **.NET CORE**, hỗ trợ side-by-side app versioning.
* Công cụ mới giúp đơn giản hóa việc phát triển web hiện đại.
* Liên kết đơn các web stack như Web UI và API Web.
* Cấu hình dựa trên môi trường đám mây sẵn có.
* Được xây dựng dựa trên cho DI (Dependency Injection).
* Tag Helpers làm cho các Razor makup trở nên tự nhiên hơn với HTML.
* Có khả năng host trên IIS hoặc self-host.

## **II.v. Sự khác nhau giữa ASP.NET và ASP.NET Core**

**ASP.NET**

* Được xây dựng chỉ để dành cho Windows.
* Có hiệu suất tốt
* Có thể chạy được trên .Net Framework hay được gọi là Full .Net Framework.
* Asp.Net hỗ trợ Web Forms, Asp.Net MVC và ASP.NET web Pages.
* Chỉ sử dụng IIS phụ thuộc vào System.web.dll.
* Sử dụng ngôn ngữ C#, VB, WCF, WPF và WF.
* Asp.Net MVC có các ứng dụng Web.config, Global.asax, Application Start.
* Hỗ trợ vùng chứa không được đánh giá quá cao.
* Tất cả các phiên bản chính thức đều được hỗ trợ.
* Người dùng cần biên dịch lại sau khi thay đổi mã.
* Người dùng không mất nhiều thời gian để tìm hiểu.

**ASP.NET Core**

* Được xây dựng dành cho cả Windows, Mac và Linux.
* Có hiệu suất cao hơn cả ASP.NET 4x.
* Có thể chạy trên .Net Core và Full .Net Framework.
* Asp.Net Core hỗ trợ cho các trang Web MVC, Web API và Asp.Net được thêm vào ban đầu trong .Net Core 2.0, không hỗ trợ cho Web Forms.
* Asp.Net Core không phụ thuộc vào IIS và System.web.dll.
* Chỉ hỗ trợ ngôn ngữ C#, F# và VB trong thời gian ngắn, không hỗ trợ WCF, WPF và WF. Tuy nhiên, Asp.Net Core lại có thể hỗ trợ cho các thư viện WCP có sẵn.
* Asp.Net Core đang hỗ trợ Appsettings.json, không hỗ trợ tệp Web.config và Global.asax.
* Hỗ trợ vùng chứa phù hợp cho các triển khai như Docker.
* Hỗ trợ Core từ Visual Studio 2015 cập nhật lần thứ 3 và phiên bản Visual Studio 2017.
* Khi làm mới Core Browser sẽ tự động biên dịch và thực thi mã mà không cần phải dịch lại.
* Cần nhiều thời gian tìm hiểu trước khi sử dụng vì Asp.Net Core phức tạp hơn với nhiều tính năng mở hơn.

# **Chapter III. ASP.NET Core MVC**

## **Mô hình MVC là gì?**

-Mô hình MVC là một kiến trúc thiết kế của một phần mềm công nghê thông tin, trong đó có cả PHP. Khi một phần mềm sử dụng mô hình này thì nó sẽ phải chia ra thành ba phần chính Model-View-Controller (MVC) với  3 nhiệm vụ khác nhau:

–**Model**: là thành phần chứa tất cả các phương thức xử lý, truy xuất database, đối tượng mô tả dữ liệu như các Class, hàm xử lý…

–**View**: Là phần đảm nhận việc hiển thị thông tin, tương tác với người dùng. Hoặc chúng ta có thể hiểu nôm na đó là phần giao diện của ứng dụng.

–**Controller**: Controller là phần xử lý và điều hướng các hành động của client, từ đó đưa ra các xử lý với database nếu có. Hay nói các khác thì **controller là cầu nối giữa view và Model**.

## **SP.NET Core MVC**

ASP.NET Core MVC là một framework “nhẹ”, opensource, giúp tối ưu hóa hiệu năng của ứng dụng với  [ASP.NET Core](https://sinhvientot.net/asp-net-core-buoc-dot-pha-trong-cong-nghe-net/).

ASP.NET Core MVC cung cấp các tính năng dựa trên mô hình xây dựng website động cho phép phân chia rõ ràng các khối lệnh. Nó cung cấp cho bạn toàn quyền kiểm soát đánh dấu, hỗ trợ phát triển  với TDD-friendly và sử dụng các tiêu chuẩn web mới nhất.

