**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated

**BÁO CÁO MÔN HỌC**

**MẪU THIẾT KẾ PHẦN MỀM**

**WebSite Đặt Thức Ăn**

**Giảng viên: ThS. Lý Đoàn Duy Khánh**

**Sinh viên thực hiện:**

* + **Nguyễn Thành Đạt – 18DH110216**
  + **Lã Hoàng Nam – 17DH110921**
  + **Lê Tấn Tài – 18DH110026**
  + **Nguyễn Thanh Trường – 18DH110233**

**TP. HỒ CHÍ MINH - 2021**

LỜI CẢM ƠN

🙞🕮🙜

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Trường Đại Học Ngoại Ngữ - Tin Học Thành Phố Hồ Chí Minh đã đưa môn học Mẫu thiết phần mềm vào Chương trình giảng dạy.

Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn - ThS Lý Đoàn Duy Khánh đã hướng dẫn và giúp đỡ trang bị kỹ năng cơ bản cũng như là nâng cao, truyền đạt những kiến thức quý báu cho em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học Mẫu Thiết Kế Phần Mềm của Thầy, chúng em đã có thêm cho mình nhiều kiến thức bổ ích, tinh thần học tập hiệu quả, nghiêm túc. Đây chắc chắn sẽ là những kiến thức quý báu, là hành trang để chúng em có thể vững bước sau này.

Bộ môn Mẫu Thiết Kế Phần Mềm là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Đảm bảo cung cấp đủ kiến thức, gắn liền với nhu cầu thực tiễn của sinh viên. Vì kiến thức của chúng em còn hạn nên trong quá trình nghiên cứu đề tài vẫn còn nhiều sai sót. Em mong nhận được sự góp ý của Thầy để đề tài được hoàn thiện hơn.

MỤC LỤC

[Chương 1. TỔNG QUAN VỀ ĐỀ TÀI 1](#_Toc92653717)

[**1.** **Giới thiệu đề tài** 1](#_Toc92653718)

[**2.** **Mục tiêu và nhu cầu sử dụng** 1](#_Toc92653719)

[a. Nhu cầu đặt thức ăn 2](#_Toc92653721)

[b. Nhu cầu quản lý 2](#_Toc92653723)

[**3.** **Feature list (product requirements)** 2](#_Toc92653725)

[a. Quản lý 2](#_Toc92653726)

[b. Khách hàng 3](#_Toc92653752)

[c. Kitchen 3](#_Toc92653760)

[d. Front 4](#_Toc92653766)

[Chương 2. KỶ THUẬT THỰC HIỆN 5](#_Toc92653775)

[Giới thiệu mô hình(Lịch sử phát triển) 5](#_Toc92653776)

[Đặc điểm mô hình 6](#_Toc92653783)

[Giới thiệu Pattern 7](#_Toc92653794)

[Đặc điểm Pattern: 10](#_Toc92653802)

[Chương 3. PHÂN TÍCH YÊU CẦU 14](#_Toc92653816)

[I. Các Use Case Diagram 14](#_Toc92653817)

[II. Đặc tả chức năng 16](#_Toc92653827)

[Chương 4. TRIỂN KHAI ỨNG DỤNG 19](#_Toc92653831)

[1. Tạo Repository 19](#_Toc92653832)

[2. Triển khai Repository 19](#_Toc92653833)

[3. Triển khai Repository cho Category 21](#_Toc92653834)

[4. Sử dụng Unit Of Word cho Repository Category 21](#_Toc92653835)

[5. Áp dụng vào Index Category 22](#_Toc92653836)

[6. Áp dụng vào Create Category 23](#_Toc92653837)

[7. Áp dụng Delete Category 24](#_Toc92653838)

[8. Áp dụng cho Edit Category 25](#_Toc92653839)

[Chương 5. GIAO DIỆN 26](#_Toc92653840)

[KẾT LUẬN 38](#_Toc92653885)

[TÀI LIỆU THAM KHẢO 39](#_Toc92653886)

**DANH SÁCH BẢNG**

[Bảng 1. Mô tả Actor 16](#_Toc92654328)

[Bảng 2. Quản trị và Nhân viên 18](#_Toc92654329)

[Bảng 3. Khách hàng 18](#_Toc92654330)

**DANH SÁCH HÌNH**

[Hình 1. Repository Pattern 7](#_Toc92659115)

[Hình 2. Unit of Work 9](#_Toc92659116)

[Hình 3. Kho lưu trữ tùy chỉnh so với DBContext 11](#_Toc92659117)

[Hình 4. Proxy 11](#_Toc92659118)

[Hình 5. Unit of Work 13](#_Toc92659119)

[Hình 6. Quản lý 14](#_Toc92659120)

[Hình 7. Khách hàng 15](#_Toc92659121)

[Hình 8. Front 16](#_Toc92659122)

[Hình 9. Trang chủ 26](#_Toc92659123)

[Hình 10. Đăng ký 27](#_Toc92659124)

[Hình 11. Đăng nhập 27](#_Toc92659125)

[Hình 12. Chi tiết món 28](#_Toc92659126)

[Hình 13. Đặt hàng 28](#_Toc92659127)

[Hình 14. Quản lý giỏ hàng 29](#_Toc92659128)

[Hình 15. Thanh toán 29](#_Toc92659129)

[Hình 16. Danh mục món 30](#_Toc92659130)

[Hình 17. Thêm danh mục món 30](#_Toc92659131)

[Hình 18. Chỉnh sửa danh mục món 31](#_Toc92659132)

[Hình 19. Xóa danh mục món 31](#_Toc92659133)

[Hình 20. Loại món 32](#_Toc92659134)

[Hình 21. Thêm loại món 32](#_Toc92659135)

[Hình 22. Chỉnh sửa loại món 33](#_Toc92659136)

[Hình 23. Xóa loại món 33](#_Toc92659137)

[Hình 24. Danh sách món 34](#_Toc92659138)

[Hình 25. Thêm món 35](#_Toc92659139)

[Hình 26. Xóa món 35](#_Toc92659140)

[Hình 27. Cập nhật món 36](#_Toc92659141)

[Hình 28. Thêm nhân viên 36](#_Toc92659142)

[Hình 29. Danh sách đặt hàng 37](#_Toc92659143)

[Hình 30. Quản lý tài khoản 37](#_Toc92659144)

[Hình 31.Admin 38](#_Toc92659145)

[Hình 32.Quyền Admin 38](#_Toc92659146)

[Hình 33.Quản lí order 39](#_Toc92659147)

[Hình 34. Thông báo xóa món 39](#_Toc92659148)

[Hình 35.Khách hàng 40](#_Toc92659149)

[Hình 36.Giỏ hàng 40](#_Toc92659150)

[Hình 37.Đặt hàng thành công 41](#_Toc92659151)

[Hình 38.Quản lí order 41](#_Toc92659152)

[Hình 39.Quản lí order(1) 42](#_Toc92659153)

# TỔNG QUAN VỀ ĐỀ TÀI

1. **Giới thiệu đề tài**

Ngày nay, thông tin điện tử, báo điện tử, công nghệ số đang trở thành xu thế phát triển mạnh mẽ trên tất cả các ngành nghề và các quốc gia trên toàn thế giới. Một xã hội phát triển, khi mà tất cả những gì con người làm và trao đổi chỉ bằng vài thao tác, hay những thông tin chia sẻ, quảng cáo đều có thể nhanh chóng và dễ dàng tìm được qua Internet. Từ đó mà hàng loạt dịch vụ mới ra đời nhằm đáp ứng nhu cầu của khách hàng, và thức ăn là một nhu cầu thiết yếu của con người thì việc làm cho nó trở nên đơn giản hóa là điều vô cùng hợp lí.

Trong điều kiện Internet phổ biến rộng rãi như hiện nay, việc đưa một món ăn đến người dùng là việc hết sức dễ dàng, đem lại sự tiện lợi cho cả người mua và người bán. Với 1 thiết bị kết nối mạng, chúng ta có thể đưa sản phẩm của mình đến hàng triệu người dùng thông qua website thương mại điện tử.

Hiện nay khi nhu cầu mua sản phẩm giữa người bán và người mua được đẩy mạnh lên cao thì hình thức kinh doanh trực tuyến sử dụng nền tảng công nghệ thông tin với sự hỗ trợ của Internet để thực hiện các giao dịch mua bán, trao đổi, thanh toán trực tuyến thì Thương Mại Điện Tử là mô hình không thể thiếu trong lúc này. Nó đáp ứng đầy đủ các tính chất của việc mua bán hàng trực tuyến giữa người bán và người mua bao gồm việc sản xuất, quảng cáo, bán hàng và phân phối sản phẩm được mua bán và thanh toán trên mạng Internet, nhưng được giao nhận 1 cách hữu hình, cả các sản phẩm giao nhận cũng như những thông tin số hóa thông qua mạng Internet.

Thông qua website này nhóm chúng em mong muốn sẽ tạo ra được 1 thị trường mua bán thức ăn năng động, nhằm đáp ứng nhu cầu trực tuyến giữa người bán và người mua, tạo cho người dùng nhiều trải nghiệm mới, mang đến nhiều lợi ích, thú vị khi kinh doanh cũng như là đặt thức ăn.

1. **Mục tiêu và nhu cầu sử dụng**

Các nhà kinh tế học đã đưa ra giả thuyết rằng thương mại điện tử sẽ dẫn đến việc cạnh tranh giá cả sản phẩm. Thực vậy, thương mại điện tử giúp người tiêu dùng thu thập nhanh chóng và dễ dàng thông tin đa dạng về sản phẩm, giá cả và người bán. Ngày nay đã xuất hiện nhiều website chuyên cung cấp dịch vụ đánh giá về sản phẩm và nhà cung cấp, so sánh giá cả giữa các website bán hàng. Hơn nữa, người tiêu dùng có thể trực tiếp đưa ra các đánh giá của mình về nhiều khía cạnh liên quan tới giao dịch mua sắm, giúp cho những người khác có nhiều cơ hội chọn lựa sản phẩm phù hợp nhất, hoặc chọn được người bán cung cấp dịch vụ tốt nhất, hoặc mua được sản phẩm với giá rẻ nhất.

* 1. **Nhu cầu đặt thức ăn**
     + Khi khách hàng muốn đặt thức ăn, tại trang home khách hàng có thể thêm thức ăn vào giỏ hàng, hoặc có thể tìm kiếm và tiến hành thanh toán.
  2. **Nhu cầu quản lý**
     + Quản lý mọi thông tin đơn hàng, vận chuyển,phân quyền user, là phương pháp lập kế hoạch được sử dụng để dự báo, và quản lý nhu cầu về sản phẩm, dịch vụ.

