TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO GIỮA KỲ MÔN**

**LẬP TRÌNH WEB VÀ ỨNG DỤNG**

*Người hướng dẫn:* **THS.NCS. VŨ ĐÌNH HỒNG**

*Người thực hiện:***ĐINH PHƯƠNG MY – 52100703**

**ĐẶNG NHƯ QUỲNH – 52100099**

Khoá**: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO GIỮA KỲ MÔN**

**LẬP TRÌNH WEB VÀ ỨNG DỤNG**

*Người hướng dẫn:* **THS.NCS. VŨ ĐÌNH HỒNG**

*Người thực hiện:***ĐINH PHƯƠNG MY – 52100703**

**ĐẶNG NHƯ QUỲNH – 52100099**

Khoá**: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

Lời nói đầu tiên, nhóm em xin được gửi lời cảm ơn chân thành đến toàn bộ giảng viên Trường Đại học Tôn Đức Thắng nói chung cũng như toàn bộ giảng viên Khoa Công nghệ thông tin nói riêng vì đã tạo điều kiện cho nhóm em được học bộ môn Lập trình Web và ứng dụng. Và đặc biệt nhóm em gửi lời cảm ơn chân thành nhất đến thầy Vũ Đình Hồng – giảng viên giảng dạy và hướng dẫn cho đề tài môn môn Lập trình Web và ứng dụng của nhóm em. Trong suốt quá trình học tập và thực hiện bài báo cáo, thầy luôn giúp đỡ, chỉ bảo tận tình để nhóm em có thể tìm ra cách giải quyết những vướng mắc gặp phải và hoàn thiện đề tài này một cách tốt nhất. Lượng kiến thức này nhóm em sẽ làm hành trang để áp dụng vào công việc sau này. Không thể nói gì hơn nữa, một lần nữa, bằng cả tấm lòng, nhóm em xin chân thành gửi lời cảm ơn sâu sắc đến thầy - người đã dìu dắt lớp trong suốt chặng đường vừa qua!

**BÁO CÁO ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Em xin cam đoan đây là công trình nghiên cứu của riêng nhóm em và được sự hướng dẫn khoa học của thầy Vũ Đình Hồng. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong báo cáo còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào em xin hoàn toàn chịu trách nhiệm về nội dung báo cáo của nhóm mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do nhóm em gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 27 tháng 04 năm 2023*

*Sinh viên thực hiện*

*(ký tên và ghi rõ họ tên)*

*Đinh Phương My*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Trong báo cáo này, nhóm em sẽ trình bày một khía cạnh quan trọng của phát triển ứng dụng web - mô hình MVC, cùng với cơ chế hoạt động của ASP.NET Core và việc áp dụng các kiến thức này để xây dựng một trang web minh họa.

Để thực hiện điều này, báo cáo được phân thành bốn chương khác nhau với mục đích và nội dung riêng biệt:

Chương 1 sẽ giới thiệu chi tiết về mô hình MVC (Model-View-Controller) và cách nó được áp dụng trong phát triển web. Nhóm em sẽ giải thích mô hình này và những ưu điểm của nó, cũng như cách sử dụng nó để tăng tính linh hoạt và hiệu quả cho quá trình phát triển ứng dụng web.

Chương 2 sẽ trình bày về ASP.NET Core, một framework phát triển ứng dụng web mã nguồn mở và đa nền tảng. Nhóm em sẽ giới thiệu cơ chế hoạt động của ASP.NET Core, cùng với những tính năng và lợi ích của nó để xây dựng các ứng dụng web mạnh mẽ và bảo mật.

Chương 3 sẽ tập trung vào việc triển khai một trang web minh họa bằng cách sử dụng các kiến thức về mô hình MVC và ASP.NET Core mà nhóm em đã tìm hiểu. Nhóm em sẽ giải thích cách thiết kế, phát triển và triển khai trang web, đồng thời cung cấp các hướng dẫn chi tiết về các công nghệ và công cụ sử dụng để thực hiện điều này.

Cuối cùng, chương 4 sẽ là phần kết luận, tổng kết lại các nội dung và kết quả của báo cáo. Nhóm em sẽ đưa ra những nhận xét và suy nghĩ của mình về mô hình MVC, ASP.NET Core và các kinh nghiệm của nhóm em trong quá trình triển khai trang web minh họa.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc133698165)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc133698166)

[TÓM TẮT iv](#_Toc133698167)

[MỤC LỤC 2](#_Toc133698168)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4](#_Toc133698169)

[CHƯƠNG 1 – Giới thiệu về MÔ HÌNH MVC 5](#_Toc133698170)

[1.1 Mô hình MVC là gì? 5](#_Toc133698171)

[1.2 Thành phần trong mô hình MVC 5](#_Toc133698172)

[1.2.1 Model 5](#_Toc133698173)

[1.2.2 View 5](#_Toc133698174)

[1.2.3 Controller 5](#_Toc133698175)

[1.3 Luồng xử lý trong mô hình MVC 6](#_Toc133698176)

[1.4 Ưu nhược điểm của mô hình MVC 7](#_Toc133698177)

[1.4.1 Ưu điểm 7](#_Toc133698178)

[1.4.2 Nhược điểm 7](#_Toc133698179)

[CHƯƠNG 2 – ASP.NET Core 9](#_Toc133698180)

[2.1 Giới thiệu ASP.NET Core 9](#_Toc133698181)

[2.2 Lịch sử phát triển 9](#_Toc133698182)

[2.3 Cấu trúc của ASP.NET 9](#_Toc133698183)

[2.3.1 Application Framework 10](#_Toc133698184)

[2.3.2 Utility Frameworks 11](#_Toc133698185)

[2.3.3 Platform 11](#_Toc133698186)

[2.4 Cách thức hoạt động 13](#_Toc133698187)

[2.5 Ưu điểm và nhược điểm 14](#_Toc133698188)

[2.5.1 Ưu điểm 15](#_Toc133698189)

[2.5.2 Nhược điểm 15](#_Toc133698190)

[CHƯƠNG 3 - Triển khai WEB MINH HỌA 16](#_Toc133698191)

[3.1 Công cụ hỗ trợ và cài đặt 16](#_Toc133698192)

[3.1.1 Công cụ hỗ trợ 16](#_Toc133698193)

[3.1.2 Cài đặt 16](#_Toc133698194)

[3.2 Xây dựng giao diện 16](#_Toc133698195)

[3.3 Xây dựng các chức năng 17](#_Toc133698196)

[3.3.1 Thêm 18](#_Toc133698197)

[3.3.2 Sửa 19](#_Toc133698198)

[3.3.3 Xóa 20](#_Toc133698199)

[CHƯƠNG 4 - KẾT LUẬN 21](#_Toc133698200)

[4.1 Kết quả đạt được 21](#_Toc133698201)

[4.2 Hướng phát triển 21](#_Toc133698202)

[TÀI LIỆU THAM KHẢO 22](#_Toc133698203)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1.3 Luồng xử lý căn bản của mô hình MVC 7](#_Toc133697800)

[Hình 2.3 Cấu trúc ASP.Net Core 11](#_Toc133697801)

[Hình 2.4 Cách thức hoạt động ASP.Net Core 14](#_Toc133697802)

[Hình 3.2.1 Hình ảnh giao diện khi mới truy cập vào trang web 17](#_Toc133697803)

[Hình 3.2.2 Hình ảnh giao diện Product 18](#_Toc133697804)

[Hình 3.2.3 Hình ảnh giao diện About 18](#_Toc133697805)

[Hình 3.2.4 Hình ảnh giao diện Contact 18](#_Toc133697806)

[Hình 3.3.1.1 Hình ảnh giao diện khi thêm sản phẩm 19](#_Toc133697807)

[Hình 3.3.1.2 Hình ảnh khi thêm dữ liệu thành công 19](#_Toc133697808)

[Hình 3.3.2.1 Hình ảnh giao diện khi sửa sản phẩm 20](#_Toc133697809)

[Hình 3.3.2.2 Hình ảnh sửa dữ liệu thành công 20](#_Toc133697810)

[Hình 3.3.3.1 Hình ảnh giao diện khi xóa sản phẩm 21](#_Toc133697811)

[Hình 3.3.3.2 Hình ảnh xóa dữ liệu thành công 21](#_Toc133697812)

**DANH MỤC BẢNG**

CHƯƠNG 1 – Giới thiệu về MÔ HÌNH MVC

1.1 Mô hình MVC là gì?

Mô hình thiết kế hay kiến trúc phần mềm MVC (Model View Controller) được phát triển từ năm 1970 bởi tiến sĩ Trygve Reenskaug tại trung tâm nghiên cứu Xerox Palo Alto. Đây là mô hình phân bố source code thành 3 phần độc lập: Model, View và Controller. Mỗi phần có nhiệm vụ riêng biệt, giúp cho các nhà thiết kế, lập trình web hoàn thiện sản phẩm dễ dàng và nhanh hơn.

Mô hình MVC là lựa chọn phù hợp cho lập trình hướng đối tượng. Nó giúp cho việc xử lý các yêu cầu kỹ thuật trong quá trình phát triển, vận hành và bảo trì hệ thống hay phần mềm trở nên đơn giản hơn.

Hiện nay, mô hình MVC được sử dụng rộng rãi trong lập trình web, đặc biệt là trong các PHP Framework. Nó cung cấp các thành phần cơ bản để thiết kế giao diện người dùng cho máy tính, thiết bị di động cùng các ứng dụng web.

1.2 Thành phần trong mô hình MVC

Mô hình MVC sẽ được chia thành 3 phần chính gồm Model, View và Controller, mỗi thành phần sẽ có một nhiệm vụ riêng.

