|  |  |
| --- | --- |
|  |  |
| **LÊ VĂN AN** | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC** |
| **NGÀNH KỸ THUẬT PHẦN MỀM** |
| **PHÁT TRIỂN ỨNG DỤNG WEB ĐẶT ĐỒ ĂN CHO NHÀ HÀNG SỬ DỤNG NEXT.JS VÀ .NET CORE** |
|  |
|  |
| **CBHD : TS. Phạm Văn Hà** |
| **KỸ THUẬT PHẦN MỀM** | **Sinh viên : Lê Văn An** |
| **Mã số sinh viên: 2021601519** |
|  |
|  |
|  |
| **Hà Nội – Năm 2024** |
|  |

# **LỜI MỞ ĐẦU**

Em xin được phép bày tỏ lòng biết ơn đến tất cả các cá nhân và tổ chức đã tạo điều kiện hỗ trợ, giúp đỡ em trong suốt quá trình học tập và nghiên cứu đề tài “ *Phát triển ứng dụng web đặt đồ ăn cho nhà hàng sử dụng Next.js và .NET Core*”. Em cũng xin chân thành cảm ơn các thầy cô ở khoa công nghệ thông tin đã truyền đạt vốn kiến thức quý báu cho em trong suốt thời gian học tập tại trường. Nhờ có những lời hướng dẫn, dạy bảo của các thầy, các cô nên đề tài tốt nghiệp: “*Phát triển ứng dụng web đặt đồ ăn cho nhà hàng sử dụng Next.js và .Net Core*” của em mới có thể hoàn thiện. Một lần nữa, em xin chân thành cảm ơn TS. Phạm Văn Hà - người đã trực tiếp giúp đỡ, quan tâm, hướng dẫn em hoàn thành tốt bài báo cáo này trong thời gian qua. Bài báo cáo đồ án của em thực hiện trong khoảng thời gian 9 tuần.

Bài báo cáo của em bao gồm 4 chương:

*Chương 1: Trình bày tổng quan về đề tài, công nghệ và cơ sở lý thuyết.*

*Chương 2: Trình bày về việc khảo sát và đặc tả ứng dụng web đặt món ăn cho nhà hàn, đưa ra thiết kế chức năng, thiết kế cơ sở dữ liệu và thiết kế giao diện của hệ thống.*

*Chương 3: Triển khai ứng dụng web và kết quả đạt được.*

*Chương 4: Kiểm thử hệ thống*

Bước đầu đi vào thực tế của em còn hạn chế và gặp nhiều bỡ ngỡ nên không khỏi gặp nhiều thiếu sót, em rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô để kiến thức của em trong lĩnh vực này được hoàn thiện hơn đồng thời có điều kiện bổ sung, nâng cao kỹ năng nghề nghiệp của em.

Em xin chân thành cảm ơn!

Hà Nội, ngày … tháng … năm 2024

Sinh Viên

**Lê Văn An**

# **MỤC LỤC**

[**LỜI MỞ ĐẦU** i](#_Toc186235890)

[**MỤC LỤC** ii](#_Toc186235891)

[**DANH MỤC CÁC CHỮ VIẾT TẮT** v](#_Toc186235892)

[**DANH MỤC BẢNG BIỂU** vi](#_Toc186235893)

[**DANH MỤC HÌNH ẢNH** vii](#_Toc186235894)

[**MỞ ĐẦU** 1](#_Toc186235895)

[**1.1.** **Lý do chọn đề tài** 1](#_Toc186235896)

[**1.2.** **Mục tiêu đề tài** 2](#_Toc186235897)

[**1.3.** **Nội dung nghiên cứu** 2](#_Toc186235898)

[**1.4.**  **Phạm vi đề tài** 3](#_Toc186235899)

[**1.5.** **Bố cục đề tài** 3](#_Toc186235900)

[**CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI** 4](#_Toc186235901)

[**1.1.** **Tổng quan về đề tài** 4](#_Toc186235902)

[**1.2.** **Tổng quan NextJS** 5](#_Toc186235903)

[**1.2.1.**  **Giới thiệu về NextJS** 5](#_Toc186235904)

[**1.2.2.** **Các tính năng chính của NextJS** 6](#_Toc186235905)

[**1.2.3.** **Ưu điểm của NextJS** 8](#_Toc186235906)

[**1.2.4.** **Nhược điểm của NextJS** 9](#_Toc186235907)

[**1.3.**  **Tổng quan My SQL** 10](#_Toc186235908)

[**1.3.1.**  **MySQL là gì?** 10](#_Toc186235909)

[**1.3.2.**  **Lịch sử phát triển của MySQL** 11](#_Toc186235910)

[**1.3.3.**  **Ưu điểm của MySQL** 11](#_Toc186235911)

[**1.3.4.**  **Nhược điểm của MySQL** 11](#_Toc186235912)

[**1.4.**  **Tổng quan .NET Core và ASP.NET Core** 12](#_Toc186235913)

[**1.4.1.**  **.NET Core là gì?** 12](#_Toc186235914)

[**1.4.2.**  **ASP.NET Core** 14](#_Toc186235915)

[**1.5.**  **Kiến trúc Monolithic** 15](#_Toc186235916)

[**1.5.1.**  **Kiến trúc phần mềm là gì?** 16](#_Toc186235917)

[**1.5.2.**  **Định nghĩa** 16](#_Toc186235918)

[**1.5.3.**  **Kiến trúc phần mềm trong kỹ thuật phần mềm** 17](#_Toc186235919)

[**1.5.5.**  **Hiểu kiến trúc nguyên khối với sản phẩm phần mềm** 17](#_Toc186235920)

[**1.5.6.**  **Lợi ích của ứng dụng nguyên khối** 18](#_Toc186235921)

[**1.5.7.**  **Hạn chế của ứng dụng nguyên khối** 18](#_Toc186235922)

[**1.6.**  **Clean architecture** 19](#_Toc186235923)

[**1.6.1.**  **Clean Architecture là gì?** 19](#_Toc186235924)

[**1.6.2.**  **Lợi ích của Clean Architecture** 21](#_Toc186235925)

[**1.6.3.**  **CQRS Pattern** 21](#_Toc186235926)

[**CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ PHẦN MỀM** 23](#_Toc186235927)

[**2.1.** **Khảo sát các ứng dụng web đặt món ăn ngay tại nhà hàng, quán ăn trên thị trường hiện nay.** 23](#_Toc186235928)

[**2.2.** **Khảo sát hệ thống.** 24](#_Toc186235929)

[**2.2.1.**  **Giới thiệu chung về hệ thống.** 24](#_Toc186235930)

[**2.2.2.**  **Các yêu cầu chức năng** 25](#_Toc186235931)

[**2.3.**  **Biểu đồ use case.** 27](#_Toc186235932)

[**2.3.1.**  **Các use case.** 27](#_Toc186235933)

[**2.3.2.**  **Phân rã use case.** 29](#_Toc186235934)

[**2.3.3.**  **Mô tả chi tiết một số use case.** 30](#_Toc186235935)

[**3.3.4.** **Biểu đồ trình tự một số use case.** 32](#_Toc186235936)

[**2.4.**  **Thiết kế cơ sở dữ liệu** 35](#_Toc186235937)

[**2.4.1.**  **Biểu đồ Entity Relationship Diagram** 35](#_Toc186235938)

[**2.4.2.**  **Thiết kế các bảng trong cơ sở dữ liệu** 35](#_Toc186235939)

[**2.5.**  **Thiết kế giao diện** 37](#_Toc186235940)

[**CHƯƠNG 3: KẾT QUẢ XÂY DỰNG** 41](#_Toc186235941)

[**3.1.**  **Cài đặt** 41](#_Toc186235942)

[**3.1.1.**  **Frontend** 41](#_Toc186235943)

[**3.1.2.**  **Backend (Xử lý nghiệp vụ)** 41](#_Toc186235944)

[**3.1.3.**  **Cơ sở dữ liệu** 41](#_Toc186235945)

[**3.1.4.**  **Giao tiếp giữa các thành phần** 42](#_Toc186235946)

[**3.1.5. Triển khai Clean Architecture** 42](#_Toc186235947)

[**3.2.**  **Giao diện** 44](#_Toc186235948)

[**CHƯƠNG 4: KIỂM THỬ PHẦN MỀM** 50](#_Toc186235949)

[**4.1.**  **Kế hoạch kiểm thử** 50](#_Toc186235950)

[**4.1.1.**  **Giới thiệu** 50](#_Toc186235951)

[**4.1.2.**  **Thực hiện kiểm thử** 54](#_Toc186235952)

[**4.2.**  **Kết quả kiểm thử** 58](#_Toc186235953)

[**4.3.**  **Kết luận** 59](#_Toc186235954)

[**KẾT LUẬN** 60](#_Toc186235955)

[**1.** **Kết quả đạt được.** 60](#_Toc186235956)

[**2.** **Hướng phát triển.** 60](#_Toc186235957)

[**TÀI LIỆU THAM KHẢO** 61](#_Toc186235958)

# **DANH MỤC CÁC CHỮ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| Từ viết tắt | Tiếng anh | Tiếng Việt |
| API | Application Programming Interface | Giao diện lập trình ứng dụng |
| CQRS | Command and Query Responsibility Segregation | Phân tách Lệnh và Truy vấn |

# 

# **DANH MỤC BẢNG BIỂU**

[Bảng 4.1. Bảng định nghĩa và các từ viết tắt 50](#_Toc185950766)

[Bảng 4.2. Bảng lịch trình công việc 51](#_Toc185950767)

[Bảng 4.3. Bảng cấu hình phần cứng 51](#_Toc185950768)

[Bảng 4.4. Bảng thông tin phần mềm 51](#_Toc185950769)

[Bảng 4.5. Bảng công cụ kiểm thử 52](#_Toc185950770)

[Bảng 4.6. Bảng nhân sự 52](#_Toc185950771)

[Bảng 4.7. Bảng loại kiểm thử hộp đen 53](#_Toc185950772)

[Bảng 4.8. Bảng loại phân loại lỗi 54](#_Toc185950773)

[Bảng 4.9. Bảng điều kiện ràng buộc cho Đăng nhập 55](#_Toc185950774)

[Bảng 4.10. Bảng điều kiện ràng buộc cho Đổi mật khẩu 56](#_Toc185950775)

# **DANH MỤC HÌNH ẢNH**

[Hình 1.1. Framework Next.js 5](#_Toc185950701)

[Hình 1.2. Cơ chế Server-Side Rendering 7](#_Toc185950702)

[Hình 1.3. Cơ chế Client-Side Rendering 8](#_Toc185950703)

[Hình 1.4. Logo MySQL 10](#_Toc185950704)

[Hình 1.5. Logo .NET Core 13](#_Toc185950705)

[Hình 1.6. Logo ASP.NET Core 14](#_Toc185950706)

[Hình 1.7. Kiến trúc Monolith 16](#_Toc185950707)

[Hình 1.8. Clean Architecture 19](#_Toc185950708)

[Hình 1.9. CQRS Pattern 21](#_Toc185950709)

[Hình 2.1. Biểu đồ use case phía frontend. 27](#_Toc185950710)

[Hình 2.2. Biểu đồ use case phía backend. 28](#_Toc185950711)

[Hình 2.3. Biểu đồ phân rã use case phía frontend. 29](#_Toc185950712)

[Hình 2.4. Biểu đồ phân rã use case phía backend. 29](#_Toc185950713)