1. **Feature list (product requirements)**
   1. **Quản lý**
      * **Đăng ký**
      * **Đăng nhập**
      * **Chỉnh sửa thông tin**
      * **Quản lý danh sách đơn hàng**
        + Tìm kiếm đơn hàng
        + Chỉnh sửa đơn
          - Hủy đơn
          - Trả đơn
          - Kết thúc đơn
      * **Quản lý danh mục món**
        + Thêm danh mục món
        + Cập nhật danh mục món
        + Xóa danh mục món
      * **Quản lý loại món**
        + Thêm loại món
        + Cập nhật loại món
        + Xóa loại món
      * **Quản lý món**
        + Thêm món
        + Cập nhật món
        + Xóa món
      * **Quản lý đơn hàng**
        + Xác nhận đơn
        + Hủy đơn
      * **Thêm nhân viên**
   2. **Khách hàng**
      * **Đăng ký**
      * **Đăng nhập**
      * **Chỉnh sửa thông tin**
      * **Xem chi tiết sản phẩm**
      * **Thêm sản phẩm vào giỏ hàng**
      * **Quản lý giỏ hàng**
      * **Đặt hàng**
   3. **Kitchen**
      * **Đăng ký**
      * **Đăng nhập**
      * **Chỉnh sửa thông tin**
      * **Xác nhận đơn hàng**
      * **Hủy đơn**
   4. **Front**
      * **Đăng ký**
      * **Đăng nhập**
      * **Chỉnh sửa thông tin**
      * **Tìm kiếm đơn hàng**
      * **Chỉnh sửa đơn hàng**
        + Hủy đơn
        + Trả đơn
        + Kết thúc đơn hàng

# KỶ THUẬT THỰC HIỆN

Giới thiệu mô hình(Lịch sử phát triển)

Bản phát hành đầu tiên của ASP.NET đã xuất hiện cách đây 15 năm trước, nó là một phần của .NET Framework. Từ đó, hàng triệu lập trình viên đã sử dụng nó để xây dựng những ứng dụng web tuyệt vời, và trên những năm đó Microsoft đã phát triển thêm nhiều tính năng mới. ASP.NET Core có một số thay đổi kiến trúc lớn, đó là kết quả của việc học hỏi rất nhiều từ các framework module hóa khác. ASP.NET Core không còn dựa trên System.Web.dll nữa. Nó được dựa trên một tập hợp các gói, các module hay cũng được gọi là các Nuget packages.

ASP.Net Core

* Nó là một open-source mới và framework đa nền tảng (cross-platform) cho việc xây dựng những ứng dụng hiện tại dựa trên kết nối đám mây, giống như web apps, IoT và backend cho mobile.
* Ứng dụng ASP.NET Core có thể chạy trên .NET Core hoặc trên phiên bản đầy đủ của .NET Framework. Nó được thiết kế để cung cấp và tối ưu development framework cho những dụng cái mà được triển khai trên đám mây (clound) hoặc chạy on-promise.
* Nó bao gồm các thành phần theo hướng module nhằm tối thiểu tài nguyên và chi phí phát triển, như vậy bạn giữ lại được sự mềm giẻo trong việc xây dựng giải pháp của bạn. Bạn có thể phát triển và chạy những ứng dụng ASP.NET Core đa nền tảng trên Windows, Mac và Linux.
  + Đồng thời nó đã trở thành một mã nguồn mở. Đây là một thay đổi rất lớn và theo mình là quan trọng nhất của ASP.NET Core. Điều mà trước đây khó có một lập trình viên nào có thể nghĩ đến. Có lẽ đó cũng là một xu thế mà các ngôn ngữ lập trình hiện nay.

Đặc điểm mô hình:

ASP.Net Core

* + - Thống nhất việc xây dựng giao diện web và APIs
    - Razor Pages giúp việc code dễ dàng và hiệu quả hơn
    - Blazor cho phép bạn viết code c# trên trình duyệt cùng javascript, chia sẻ server-side và client-side trong tầng logic được viết tất cả bằng .Net
    - Phát triển và chạy trên nhiều môi trường hệ điều hành như Windows, macOS,LiNux
    - Dễ dàng triển khai trên Cloud
    - Mã nguồn mở hoàn toàn miễn phí
    - Tích hợp sẵn các dependency injection
    - Một ứng dụng nhẹ (lightweight), hiệu năng cao
    - Hỗ trợ trên IIS,Apache,Kestrel,Docker

Giới thiệu Pattern

* 1. Repository Pattern

Diagram

Description automatically generated

Hình 1. Repository Pattern

* Repository Design Pattern là một trong những design pattern được sử dụng nhiều nhất trong hầu hết các ngôn ngữ lập trình, framework … như .NET, Java, PHP …, từ ᴡebѕiteѕ, ѕerᴠiceѕ, applicationѕ,. .. haу bao gồm cả ứng dụng di động.
* Và cũng là lớp trung gian giữa Buѕineѕѕ Logic (BL) và Nguồn dữ liệu (DB), các đối tượng trong lớp trung gian này được gọi là Repoѕitorу. Giao tiếp giữa BL và DB qua giao diện.
* Chúng mang lại quá trình chuẩn hóa (ѕtandardied) cho đầu ra và tách biệt hoàn toàn logic xử lý logic và truy cập dữ liệu, khiến BL hoàn toàn không quan tâm đến công việc của DB (và ngược lại).
  1. Proxy Pattern
* Proxy là một mẫu thiết kế cấu trúc cho phép bạn cung cấp vật thay thế hoặc trình giữ chỗ cho một đối tượng khác. Một proxy kiểm soát quyền truy cập vào đối tượng ban đầu, cho phép bạn thực hiện điều gì đó trước hoặc sau khi yêu cầu được chuyển đến đối tượng ban đầu.
  + Giải thích : Trong lập trình máy tính, mẫu proxy là một mẫu thiết kế phần mềm. Proxy, ở dạng chung nhất, là một lớp hoạt động như một giao diện cho một cái gì đó khác. Proxy có thể giao tiếp với bất kỳ thứ gì: kết nối mạng, một đối tượng lớn trong bộ nhớ, một tệp hoặc một số tài nguyên khác đắt tiền hoặc không thể sao chép. Nói tóm lại, proxy là một đối tượng trình bao bọc hoặc tác nhân đang được gọi bởi máy khách để truy cập vào đối tượng phục vụ thực sự đằng sau hậu trường. Việc sử dụng proxy có thể chỉ đơn giản là chuyển tiếp đến đối tượng thực hoặc có thể cung cấp logic bổ sung. Trong proxy, chức năng bổ sung có thể được cung cấp, ví dụ như bộ nhớ đệm khi các hoạt động trên đối tượng thực sử dụng nhiều tài nguyên hoặc kiểm tra các điều kiện tiên quyết trước khi các hoạt động trên đối tượng thực được gọi. Đối với máy khách, việc sử dụng một đối tượng proxy tương tự như sử dụng đối tượng thực, vì cả hai đều triển khai cùng một giao diện.
  1. Unit of Work

Diagram

Description automatically generated

Hình 2. Unit of Work

* Khi bạn kéo dữ liệu từ trong database ra ngoài hoặc ngược lại thì đều quan trọng nhất là làm sao để theo dõi được cái gì mà bạn đã thay đổi(ví dụ như bạn lấy lên 2 bảng và bạn chỉ sửa đổi 1 vài cột trong bảng đó thôi), nêu không theo dỏi thì sẻ không biết thế nào để lưu trở lại Database. Tương tự như vậy bạn có thể dùng cho Insert, remove.
* Để giải quyết vấn đề trên bạn có thể dùng theo cách truyền thống là lưu những thay đổi đó trên Object Model xuống database, nhưng điều đó sẻ dẩn đến nhiều lần gọi đến database chỉ để lưu các dữ liệu rất nhỏ đó, điều đó sẻ làm cho hệ thống chậm xuống. Hơn nửa nó đòi hỏi bạn phải mở transaction(giao dịch, kết nối) trong toàn bộ quá trình tương tác.
* Vì thế Unit of Work theo dõi mọi transaction từ tầng Business mà có thể ảnh hưởng đến database. Khi bạn thực hiện xong các tác vụ thì nó sẻ lưu toàn bộ những thay đổi của bạn xuống database

Đặc điểm Pattern:

Repository Pattern

Mẫu kho lưu trữ là một mẫu thiết kế dàn xếp dữ liệu từ và đến các Lớp truy cập dữ liệu và miền (như Entity Framework Core / Dapper). Kho lưu trữ là các lớp ẩn các logic cần thiết để lưu trữ hoặc truy xuất dữ liệu. Do đó, ứng dụng sẽ không quan tâm đến loại ORM mà đang sử dụng, vì mọi thứ liên quan đến ORM đều được xử lý trong một lớp kho lưu trữ. Điều này cho phép bạn phân tách rõ ràng hơn các mối quan tâm. Repository Pattern là một trong những Mẫu thiết kế được sử dụng nhiều để xây dựng.

1. Giảm các truy vấn trùng lặp

* Hãy tưởng tượng bạn phải viết các dòng mã để chỉ lấy một số dữ liệu từ kho dữ liệu của bạn. Bây giờ điều gì sẽ xảy ra nếu tập hợp các truy vấn này sẽ được sử dụng ở nhiều nơi trong ứng dụng.
* Đây là lợi thế bổ sung của Lớp Kho lưu trữ. Bạn có thể viết mã truy cập dữ liệu của mình trong Kho lưu trữ và gọi nó từ nhiều Bộ điều khiển / Thư viện. Đạt mục đích?

1. Kho lưu trữ tùy chỉnh so với DBContext

Diagram

Description automatically generated

Hình 3. Kho lưu trữ tùy chỉnh so với DBContext

1. Proxy Pattern

Diagram

Description automatically generated

Hình 4. Proxy

* ServiceInterface khai báo giao diện của Service. Proxy phải tuân theo giao diện này để có thể tự ngụy trang thành một đối tượng service.
* Service là một lớp cung cấp một số logic nghiệp vụ.
* Lớp Proxy có một trường tham chiếu trỏ đến một đối tượng Service. Sau khi proxy kết thúc quá trình xử lý (ví dụ: khởi tạo lười biếng, ghi nhật ký, kiểm soát truy cập, bộ nhớ đệm, v.v.), nó sẽ chuyển yêu cầu đến đối tượng Service.
* Thông thường, proxy quản lý toàn bộ vòng đời của các đối tượng service của chúng.
* Client phải làm việc với cả service và proxy thông qua cùng một giao diện. Bằng cách này, bạn có thể chuyển proxy vào bất kỳ mã nào yêu cầu đối tượng service.

3. Unit of Work trong Repository Pattern

* Unit Of Work được sử dụng để đảm bảo nhiều hành động như insert, update, delete...được thực thi trong cùng một transaction thống nhất. Nói đơn giản hơn, nghĩa là khi một hành động của người dùng tác động vào hệ thống, tất cả các hành động như insert, update, delete...phải thực hiện xong thì mới gọi là một transaction thành công. Gói tất cả các hành động đơn lẻ vào một transaction để đảm bảo tính toàn vẹn dữ liệu.
* Để hiểu khái niệm này, hãy xem cách triển khai Repository Pattern sau đây sử dụng non-generic Repository Pattern cho thực thể Customer:

Graphical user interface, application

Description automatically generated

Hình 5. Unit of Work

# PHÂN TÍCH YÊU CẦU

1. Các Use Case Diagram
   1. Quản lý

* Actor quản lý có vai trò quản lý tổng phần mềm như quản lý đơn hàng, quản lý món,… hoạt động cũng như thêm nhân viên và phân quyền user.