Tính năng của ASP.NET Core MVC bao gồm:

* Routing
* Model binding
* Model validation
* Dependency injection
* Filters
* Areas
* Web APIs
* Testability
* Razor view engine
* Strongly typed views
* Tag Helpers
* View Components

**Routing**

Routing sẽ xác định URL và điều khiển thông tin tương ứng với URL mà người dùng web nhập vào. Tất cả các cấu hình Routing của một ứng dụng ASP.NET MVC được lưu trữ trong RouteTable, nó đóng vai trò định tuyến các URL để xác định các lớp xử lý tương ứng khi có request được gửi đến từ người dùng web.

Ví dụ:

routes.MapRoute(name: "Default", template: "{controller=Home}/{action=Index}/{id?}");

[Route("api/[controller]")]

public class ProductsController : Controller

{

[HttpGet("{id}")]

public IActionResult GetProduct(int id)

{

...

}

}

### **Model binding**

Trong ASP.NET Core MVC Model binding chuyển đổi dữ liệu yêu cầu từ phía client (form values, route data, query string parameters, HTTP headers) vào bên trong đối tượng để controller có thể xử lí. Kết quả là Controller của bạn không phải thực hiện công việc tìm ra dữ liệu từ đâu đến. Nó chỉ đơn giản là kiểm tra dữ liệu và tham số từ phương thức action.

public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null) { ... }

### **Model validation**

ASP.NET Core MVC  hỗ trợ ràng buộc dữ liệu cho các thuộc tính trong model của bạn, các thuộc tính sẽ được kiểm tra ở client xem có hợp lệ không trước khi giá trị của thuộc tính đó được gửi về server. Cũng như trên server trước khi action của controller gọi.

using System.ComponentModel.DataAnnotations;

public class LoginViewModel

{

[Required]

[EmailAddress]

public string Email { get; set; }

[Required]

[DataType(DataType.Password)]

public string Password { get; set; }

[Display(Name = "Remember me? on sinhvientot.net")]

public bool RememberMe { get; set; }

}

Và model được gọi trong action của controller:

public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null)

{

if (ModelState.IsValid)

{

// work with the model

}

// At this point, something failed, redisplay form

return View(model);

}

### **Dependency injection**

Trong ASP.NET Core MVC controller có thể gửi yêu cầu cần thiết đến các service thông cấu trúc xây dựng của chúng.

@inject SomeService ServiceName

<!DOCTYPE html>

<html lang="en">

<head>

<title>@ServiceName.GetTitle</title>

</head>

<body>

<h1>@ServiceName.GetTitle</h1>

</body>

</html>

### **Filters**

Filters giúp các lập trình viên đóng gói “cross-cutting concerns”. giống như là xử lý ngoại lệ và phân quyền. Filter được kích hoạt để chạy trước và sau các action của controller. Ví dụ về phân quyền [Authorize] được đặt trước action.

[Authorize]

public class AccountController : Controller

{

### **Areas**

Khi một website ASP.NET MVC trở nên quá lớn quá và phức tạp,số Controller chắc chắn sẽ tăng lên, với nhiều controller như vậy bạn sẽ thấy chúng có thể thuộc về một nhóm như phần Administrator, phần sản phẩm, phần nhóm sản phẩm… Areas cho phép bạn chia các Controllers,models và views tới các vị trí khác nhau trong solution với cùng một thư mục độc lập.

### **Web APIs**

Ngoài việc là một nền tảng tuyệt vời để xây dựng các trang web. ASP.NET Core MVC hỗ trợ rất nhiều cho việc xây dựng API Web. Bạn có thể xây dựng các dịch vụ tiếp cận nhiều khách hàng bao gồm trình duyệt web và thiết bị di động.

### **Testability**

The framework’s use of interfaces and dependency injection make it well-suited to unit testing

Việc xây dựng các framework’s sử dụng các interface và DI làm cho việc kiểm tra ứng dụng trở nên dễ dàng hơn và framework’s  bao gồm các tính năng (like a TestHost and InMemory provider for Entity Framework) chúng giúp ích cho lập trình viên rất nhiều.

### **Razor view engine**

ASP.NET Core MVC views sử dụng Razor view engine để render các code html đến view.Razor được sử dụng để tự động tạo nội dung web trên máy chủ.