[Hình 2.6. Biểu đồ trình tự Use case Xem món ăn. 32](#_Toc185950714)

[Hình 2.7. Biểu đồ trình tự Use case Bảo trì món ăn. 34](#_Toc185950719)

[Hình 2.8. Biểu đồ Entity Relationship Diagram 35](#_Toc185950720)

[Hình 2.9. Bảng Tables 35](#_Toc185950721)

[Hình 2.10. Bảng Accounts 36](#_Toc185950722)

[Hình 2.11. Bảng Dishes 36](#_Toc185950723)

[Hình 2.12. Bảng DishSnapshots 36](#_Toc185950724)

[Hình 2.13. Bảng Guests 36](#_Toc185950725)

[Hình 2.14. Bảng Orders 37](#_Toc185950726)

[Hình 2.15. Bảng RefreshTokens 37](#_Toc185950727)

[Hình 2.16. Giao diện mockup Đăng nhập 37](#_Toc185950728)

[Hình 2.17. Giao diện mockup Trang chủ 38](#_Toc185950729)

[Giao diện mockup Quản lý đơn hàng: 38](#_Toc185950730)

[Hình 2.18. Giao diện mockup Quản lý đơn hàng 38](#_Toc185950731)

[Hình 2.19. Giao diện mockup Quản lý bàn ăn 39](#_Toc185950732)

[Hình 2.20. Giao diện mockup Thống kê 39](#_Toc185950734)

[Hình 2.21. Giao diện mockup Menu 40](#_Toc185950735)

[Hình 3.1. Giao diện đăng nhập 44](#_Toc185950736)

[Hình 3.2. Màn hình trang chủ 45](#_Toc185950737)

[Hình 3.3. Màn hình quản lý đơn hàng 45](#_Toc185950738)

[Hình 3.4. Màn hình chi tiết quản lý đơn hàng tại một bàn 45](#_Toc185950739)

[Hình 3.5. Màn hình chi tiết cập nhật đơn hàng 46](#_Toc185950740)

[Hình 3.6. Màn hình quản lý bàn ăn 46](#_Toc185950741)

[Hình 3.7. Màn hình quản lý món ăn 47](#_Toc185950742)

[Hình 3.8. Màn hình thống kê doanh thu 47](#_Toc185950743)

[Hình 3.9. Màn hình quản lý nhân viên 48](#_Toc185950745)

[Hình 3.10. Màn hình chi tiết menu món ăn 48](#_Toc185950746)

[Hình 3.11. Màn hình đơn đặt hàng 49](#_Toc185950748)

[Hình 4.1. Giao diện Đăng nhập 54](#_Toc185950749)

[Hình 4.2. Danh sách Test case Đăng nhập 55](#_Toc185950750)

[Hình 4.3. Kết quả kiểm thử Đăng nhập bằng Selenium 56](#_Toc185950751)

[Hình 4.4. Giao diện Đổi mật khẩu. 56](#_Toc185950752)

[Hình 4.5. Danh sách Test case Đổi mật khẩu 57](#_Toc185950753)

[Hình 4.6. Kết quả kiểm thử Đổi mật khẩu bằng Selenium 57](#_Toc185950754)

[Hình 4.7. Giao diện. thử thông báo realtime cho người dùng về trạng thái đơn đặt hàng 58](#_Toc185950755)

[Hình 4.8. Danh sách Test case Kiểm thử thông báo realtime cho người dùng về trạng thái đơn đặt hàng 58](#_Toc185950756)

[Hình 4.9. Kết quả kiểm thử Thông báo realtime bằng Selenium 58](#_Toc185950757)

[Hình 4.10. Test report 59](#_Toc185950758)

# **MỞ ĐẦU**

## **1.1. Lý do chọn đề tài**

Hiện nay công nghệ thông tin không ngừng phát triển mạnh mẽ và hiện đại, các nền tảng số ngày càng tiếp cận tới người dùng dễ dàng hơn, trong đó có các hệ thống đặt đồ món ăn dần được phổ biến trên thị trường hiện nay với nhiều nền tảng công nghệ.

Đề tài "Phát triển ứng dụng web đặt đồ ăn cho nhà hàng sử dụng Next.js và .NET Core" nhằm mục đích xây dựng một nền tảng hiện đại, thân thiện với người dùng, và đáp ứng tốt nhu cầu thực tế của các nhà hàng. Với sự kết hợp giữa Next.js - một framework phát triển giao diện người dùng nhanh và tối ưu, cùng .NET Core - nền tảng mạnh mẽ cho việc xây dựng backend, ứng dụng sẽ cung cấp một hệ thống toàn diện bao gồm các chức năng như đặt món trực tuyến, quản lý thực đơn, xử lý thanh toán, và theo dõi trạng thái đơn hàng.

Bên cạnh đó, việc đặt món qua mạng giúp cửa hàng có thể tiết kiệm ngân sách nhân viên, tối ưu quản lý nghiệp vụ. Hơn nữa, mỗi khách hàng chúng ta hiện nay ngày càng ưu chuộng việc tiện lợi, nhanh chóng và mất ít công sức hơn. Đối với thị trường Việt Nam, việc có các ứng dụng đặt món ngay tại cửa hàng online đang phổ biến hơn vì sự tiện lợi và nhanh chóng của nó.

Với mong muốn củng cố và nâng cao kiến thức về lập trình, áp dụng kiến thức để xây dựng một ứng dung giúp người dùng có thể dễ dàng sử dụng đặt món ăn, tham khảo, lựa chọn các thực phẩm mình mong muốn với mức giá chuẩn xác, sử dụng hệ thống với công nghệ mới nhanh và mượt mà hơn em đã chọn đề tài “Phát triển ứng dụng web đặt đồ ăn cho nhà hàng sử dụng Next.js và .Net Core”.

Đề tài mang tính thực tiễn cao, phù hợp với xu hướng hiện tại và mở ra cơ hội nghiên cứu sâu hơn về việc ứng dụng công nghệ trong ngành dịch vụ. Với những lợi ích thiết thực mà đề tài mang lại, nó không chỉ là một bài tập tốt nghiệp mà còn có thể triển khai thực tế, mang lại giá trị lâu dài cho cả nhà hàng và khách hàng.

## **1.2. Mục tiêu đề tài**

Làm chủ Nextjs và .NET Core, phân tích, thiết kế, phát triển ứng dụng web đặt đồ ăn cho nhà hàng, đáp ứng các yêu cầu cơ bản của một trang đặt đồ ăn.

Ứng dụng sẽ bao gồm hai phân hệ chính:

Phân hệ khách hàng: Cho phép người dùng xem thực đơn, chọn món, đặt hàng và theo dõi trạng thái đơn hàng. Giao diện được thiết kế trực quan, thân thiện, tương thích với nhiều loại thiết bị như máy tính và điện thoại.

Phân hệ quản trị: Dành cho quản lý nhà hàng để theo dõi và xử lý đơn hàng, cập nhật thực đơn, quản lý tài khoản nhân viên, và xem báo cáo doanh thu.

Mục tiêu chính của đề tài là cung cấp một giải pháp hoàn chỉnh, dễ sử dụng và có khả năng mở rộng trong tương lai. Việc sử dụng các công nghệ tiên tiến như Next.js và .NET Core không chỉ đảm bảo hiệu suất và tính bảo mật cao, mà còn giúp ứng dụng dễ dàng tích hợp với các hệ thống khác hoặc nâng cấp để đáp ứng các yêu cầu mới.

## **1.3. Nội dung nghiên cứu**

Trong phạm vi đề tài và bài báo cáo, em tiến hành các nội dung sau:

* Khảo sát và phân tích yêu cầu quy trình đặt món ăn.
* Sử dụng API Cloudinary.
* Sử dụng kiến trúc Monolithic và Clean Architecture
* Phân tích thiết kế hệ thống.
* Thiết kế cơ sở dữ liệu.
* Lập trình backend API với .NET Core.
* Lập trình giao diện sử dụng Nextjs và kết nối tới API.
* Có kế hoạch và đánh giá kiểm thử hệ thống

## **1.4. Phạm vi đề tài**

Đề tài "Phát triển ứng dụng web đặt đồ ăn cho nhà hàng sử dụng Next.js và .NET Core" tập trung vào việc xây dựng hệ thống đặt đồ ăn trực tuyến dành cho nhà hàng.

* Các tính năng chính như hiển thị thực đơn, đặt món, quản lý đơn hàng và theo dõi trạng thái đơn hàng, quản lý nhân viên, quản lý món ăn.
* Ứng dụng chỉ hỗ trợ thanh toán trực tiếp tại nhà hàng khi khách nhận món, chưa tích hợp các cổng thanh toán trực tuyến.
* Phần báo cáo thống kê trong hệ thống giới hạn ở việc hiển thị doanh thu theo ngày, tuần và tháng mà không đi sâu vào các chỉ số phân tích nâng cao như hành vi người dùng hay xu hướng tiêu dùng.
* Ngoài ra, ứng dụng được thiết kế hướng đến quy mô nhỏ và vừa, chưa hỗ trợ tích hợp đa nhà hàng hoặc hệ thống quản lý chuỗi.

## **1.5. Bố cục đề tài**

Ngoài các phần Mở đầu, Kết luận và Tài liệu tham khảo, báo cáo đồ án được bố cục thành ba chương chính sau.

* Chương 1: Trình bày tổng quan về đề tài, công nghệ và cơ sở lý thuyết.
* Chương 2: Trình bày về việc khảo sát và đặc tả ứng dụng web đặt món ăn cho nhà hàn, đưa ra thiết kế chức năng, thiết kế cơ sở dữ liệu và thiết kế giao diện của hệ thống.
* Chương 3: Triển khai ứng dụng web và kết quả đạt được.
* Chương 4: Kiểm thử hệ thống

# **CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI**

## **Tổng quan về đề tài**

Ngày nay, với sự phát triển không ngừng của công nghệ, các doanh nghiệp trong ngành dịch vụ ăn uống đang dần chuyển đổi sang mô hình số hóa để đáp ứng nhu cầu của khách hàng và nâng cao hiệu quả hoạt động. Một trong những xu hướng phổ biến nhất là xây dựng các ứng dụng web cho phép khách hàng dễ dàng đặt đồ ăn trực tuyến. Ứng dụng không chỉ giúp tiết kiệm thời gian, tạo sự tiện lợi cho khách hàng mà còn giúp nhà hàng quản lý đơn hàng và dữ liệu một cách hiệu quả hơn.

Đề tài "Phát triển ứng dụng web đặt đồ ăn cho nhà hàng sử dụng Next.js và .NET Core" nhằm mục đích xây dựng một nền tảng hiện đại, thân thiện với người dùng, và đáp ứng tốt nhu cầu thực tế của các nhà hàng. Với sự kết hợp giữa Next.js - một framework phát triển giao diện người dùng nhanh và tối ưu, cùng .NET Core - nền tảng mạnh mẽ cho việc xây dựng backend, ứng dụng sẽ cung cấp một hệ thống toàn diện bao gồm các chức năng như đặt món trực tuyến, quản lý thực đơn, xử lý thanh toán, và theo dõi trạng thái đơn hàng.

Ứng dụng sẽ bao gồm hai phân hệ chính:

Phân hệ khách hàng: Cho phép người dùng xem thực đơn, chọn món, đặt hàng và theo dõi trạng thái đơn hàng. Giao diện được thiết kế trực quan, thân thiện, tương thích với nhiều loại thiết bị như máy tính và điện thoại.