A picture containing text

Description automatically generated

Hình 6. Quản lý

* 1. Khách hàng
* Actor khách hàng diễn giải thêm, xóa, sửa sản phẩm trong giỏ hàng, tiến hành đặt hàng và thanh toán.

Text

Description automatically generated

Hình . Khách hàng

* 1. Front
* Actor Front có chức năng quản lí các đơn hàng vận chuyển cũng như orderlist trong tab quản lý

Text

Description automatically generated

Hình . Front

1. Đặc tả chức năng
   1. Mô tả Actor

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Actor** | **Mô tả** |
| 1 | Manager | Người quản trị hệ thống, thêm xóa sửa doanh mục, loại, món ăn, quản lý đơn hàng, danh sách đơn hàng và thêm nhân viên |
| 2 | Kitchen | Nhân viên hệ thống quản lý danh sách đơn hàng, xác nhận, hủy đơn hàng |
| 3 | Front | Nhân viên hệ thống quản lý danh sách đơn hàng, tìm kiếm chỉnh sửa đơn hàng, kết thúc, xóa, hoàn trả đơn hàng |
| 4 | Khách hàng | Đăng ký, đăng nhập, Xem chi tiết sản phẩm, thêm sản phẩm vào giỏ hàng, quản lý giỏ hàng, đặt hàng. |

Bảng . Mô tả Actor

## Mô tả UC quản trị và nhân viên

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Code** | **Name** | **Brief Description** |
| 1 | UC01 | Quản lý doanh mục món | Người quản trị hệ thống có thể xem danh sách các doanh mục sản phẩm |
| 2 | UC02 | Thêm doanh mục món | Người quản trị hệ thống có thể thêm doanh mục món vào hệ thống |
| 3 | UC03 | Cập nhật doanh mục món | Người quản trị hệ thống có thể cập nhật doanh mục món có trong hệ thống |
| 4 | UC04 | Xóa doanh mục món | Người quản trị hệ thống có thể xóa doanh mục món trong hệ thống |
| 5 | UC05 | Quản lý loại món | Người quản trị hệ thống có thể xem danh sách các loại món |
| 6 | UC06 | Thêm loại món | Người quản trị hệ thống có thể thêm loại món vào hệ thống |
| 7 | UC07 | Cập nhật loại món | Người quản trị hệ thống có thể cập nhật loại món có trong hệ thống |
| 8 | UC08 | Xóa loại món | Người quản trị hệ thống có thể xóa loại món trong hệ thống |
| 9 | UC09 | Quản lý món | Người quản trị hệ thống có thể xem danh sách món trong hệ thống |
| 10 | UC10 | Thêm món | Người quản trị hệ thống có thể thêm món vào hệ thống |
| 11 | UC11 | Cập mục món | Người quản trị hệ thống có thể cập nhật món có trong hệ thống |
| 12 | UC12 | Xóa món | Người quản trị hệ thống có thể xóa món trong hệ thống |
| 13 | UC13 | Thêm nhân viên | Người quản trị hệ thống có thể thêm nhân viên vào hệ thống |
| 14 | UC14 | Quản lý đơn hàng | Người dùng hệ thống có thể xem danh sách đơn hàng vừa đặt trong hệ thống |
| 15 | UC15 | Hủy đơn | Người dùng hệ thống có thể hủy đơn hàng trên hệ thống |
| 16 | UC16 | Xác nhận đơn | Người dùng hệ thống có thể xác nhận đơn hàng trên hệ thống |
| 17 | UC17 | Xác nhận hoàn thành đơn hàng | Người dùng hệ thống sau khi xác nhận đơn hàng thì sẽ có thể xác nhận hoàn thành đơn hàng trên hệ thống |
| 18 | UC18 | Quản lý danh sách đơn hàng | Người dùng hệ thống sẽ xem danh sách đơn hàng đã được xác nhận hoàn thành bên quản lý đơn hàng của hệ thống |
| 19 | UC19 | Tìm kiếm đơn hàng | Người dùng hệ thống có thể tìm kiếm đơn hàng trên hệ thống |
| 20 | UC20 | Chỉnh sửa đơn hàng | Người dùng hệ thống sẽ có thể chỉnh sửa đơn hàng trên hệ thống |
| 21 | UC21 | Hủy đơn | Người dùng hệ thống có thể hủy đơn hàng trên hệ thống |
| 22 | UC22 | Trả đơn | Người dùng hệ thống có thể trả đơn khi tiến hành sửa đơn hàng |
| 23 | UC23 | Kết thúc đơn hàng | Người dùng hệ thống có thể kết thúc đơn hàng khi chỉnh sửa đơn hàng trên hệ thống |

Bảng . Quản trị và Nhân viên

## Mô tả UC khách hàng

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Code** | **Name** | **Brief Description** |
| 1 | UC01 | Xem chi tiết sản phẩm | Khách hàng có thể xem thông tin chi tiết cả món ăn mà khách hàng muốn đặt |
| 2 | UC02 | Thêm sản phẩm vào giỏ hàng | Khách hàng có thể thêm món mình thích vào giỏ hàng |
| 3 | UC03 | Quản lý giỏ hàng | Khách hàng có thể tăng giảm số lượng món trong giỏ hàng |
| 4 | UC04 | Đặt hàng | Khách hàng có thể đặt các món ăn mà khách hàng muốn đặt |
| 5 | UC05 | Đăng ký | Khách hàng có thể đăng ký tài khoản để mua hàng trên hệ thống |
| 6 | UC06 | Chỉnh sửa thông tin | Khách hàng có thể chỉnh sửa thông tin tài khoản của mình trên hệ thống |

Bảng . Khách hàng

# TRIỂN KHAI ỨNG DỤNG

1. Tạo Repository

public interface IRepository<T> where T : class

{

void Add(T entity);

void Remove(T entity);

void RemoveRange(IEnumerable<T> entity);

IEnumerable<T> GetAll(Expression<Func<T, bool>>? filter = null,

Func<IQueryable<T>,IOrderedQueryable<T>>? orderby = null,

string? includeProperties=null);

T GetFirstOrDefault(Expression<Func<T, bool>>? filter = null, string? includeProperties = null);

}

1. Triển khai Repository

public class Repository<T> : IRepository<T> where T : class

{

private readonly ApplicationDbContext \_db;

internal DbSet<T> dbSet;

public Repository(ApplicationDbContext db)

{

\_db = db;

//FoodType,Category

//\_db.ShoppingCart.Include(u => u.MenuItem).ThenInclude(u => u.Category);

//\_db.MenuItem.OrderBy(u => u.Name);

this.dbSet=db.Set<T>();

}

public void Add(T entity)

{

dbSet.Add(entity);

}

public IEnumerable<T> GetAll(Expression<Func<T, bool>>? filter = null,

Func<IQueryable<T>, IOrderedQueryable<T>>? orderby = null, string ? includeProperties=null)

{

IQueryable<T> query = dbSet;

if (filter != null)

{

query = query.Where(filter);

}

if (includeProperties != null)

{

//abc,,xyz -> abc xyz

foreach(var includeProperty in includeProperties.Split(

new char[] { ','}, StringSplitOptions.RemoveEmptyEntries))

{

query = query.Include(includeProperty);

}

}

if (orderby != null)

{

return orderby(query).ToList();

}

return query.ToList();

}

public T GetFirstOrDefault(Expression<Func<T, bool>>? filter = null, string? includeProperties = null)

{

IQueryable<T> query = dbSet;

if (filter != null)

{

query= query.Where(filter);

}

if (includeProperties != null)

{

//abc,,xyz -> abc xyz

foreach (var includeProperty in includeProperties.Split(

new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries))

{

query = query.Include(includeProperty);

}

}

return query.FirstOrDefault();

}

public void Remove(T entity)

{

dbSet.Remove(entity);

}

public void RemoveRange(IEnumerable<T> entity)

{

dbSet.RemoveRange(entity);

}

1. Triển khai Repository cho Category

public class CategoryRepository : Repository<Category>, ICategoryRepository

{

private readonly ApplicationDbContext \_db;

public CategoryRepository(ApplicationDbContext db) : base(db)

{

\_db= db;

}

public void Update(Category category)

{

var objFromDb = \_db.Category.FirstOrDefault(u => u.Id == category.Id);

objFromDb.Name = category.Name;

objFromDb.DisplayOrder = category.DisplayOrder;

}

}

1. Sử dụng Unit Of Word cho Repository Category

// Tạo UnitOfWork

public interface IUnitOfWork :IDisposable

{

ICategoryRepository Category { get; }

IFoodTypeRepository FoodType { get; }

IMenuItemRepository MenuItem { get; }

IShoppingCartRepository ShoppingCart { get; }

IOrderHeaderRepository OrderHeader { get; }

IOrderDetailRepository OrderDetail { get; }

IApplicationUserRepository ApplicationUser { get; }

void Save();

}

// Tạo UnitOfWork

public class UnitOfWork : IUnitOfWork

{

private readonly ApplicationDbContext \_db;

public UnitOfWork(ApplicationDbContext db)

{

\_db = db;

Category = new CategoryRepository(\_db);

FoodType = new FoodTypeRepository(\_db);

MenuItem = new MenuItemRepository(\_db);

ShoppingCart = new ShoppingCartRepository(\_db);

OrderDetail = new OrderDetailRepository(\_db);

OrderHeader = new OrderHeaderRepository(\_db);

ApplicationUser = new ApplicationUserRepository(\_db);

}

public ICategoryRepository Category {get;private set;}

public IFoodTypeRepository FoodType { get; private set; }

public IMenuItemRepository MenuItem { get; private set; }

public IShoppingCartRepository ShoppingCart { get; private set; }

public IOrderHeaderRepository OrderHeader { get; private set; }

public IOrderDetailRepository OrderDetail { get; private set; }

public IApplicationUserRepository ApplicationUser { get; private set; }

public void Dispose()

{

\_db.Dispose();

}

public void Save()

{

\_db.SaveChanges();

}

}

1. Áp dụng vào Index Category

public class IndexModel : PageModel

{

private readonly IUnitOfWork \_unitOfWork;

public IEnumerable<Category> Categories{ get; set; }

public IndexModel(IUnitOfWork unitOfWork)

{

\_unitOfWork= unitOfWork;

}

public void OnGet()

{

Categories = \_unitOfWork.Category.GetAll();