1.2.1 Model

Là thành phần chịu trách nhiệm cho việc lưu trữ và xử lý dữ liệu của ứng dụng. Nó đại diện cho các đối tượng và các thao tác trên chúng. Thông thường, các đối tượng được lưu trữ trong cơ sở dữ liệu, và các thao tác trên chúng được thực hiện thông qua các truy vấn SQL.

1.2.2 View

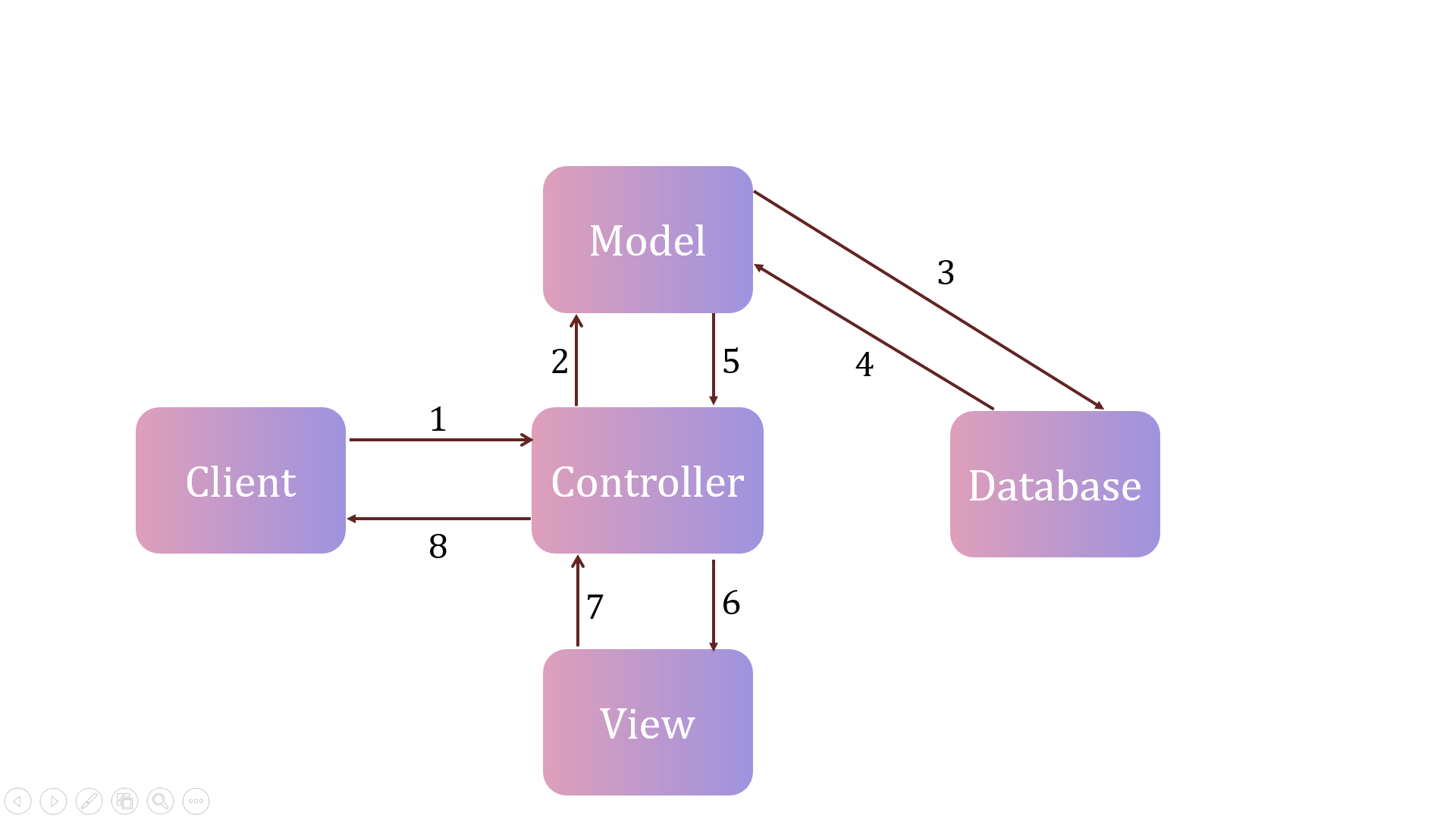
Là thành phần chịu trách nhiệm cho việc hiển thị các dữ liệu cho người dùng. Nó đại diện cho giao diện người dùng của ứng dụng. Các View có thể được hiển thị dưới dạng các trang HTML, các tệp tin CSS, JavaScript, vv..

1.2.3 Controller

Là thành phần chịu trách nhiệm cho việc điều hướng các yêu cầu của người dùng đến các thành phần phù hợp của ứng dụng. Nó đại diện cho các đối tượng quản lý các tương tác người dùng và cập nhật các đối tượng Model và View tương ứng.

1.3 Luồng xử lý trong mô hình MVC

MVC là một kiến trúc được áp dụng rộng rãi trong phát triển ứng dụng web. Luồng xử lý căn bản và phổ biến nhất của MVC bao gồm các bước sau:



Hình 1.3 Luồng xử lý căn bản của mô hình MVC

* Bước 1: Người dùng gửi yêu cầu đến server thông qua Controller. Controller nhận yêu cầu và tiếp nhận dữ liệu đầu vào.
* Bước 2: Controller xử lý dữ liệu đầu vào và quyết định luồng xử lý tiếp theo của yêu cầu. Nếu cần truy xuất dữ liệu từ database, Controller gọi tới Model để lấy dữ liệu. Nếu không, Controller trả về kết quả ngay lập tức.
* Bước 3: Model tương tác với Database để truy xuất dữ liệu phù hợp với yêu cầu.
* Bước 4: Database trả về dữ liệu cho Model.
* Bước 5: Model trả về dữ liệu cho Controller để xử lý tiếp.
* Bước 6: Controller gọi đến View phù hợp với yêu cầu và kèm theo dữ liệu. View chịu trách nhiệm hiển thị dữ liệu phù hợp với yêu cầu.
* Bước 7: View trả về kết quả cho Controller (HTML, XML hoặc JSON…).
* Bước 8: Sau khi hoàn tất, Controller trả về kết quả cho người dùng.

Mỗi bước trong luồng xử lý có thể liên kết với các bước khác để xử lý yêu cầu của người dùng. Sử dụng kiến trúc MVC giúp cho quá trình phát triển ứng dụng web trở nên dễ dàng quản lý và bảo trì, cũng như tách biệt giữa các thành phần của ứng dụng.

1.4 Ưu nhược điểm của mô hình MVC

1.4.1 ****Ưu điểm****

* Dễ dàng bảo trì và mở rộng ứng dụng: Các thành phần của ứng dụng được tách biệt ra thành các phần riêng lẻ, điều này giúp cho việc bảo trì và mở rộng ứng dụng trở nên dễ dàng hơn. Nếu có thay đổi trong cơ sở dữ liệu hoặc giao diện người dùng, chỉ cần thay đổi ở thành phần tương ứng mà không ảnh hưởng đến các thành phần khác.
* Tăng tính linh hoạt: Các thành phần của ứng dụng có thể được tái sử dụng và kết hợp lại với nhau để tạo ra nhiều tính năng khác nhau.
* Dễ dàng phát triển: Với Mô hình MVC, các nhà phát triển có thể làm việc độc lập với nhau trên các thành phần khác nhau của ứng dụng mà không cần quan tâm đến các thành phần khác. Điều này giúp cho việc phát triển ứng dụng trở nên nhanh chóng và hiệu quả hơn.
* Cải thiện bảo mật: Với Mô hình MVC, các thành phần của ứng dụng được tách biệt ra, điều này giúp cho việc xử lý các yêu cầu của người dùng trở nên an toàn hơn. Các đối tượng Model không thể được truy cập trực tiếp từ bên ngoài, các yêu cầu của người dùng phải được điều hướng qua Controller để được xử lý và kiểm tra trước khi được cập nhật vào Model.

1.4.2 Nhược điểm

* Khó khăn trong việc triển khai: Đối với các dự án lớn và phức tạp, triển khai mô hình MVC có thể gặp khó khăn và yêu cầu kỹ năng cao từ các nhà phát triển. Các thành phần của ứng dụng phải được định nghĩa và tổ chức một cách chặt chẽ để đảm bảo tính nhất quán và hiệu quả của toàn bộ ứng dụng.
* Tốn nhiều thời gian và công sức để quản lý và tổ chức các file: Vì mô hình MVC yêu cầu các thành phần của ứng dụng được tổ chức theo cách đặc biệt, do đó quản lý và tổ chức các file có thể là một thách thức cho các nhà phát triển.
* Không thể xem trước các trang web: Trong mô hình MVC, các trang web được tạo ra bằng cách kết hợp các thành phần khác nhau, do đó, không thể xem trước toàn bộ trang web trực tiếp như trong ASP.NET Web Forms.
* Khó khăn khi bắt đầu: Mô hình MVC đòi hỏi sự hiểu biết về kiến trúc phần mềm và các kỹ thuật phát triển web, do đó, nó có thể khó khăn cho các nhà phát triển mới bắt đầu hoặc không có nhiều kinh nghiệm.
* Cần nhiều tài nguyên hơn: Mô hình MVC yêu cầu các thành phần được phát triển và kiểm thử riêng biệt, do đó, nó cần nhiều tài nguyên hơn để triển khai và phát triển ứng dụng so với các kiểu kiến trúc khác.

CHƯƠNG 2 – ASP.NET Core

2.1 Giới thiệu ASP.NET Core

ASP.NET Core là một framework phát triển ứng dụng web được phát triển bởi Microsoft, được thiết kế để chạy trên nhiều nền tảng khác nhau, bao gồm Windows, Linux và macOS. Nó là phiên bản tiếp theo của ASP.NET Framework và được thiết kế lại để cung cấp nhiều cải tiến về hiệu suất, bảo mật và tính linh hoạt.