Phân hệ quản trị: Dành cho quản lý nhà hàng để theo dõi và xử lý đơn hàng, cập nhật thực đơn, quản lý tài khoản nhân viên, và xem báo cáo doanh thu.

Mục tiêu chính của đề tài là cung cấp một giải pháp hoàn chỉnh, dễ sử dụng và có khả năng mở rộng trong tương lai. Việc sử dụng các công nghệ tiên tiến như Next.js và .NET Core không chỉ đảm bảo hiệu suất và tính bảo mật cao, mà còn giúp ứng dụng dễ dàng tích hợp với các hệ thống khác hoặc nâng cấp để đáp ứng các yêu cầu mới.

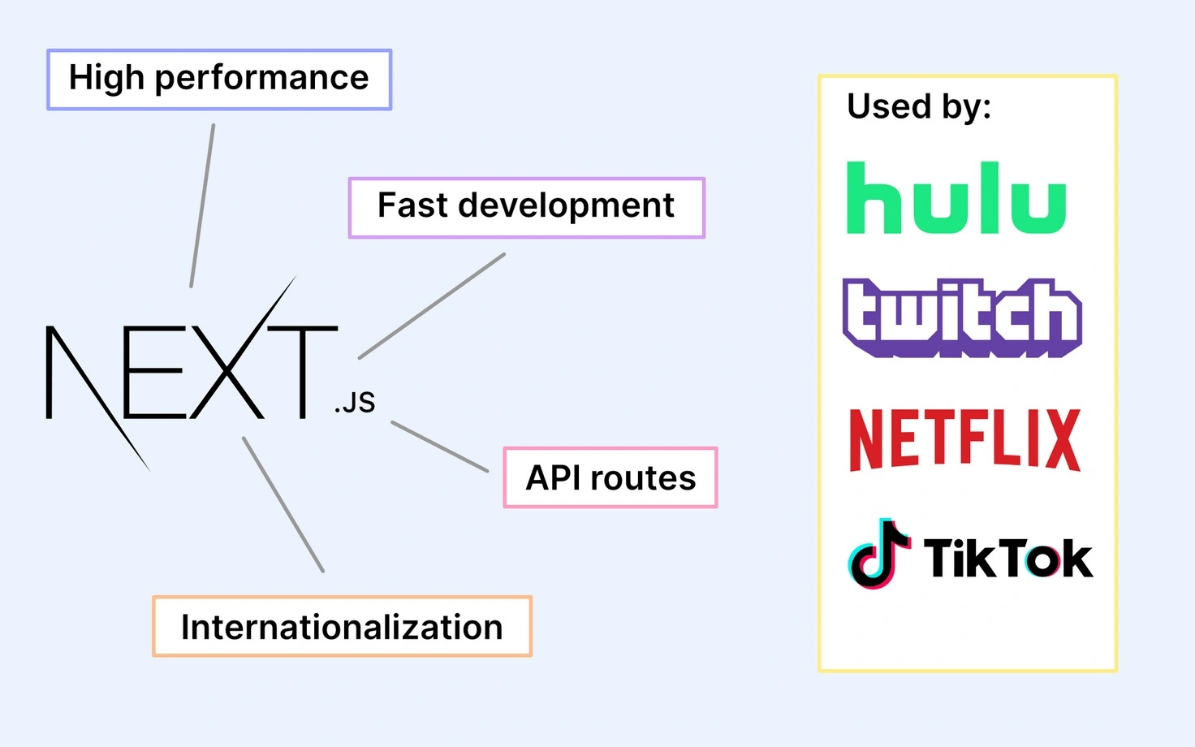
Đề tài mang tính thực tiễn cao, phù hợp với xu hướng hiện tại và mở ra cơ hội nghiên cứu sâu hơn về việc ứng dụng công nghệ trong ngành dịch vụ. Với những lợi ích thiết thực mà đề tài mang lại, nó không chỉ là một bài tập tốt nghiệp mà còn có thể triển khai thực tế, mang lại giá trị lâu dài cho cả nhà hàng và khách hàng.

## **1.2. Tổng quan NextJS**

### **1.2.1. Giới thiệu về NextJS**

NextJS là một framework có mã nguồn mở được xây dựng trên nền tảng của React, cho phép chúng ta xây dựng các trang web tĩnh có tốc độ siêu nhanh và thân thiện với người dùng, cũng như xây dựng các ứng dụng web React.

NextJS được ra đời vào năm 2016, thuộc sở hữu của Vercel. NextJS bắt đầu trở nên phổ biến vào năm 2018 và tiếp tục tăng trưởng mạnh mẽ trong cộng đồng phát triển web. Sự kết hợp của các tính năng như Server-side Rendering (SSR) với Static Site Generation (SSG) đã giúp NextJS trở thành sự lựa chọn hấp dẫn cho nhiều dự án.



Hình 1.1. Framework Next.js

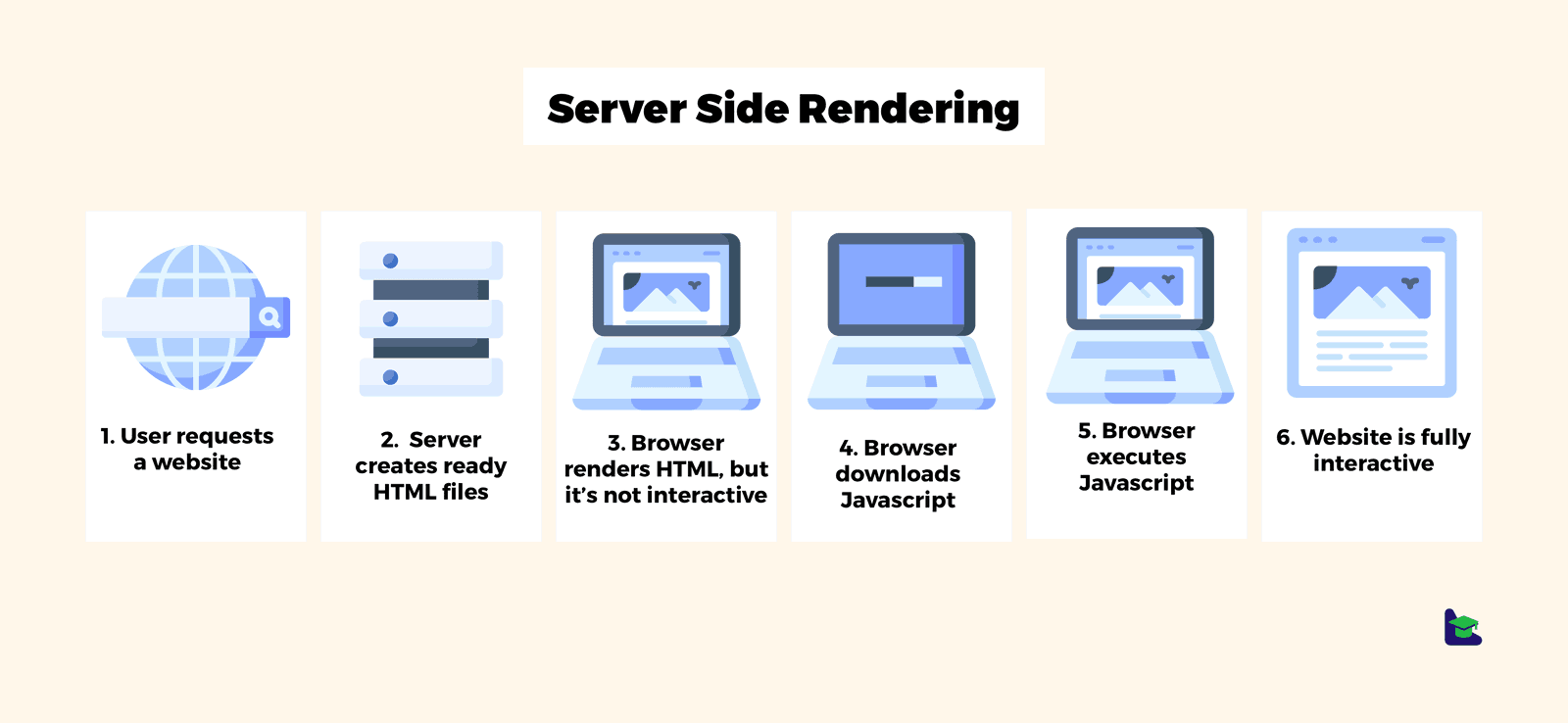
### **1.2.2. Các tính năng chính của NextJS**

1. Routing trong NextJS

* Automatic Routing: NextJS sẽ tự động tạo các router dựa trên cấu trúc thư mục của chúng ta. Ví dụ, nếu bạn tạo một file có tên là about.js ở thư mục pages. NextJS sẽ tạo router là /about.
* Nested Routing: Chúng ta có thể tạo các thư mục con để tạo các router lồng nhau. Ví dụ, nếu bạn tạo một folder có tên blog nằm trong folder pages, bên trong folder blog lại có file post.js, đường dẫn sẽ là pages/blog/post.js, thì router mà NextJS tạo ra sẽ là /blog/post.
* Dynamic Routes: Có thể tạo các router động bằng cách sử dụng cặp dấu [] trong tên file. Ví dụ nếu đường dẫn là pages/blog/[slug].js thì NextJS sẽ tạo ra các router như /blog/blog-dau-tien hoặc /blog/blog-thu-hai. Với slug là một giá trị bất kì do bạn truyền vào.
* Link Component: Để tạo liên kết giữa các trang, ta sử dụng component Link được cung cấp sẵn bởi NextJS ở thư viện next/link. Sử dụng Link thay cho thẻ a giúp tránh việc tải lại trang và tối ưu hóa hiệu suất.
* Query Parameter: Có thể truyền dữ liệu giữa các trang sử dụng query parameters trong router bằng cách sử dụng ký tự dấu chấm hỏi ? trong tên file. Ví dụ, pages/product.js có thể có các router như /product?productId=0001.

1. Rendering trong NextJS

* Server-side Rendering (SSR): Từ những năm 2000, SSR đã được sử dụng rất phổ biến, gọi nó là SSR vì hầu hết các logic phức tạp trên trang web sẽ được xử lý ở phía server.
* Khi user truy cập vào một trang sử dụng SSR của NextJS, browser sẽ gửi một request đến server
* Trước khi gửi kết quả ra trình duyệt, NextJS chạy hàm getServerSideProps() để lấy dữ liệu cần thiết cho trang. Hàm này thường sẽ dùng để gọi API hoặc truy vấn vào cơ sở dữ liệu để lấy dữ liệu cần thiết.
* Với dữ liệu cần thiết đã được lấy từ bước trước, NextJS tạo ra một phiên bản đã render đầy đủ cả nội dung và dữ liệu. Sau đó trả phiên bản này ra browser dưới dạng HTML
* Browser nhận được code HTML từ server và hiển thị lên giao diện. Nội dung của trang sẽ được hiển thị ngay lập tức, sau đó JS được tải xuống và đổ lên để trang có thể tương tác được, ví dụ như hàm onClick().