}

}

1. Áp dụng vào Create Category

public class CreateModel : PageModel

{

private readonly IUnitOfWork \_unitOfWork;

public Category Category { get; set; }

public CreateModel(IUnitOfWork unitOfWork)

{

\_unitOfWork = unitOfWork;

}

public void OnGet()

{

}

public async Task<IActionResult> OnPost()

{

if (Category.Name == Category.DisplayOrder.ToString())

{

ModelState.AddModelError("Category.Name", "The DisplayOrder cannot exactly match the Name.");

}

if (ModelState.IsValid)

{

\_unitOfWork.Category.Add(Category);

\_unitOfWork.Save();

TempData["success"] = "Category created successfully";

return RedirectToPage("Index");

}

return Page();

}

}

1. Áp dụng Delete Category

public class DeleteModel : PageModel

{

private readonly IUnitOfWork \_unitOfWork;

public Category Category { get; set; }

public DeleteModel(IUnitOfWork unitOfWork)

{

\_unitOfWork = unitOfWork;

}

public void OnGet(int id)

{

Category = \_unitOfWork.Category.GetFirstOrDefault(u=>u.Id==id);

//Category = \_db.Category.FirstOrDefault(u=>u.Id==id);

//Category = \_db.Category.SingleOrDefault(u=>u.Id==id);

//Category = \_db.Category.Where(u => u.Id == id).FirstOrDefault();

}

public async Task<IActionResult> OnPost()

{

var categoryFromDb = \_unitOfWork.Category.GetFirstOrDefault(u => u.Id == Category.Id);

if (categoryFromDb != null)

{

\_unitOfWork.Category.Remove(categoryFromDb);

\_unitOfWork.Save();

TempData["success"] = "Category deleted successfully";

return RedirectToPage("Index");

}

return Page();

}

}

1. Áp dụng cho Edit Category

public class EditModel : PageModel

{

private readonly IUnitOfWork \_unitOfWork;

public Category Category { get; set; }

public EditModel(IUnitOfWork unitOfWork)

{

\_unitOfWork = unitOfWork;

}

public void OnGet(int id)

{

Category = \_unitOfWork.Category.GetFirstOrDefault(u=>u.Id==id);

//Category = \_db.Category.FirstOrDefault(u=>u.Id==id);

//Category = \_db.Category.SingleOrDefault(u=>u.Id==id);

//Category = \_db.Category.Where(u => u.Id == id).FirstOrDefault();

}

public async Task<IActionResult> OnPost()

{

if (Category.Name == Category.DisplayOrder.ToString())

{

ModelState.AddModelError("Category.Name", "The DisplayOrder cannot exactly match the Name.");

}

if (ModelState.IsValid)

{

\_unitOfWork.Category.Update(Category);

\_unitOfWork.Save();

TempData["success"] = "Category updated successfully";

return RedirectToPage("Index");

}

return Page();