<ul>

@for (int i = 0; i < 5; i++) {

<li>List item @i</li>

}

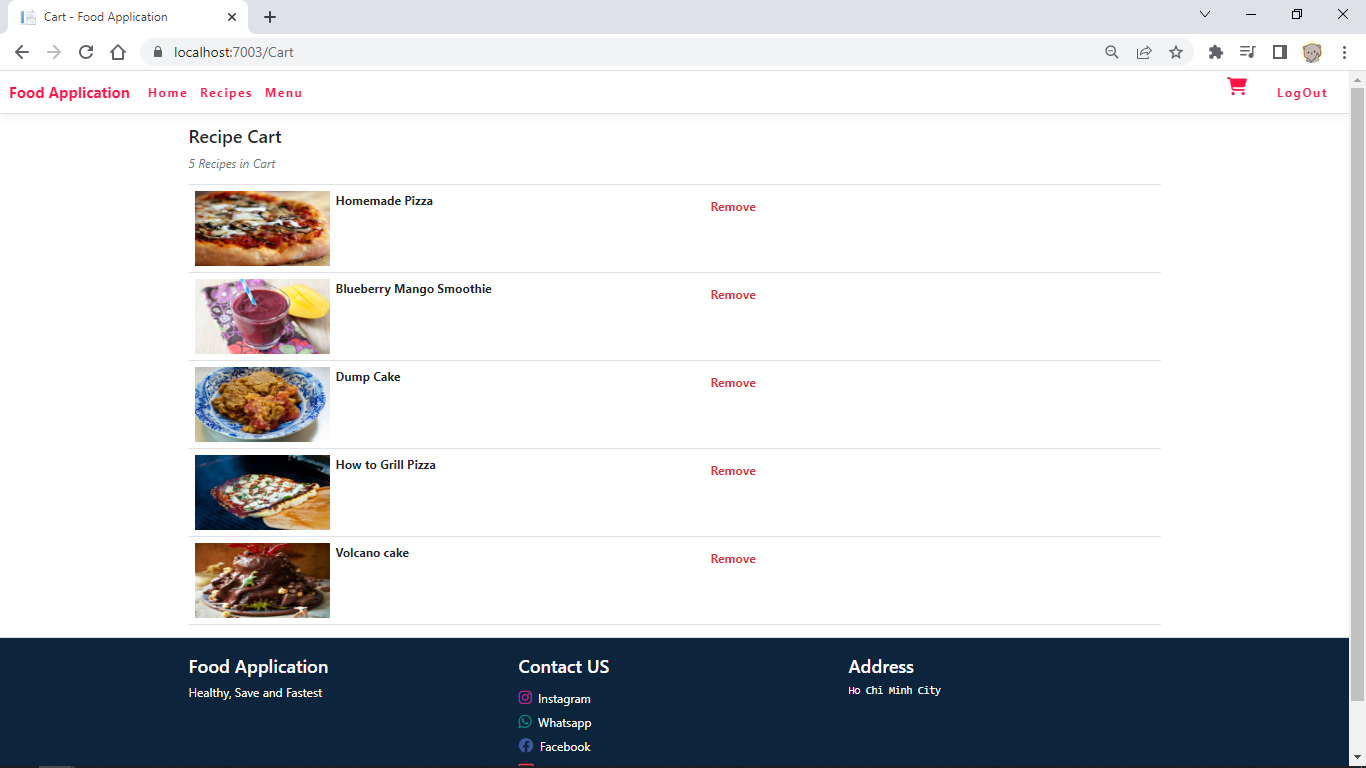
</ul>

### **Tag Helpers**

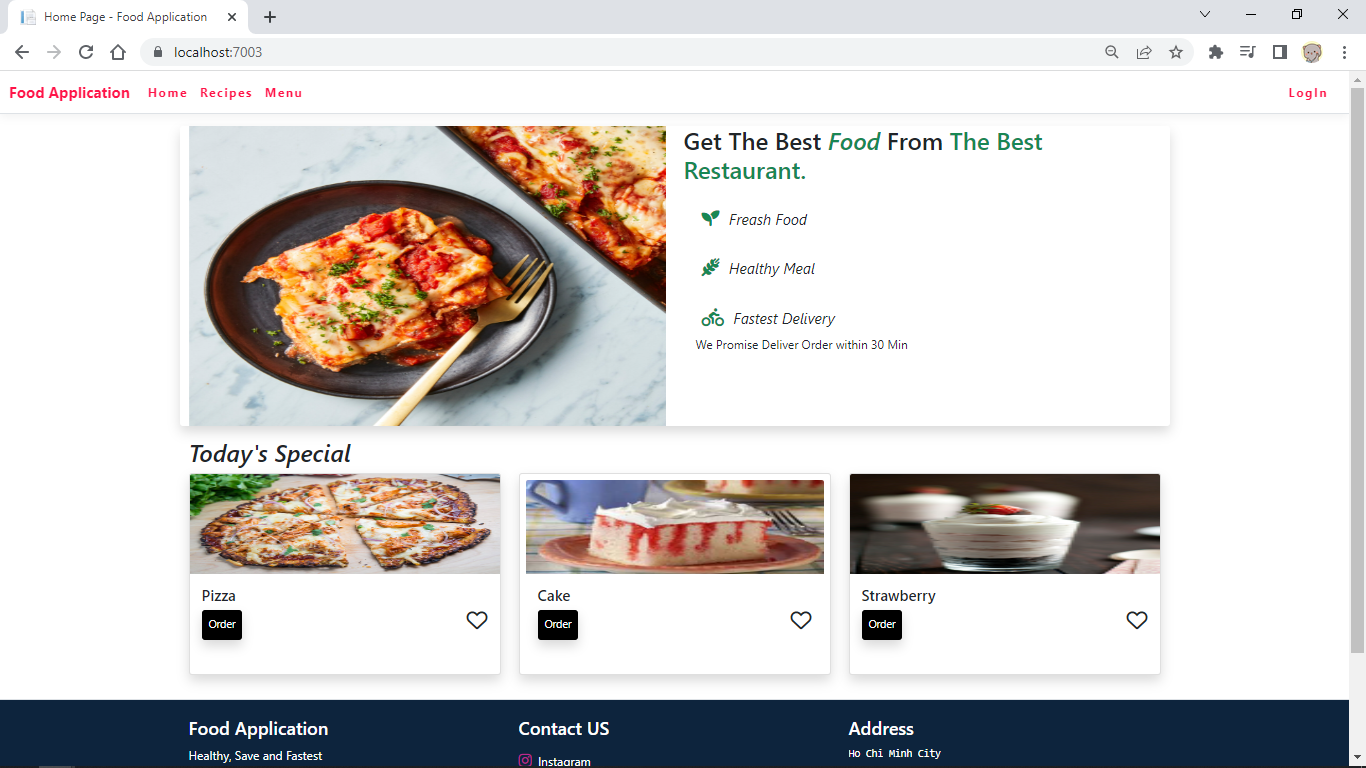
Tag Helpers cho phép mã phía máy chủ tham gia vào việc tạo và hiển thị các phần tử HTML trong các tệp razor.

Đây là một số khái niệm cơ bản trong ASP.NET Core MVC cho người mới học cần nắm rõ để có thể hiểu hết bản chất khi xây dựng ứng dụng.