Hình 1.2. Cơ chế Server-Side Rendering

* Client-side Rendering (CSR): Sở dĩ nó được gọi là CSR vì việc render HTML sẽ được thực thi ở phía client. Hay chúng ta còn gọi là Single Page App (SPA).
* Chuyển việc xử lý dữ liệu sang cho client giúp server nhẹ việc hơn.
* Trang chỉ cần load một lần duy nhất, khi user muốn lấy dữ liệu mới từ server chỉ cần gọi đến server thông qua AJAX
* Trang web không cần load lại nhiều khi user chuyển trang, đem đến trải nghiệm tốt hơn cho người dùng



Hình 1.3. Cơ chế Client-Side Rendering

1. Styling trong NextJS

* CSS Modules: Để style cho ứng dụng NextJS, cách dễ nhất là bạn có thể tạo các file CSS/SCSS riêng lẻ cho từng component hoặc sử dụng file chung cho toàn dự án.
* CSS-in-JS Libraries: Ngoài ra bạn cũng có thể sử dụng các thư viện CSS-in-JS như Styled-Components để viết trực tiếp css vào code.
* CSS Frameworks: NextJS cũng hỗ trợ sử dụng cùng các CSS framework như TailwindCSS, Bootstrap hoặc MaterialUI.

### **1.2.3. Ưu điểm của NextJS**

NextJS hiện tại đang được sử dụng ở rất nhiều dự án khác nhau, sở dĩ NextJS được tin dùng như vậy là vì một số lý do sau:

* Sử dụng SSR và SSG: Giúp cải thiện tốc độ tải trang và khả năng SEO.
* Có nhiều tính năng giúp tối ưu hoá hiệu suất như Code Splitting, Lazy Loading, Image Optimization,...
* Fast Refresh: Tính năng giúp tự động làm mới giao diện mà không cần load lại toàn bộ trang.
* Tự động tạo file CSS dành riêng cho mỗi trang, giúp tránh xung đột trong việc sử dụng và quản lý các file CSS.
* Hỗ trợ TypeScript: NextJS cũng hỗ trợ sử dụng Typescript giúp cải thiện tính rõ ràng cho code và thuận tiện cho việc debug về sau.
* Cộng đồng lớn: NextJS có một cộng đồng sử dụng đông đảo, điều này được chứng minh ở trên chính trang Github của NextJS khi nó đang đạt khoảng hơn 100k sao. Điều này giúp cho NextJS có thêm nhiều nguồn tài liệu phong phú và các plugin hữu ích.
* Hệ sinh thái mạnh mẽ: NextJS kết hợp tốt với các thư viện và công cụ như Redux, React Query, Apollo Client và nhiều thư viện khác nằm trong hệ sinh thái của React.
* Tích hợp tốt với React: Nếu bạn đã quen với việc sử dụng React trước đó thì việc làm quen với NextJS sẽ đơn giản hơn rất nhiều.

### **1.2.4. Nhược điểm của NextJS**

Mặc dù có nhiều ưu điểm nêu trên, NextJS vẫn có những khuyết điểm nên chúng ta cần xem xét trước khi sử dụng nó:

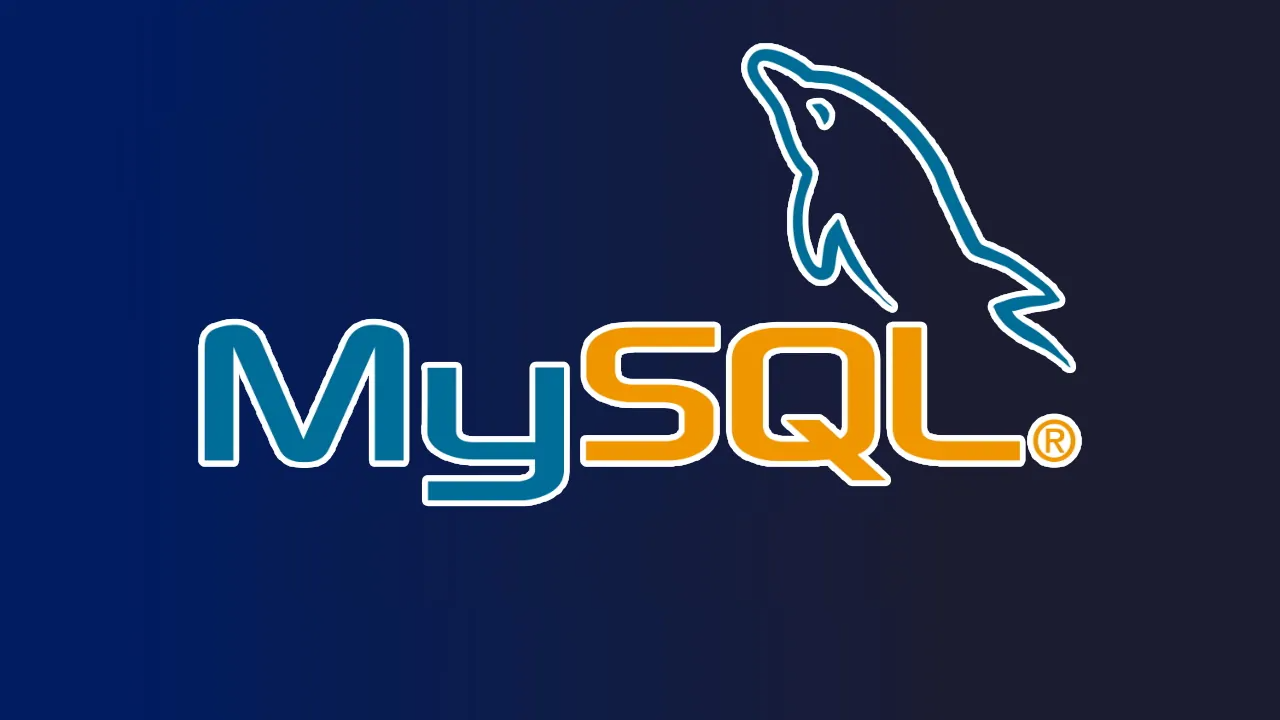
* Khó học cho người mới: Nếu chưa có hiểu biết cơ bản về Web Fundamentals, JavaScript và React thì việc học NextJS sẽ hơi khó khăn. Nhất là khi bạn gặp các khái niệm như SSR hay SSG.
* Khó khăn trong việc tích hợp với một số thư viện bên ngoài: Một số thư viện và plugin có thể cần phải điều chỉnh hoặc tùy chỉnh để hoạt động tốt với Next.js. Ví dụ như để sử dụng Redux trong ứng dụng NextJS, các bạn cần cài thêm thư viện next-redux-wrapper để quản lý state trên cả server và client.
* Phụ thuộc vào hệ sinh thái của React: Next.js phụ thuộc vào React, vì vậy nếu bạn không quen thuộc với React hoặc không muốn sử dụng React thì NextJS không phải là lựa chọn tốt.
* Đòi hỏi chạy trên server NodeJS: Để deploy ứng dụng NextJS, bạn cần có một máy chủ NodeJS, việc này có thể làm tăng chi phí và quá trình triển khai sẽ trở nên phức tạp hơn.
* Cấu trúc dự án phức tạp: Với các dự án lớn, việc quản lý cấu trúc dự án không cẩn thận lúc ban đầu sẽ dẫn đến việc khó quản lý sau này.
* Tuỳ vào tính chất của dự án và yêu cầu mà các bạn có thể cân nhắc có nên sử dụng NextJS trong dự án của mình hay không. Một số dự án thường được sử dụng NextJS để triển khai như: Trang tin tức, Blog, Landing Page,...

## **1.3. Tổng quan My SQL**

### **1.3.1. MySQL là gì?**

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System - RDBMS) mã nguồn mở, sử dụng ngôn ngữ truy vấn cấu trúc (SQL) để quản lý và thao tác dữ liệu. MySQL tổ chức và lưu trữ dữ liệu dưới dạng các bảng, trong đó các bảng có thể liên kết với nhau thông qua các khóa (keys).

MySQL được sử dụng rộng rãi trong việc phát triển các ứng dụng web. Nhiều công ty lớn như: Facebook, Twitter, và YouTube sử dụng MySQL để quản lý dữ liệu.



Hình 1.4. Logo MySQL

### **1.3.2. Lịch sử phát triển của MySQL**

MySQL được tạo ra bởi ba nhà phát triển: Michael Widenius, David Axmark và Allan Larsson vào năm 1995. Họ muốn tạo ra một hệ quản trị cơ sở dữ liệu nhanh, đáng tin cậy và dễ sử dụng cho các dự án của mình.

Năm 2008, công ty Sun Microsystems mua lại MySQL AB, công ty phát triển MySQL, với giá 1 tỷ USD. Sau đó, vào năm 2010, Oracle Corporation mua lại Sun Microsystems, và từ đó, MySQL trở thành một phần của Oracle.

### **1.3.3. Ưu điểm của MySQL**

* MySQL là phần mềm mã nguồn mở, cho phép người dùng sử dụng và tùy chỉnh mà không cần phải trả phí.
* Được thiết kế để xử lý dữ liệu nhanh chóng và hiệu quả, MySQL phù hợp với các ứng dụng web đòi hỏi tốc độ cao và khả năng phản hồi nhanh.
* Có nhiều tài liệu, hướng dẫn về cú pháp SQL phù hợp với cả người mới bắt đầu.
* Cung cấp nhiều tính năng bảo mật như xác thực người dùng, mã hóa dữ liệu và quản lý quyền truy cập chi tiết, giúp bảo vệ dữ liệu quan trọng.
* MySQL có thể xử lý từ các cơ sở dữ liệu nhỏ đến các hệ thống lớn với hàng triệu bản ghi, dễ dàng mở rộng khi nhu cầu tăng lên mà không cần thay đổi cấu trúc hệ thống.
* Hỗ trợ các tính năng transaction đảm bảo tính toàn vẹn và nhất quán của dữ liệu, giúp quản lý các thao tác dữ liệu phức tạp.

### **1.3.4. Nhược điểm của MySQL**

* So với các hệ quản trị cơ sở dữ liệu như PostgreSQL, MySQL thiếu một số tính năng nâng cao như hỗ trợ đầy đủ cho các truy vấn phức tạp, hệ thống kiểu dữ liệu phong phú hơn, và các tính năng phân tích dữ liệu tiên tiến.
* MySQL có thể gặp vấn đề về hiệu suất, thời gian phản hồi chậm khi xử lý lượng dữ liệu cực kỳ lớn hoặc các truy vấn rất phức tạp,
* Một số tính năng không thể tùy chỉnh hoặc mở rộng theo nhu cầu cụ thể của doanh nghiệp, hạn chế khả năng thích ứng với các yêu cầu đặc thù.
* Việc quản lý quyền truy cập chi tiết cho nhiều người dùng và vai trò khác nhau có thể trở nên phức tạp và khó khăn.
* Mặc dù có cộng đồng lớn, nhưng việc không có hỗ trợ chính thức miễn phí từ nhà phát triển có thể gây khó khăn khi gặp các vấn đề phức tạp hoặc lỗi hệ thống nghiêm trọng.
* MySQL có một số mở rộng và khác biệt so với chuẩn SQL, điều này có thể gây khó khăn khi chuyển đổi hoặc tích hợp với các hệ quản trị cơ sở dữ liệu khác.

## **1.4. Tổng quan .NET Core và ASP.NET Core**

### **1.4.1. .NET Core là gì?**

.NET Core là một nền tảng phát triển phần mềm đa nền tảng, mã nguồn mở do Microsoft phát triển. Đây là bước tiến vượt bậc từ .NET Framework truyền thống, mang lại sự linh hoạt và hiệu năng cao hơn, phù hợp với các xu hướng phát triển phần mềm hiện đại. .NET Core được thiết kế để hoạt động trên nhiều hệ điều hành như Windows, Linux và macOS, đáp ứng nhu cầu phát triển ứng dụng trên nhiều môi trường khác nhau và có thể được sử dụng để xây dựng các ứng dụng thiết bị, đám mây và IoT.