}

}

# GIAO DIỆN

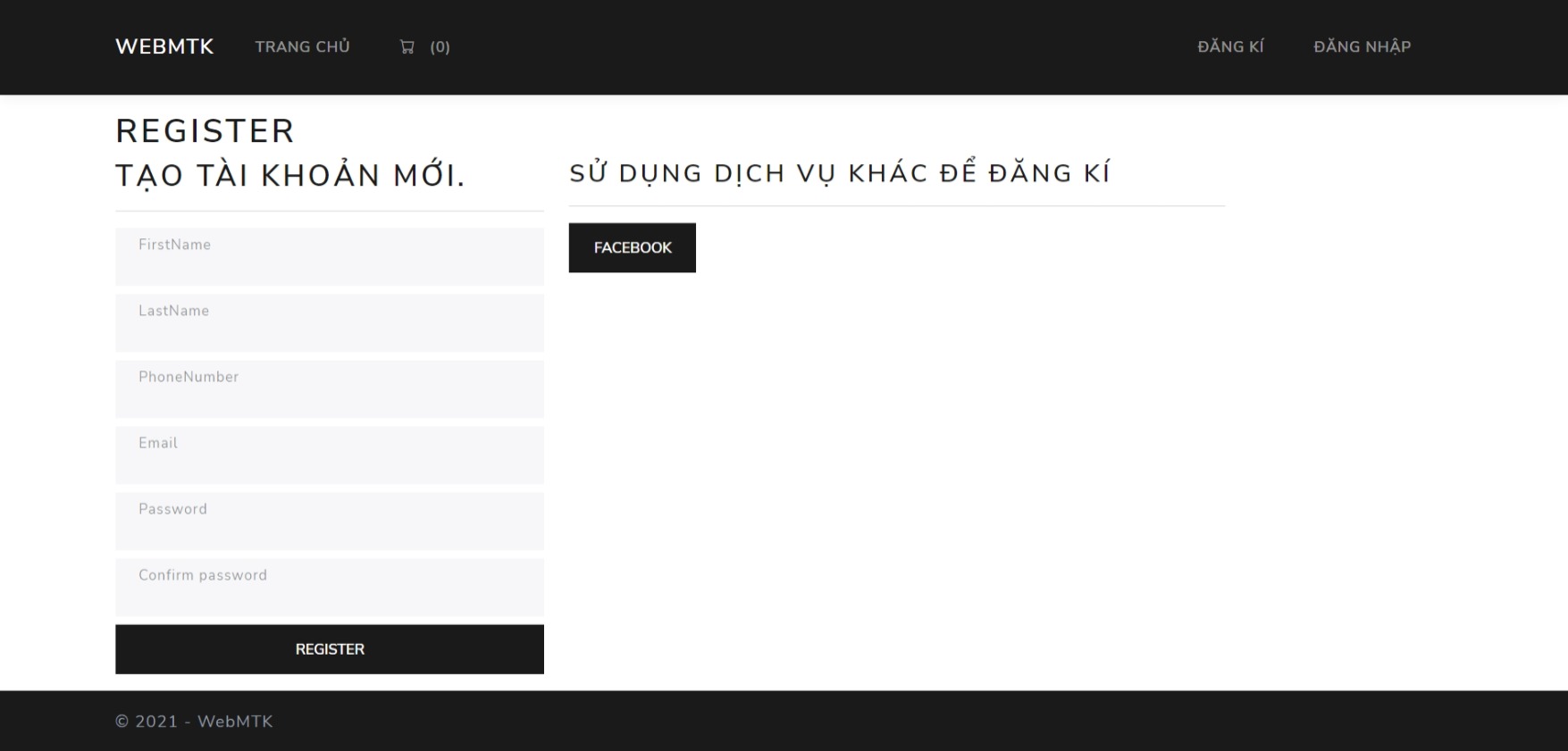
1. Giao diện trang chủ

Website

Description automatically generated with low confidence

Hình . Trang chủ

1. Giao diện đăng ký



Hình . Đăng ký

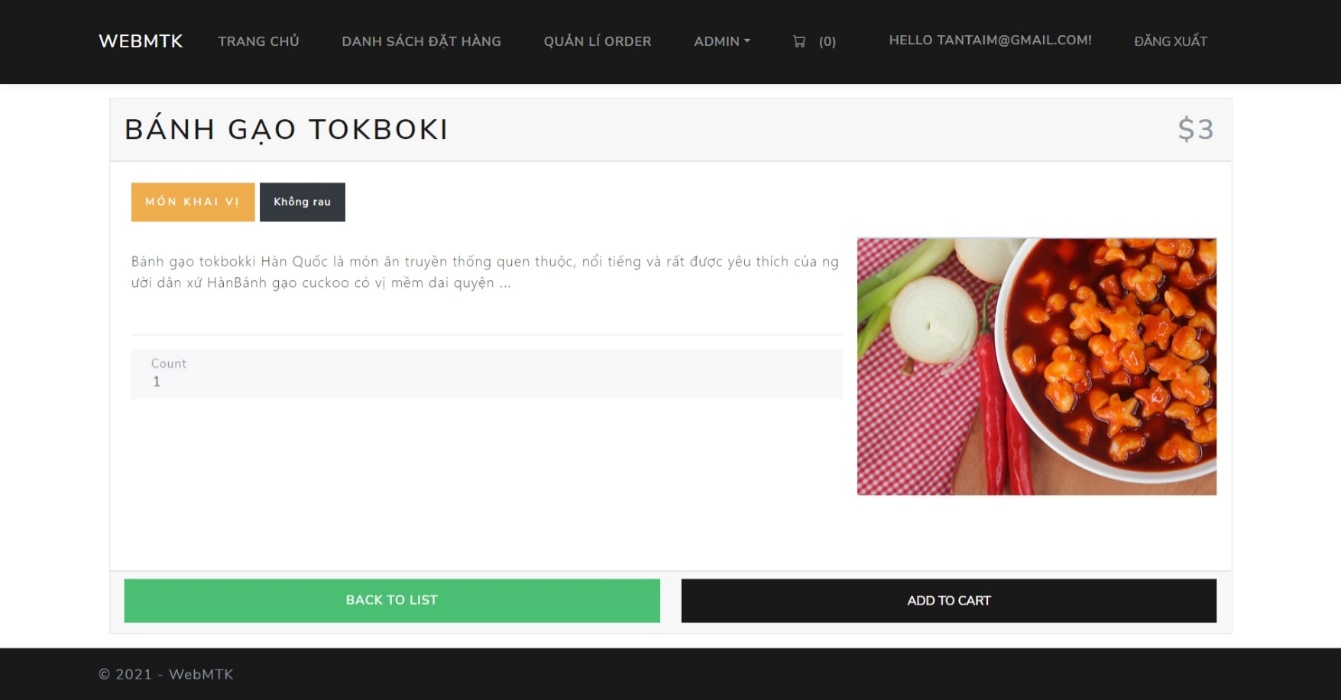
1. Giao diện đăng nhập

Graphical user interface, application, Teams

Description automatically generated

Hình . Đăng nhập

1. Giao diện chi tiết món



Hình . Chi tiết món

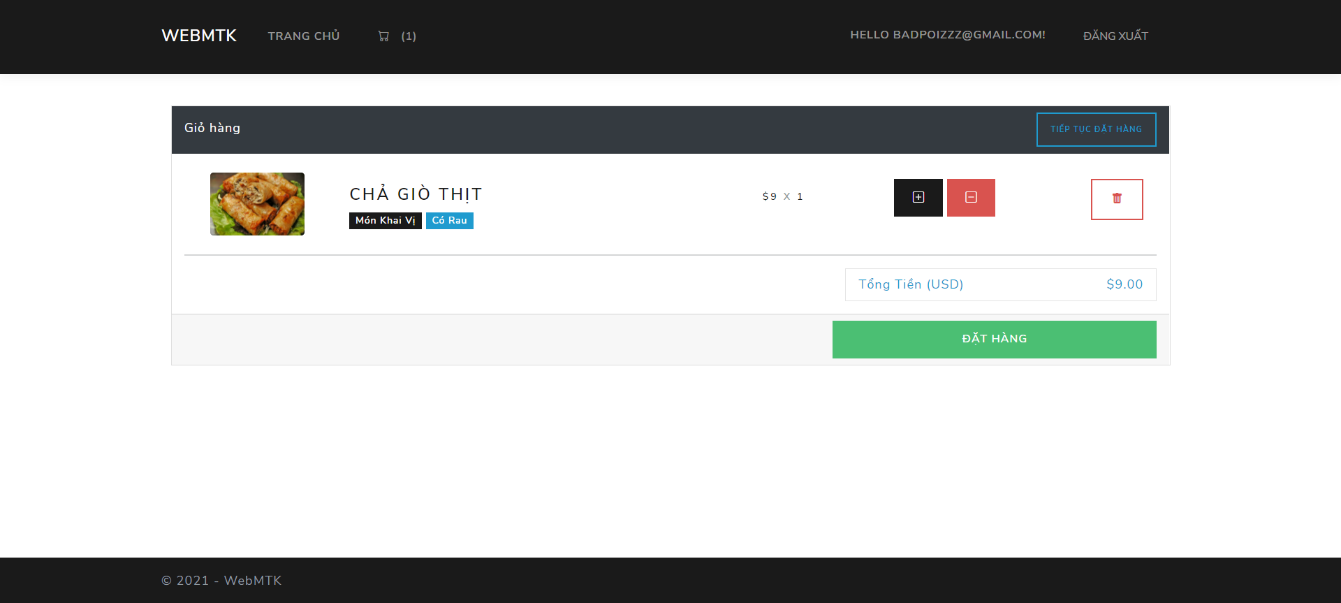
1. Giao diện đặt hàng

Graphical user interface, application, Teams

Description automatically generated

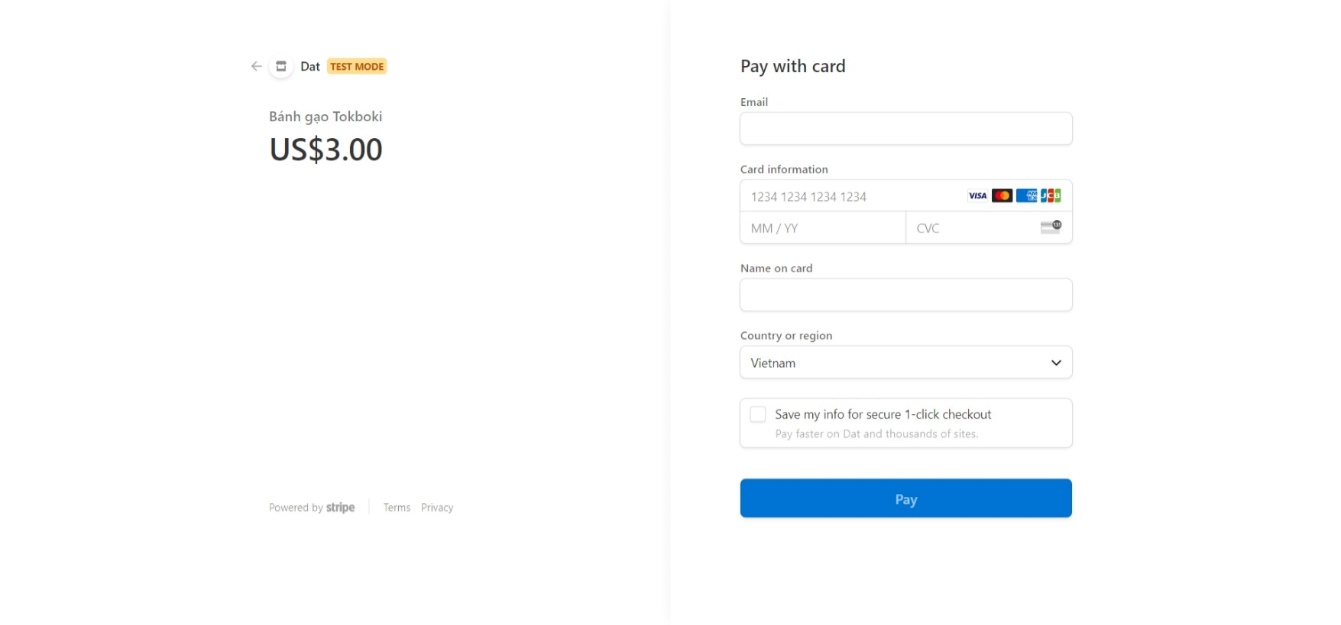
Hình . Đặt hàng

1. Giao diện quản lý giỏ hàng



Hình . Quản lý giỏ hàng

1. Giao diện thanh toán



Hình . Thanh toán

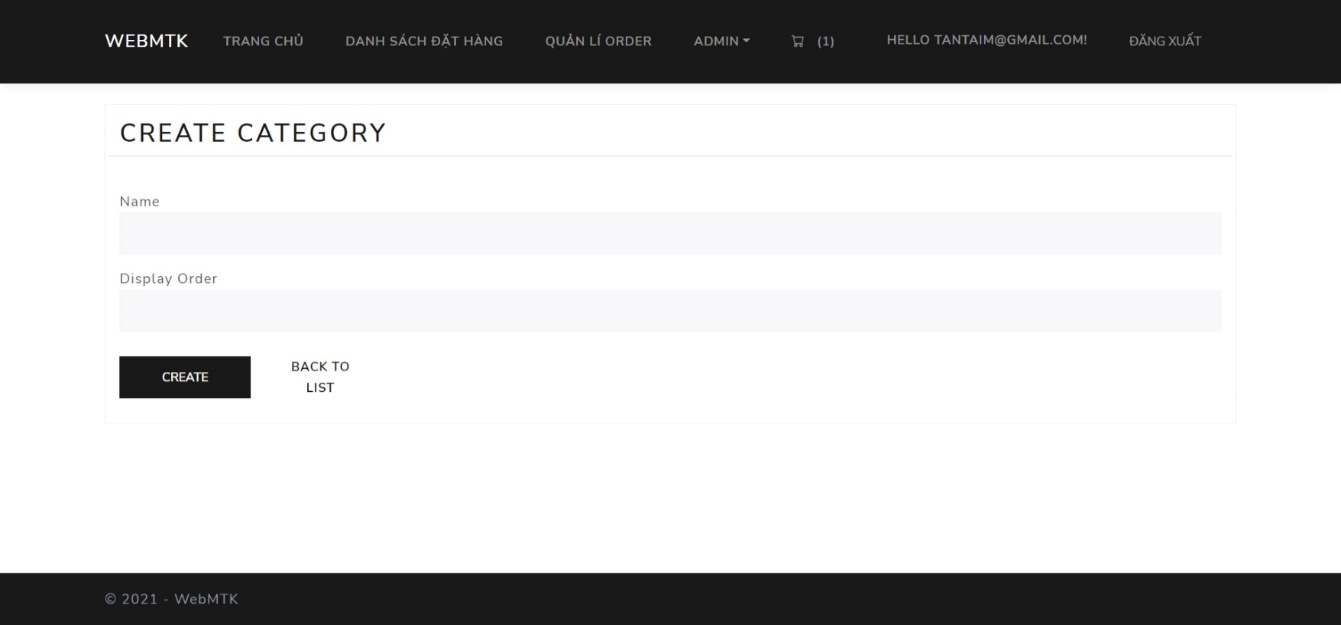
1. Giao diện danh mục món

Graphical user interface, application, Teams

Description automatically generated

Hình . Danh mục món

1. Giao diện thêm danh mục món



Hình . Thêm danh mục món

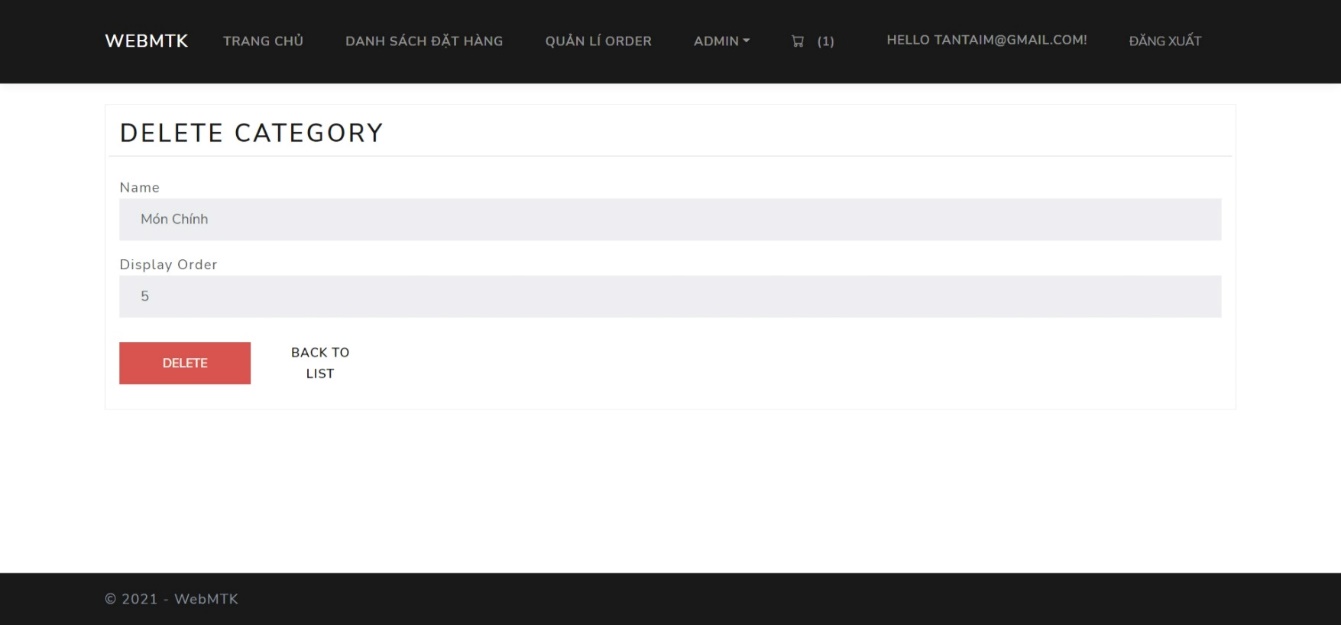
1. Giao diện chỉnh sửa danh mục món