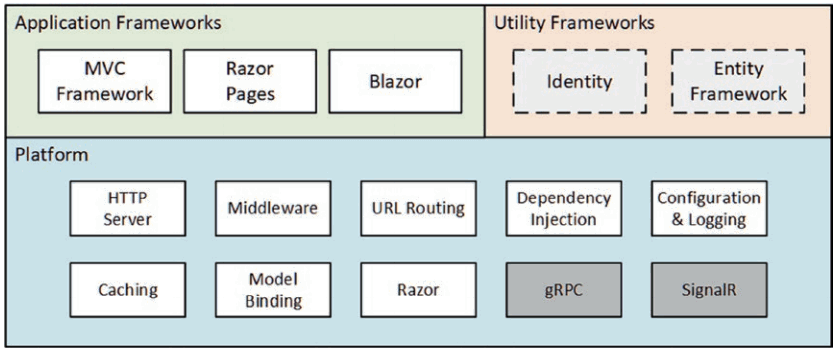
2.2 Lịch sử phát triển

ASP.NET Core là một framework phát triển ứng dụng web đa nền tảng, được phát triển dựa trên các phiên bản trước đó của ASP.NET, bao gồm ASP.NET Web Forms, ASP.NET MVC và ASP.NET Web API. Phiên bản đầu tiên của ASP.NET Core được ra mắt vào năm 2016 với tên gọi ASP.NET 5. Sau đó, Microsoft đã quyết định đổi tên nó thành ASP.NET Core và ra mắt phiên bản đầu tiên của nó vào cùng năm.

Kể từ đó, ASP.NET Core đã trải qua nhiều cải tiến và bổ sung tính năng. Phiên bản 2.0 được phát hành vào năm 2017, phiên bản 2.1 được phát hành vào năm 2018, phiên bản 2.2 được phát hành vào năm 2018, phiên bản 3.0 được phát hành vào năm 2019, phiên bản 3.1 được phát hành vào năm 2019, và phiên bản mới nhất là 6.0 được phát hành vào năm 2021. Các phiên bản này đều cung cấp nhiều tính năng mới và cải thiện hiệu suất của ASP.NET Core.

2.3 Cấu trúc của ASP.NET

Cấu trúc Asp.net Core bao gồm 3 phần: Application Frameworks, Utility Frameworks, và Platform



Hình 2.3 Cấu trúc ASP.Net Core

(Nguồn: [Cấu trúc của ASP.Net Core](https://tuhocict.com/cau-truc-asp-net-core/))

2.3.1 Application Framework

Application Frameworks chứa những cái tên có lẽ tương đối quen thuộc như MVC Framework, Razor Pages hay Blazor. Đây là những framework giúp bạn xây dựng các dạng khác nhau của ứng dụng web. Các framework này giúp giải quyết các loại vấn đề khác nhau, và giải quyết cùng một vấn đề theo những cách thức khác nhau. Đây là phần nổi của Asp.net Core.

Nhìn nhận theo một cách khác, đây là những mẫu kiến trúc khác nhau cho ứng dụng web mà Microsoft hỗ trợ. Có thể lựa chọn bất kỳ framework nào phù hợp với yêu cầu của dự án và sở thích cá nhân. Do vậy, chúng không loại trừ nhau mà còn tương trợ lẫn nhau. Trong quá trình phát triển ứng dụng, có thể chuyển đổi từ framework này sang framework khác, hoặc kết hợp chúng.

* **MVC**là framework xây dựng theo nguyên lý phân chia nhiệm vụ (Separation of Concerns, SoC). MVC phân chia chức năng thành các phần của ứng dụng ra làm các nhóm độc lập, gọi là **M**odel, **V**iew và **C**ontroller. Trong Asp.net Core, cùng với sự phổ biến của ứng dụng đơn trang (Single-Page Application, SPA), MVC không còn đóng vai trò một framework chính nữa.
* **Razor Pages** là một framework đơn giản hướng tới xây dựng các trang web tự thân độc lập. Mô hình của Razor Pages đơn giản hơn so với MVC. Razor Pages có thể xem là framework tương tự với Asp.net Web Pages.
* **Blazor**là framework dành cho phát triển ứng dụng đơn trang (SPA) nhưng sử dụng ngôn ngữ C# thay cho JavaScript chạy trên trình duyệt. Đây là framework mới nhất và đang nhận được nhiều sự quan tâm. Blazor bao gồm hai mô hình chính: [Blazor Server](https://tuhocict.com/mo-hinh-hoat-dong-cua-blazor-server/) và [Blazor WebAssembly](https://tuhocict.com/mo-hinh-hoat-dong-cua-blazor-webassembly/). Ngoài ra còn có các mô hình triển khai Blazor khác đang được phát triển.

2.3.2 Utility Frameworks

Khối Utility Frameworks hai framework (không bắt buộc) nhưng lại được sử dụng gần như trong mọi ứng dụng Asp.net Core: Identity và Entity Framework. Khối này chứa những framework hỗ trợ cho ứng dụng, bao gồm bảo mật và cơ sở dữ liệu.

* **Entity Framework Core** là một ORM (Object-Relational Mapping) giúp ứng dụng tương tác với cơ sở dữ liệu. Entity Framework Core giúp ánh xạ (hai chiều) giữa các bảng cơ sở dữ liệu (ví dụ, SQL Server) với các domain class của ứng dụng. Entity Framework Core không gắn với Asp.net Core mà có thể sử dụng trong bất kỳ ứng dụng .NET nào. Entity Framework Core có thể xem là phiên bản hiện đại đa nền tảng của [Entity Framework](https://tuhocict.com/huong-dan-tu-hoc-lap-trinh-ado-net-va-entity-framework/) – ORM chính của .NET Framework cổ điển.
* **Asp.net Core Identity** là framework dành cho xác thực (authentication) và xác minh quyền (authorization) người dùng trong ứng dụng. Các ứng dụng đơn giản minh họa thường không sử dụng Identity nhưng đây là lại framework quan trọng hàng đầu với các ứng dụng thực tế.

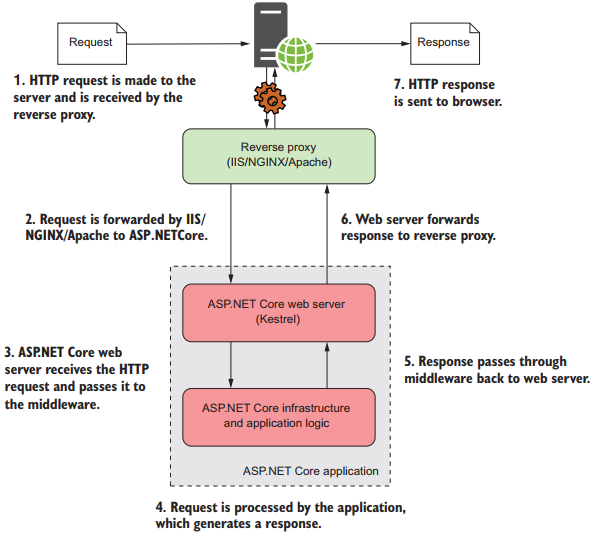
2.3.3 Platform

Khối Platform là những gì tạo nên nền tảng chung nhất mà mọi loại ứng dụng Asp.net Core đều cần sử dụng đến. Phần platform chứa nhiều thành phần cấp thấp cần thiết cho việc nhận và xử lý truy vấn HTTP, cũng như tạo ra các phản hồi phù hợp. Các thành phần của platform khá đa dạng, bao gồm:

* **HTTP Sserver**, còn gọi là built-in server với tên gọi Kestrel, có nhiệm vụ tiếp nhận truy vấn HTTP. Kestrel có thể hoạt động độc lập (tích hợp trong một ứng dụng khác) hoặc phối hợp với một web server thông thường (Apache, NGinx, IIS).
* **Middleware** là nhóm thành phần có nhiệm vụ xử lý truy vấn HTTP. Các middleware được xếp theo chuỗi. Khi truy vấn chạy qua mỗi khâu của chuỗi middle sẽ được xem xét xử lý.
* **URL Routing** là cơ chế ánh xạ chuỗi truy vấn HTTP sang thực thi một phương thức nào đó. Do vậy, mỗi URL (địa chỉ bạn gửi về server) sẽ tương ứng với thực thi một phương thức trên server.
* **Razor**là cơ chế sinh ra HTML từ dữ liệu và logic của chương trình. Razor được gọi là view engine trong Asp.net Core. Razor sử dụng loại cú pháp đặc biệt kết hợp giữa C# và HTML.
* **Model Binding** là cơ chế ánh xạ dữ liệu chứa trong truy vấn HTTP sang tham số của phương thức cần thực thi. Nhờ cơ chế Model Binding, việc xây dựng các phương thức trong Asp.net Core đơn giản như bất kỳ phương thức c# thông thường nào.
* **Dependency Injection** là cơ chế cho phép tự động sinh và chèn object vào một object khác. Asp.net Core xây dựng sẵn cơ chế này cho bạn mà không cần đến một thư viện thứ ba (như Ninject, Unity).
* **Configuration & Logging** là cơ chế hỗ trợ cấu hình và lưu vết quá trình thực thi ứng dụng. gRPC SignalR
* **Caching**là cơ chế lưu tạm để tăng hiệu suất cho ứng dụng.
* Nhìn chung, trong quá trình học, bạn sẽ lần lượt làm quen với tất cả các thành phần trên. Một số trong đó sẽ được xem xét chi tiết hơn.