Hình 1.5. Logo .NET Core

1. Đặc điểm của .NET Core:

* Đa nền tảng: Chạy trên các hệ điều hành Windows, macOS và Linux.
* Nhất quán trên các kiến ​​trúc: có thể chạy mã nguồn của bạn với cùng một hành vi trên nhiều kiến ​​trúc hệ thống, bao gồm x64, x86 và ARM.
* Các công cụ dòng lệnh: Bao gồm các công cụ dòng lệnh dễ sử dụng, có thể được sử dụng để phát triển cục bộ và trong các tình huống tích hợp liên tục.
* Triển khai linh hoạt: có thể cài đặt song song (cài đặt toàn người dùng hoặc toàn hệ thống). Có thể được sử dụng với các container Docker
* Tương thích: .NET Core tương thích với .NET Framework, Xamarin và Mono, thông qua .NET Standard.
* Nguồn mở: Nền tảng .NET Core là nguồn mở, sử dụng giấy phép MIT và Apache 2. .NET Core là một dự án .NET Foundation.
* Được hỗ trợ bởi Microsoft: .NET Core được Microsoft hỗ trợ, theo Hỗ trợ .NET Core.

1. Thành phần của .NET Core

* .NET Core runtime: cung cấp một hệ thống kiểu, tải lắp ráp, trình thu gom rác, interop gốc và các dịch vụ cơ bản khác. Các thư viện khung .NET Core cung cấp các kiểu dữ liệu nguyên thủy, các kiểu thành phần ứng dụng và các tiện ích cơ bản.
* ASP.NET Core runtime: cung cấp khung để xây dựng các ứng dụng kết nối internet, điện toán đám mây hiện đại, chẳng hạn như ứng dụng web, ứng dụng IoT và phụ trợ di động.
* .NET Core SDK và trình biên dịch ngôn ngữ (Roslyn và F #) cho phép trải nghiệm nhà phát triển .NET Core.
* Dotnet command, được sử dụng để khởi chạy các ứng dụng .NET Core và các lệnh CLI. Nó chọn thời gian chạy và lưu trữ thời gian chạy, cung cấp chính sách tải lắp ráp và khởi chạy các ứng dụng và công cụ.

### **1.4.2. ASP.NET Core**



Hình 1.6. Logo ASP.NET Core

1. Khái niệm

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 đang hướng tới.

1. Đặc điểm

* Hợp nhất việc xây dựng web UI và web APIs
* Tích hợp những client-side frameworks hiện đại và những luồng phát triển
* Hệ thống cấu hình dựa trên môi trường đám mây thật sự
* Dependency injection được xây dựng sẵn
* HTTP request được tối ưu nhẹ hơn
* Có thể host trên IIS hoặc self-host trong process của riêng bạn
* Được xây dựng trên .NET Core, hỗ trợ thực sự app versioning
* Chuyển các thực thể, thành phần, module như những NuGet packages
* Những công cụ mới để đơn giản hóa quá trình phát triển web hiện đại
* Xây dựng và chạy đa nền tảng (Windows, Mac và Linux)
* Mã nguồn mở và tập trung vào cộng đồng

## **1.5. Kiến trúc Monolithic**

### **1.5.1. Kiến trúc phần mềm là gì?**

Trong ngành công nghiệp phần mềm, khi chúng ta muốn phát triển một sản phẩm phần mềm thì chúng ta cần kiến trúc tốt bằng cách sử dụng kiến trúc đó, chúng ta có thể phát triển một sản phẩm chất lượng cao, sản phẩm có khả năng kiểm tra cao, khả năng mở rộng cao và khả năng bảo trì cao.

A diagram of a structure

Description automatically generated

Hình 1.7. Kiến trúc Monolith

### **1.5.2. Định nghĩa**

“Kiến trúc phần mềm đơn giản là tổ chức của hệ thống, tổ chức này bao gồm tất cả các thành phần và cách chúng sẽ tương tác với nhau với môi trường mà chúng hoạt động và các nguyên tắc được sử dụng để xây dựng ứng dụng. Trong nhiều trường hợp, nó thường mang nghĩa sự phát triển của sản phẩm phần mềm trong tương lai”.

Kiến trúc phần mềm được thiết kế với sứ mệnh cụ thể rằng sản phẩm phải có khả năng kiểm thử cao, có khả năng mở rộng và bảo trì cao, đồng thời hành vi và cấu trúc của phần mềm phải tuân theo tiêu chuẩn và mang lại cho chúng ta kết quả tốt nhất có thể.

### **1.5.3. Kiến trúc phần mềm trong kỹ thuật phần mềm**

Kiến trúc phần mềm trong công nghệ phần mềm trình bày cấu trúc của hệ thống. Kiến trúc tập trung vào các thành phần của hệ thống và mối quan hệ của chúng và cách các thành phần này sẽ tương tác với nhau và mang lại cho chúng ta sản phẩm phần mềm với chất lượng tốt nhất

**1.5.4. Kiến trúc nguyên khối (Monolithic) là gì?**

Kiến trúc nguyên khối (monolithic) là mô hình truyền thống trong thiết kế các sản phẩm phần mềm, kiến trúc nguyên khối có nghĩa là kết hợp ứng dụng thành một đơn vị (khối) và các thành phần ứng dụng được liên kết chặt chẽ với nhau và phụ thuộc lẫn nhau (nghĩa là để ứng dụng chạy trơn tru các thành phần đó phải có mặt).

Ứng dụng nguyên khối là ứng dụng một tầng, nghĩa là nhiều thành phần của ứng dụng được kết hợp thành một ứng dụng lớn duy nhất và kiến trúc của ứng dụng được kết hợp chặt chẽ

Trong ứng dụng Nguyên khối nếu chúng ta muốn thay đổi một thành phần thì các thành phần phụ thuộc khác cũng yêu cầu viết lại các đoạn code sau khi thay đổi, ứng dụng của chúng ta cần được biên dịch lại. Hơn nữa, chúng ta chạy đi chạy lại quy trình kiểm tra và chúng ta lặp lại quy trình này sau mỗi lần thay đổi. Toàn bộ quá trình làm tăng chi phí và hạn chế sự linh hoạt và tốc độ của quá trình phát triển sản phẩm phần mềm.

### **1.5.5. Hiểu kiến trúc nguyên khối với sản phẩm phần mềm**

Giả sử chúng ta có ứng dụng cung cấp dịch vụ cho người dùng cuối và các dịch vụ này như Cyber Security, Khoa học dữ liệu, Phát triển phần mềm và Kỹ thuật DevOps, đồng thời các dịch vụ có thể được bán cho người dùng cuối theo các tiêu chí nhất định và người dùng cuối trả tiền cho dịch vụ sử dụng phương thức thanh toán trực tuyến và trong ứng dụng, chúng ta cung cấp thông tin đăng nhập của người dùng và lịch sử giao dịch, lịch sử dự án và trong dự án, chúng ta có tính năng cuộc gọi video trực tuyến sẽ được sử dụng để xử lý cuộc gọi giữa các kỹ sư và tất cả các bên liên quan của dự án

Các thành phần của ứng dụng có thể được liệt kê như sau:

* Authentication
* Services
* Video Calling
* Online Payments
* Reports
* Jira Project API Integration

Giả sử rằng chúng ta phát triển các dự án như vậy bằng cách sử dụng kiến trúc nguyên khối mang lại cho chúng ta sản phẩm được kết hợp chặt chẽ và xem xét dự án trên nếu chúng ta muốn thay đổi mô-đun Authentication thì các thành phần khác phụ thuộc vào mô-đun này sau khi thay đổi, thành phần phụ thuộc cần viết lại chức năng được sử dụng để duy trì trạng thái của người dùng và vì một thay đổi mà chúng ta cần kiểm tra lại toàn bộ ứng dụng và toàn bộ quá trình này làm tăng chi phí và thời gian phát triển và có thể làm giảm hiệu quả của ứng dụng.

### **1.5.6. Lợi ích của ứng dụng nguyên khối**

* Ứng dụng nguyên khối có thể có thông lượng tốt hơn ứng dụng mô-đun.
* Một ứng dụng nhỏ với ít mô-đun hơn sẽ dễ kiểm tra và gỡ lỗi hơn từ đó có thể giảm chi phí phát triển.
* Kiến trúc nguyên khối tốt hơn cho các ứng dụng nhẹ quy mô nhỏ.

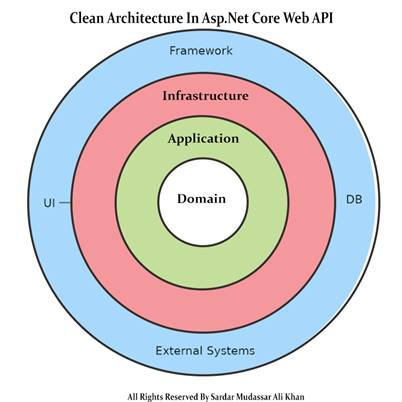
### **1.5.7. Hạn chế của ứng dụng nguyên khối**

* Kiến trúc nguyên khối không phù hợp với các ứng dụng phức tạp.
* Khi độ phức tạp của ứng dụng tăng lên thì quy mô nhóm phát triển cũng tăng theo.
* Codebase của ứng dụng thường sẽ khó hiểu và khó sửa đổi
* Đối với bất kỳ ứng dụng nguyên khối nào, nhà phát triển cần kiểm tra và triển khai toàn bộ ứng dụng sau một thay đổi nhỏ.
* Với ứng dụng nguyên khối, chi phí và thời gian phát triển cũng tăng theo những thay đổi trong cấu trúc ứng dụng.
* Các ứng dụng nguyên khối hạn chế về khả năng mở rộng và độ tin cậy
* Một module nhỏ, lớn có thể khiến cả ứng dụng bị sập.

## **1.6. Clean architecture**

### **1.6.1. Clean Architecture là gì?**

Clean Architecture bao gồm các tầng Domain Layer, Application Layer, Infrastructure Layer và Framework Layer. Trong đó, Domain Layer và Application Layer luôn là trung tâm của thiết kế và được gọi là lõi (core) của hệ thống. Phần lõi này độc lập với việc truy cập dữ liệu và các mối quan tâm về hạ tầng. Mục tiêu này có thể đạt được bằng cách sử dụng các Interface và sự trừu tượng trong lõi hệ thống, nhưng việc triển khai sẽ được thực hiện bên ngoài lõi hệ thống.



Hình 1.8. Clean Architecture

Các tầng trong Clean Architecture:

Clean Architecture bao gồm các tầng Domain Layer, Application Layer, Infrastructure Layer và Presentation Layer. Domain Layer và Application Layer luôn là trung tâm của thiết kế và được gọi là lõi của hệ thống.

Trong Clean Architecture, tất cả các phụ thuộc trong ứng dụng đều là độc lập/hướng vào trong (Independent/Inwards), và phần lõi của hệ thống không phụ thuộc vào bất kỳ tầng nào khác. Vì vậy, trong tương lai, nếu muốn thay đổi giao diện người dùng (UI) hoặc framework của hệ thống, bạn có thể thực hiện điều đó một cách dễ dàng vì các thành phần khác không phụ thuộc vào phần lõi của hệ thống.