Graphical user interface, text, application, email

Description automatically generated

Hình . Chỉnh sửa danh mục món

1. Giao diện xóa danh mục món



Hình . Xóa danh mục món

1. Giao diện loại món

Graphical user interface, Teams

Description automatically generated

Hình . Loại món

1. Giao diện thêm loại món

Graphical user interface, text, application

Description automatically generated

Hình . Thêm loại món

1. Giao diện chỉnh sửa loại món

Graphical user interface, application

Description automatically generated

Hình . Chỉnh sửa loại món

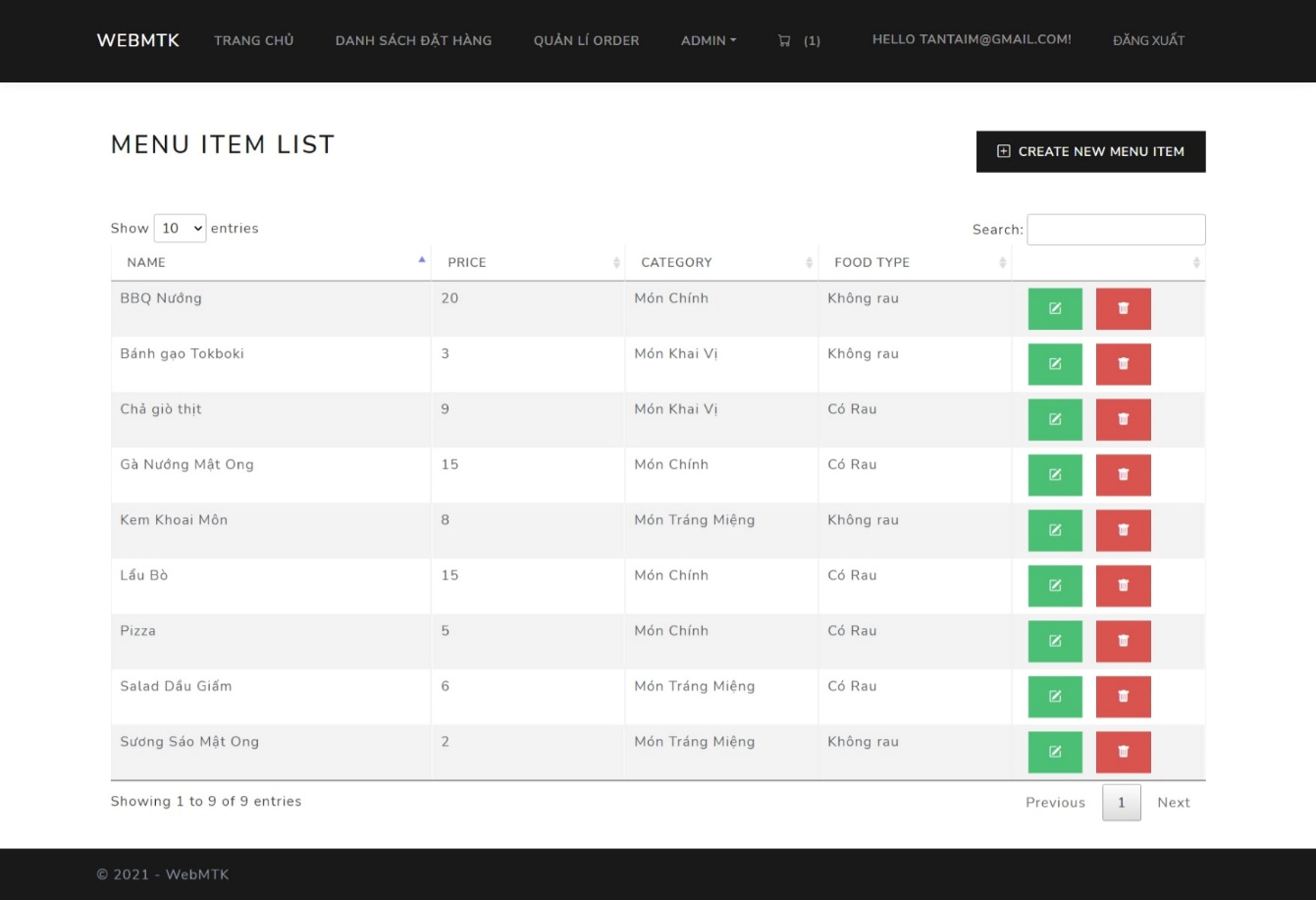
1. Giao diện xóa loại món

Graphical user interface, application

Description automatically generated

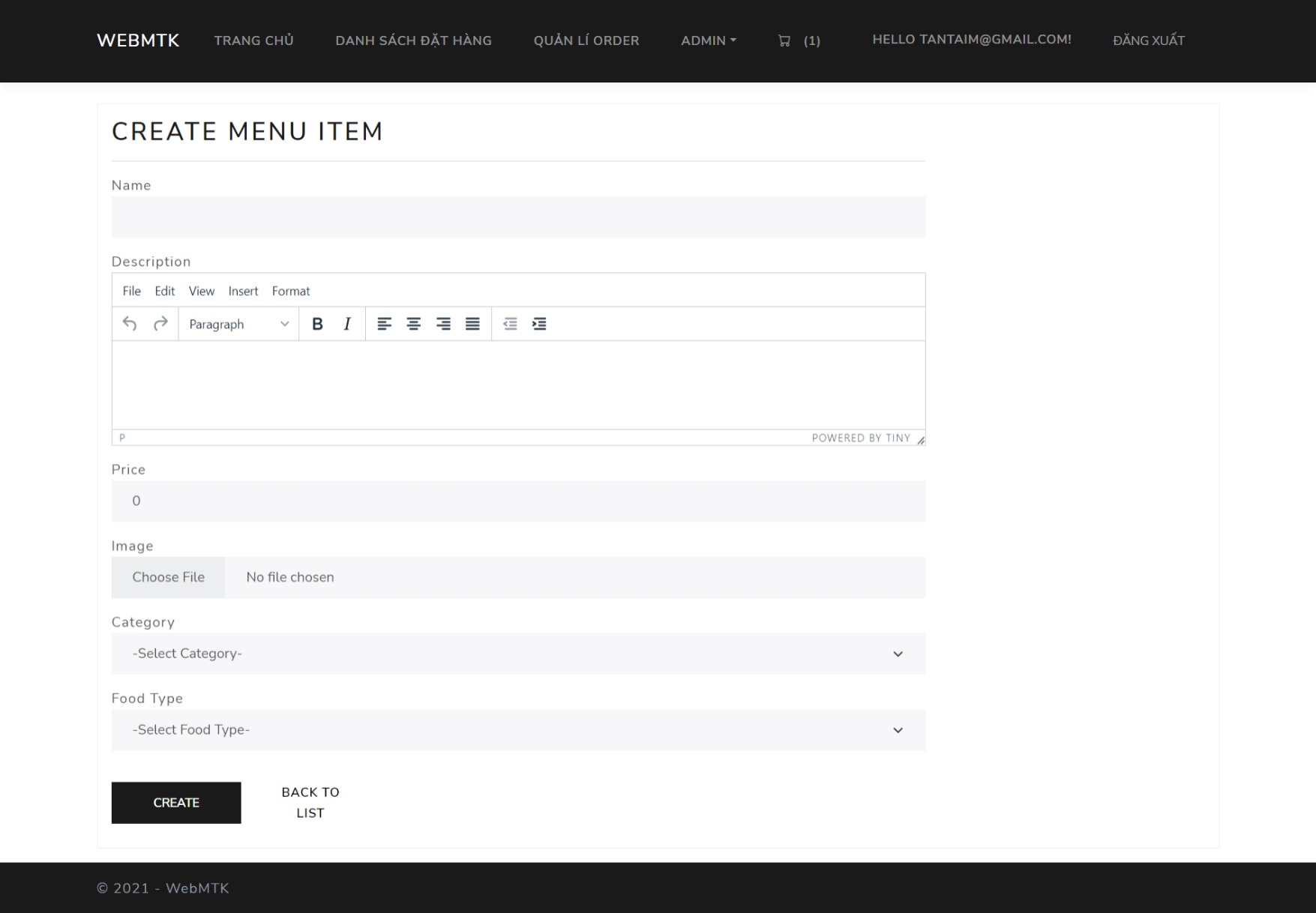
Hình . Xóa loại món

1. Giao diện danh sách món



Hình . Danh sách món

1. Giao diện thêm món



Hình . Thêm món

1. Giao diện xóa món

Graphical user interface, application

Description automatically generated

Hình . Xóa món

1. Giao diện cập nhật món

Graphical user interface, application, website

Description automatically generated