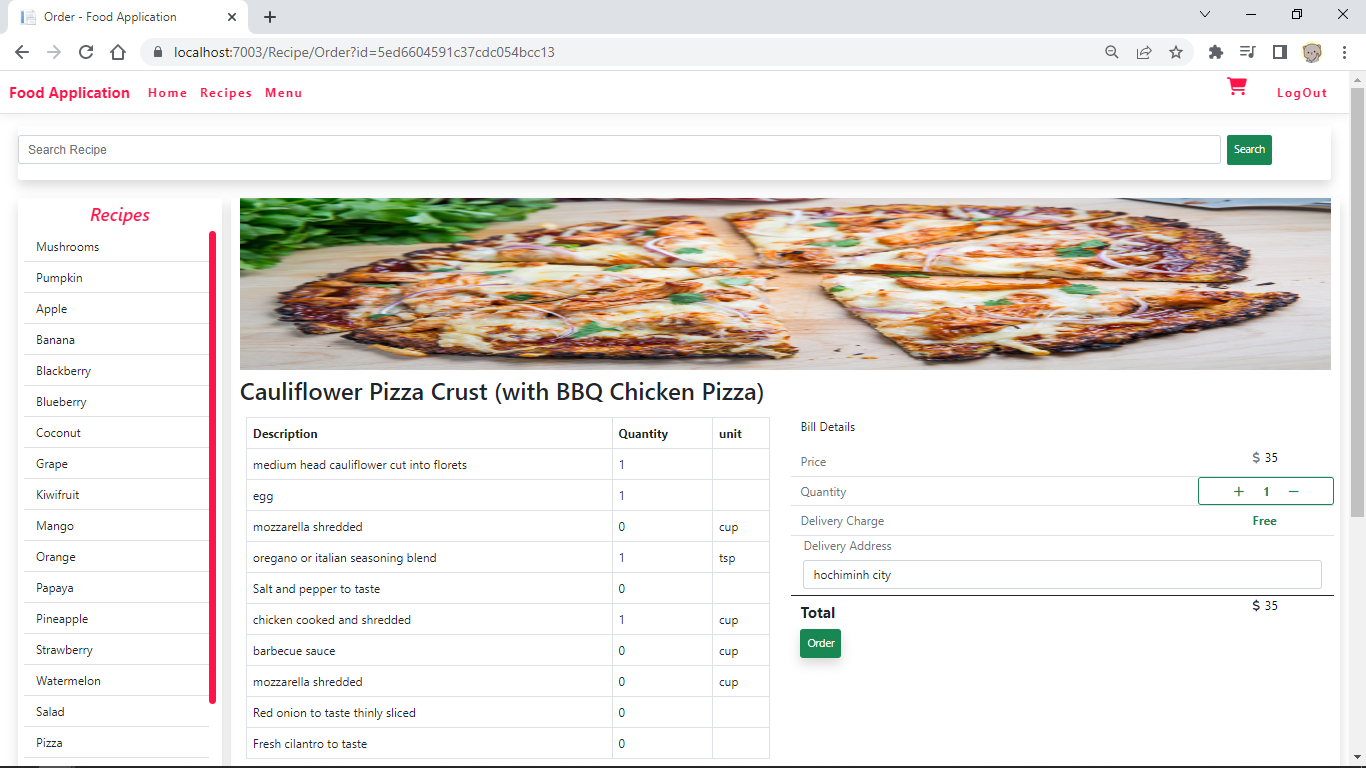
# **Chapter VI. Food Application project**