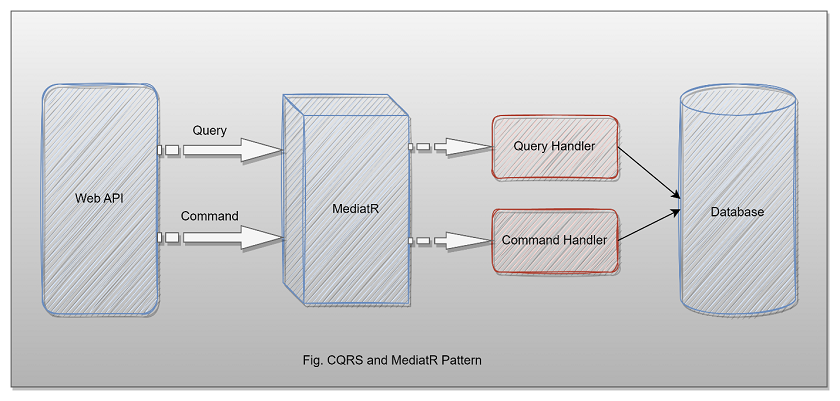
1. Domain Layer trong Clean Architecture chứa logic nghiệp vụ chính (enterprise logic), như các entity và các quy tắc của chúng. Tầng này nằm ở trung tâm của kiến trúc, nơi chứa các thực thể ứng dụng (application entities), bao gồm các lớp mô hình ứng dụng hoặc mô hình cơ sở dữ liệu. Sử dụng cách tiếp cận Code-First trong phát triển ứng dụng với ASP.NET Core, các thực thể này sẽ được dùng để tạo bảng trong cơ sở dữ liệu.
2. Application Layer chứa logic nghiệp vụ (business logic). Tất cả logic nghiệp vụ sẽ được viết trong tầng này. Ở đây, các interface của dịch vụ (service interfaces) được tách biệt khỏi phần triển khai (implementation) nhằm đảm bảo kết hợp lỏng lẻo (loose coupling) và tách biệt trách nhiệm (separation of concerns).
3. Infrastructure Layer chứa các đối tượng mô hình (model objects), nơi quản lý toàn bộ migration cơ sở dữ liệu và các đối tượng DbContext. Trong tầng này, các repository cho các đối tượng mô hình domain cũng được đặt.
4. Presentation Layer

* Đối với API, Presentation Layer là nơi cung cấp dữ liệu đối tượng từ cơ sở dữ liệu thông qua HTTP request dưới dạng JSON Object.
* Đối với các ứng dụng giao diện người dùng (UI), tầng này sẽ hiển thị dữ liệu bằng cách tiêu thụ các API.

### **1.6.2. Lợi ích của Clean Architecture**

* Triển khai dễ dàng: Bạn có thể triển khai kiến trúc này với bất kỳ ngôn ngữ lập trình nào.
* Tính độc lập: Domain Layer và Application Layer luôn là trung tâm của thiết kế và không phụ thuộc vào các hệ thống bên ngoài.
* Thay đổi dễ dàng: Cho phép thay đổi các hệ thống bên ngoài mà không ảnh hưởng đến lõi của hệ thống.
* Môi trường kiểm thử cao: Dễ dàng kiểm thử mã nguồn của bạn nhanh chóng và hiệu quả.
* Sản phẩm chất lượng cao và có khả năng mở rộng tốt: Bạn có thể tạo ra một sản phẩm có chất lượng cao và dễ dàng mở rộng.

### **1.6.3. CQRS Pattern**



Hình 1.9. CQRS Pattern

CQRS là viết tắt của Command and Query Responsibility Segregation (Phân tách trách nhiệm lệnh và truy vấn) và được sử dụng để tách biệt các thao tác đọc (queries) và ghi (commands).

* Queries thực hiện các thao tác đọc dữ liệu.
* Commands thực hiện các thao tác ghi dữ liệu như tạo (create), cập nhật (update), xóa (delete) và trả về dữ liệu.

1. CQRS Pattern

Thông thường, trong các ứng dụng, chúng ta sử dụng một mô hình dữ liệu duy nhất để đọc và ghi dữ liệu, điều này hoạt động tốt trong các ứng dụng nhỏ và giúp thực hiện các thao tác CRUD dễ dàng. Tuy nhiên, khi ứng dụng trở nên phức tạp hơn, các truy vấn có thể trả về nhiều loại dữ liệu khác nhau dưới dạng các đối tượng, điều này khiến việc quản lý trở nên khó khăn với các DTO khác nhau. Đồng thời, cùng một mô hình cũng được sử dụng cho các thao tác ghi dữ liệu, dẫn đến mô hình trở nên phức tạp.

Ngoài ra, khi sử dụng cùng một mô hình cho cả thao tác đọc và ghi, việc quản lý bảo mật cũng trở nên khó khăn khi ứng dụng lớn, vì có thể xảy ra tình trạng lộ dữ liệu không đúng ngữ cảnh do khối lượng công việc trên cùng một mô hình.

CQRS giúp tách biệt các thao tác, làm cho ứng dụng dễ mở rộng và linh hoạt hơn ở quy mô lớn.

1. Khi nào nên sử dụng CQRS

* Ứng dụng lớn và phức tạp: Chúng ta nên sử dụng CQRS khi ứng dụng lớn và cần truy cập dữ liệu song song. CQRS giúp giảm xung đột hợp nhất (merge conflict) khi thực hiện nhiều thao tác với dữ liệu.
* Mô hình dữ liệu phức tạp: Trong ngữ cảnh Domain-Driven Design (DDD), nếu mô hình dữ liệu domain phức tạp và cần thực hiện nhiều thao tác ưu tiên như xác thực (validation) và thực thi logic nghiệp vụ, thì CQRS cung cấp tính nhất quán và linh hoạt cần thiết.

# **CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ PHẦN MỀM**

## **2.1. Khảo sát các ứng dụng web đặt món ăn ngay tại nhà hàng, quán ăn trên thị trường hiện nay.**

Hiện nay, ứng dụng web đặt món ăn trực tiếp tại nhà hàng đang ngày càng phổ biến trên thị trường, đáp ứng nhu cầu hiện đại hóa và nâng cao trải nghiệm của khách hàng. Các ứng dụng này cho phép thực khách dễ dàng duyệt menu, đặt món ăn, và thanh toán trực tuyến ngay tại bàn, tiết kiệm thời gian và tăng cường sự tiện lợi. Một số ứng dụng còn tích hợp các tính năng như gọi phục vụ, đánh giá món ăn, và chương trình khuyến mãi, giúp tăng sự tương tác giữa khách hàng và nhà hàng. Trên thị trường, các nhà phát triển thường tập trung vào thiết kế giao diện thân thiện, tích hợp công nghệ QR code, và đảm bảo bảo mật thông tin thanh toán. Tuy nhiên, thách thức lớn nhất vẫn là khả năng tương thích với hệ thống hiện tại của nhà hàng và việc đào tạo nhân viên sử dụng hiệu quả. Sự phát triển của loại hình này không chỉ giúp nhà hàng tối ưu hóa quy trình phục vụ mà còn nâng cao sự hài lòng và trung thành của khách hàng.

Ứng dụng web của em mang đến giải pháp toàn diện và hiện đại cho các nhà hàng với nhiều tính năng nổi bật, giúp tối ưu hóa quy trình vận hành và nâng cao trải nghiệm khách hàng. Một trong những tính năng đặc biệt của ứng dụng là theo dõi trạng thái món ăn theo thời gian thực (realtime), cho phép khách hàng nắm bắt được tiến độ chế biến và sẵn sàng nhận món ngay khi hoàn tất. Tính năng quét QR và đổi QR linh hoạt giúp khách hàng dễ dàng truy cập menu, đặt món hoặc đổi bàn mà không cần sự can thiệp từ nhân viên, tạo nên sự tiện lợi tối đa. Bên cạnh đó, hệ thống cũng hỗ trợ theo dõi doanh thu trực quan, cung cấp cho nhà hàng các báo cáo chi tiết về hiệu quả kinh doanh theo từng thời điểm, từ đó hỗ trợ quản lý đưa ra quyết định chiến lược. Với giao diện thân thiện và công nghệ tiên tiến, ứng dụng này không chỉ cải thiện hiệu suất vận hành mà còn tăng cường sự hài lòng và trung thành của khách hàng.

## **2.2. Khảo sát hệ thống.**

### **2.2.1. Giới thiệu chung về hệ thống.**

Ứng dụng web đặt món ăn của nhà hàng Trà Đá là giải pháp hiện đại, được thiết kế nhằm nâng cao hiệu quả quản lý và mang lại trải nghiệm tiện lợi, nhanh chóng cho cả khách hàng và nhân viên. Với giao diện thân thiện và nhiều tính năng nổi bật, ứng dụng giúp nhà hàng tối ưu hóa quy trình vận hành và phục vụ.

Ứng dụng tích hợp Light/Dark Mode Toggle, cho phép người dùng tùy chỉnh giao diện theo sở thích. Khách hàng có thể đăng ký tài khoản, đăng nhập, đăng xuất, đổi mật khẩu và dễ dàng quét mã QR tại bàn để xem thực đơn, đặt món ăn mà không cần sự hỗ trợ trực tiếp từ nhân viên.

Đối với nhà hàng, ứng dụng hỗ trợ quản lý bàn ăn, cho phép thêm, sửa, xóa bàn, đổi QR Code, ẩn bàn, và chuyển đổi trạng thái sử dụng linh hoạt. Tính năng quản lý món ăn giúp cập nhật thực đơn nhanh chóng, từ thêm, sửa, xóa đến chuyển đổi trạng thái món ăn. Ngoài ra, quản lý đơn hàng hỗ trợ theo dõi và thay đổi trạng thái đơn hàng, thậm chí đặt đơn hộ khách khi cần.

Ứng dụng còn cung cấp biểu đồ phân tích doanh thu, giúp nhà hàng dễ dàng theo dõi hiệu quả kinh doanh theo thời gian thực. Quản lý tài khoản nhân viên với phân quyền chi tiết giúp đảm bảo hoạt động ổn định và bảo mật thông tin. Hơn nữa, tính năng realtime trạng thái món ăn cho phép khách hàng biết được tình trạng chuẩn bị của món, mang lại trải nghiệm dịch vụ chuyên nghiệp và minh bạch.

Ứng dụng đặt món ăn này không chỉ hỗ trợ hoạt động quản lý nhà hàng hiệu quả mà còn giúp khách hàng tận hưởng bữa ăn một cách thuận tiện, hiện đại và thoải mái hơn bao giờ hết.