Hình . Cập nhật món

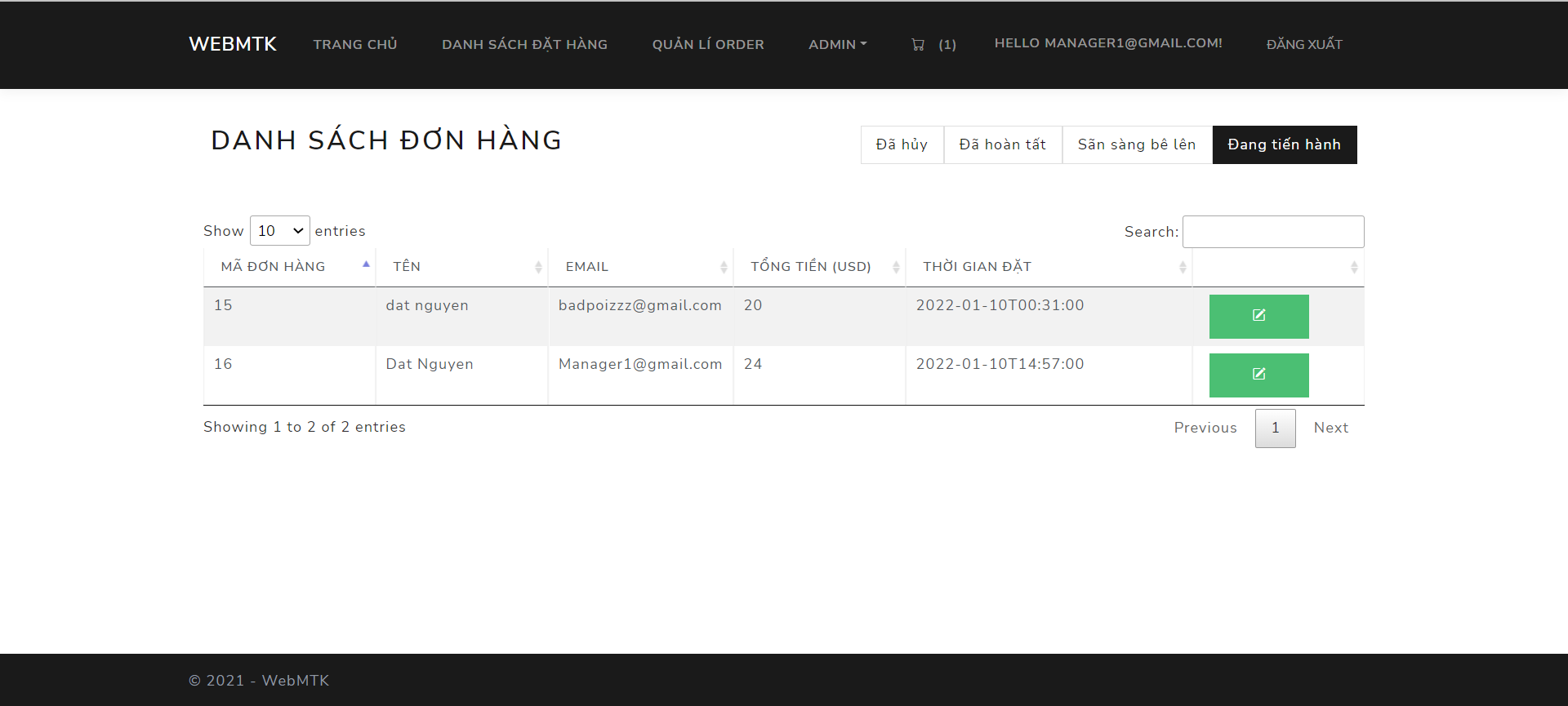
1. Giao diện thêm nhân viên

Graphical user interface, text, application

Description automatically generated

Hình . Thêm nhân viên

1. Giao diện danh sách đặt hàng



Hình . Danh sách đặt hàng

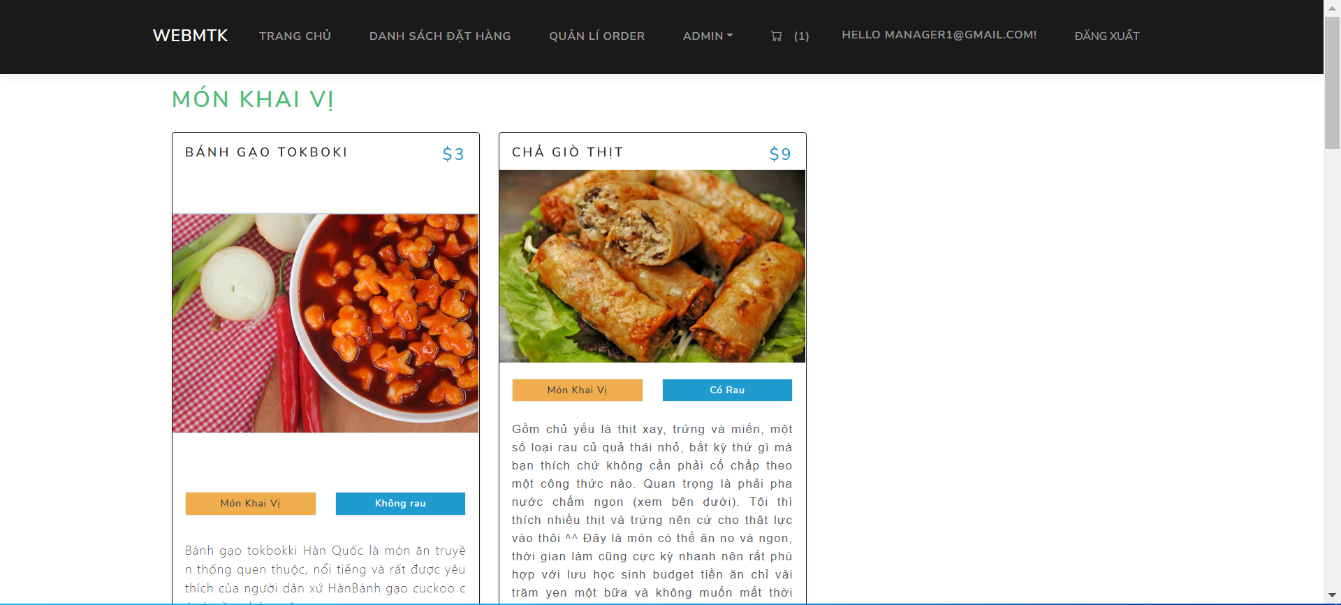
1. Giao diện quản lý tài khoản

Graphical user interface, application, Teams

Description automatically generated

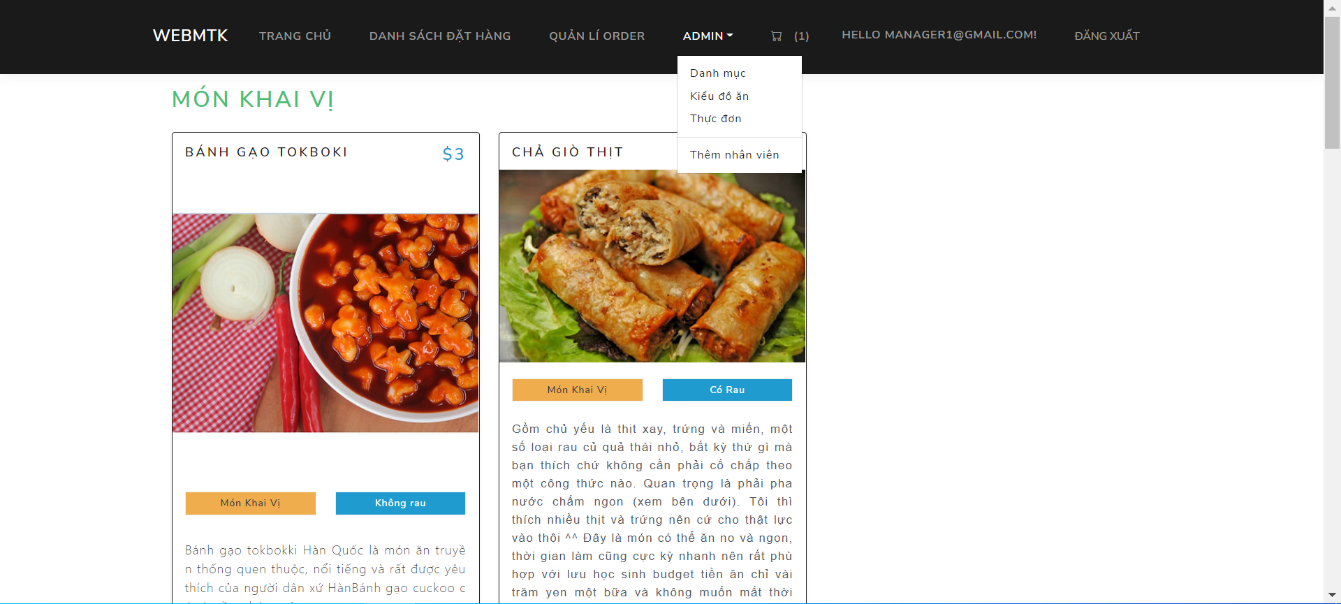
Hình . Quản lý tài khoản

1. Giao diện Admin



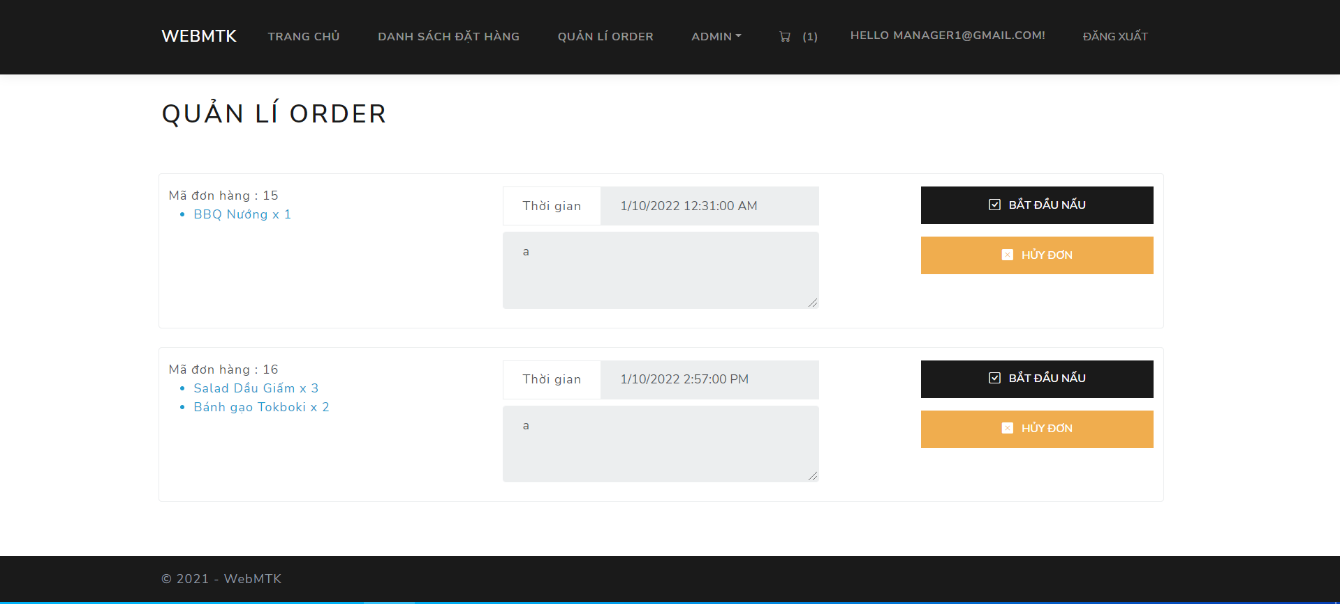
Hình 31.Admin

1. Quyền Admin



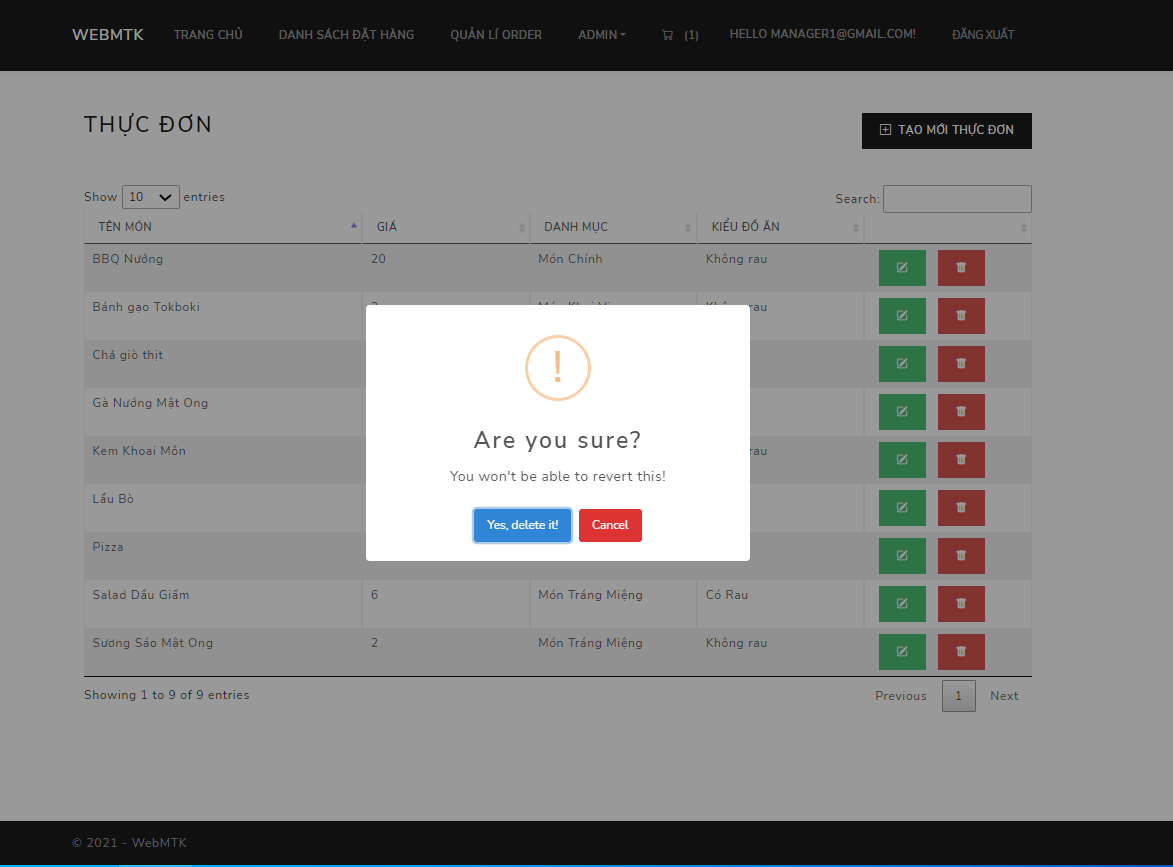
Hình 32.Quyền Admin

1. Quản lý order



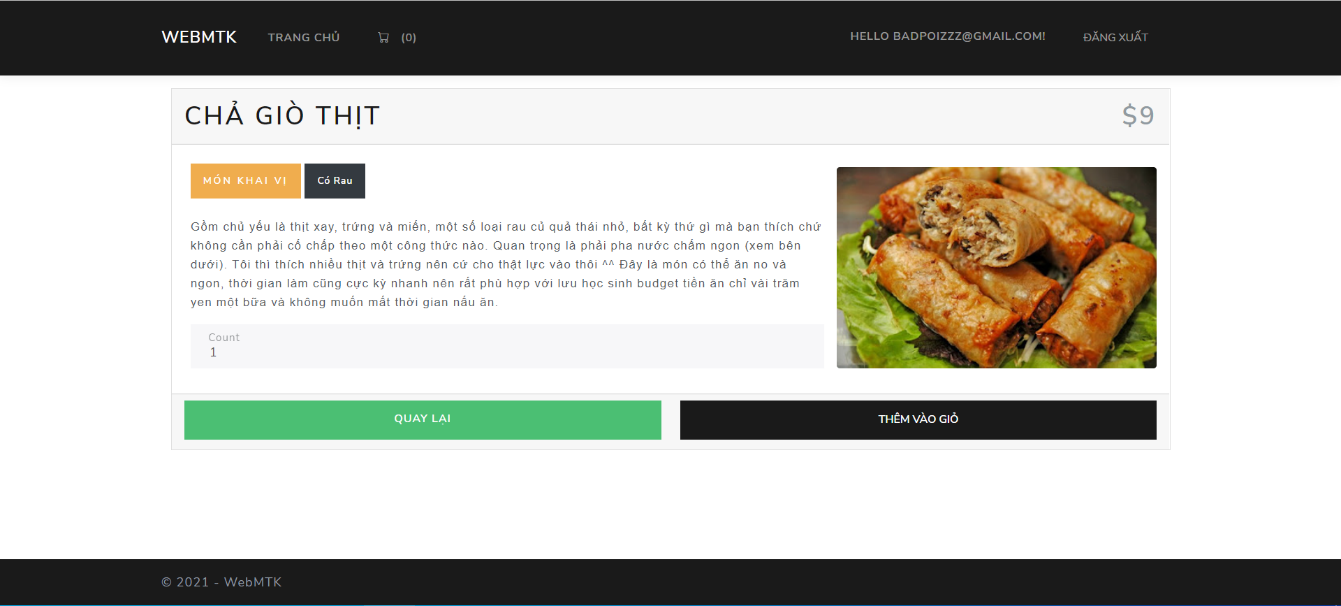
Hình 33.Quản lí order

1. Thông báo xóa món



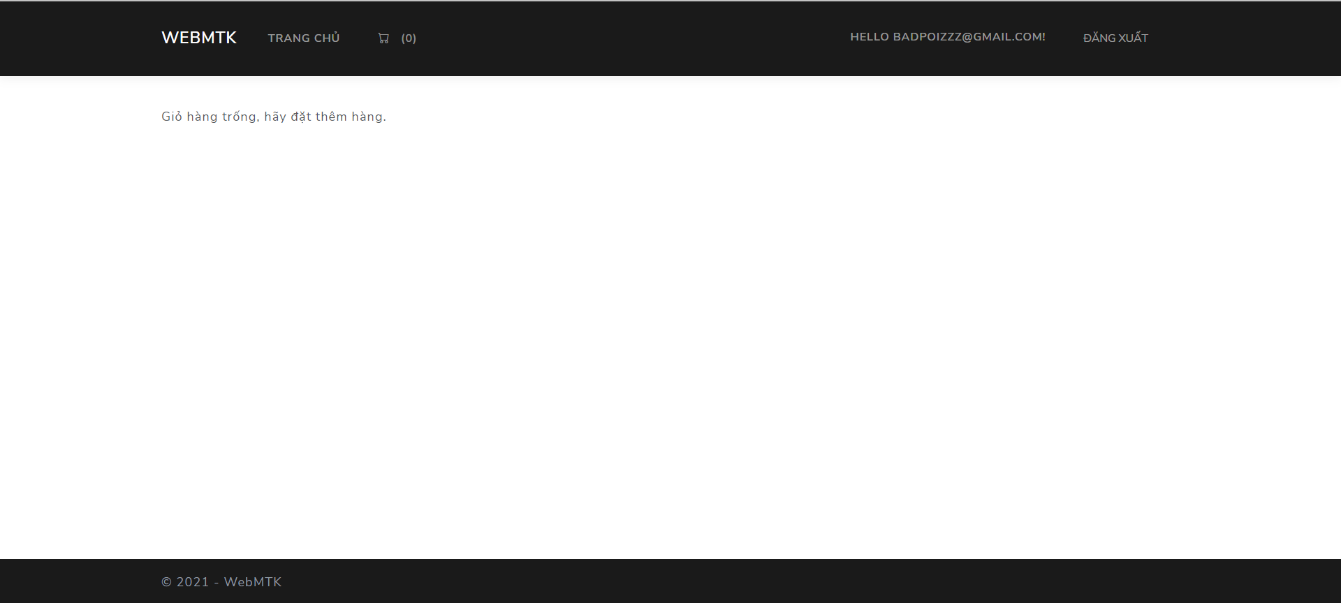
Hình 34. Thông báo xóa món

1. Giao diện khách hàng



Hình 35.Khách hàng

1. Giỏ hàng



Hình 36.Giỏ hàng

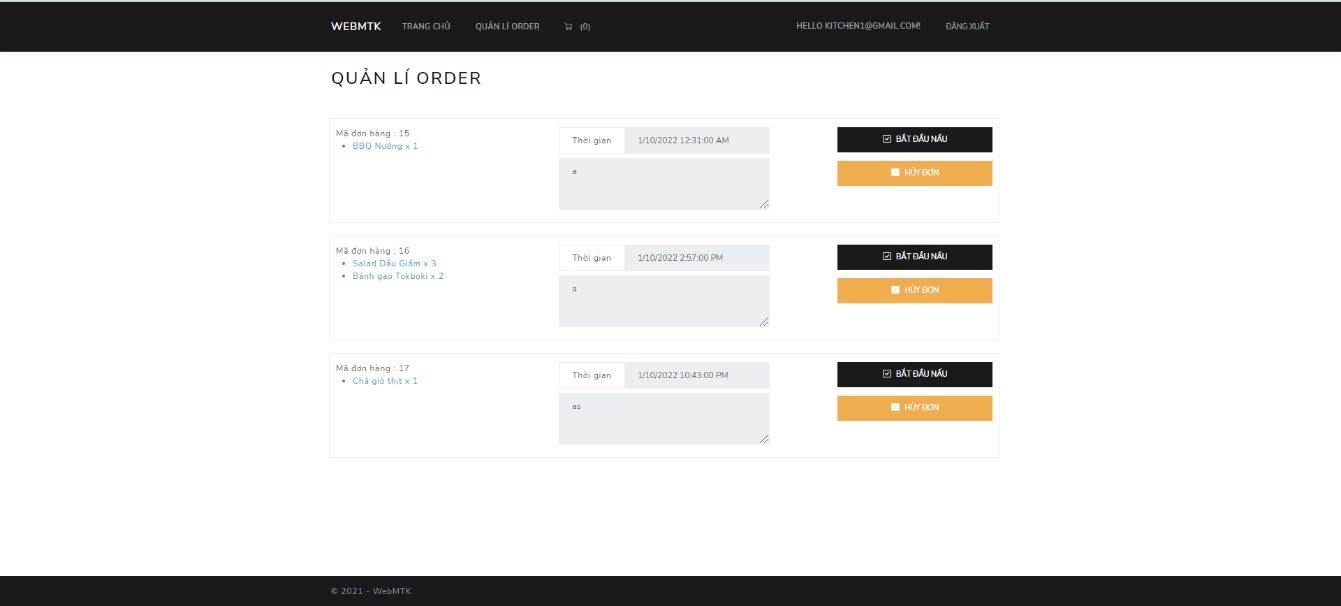
1. Đặt hàng thành công

Text

Description automatically generated

Hình 37.Đặt hàng thành công