2.4 Cách thức hoạt động

Ứng dụng ASP.NET Core có cách thức hoạt động khác biệt với ứng dụng ASP.NET truyền thống cũng như các loại ứng dụng web xây dựng trên các nền tảng khác. Sự khác biệt này giúp ứng dụng ASP.NET Core hoạt động đa nền tảng, hiệu quả và an toàn hơn. Cách thức hoạt động của ASP.NET Core bao gồm các bước sau:



Hình 2.4 Cách thức hoạt động ASP.Net Core

(Nguồn: [Cách thức hoạt động ASP.Net Core](https://tuhocict.com/cach-hoat-dong-cua-ung-dung-aspnet-core/))

* Bước 1: Yêu cầu (Request)

Khi một yêu cầu được gửi đến ứng dụng ASP.NET Core, nó được chuyển đến web server (IIS, Apache, Nginx, v.v) để được xử lý.

* Bước 2: Middleware

Sau khi nhận được yêu cầu, nó sẽ được chuyển đến Middleware Pipeline, là một chuỗi các thành phần middleware được đăng ký bởi ứng dụng. Các thành phần middleware này sẽ thực hiện các tác vụ khác nhau, chẳng hạn như kiểm tra xác thực, đăng ký phiên, ghi nhật ký, v.v.

* Bước 3: Route

Tiếp theo, yêu cầu được xử lý bởi Router Middleware, nó sẽ kiểm tra URL và gọi phương thức hành động (action) của Controller tương ứng để xử lý yêu cầu.

* Bước 4: Controller

Controller sẽ thực hiện các xử lý, tương tác với Model và View để trả về kết quả cho yêu cầu. Controller cũng có thể tương tác với các dịch vụ khác để lấy hoặc lưu trữ dữ liệu.

* Bước 5: Model

Model là thành phần chịu trách nhiệm xử lý dữ liệu và tương tác với cơ sở dữ liệu hoặc các dịch vụ khác để lấy hoặc lưu trữ dữ liệu.

* Bước 6: View

View là thành phần chịu trách nhiệm hiển thị dữ liệu cho người dùng. Nó tương tác với Model để lấy dữ liệu và hiển thị nó dưới dạng HTML hoặc các định dạng khác.

* Bước 7: Response

Cuối cùng, kết quả của yêu cầu được trả về cho người dùng thông qua phản hồi HTTP (HTTP Response). Kết quả có thể là trang web được hiển thị trực tiếp trong trình duyệt của người dùng hoặc các định dạng khác như JSON hoặc XML được sử dụng để chia sẻ dữ liệu giữa các ứng dụng khác. Quá trình xử lý này giúp đảm bảo ứng dụng ASP

2.5 Ưu điểm và nhược điểm

2.5.1 Ưu điểm

* Đa nền tảng: ASP.NET Core cho phép phát triển ứng dụng web trên nhiều nền tảng khác nhau, bao gồm Windows, Linux và macOS.
* Hiệu suất cao: ASP.NET Core được thiết kế để cung cấp hiệu suất cao, cho phép xử lý các yêu cầu HTTP nhanh chóng và hiệu quả.
* Tính mở rộng: ASP.NET Core hỗ trợ tính mở rộng, cho phép phát triển ứng dụng web với khả năng mở rộng cao, giúp ứng dụng có thể chạy tốt trên các máy chủ với tải cao.
* Tính bảo mật: ASP.NET Core cung cấp nhiều tính năng bảo mật, bao gồm chống CSRF, bảo vệ chống XSS và chống SQL Injection.
* Dễ dàng bảo trì: ASP.NET Core được thiết kế để dễ dàng bảo trì và nâng cấp, giúp các nhà phát triển tiết kiệm thời gian và công sức trong quá trình phát triển ứng dụng.
* Tích hợp linh hoạt: ASP.NET Core cho phép tích hợp với các công nghệ và các dịch vụ bên thứ ba khác, giúp cho các nhà phát triển có thể tận dụng các tính năng và công cụ của các dịch vụ này để phát triển ứng dụng web hiệu quả hơn.

2.5.2 Nhược điểm

* Có thể có sự phức tạp khi bắt đầu sử dụng vì cấu trúc và quy trình làm việc khác với các phiên bản trước đây của ASP.NET.
* Vì là một framework mới, có thể sẽ có thiếu sót trong tài liệu hướng dẫn và cộng đồng hỗ trợ còn chưa phát triển đầy đủ.
* Có một số tính năng không tương thích với các phiên bản trước đây của ASP.NET, điều này có thể gây khó khăn trong việc nâng cấp từ các ứng dụng ASP.NET cũ lên ASP.NET Core.

CHƯƠNG 3 - Triển khai WEB MINH HỌA

3.1 Công cụ hỗ trợ và cài đặt

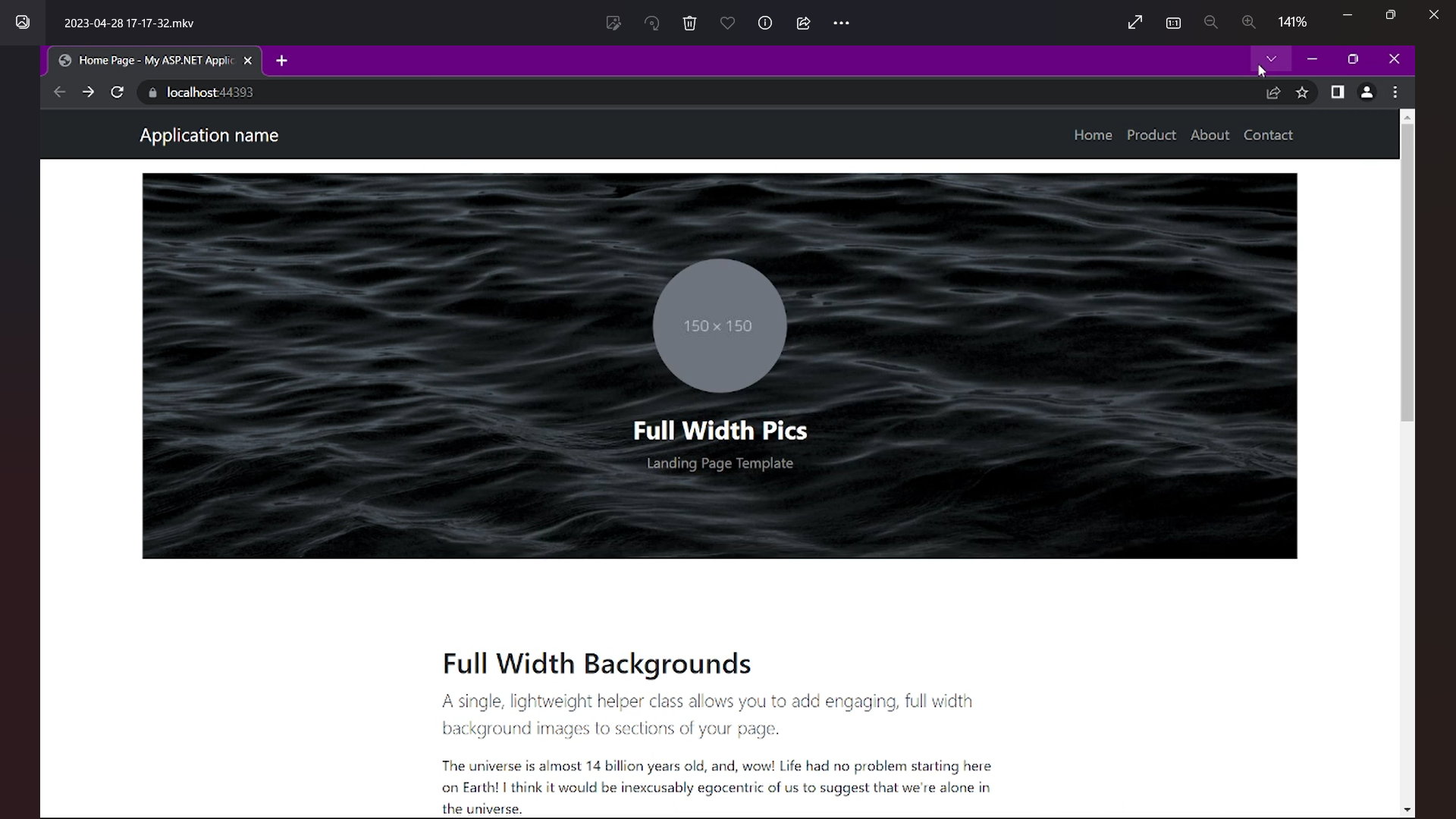
3.1.1 Công cụ hỗ trợ

3.1.2 Cài đặt

* Cài .NET Core SDK 7.0.203 SDK hoặc cao hơn [theo đường link này](https://www.microsoft.com/net/core).
* Visual Studio Code [theo đườg link này](https://code.visualstudio.com/).
* Cài VS Code C# extension [theo đường link này](https://marketplace.visualstudio.com/items?itemName=ms-vscode.csharp)*.*