### **2.2.2. Các yêu cầu chức năng**

1. Yêu cầu chức năng

* Light/Dark Mode Toggle:
* Cho phép người dùng chuyển đổi giao diện giữa chế độ sáng và tối.
* Lưu tùy chọn giao diện của người dùng để tự động áp dụng trong lần truy cập sau.
* Đăng nhập - Đăng xuất - Đổi mật khẩu
* Cung cấp chức năng đăng nhập cho nhân viên, quản trị viên và khách hàng.
* Hỗ trợ đăng xuất an toàn để bảo vệ thông tin người dùng.
* Cho phép người dùng thay đổi mật khẩu cá nhân sau khi xác thực thông tin.
* Quản lý đơn hàng
* Hiển thị danh sách các đơn hàng, bao gồm trạng thái hiện tại.
* Cho phép nhân viên thay đổi trạng thái đơn hàng (đang chuẩn bị,đã phục vụ, đã thanh toán,...).
* Hỗ trợ tính năng đặt đơn hàng hộ khách, bao gồm nhập thông tin và xác nhận đơn hàng.
* Quản lý bàn ăn
* Cung cấp các chức năng thêm, sửa, xóa thông tin bàn ăn.
* Cho phép đổi mã QR của bàn ăn và ẩn bàn khi không sử dụng.
* Hỗ trợ thay đổi trạng thái bàn (có sẵn, ẩn).
* Quản lý món ăn
* Hỗ trợ thêm mới món ăn với thông tin chi tiết như tên, giá, mô tả và hình ảnh.
* Cung cấp các chức năng sửa, xóa món ăn.
* Cho phép thay đổi trạng thái món ăn (hiển thị/ẩn trên thực đơn).
* Biểu đồ phân tích doanh thu
* Hiển thị báo cáo trực quan về doanh thu theo thời gian (ngày, tuần, tháng).
* Phân tích số lượng đơn hàng và món ăn được bán ra.
* Quản lý tài khoản nhân viên
* Hỗ trợ thêm, sửa, xóa tài khoản nhân viên.
* Cung cấp tính năng phân quyền để kiểm soát các chức năng mà nhân viên được phép truy cập.
* Realtime trạng thái món ăn
* Cập nhật trạng thái món ăn theo thời gian thực.
* Cho phép nhân viên theo dõi trạng thái để phối hợp tốt hơn trong quy trình phục vụ.
* Khách hàng: Đăng ký tài khoản, quét mã QR đặt món ăn
* Cho phép khách hàng tạo tài khoản để lưu thông tin đặt hàng.
* Hỗ trợ quét mã QR tại bàn để xem thực đơn và đặt món ăn trực tiếp.
* Upload hình ảnh liên quan
* Cho phép nhân viên và quản trị viên tải lên hình ảnh minh họa món ăn, bàn ăn hoặc các tài liệu liên quan.
* Hỗ trợ lưu trữ và quản lý hình ảnh thông qua API Cloudinary.

1. Yêu cầu phi chức năng

* Ngôn ngữ trên hệ thống: Sử dụng ngôn ngữ tiếng Việt.
* Môi trường cài đặt: Hệ thống được thiết kế để tương thích với các trình duyệt web phổ biến và hệ điều hành khác nhau, đảm bảo rằng người dùng có thể truy cập và sử dụng hệ thống một cách dễ dàng : Windows, MacOS,Linux và tương thích với các thiết bị di động thông minh như iphone, ipad và các loại smartphone.
* Độ tin cậy và bảo mật: Hệ thống đảm bảo rằng thông tin của khách hàng được bảo mật và không bị rò rỉ hoặc bị truy cập trái phép.
* Tốc độ và hiệu suất: Hệ thống được thiết kế để hoạt động nhanh chóng và đáp ứng nhu cầu của người dùng trong thời gian ngắn nhất có thể .
* Thiết kế đẹp và dễ sử dụng: Giao diện website phải đẹp, gần gũi và dễ sử dụng, đảm bảo rằng người dùng có thể tìm thấy thông tin và sản phẩm một cách dễ dàng và thuận tiện.
* Độ khả dụng: Hệ thống nên được thiết kế để có thể hoạt động 24/7 mà không gặp sự cố, đảm bảo rằng người dùng có thể truy cập và sử dụng hệ thống bất cứ khi nào họ muốn.
* Khả năng hồi phục: Khả năng hồi phục từ đề cập đến khả năng vượt qua những thời điểm quan trọng và thích nghi sau khi trải qua một số tình huống bất ngờ. Nó cũng chỉ ra việc trở lại bình thường.

## **2.3. Biểu đồ use case.**

### **2.3.1. Các use case.**



Hình 2.1. Biểu đồ use case phía frontend.

**

Hình 2.2. Biểu đồ use case phía backend.

### **2.3.2. Phân rã use case.**



Hình 2.3. Biểu đồ phân rã use case phía frontend.

**

Hình 2.4. Biểu đồ phân rã use case phía backend.

### **2.3.3. Mô tả chi tiết một số use case.**

1. Mô tả use case Xem món ăn

Use case này cho phép Khách hàng xem món ăn.

* **Luồng sự kiện:**
  + **Luồng cơ bản:**

1. Use case này bắt đầu khi Khách hàng kích vào nút “Menu” trên giao diện. Hệ thống sẽ lấy thông tin từ bảng Dishes và hiển thị thông tin các món ăn lên màn hình. Use case kết thúc.
   * **Luồng rẽ nhánh:**
2. Tại bước 1 luồng cơ bản nếu bảng Dishes không có thông tin thì không hiển thị món ăn nào.
3. Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

* **Các yêu cầu đặc biệt:** Không có.
* **Tiền điều kiện:** Không có.
* **Hậu điều kiện:** Không có.
* **Điểm mở rộng:** Không có.

1. Mô tả use case Bảo trì món ăn.

Use case này cho phép Admin có thể thêm, sửa, xóa món ăn trên trang web

* **Luồng sự kiện:**
  + **Luồng cơ bản:**

1. Use case này bắt đầu khi Người quản trị kích vào nút “Món ăn” trên menu quản trị. Hệ thống lấy thông tin chi tiết của các Món ăn từ bảng Dishes trong cơ sở dữ liệu và hiển thị lên màn hình.
2. Thêm món ăn mới:
3. Người quản trị sẽ kích vào nút “Thêm món ăn” trên cửa sổ danh sách món ăn. Hệ thống sẽ hiện 1 mẫu yêu cầu nhập thông tin chi tiết món ăn muốn thêm.
4. Người quản trị nhập thông tin sản phẩm sau đó nhấn vào nút “Thêm món ăn”. Hệ thống sẽ hiện ra thông báo xác nhận. Người quản trị nhấn “Lưu”, hệ thống sẽ tạo một bản ghi mới trong bảng Dishes và hiển thị danh sách món ăn được cập nhật.
5. Sửa thông tin món ăn:
6. Người quản trị kích vào nút “Sửa” trên một dòng món ăn. Hệ thống hiển thị một màn hình với các thông tin cũ của món ăn.
7. Người quản trị sửa các thông tin của món ăn và kích vào nút “Lưu”.
8. Xóa thông tin món ăn:
9. Người quản trị kích vào nút “Xóa” trên một dòng món ăn. Hệ thống sẽ hiển thị một màn hình yêu cầu xác nhận xóa.
10. Người quản trị kích vào nút “Lưu”. Hệ thống sẽ xóa sản phẩm được chọn khỏi bảng Dishes và hiển thị danh sách các món ăn đã cập nhật.

5) Khi người quản trị chọn chức năng khác Use case kết thúc.

* + **Luồng rẽ nhánh:**

1. Tại bước 2b hoặc 3b trong luồng cơ bản nếu người quản trị nhập thông tin món ăn không hợp lệ thì hệ thống sẽ hiển thị thông báo lỗi yêu cầu nhập lại. Người quản trị có thể nhập lại để tiếp tục hoặc kích vào nút “Hủy bỏ” để kết thúc.
2. Tại bước 2b hoặc 3b trong luồng cơ bản nếu người quản trị kích vào nút “Hủy bỏ” hệ thống sẽ bỏ qua thao tác thêm mới hoặc sửa chữa tương ứng và hiển thị danh sách các món ăn trong bảng Dishes
3. Tại bước 4b trong luồng cơ bản nếu người quản trị kích vào nút “Hủy bỏ” hệ thống sẽ bỏ qua thao tác xóa và hiển thị danh sách các món ăn trong bảng Dishes
4. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

* **Các yêu cầu đặc biệt:**

Use case này chỉ cho phép admin và nhân viên thực hiện

* **Tiền điều kiện:**

Người quản trị cần đăng nhập trước khi thực hiện use case

* **Hậu điều kiện:**

Nếu use case kết thúc thành công thì thông tin về môn học sẽ được cập nhật trong cơ sở dữ liệu.

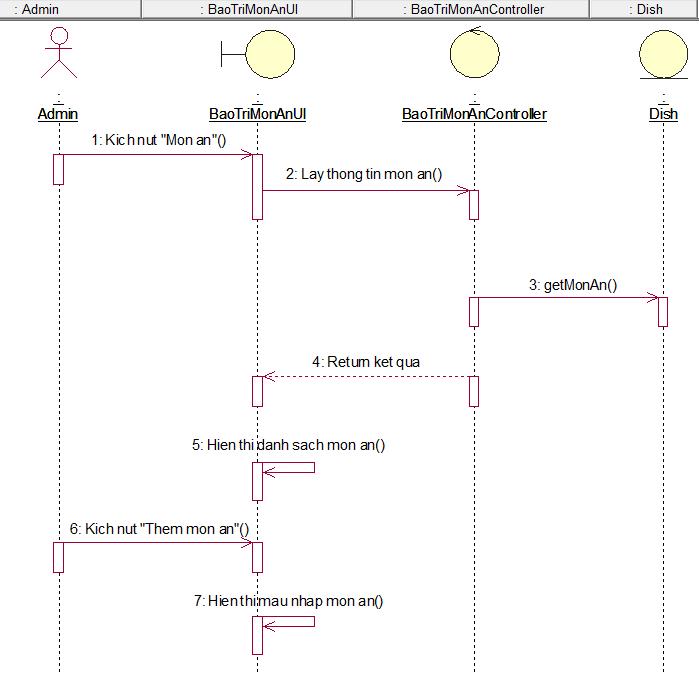
* **Điểm mở rộng:**

Không có.

### **3.3.4. Biểu đồ trình tự một số use case.**



Hình 2.6. Biểu đồ trình tự Use case Xem món ăn.



A diagram of a project

Description automatically generated

A screenshot of a computer

Description automatically generated

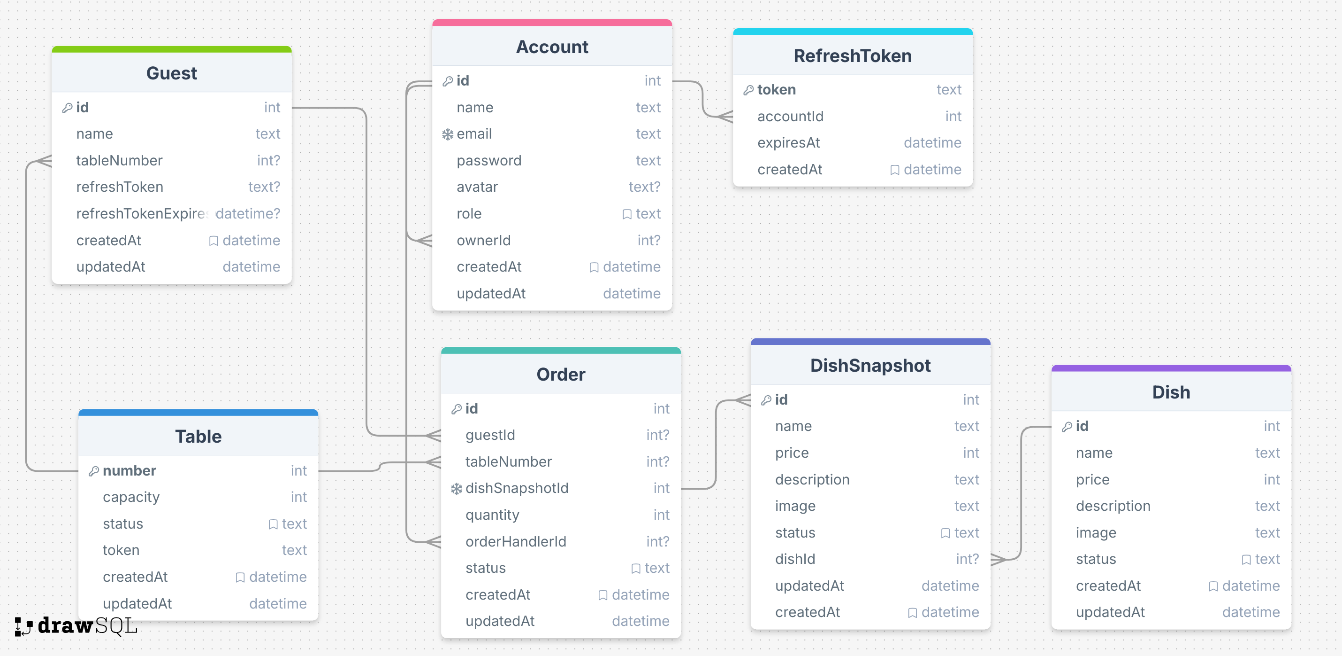
A diagram of a diagram

Description automatically generated with medium confidence

Hình 2.7. Biểu đồ trình tự Use case Bảo trì món ăn.