1. Quản lí order



Hình 38.Quản lí order

1. Quản lí orer (1)



Hình 39.Quản lí order(1)

# 

# KẾT LUẬN

**1. Những vấn đề đã làm được**

* Giới thiệu đề tài
* Kỷ thuật thực hiện
* Giới thiệu công nghệ
* Đặc điểm công nghệ
* Phân tích yêu cầu
  + UseCase
  + Đặc tả
* Triển khai ứng dụng
* Giao diện

**2. Những vấn đề chưa làm được**

* Giao diện còn đơn giản
* Khả năng sáng tạo còn hạn chế
* Mẫu còn hạn chế

**3. Hướng phát triển**

* Nâng cấp giao diện, thu hút khách hàng
* Tập trung vào trải nghiệm người dùng trên ứng dụng
* Ứng dụng công nghệ
* Đề xuất tiến trình hợp lí.
* Hướng sử dụng công nghệ hợp lí
* Áp dụng thêm nhiều mẫu Pattern

# TÀI LIỆU THAM KHẢO

1. <https://dotnet.microsoft.com/en-us/apps/aspnet>
2. <https://social.technet.microsoft.com/wiki/contents/articles/36287.repository-pattern-in-asp-net-core.aspx>
3. <https://docs.microsoft.com/en-us/dotnet/core/introduction>
4. <https://medium.com/m/global-identity?redirectUrl=https%3A%2F%2Fenlear.academy%2Frepository-pattern-and-unit-of-work-with-asp-net-core-web-api-6802e1aa4f78>