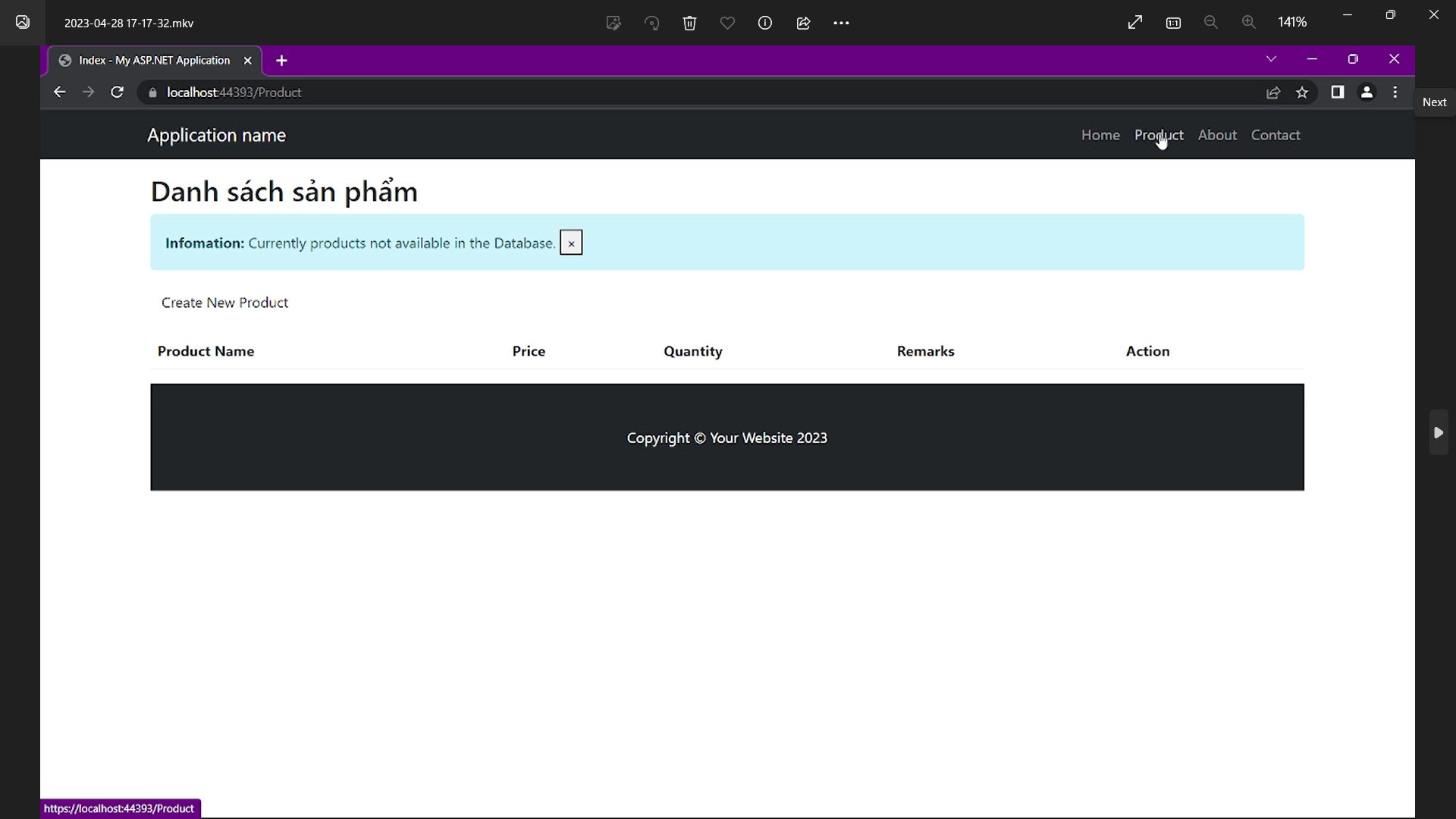
3.2 Xây dựng giao diện

Khi mới truy cập và trang web thành công thì giao diện sẽ được hiển thị như sau:



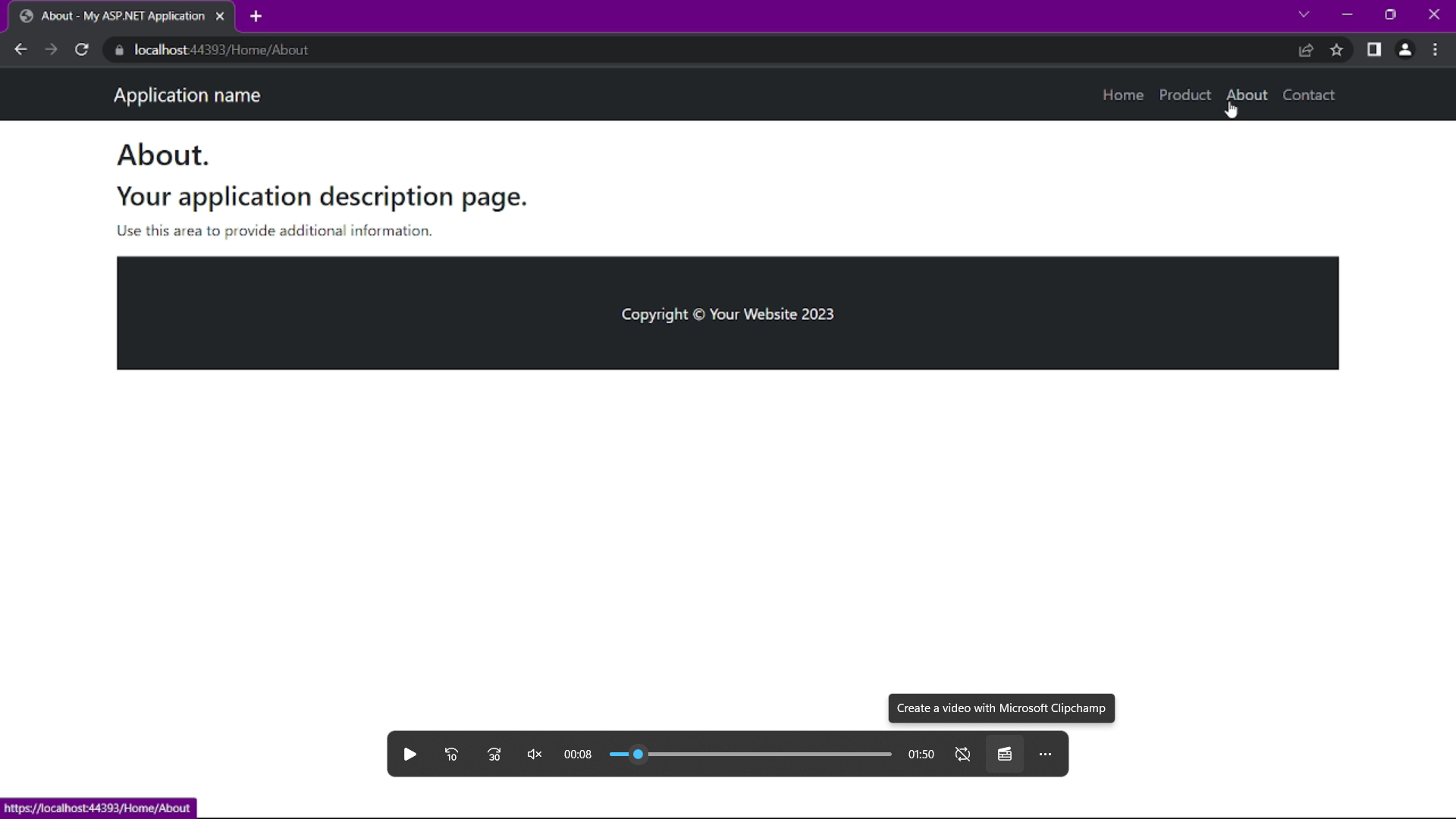
Hình 3.2.1 Hình ảnh giao diện khi mới truy cập vào trang web

Tiếp theo, khi bấm chuột sang thẻ Product thì giao diện sẽ được hiển thị như hình dưới đây. Đồng thời, giỏ hàng hiện tại là đang rỗng nên chưa có sản phẩm nào được hiển thị.