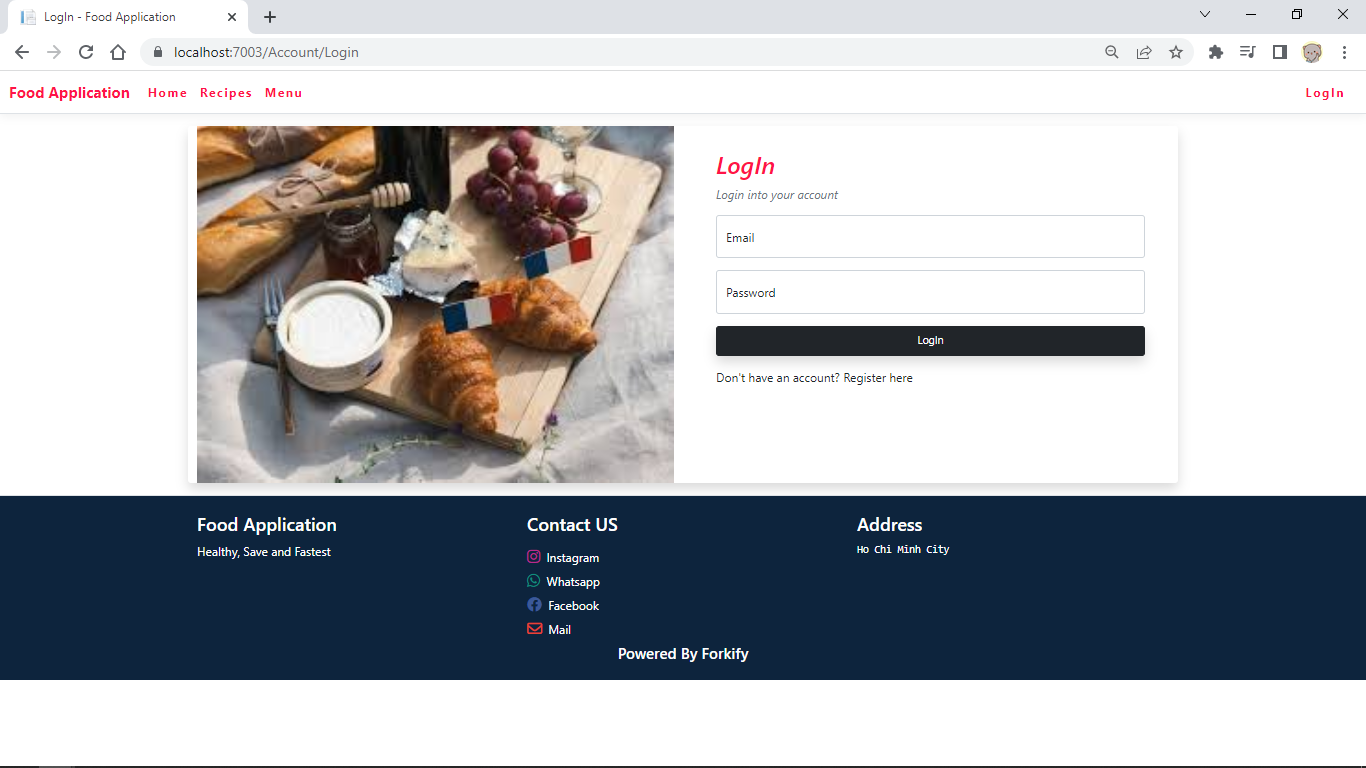
**Cart**



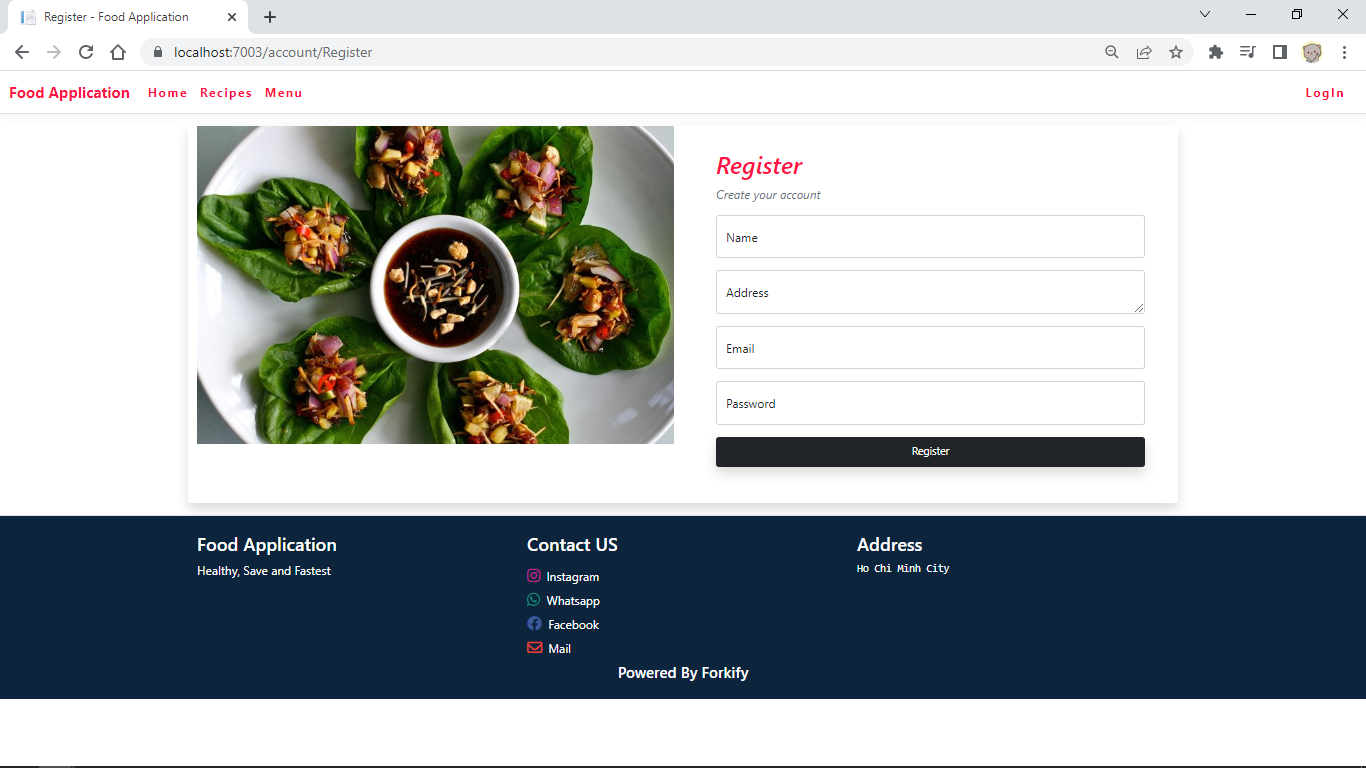
**Homepage**



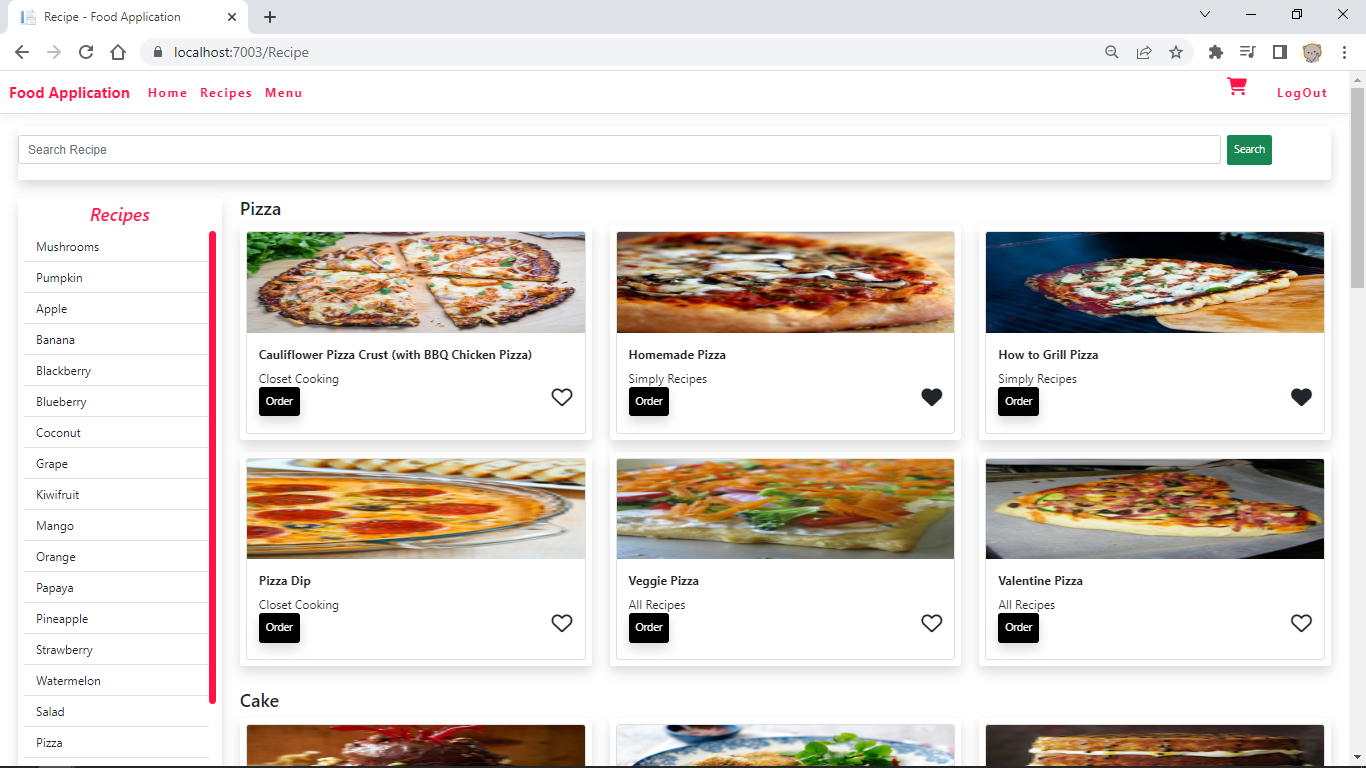
**Order**



**Login**



**Register**



**Recipes**

# **Tài liệu tham khảo**

<https://vietnix.vn/tim-hieu-mo-hinh-mvc-la-gi/>

<https://viblo.asia/p/co-ban-ve-mo-hinh-mvc-6J3ZgReRKmB>

<https://sinhvientot.net/asp-net-core-mvc-tong-quan-ve-asp-net-core-mvc/>