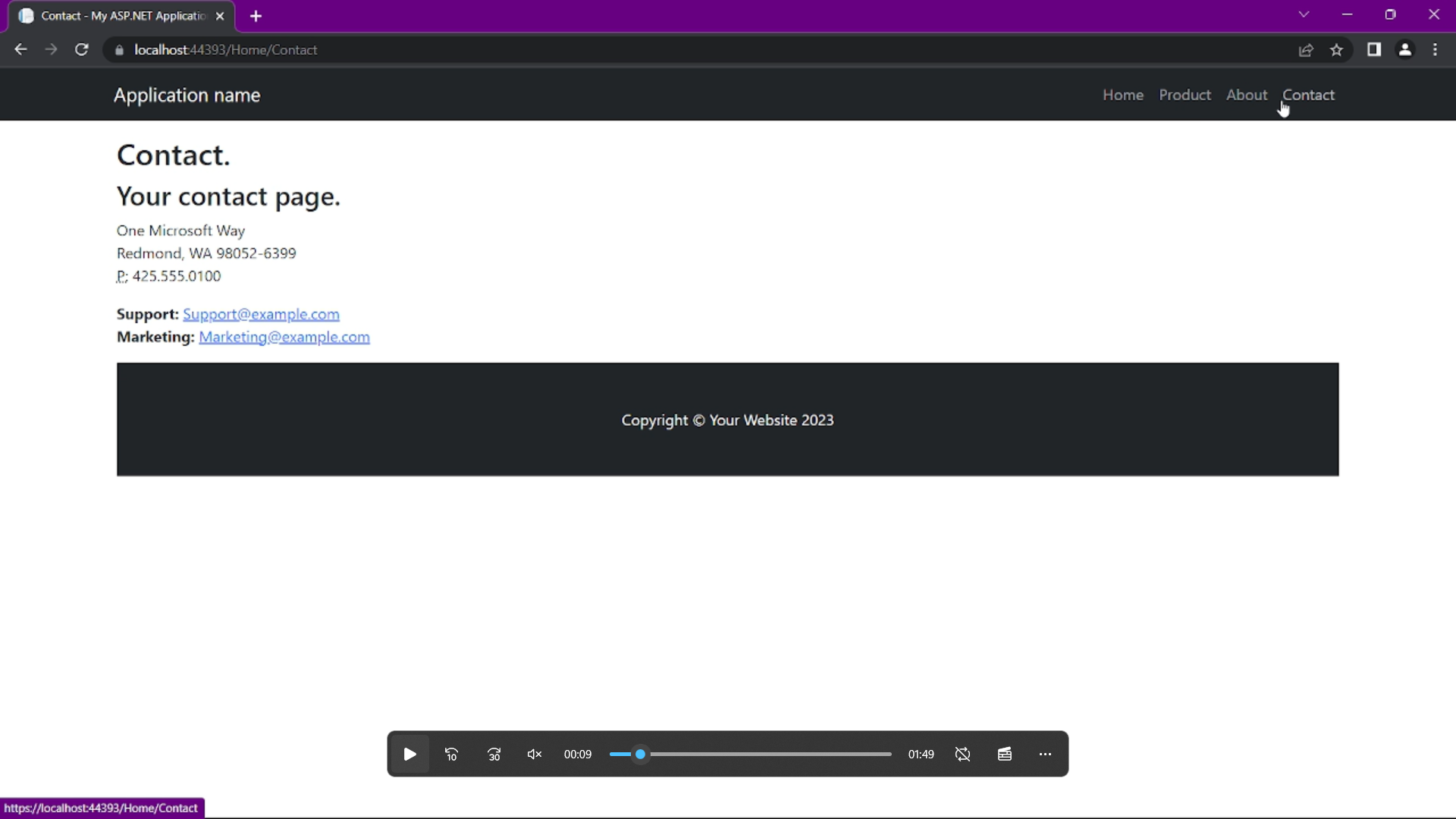


Hình 3.2.2 Hình ảnh giao diện Product

Khi người dùng nhấp chuột điều hướng sang thẻ About và Contact thì giao diện sẽ hiển thị lần lượt như sau:



Hình 3.2.3 Hình ảnh giao diện About



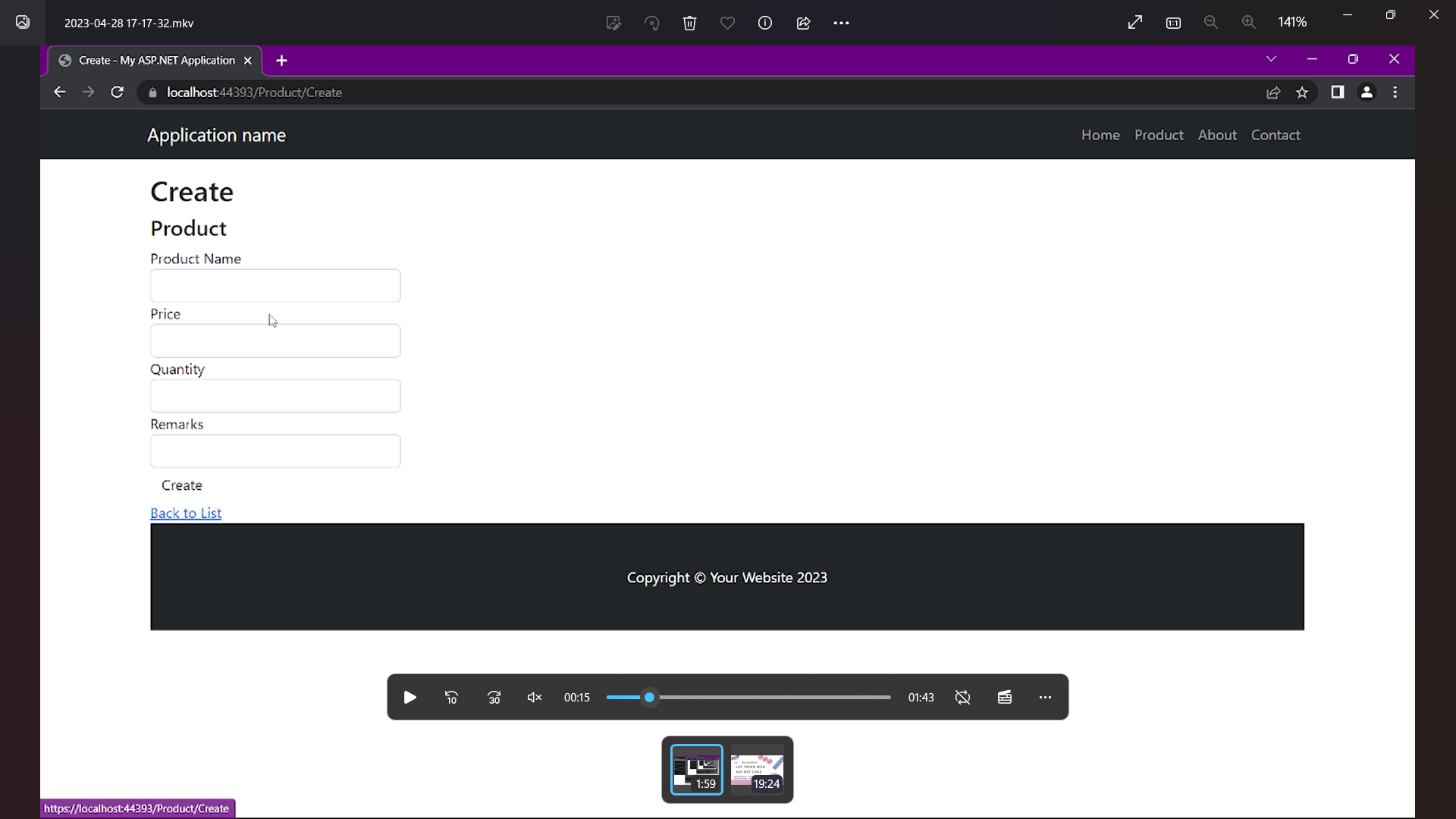
Hình 3.2.4 Hình ảnh giao diện Contact

3.3 Xây dựng các chức năng

Các chức năng như thêm, xóa, sửa sẽ được hiển thị và nằm trong thẻ Product.

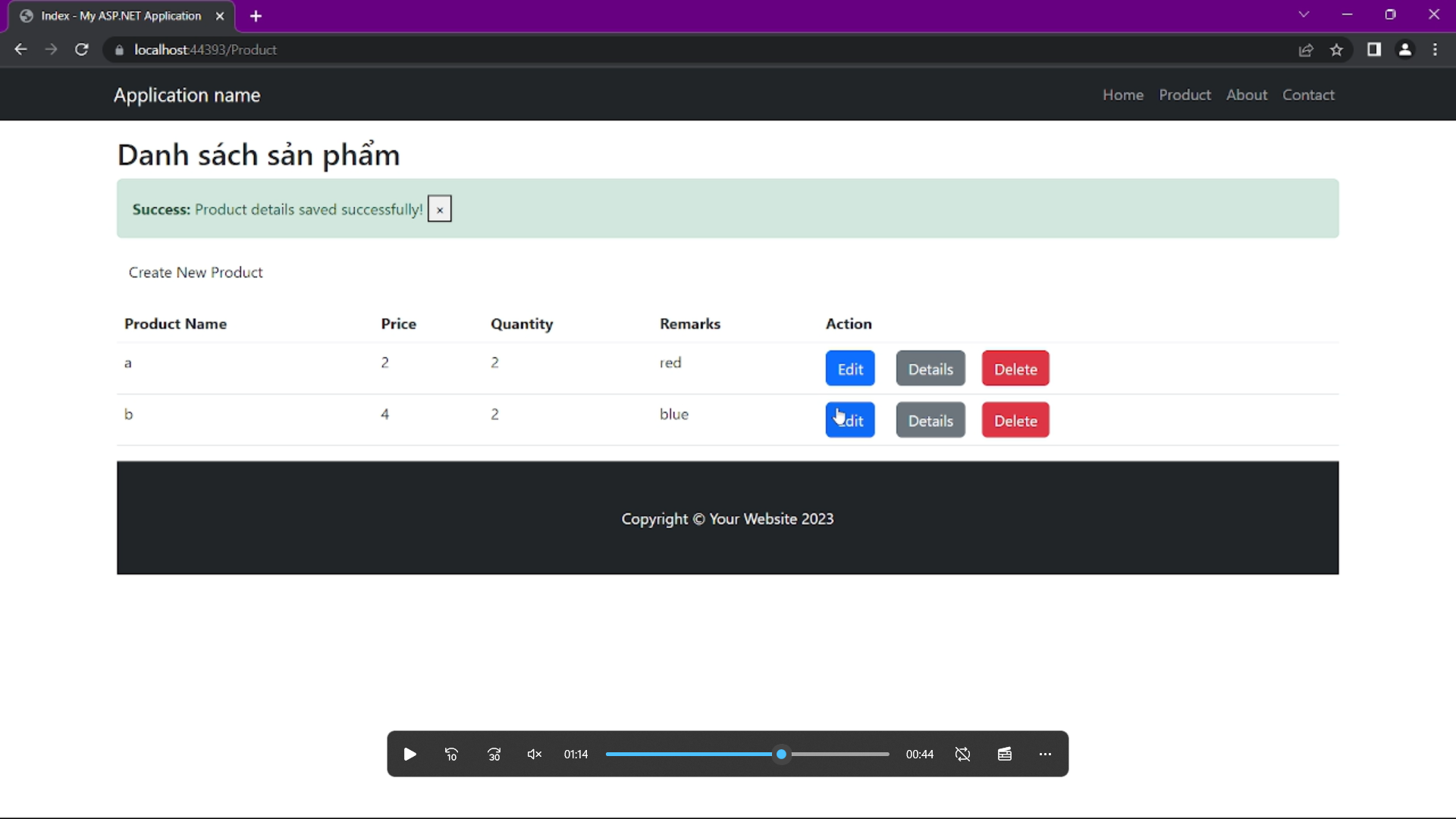
3.3.1 Thêm

Như ở giao diện Product ban đầu thì giỏ hàng đang là rỗng, khi bấm vào nút Create New Product thì giao diện sẽ được hiển thị như sau:



Hình 3.3.1.1 Hình ảnh giao diện khi thêm sản phẩm

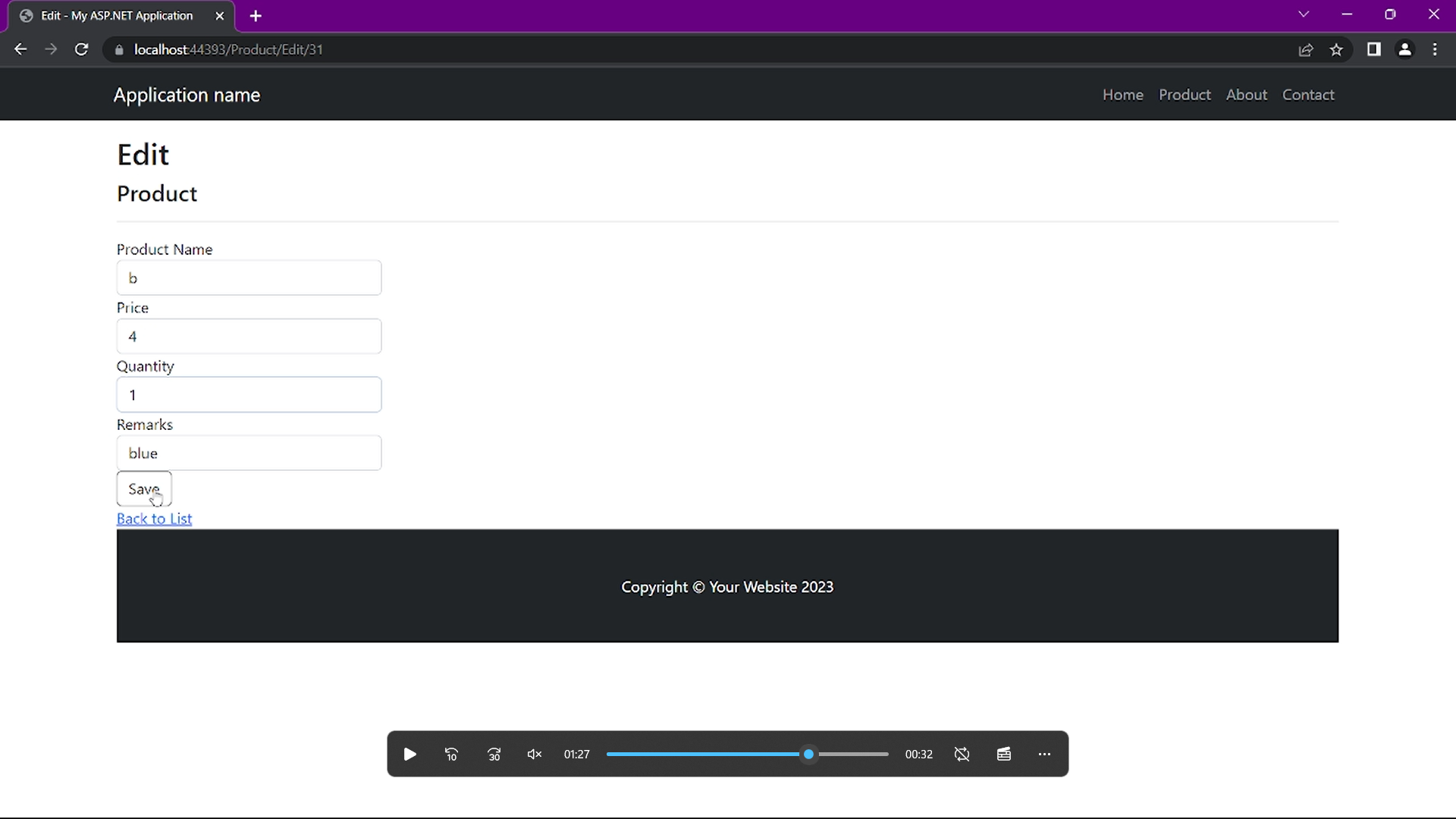
Sau khi điền đầy đủ nội dung không bị hiển thị lỗi màu đỏ và nhấp vào nút Create thì dữ liệu sẽ được gửi lên database đồng thời giỏ hàng sẽ được cập nhật, hệ thống sẽ thông báo thành công “Success: Product delete successfully!” dưới dòng Danh sách sản phẩm cho người dùng biết. Hình ảnh dưới đây là sau khi thêm 2 sản phẩm vào giỏ hàng thành công.



Hình 3.3.1.2 Hình ảnh khi thêm dữ liệu thành công

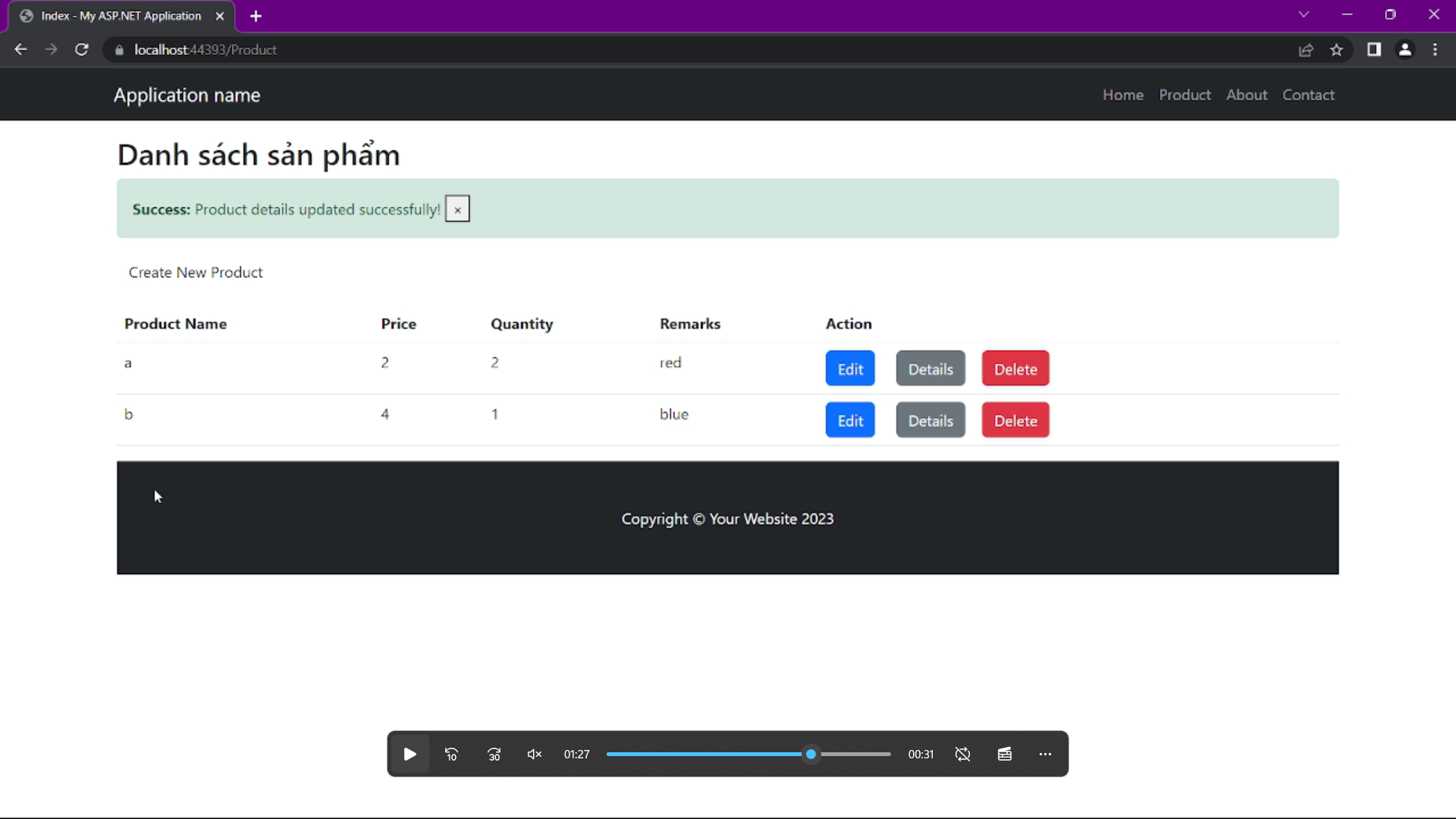
3.3.2 Sửa

Khi người dùng bấm vào nút Edit thì giao diện chỉnh sửa sẽ được hiển thị như hình dưới đây.



Hình 3.3.2.1 Hình ảnh giao diện khi sửa sản phẩm

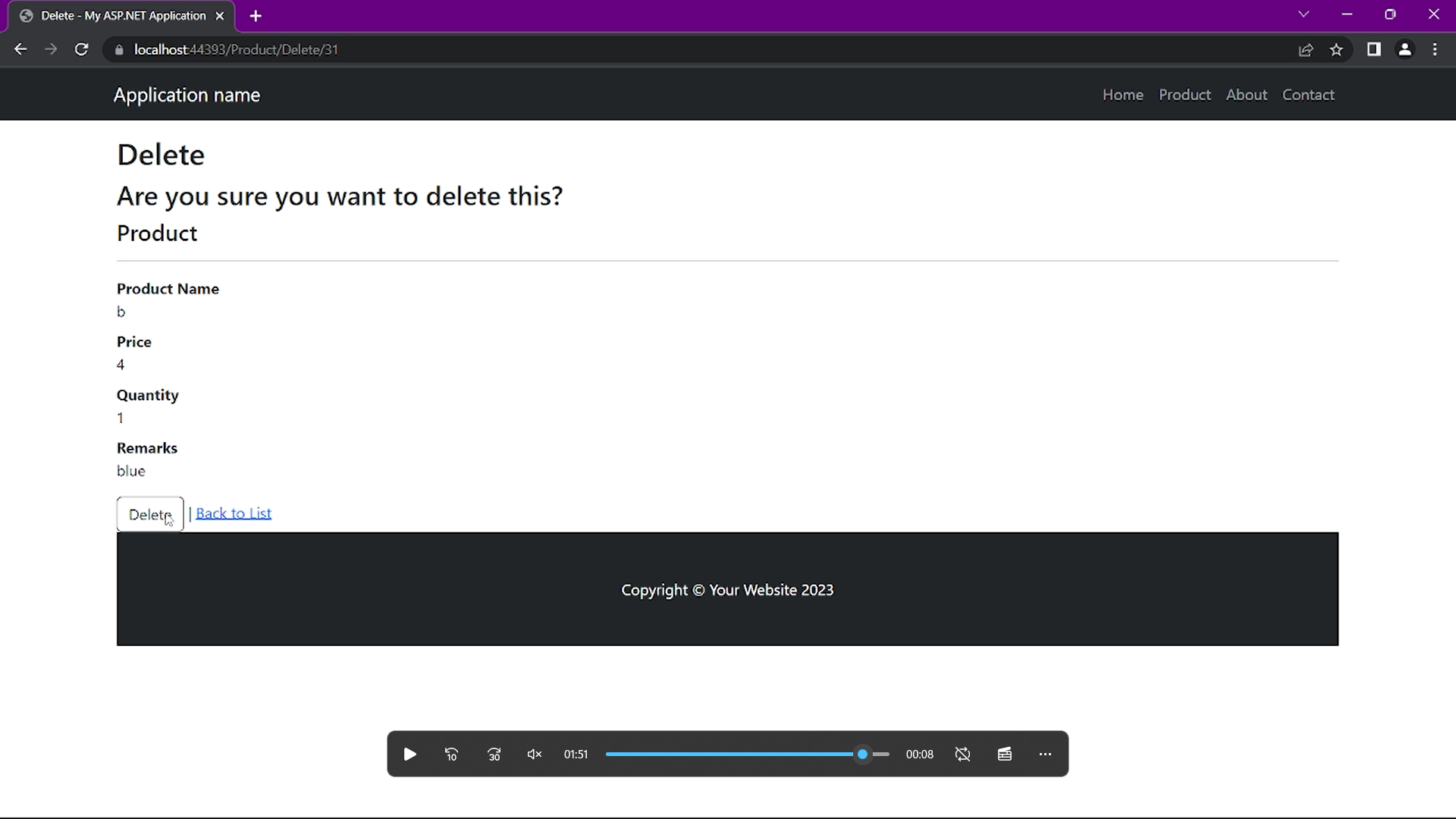
Sau khi người dùng chỉnh sửa thông tin theo nhu cầu của mình và bấm vào nút Save thì dữ liệu sẽ được cập nhật lên database đồng thời giỏ hàng cũng sẽ được cập nhật lại nội dung chỉnh sửa, hệ thống sẽ thông báo thành công “Success: Product delete successfully!” dưới dòng Danh sách sản phẩm cho người dùng biết.



Hình 3.3.2.2 Hình ảnh sửa dữ liệu thành công

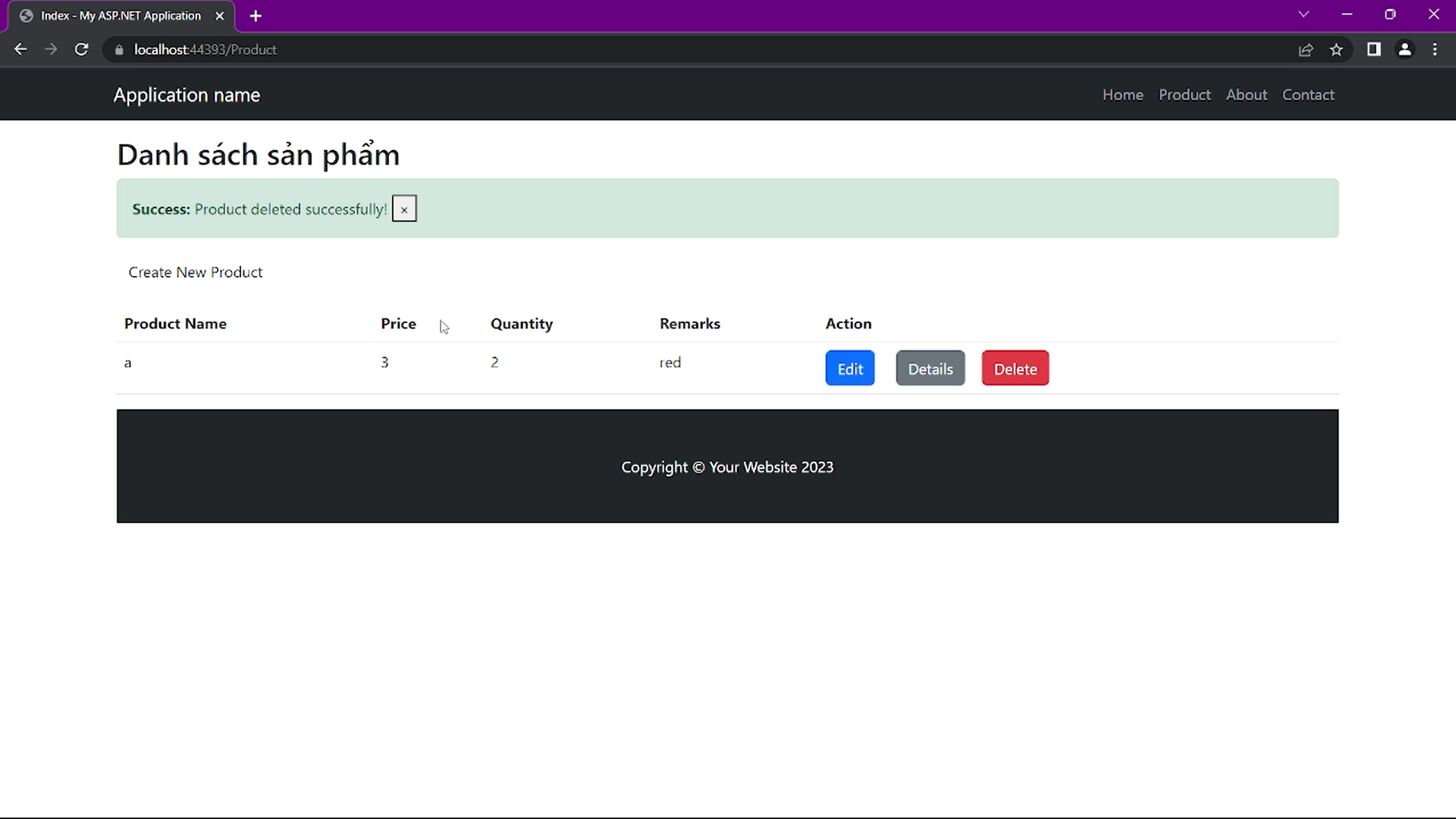
3.3.3 Xóa

Khi người dùng bấm vào nút Delete thì giao diện xóa sẽ được hiển thị như hình dưới đây kèm thông báo “Are you sure you want to detele this”.



Hình 3.3.3.1 Hình ảnh giao diện khi xóa sản phẩm

Sau khi người dùng chắc chắn xóa và bấm nút Delete thì dữ liệu đồng thời sẽ xóa khỏi database và giỏ hàng, hệ thống sẽ thông báo thành công “Success: Product delete successfully!” dưới dòng Danh sách sản phẩm cho người dùng biết. Dữ liệu đang được xóa ở đây là dữ liệu có Product Name là “b”.



Hình 3.3.3.2 Hình ảnh xóa dữ liệu thành công

CHƯƠNG 4 - KẾT LUẬN

4.1 Kết quả đạt được

* Hiểu rõ về cách hoạt động của ASP.NET Core và các khái niệm cơ bản trong framework.
* Biết cách tạo các ứng dụng web đơn giản và phức tạp sử dụng ASP.NET Core, từ việc thiết lập môi trường phát triển đến triển khai ứng dụng.
* Nắm vững các kỹ thuật phát triển ứng dụng web như xử lý yêu cầu HTTP, xử lý dữ liệu, validation, xác thực và bảo mật.
* Sử dụng các công cụ và thư viện hỗ trợ phát triển ứng dụng web trên ASP.NET Core như Entity Framework, LINQ, SignalR, Identity, ...
* Hiểu về các kỹ thuật tối ưu hóa hiệu suất ứng dụng, sử dụng các công cụ và kỹ thuật để giảm thiểu thời gian phản hồi và tăng khả năng mở rộng của ứng dụng.
* Có thể sử dụng ASP.NET Core để phát triển các ứng dụng web chạy trên các nền tảng khác nhau như Windows, Linux và macOS.

4.2 Hướng phát triển

* Xây dựng ứng dụng web theo hướng phức tạp để hoàn thiện hơn khi sử dụng ASP.NET Core: xây dựng các trang đăng nhập, đăng ký, đăng xuất.
* Xây dựng giao diện web đẹp hơn, bắt mắt hơn.

TÀI LIỆU THAM KHẢO

**Tiếng Việt**

1. [Tự học ICT, *Cấu trúc ASP.NET Core*, truy cập vào ngày 20 tháng 4 năm 2023.](https://tuhocict.com/cau-truc-asp-net-core/)
2. [Tự học ICT,](https://tuhocict.com/cach-hoat-dong-cua-ung-dung-aspnet-core/) *[Cách hoạt động của ứng dụng ASP.NET Core,](https://tuhocict.com/cach-hoat-dong-cua-ung-dung-aspnet-core/)* [truy cập vào ngày 20 tháng 4 năm 2023](https://tuhocict.com/cach-hoat-dong-cua-ung-dung-aspnet-core/)