## **2.4. Thiết kế cơ sở dữ liệu**

### **2.4.1. Biểu đồ Entity Relationship Diagram**



Hình 2.8. Biểu đồ Entity Relationship Diagram

### **2.4.2. Thiết kế các bảng trong cơ sở dữ liệu**

A screenshot of a computer

Description automatically generated

Hình 2.9. Bảng Tables

A screenshot of a computer

Description automatically generated

Hình 2.10. Bảng Accounts

A screenshot of a computer

Description automatically generated

Hình 2.11. Bảng Dishes

A screenshot of a computer

Description automatically generated

Hình 2.12. Bảng DishSnapshots

A screenshot of a computer

Description automatically generated

Hình 2.13. Bảng Guests

A screenshot of a computer

Description automatically generated

Hình 2.14. Bảng Orders

A screenshot of a computer

Description automatically generated

Hình 2.15. Bảng RefreshTokens

## **2.5. Thiết kế giao diện**

Giao diện mockup Đăng nhập:

**A screen shot of a login form

Description automatically generated**

Hình 2.16. Giao diện mockup Đăng nhập

Giao diện mockup Trang chủ*:A screenshot of a computer

Description automatically generated*

Hình 2.17. Giao diện mockup Trang chủ

Giao diện mockup Quản lý đơn hàng:

*A screenshot of a computer

Description automatically generated*

Hình 2.18. Giao diện mockup Quản lý đơn hàng

*A white rectangular object with black text

Description automatically generated*

Hình 2.19. Giao diện mockup Quản lý bàn ăn

Giao diện mockup Thống kê:

*A graph showing a graph and a diagram

Description automatically generated with medium confidence*

Hình 2.20. Giao diện mockup Thống kê

Giao diện mockup Menu:

*A screenshot of a menu

Description automatically generated*

Hình 2.21. Giao diện mockup Menu

# **CHƯƠNG 3: KẾT QUẢ XÂY DỰNG**

## **3.1. Cài đặt**

### **3.1.1. Frontend**

* Được xây dựng bằng **Next.js**, một framework React.js mạnh mẻ cho phát triển giao diện người dùng:
  + **SSR (Server-Side Rendering)**: Tối ưu hiển thị giao diện nhằm tăng tốc độ tải trang.
  + **Static Site Generation (SSG)**: Dùng cho những trang tính như thực đơn nhà hàng.
  + **Client-Side Rendering (CSR)**: Hỗ trợ tương tác nhanh khi thao tác người dùng.
* Frontend giao tiếp với Backend qua các API REST.
* Thiết kế giao diện thân thiện, hỗ trợ responsive trên nhiều thiết bị (PC, tablet, mobile).

### **3.1.2. Backend (Xử lý nghiệp vụ)**

Backend được phát triển bằng **.NET Core** nhờ vào sự mạnh mẻ về tính module hóa, độ bền và hiệu năng cao:

* **Xử lý logic nghiệp vụ**: Quản lý danh mục món ăn, đặt món, thanh toán.
* **API RESTful**: Cung cấp các endpoint để Frontend gửi request.
* **Bảo mật**: Triển khai Bearer token cho xác thực người dùng.
* **Middleware**: Xử lý request/response.

### **3.1.3. Cơ sở dữ liệu**

Sử dụng My SQL làm cơ sở dữ liệu chính:

* Quản lý dữ liệu về thực đơn, đơn hàng, khách hàng.
* Tính quan hệ logic giữa các bảng như: Khách hàng, Đơn hàng, Món ăn, ...
* Tối ưu hiệu năng truy vấn bằng các công nghệ như caching và indexing.

### **3.1.4. Giao tiếp giữa các thành phần**

* **Frontend - Backend**:
  + Sử dụng các API RESTful để truyền request.
  + Dữ liệu request/response được định dạng theo JSON.
* **Backend - Database**:
  + Backend tương tác với My SQL qua Entity Framework Core, cho phép xử lý ORM một cách dễ dàng.

### **3.1.5. Triển khai Clean Architecture**

├── API

│ └── Controllers

│ └── \*.cs

├── appsettings. json

├── Application

│ ├── Common

│ │ ├── Interfaces

│ │ └── Mappings

│ ├── Exceptions

│ └── Features

│ ├── Account

│ │ ├── Commands

│ │ ├── Queries

│ │ ├── Auth

│ ├── Dish

│ └── Order

├── Core

│ ├── Const

│ ├── Dtos

│ ├── Entities

│ └── Enums

├── Infrastructure

│ ├── Configurations

│ ├── Converters

│ ├── Data

│ ├── Migrations

│ └── Services

**Chi tiết các thư mục**

* **API**: Chứa các controller xử lý yêu cầu HTTP từ người dùng.
  + Controllers/: Các lớp API Controller (ví dụ: OrderController, DishController).
* **appsettings.json**: File cấu hình cho ứng dụng, chứa thông tin như connection string, key bảo mật.
* **Application**: Tầng logic nghiệp vụ chính của ứng dụng.
  + Common/Interfaces: Các interface để định nghĩa hợp đồng giữa các tầng.
  + Common/Mappings: Các ánh xạ DTO hoặc Entity.
  + Exceptions/: Các lớp xử lý ngoại lệ.
  + Features/: Tách biệt theo tính năng (Account, Dish, Order) với các lớp con như Commands, Queries, Auth.
* **Core**: Tầng chứa các định nghĩa cốt lõi không phụ thuộc vào framework.
  + Const/: Các hằng số sử dụng chung trong ứng dụng.
  + Dtos/: Các đối tượng truyền dữ liệu.
  + Entities/: Các thực thể cốt lõi như Dish, Order.
  + Enums/: Các enum định nghĩa trạng thái hoặc kiểu dữ liệu.
* **Infrastructure**: Tầng triển khai các dịch vụ phụ thuộc framework.
  + Configurations/: Cấu hình cho database, email, hoặc logging.
  + Converters/: Các bộ chuyển đổi dữ liệu hoặc định dạng.
  + Data/: Các lớp truy cập dữ liệu, repository.
  + Migrations/: Các file migration cho cơ sở dữ liệu.
  + Services/: Các dịch vụ như gửi email, …

## **3.2. Giao diện**

* + 1. Giao diện phía quản trị

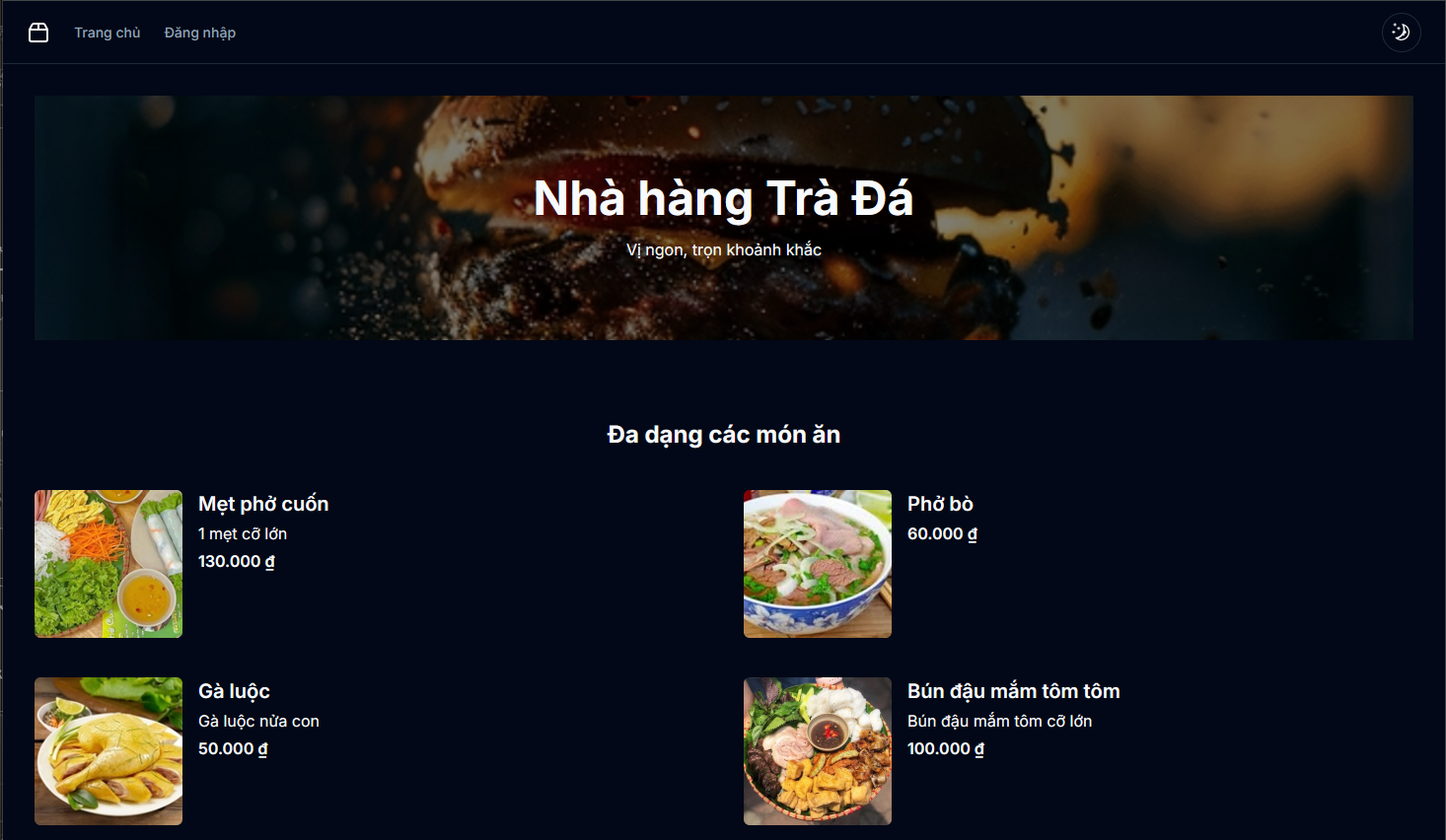
Giao diện đăng nhập:

A screenshot of a login form

Description automatically generated

Hình 3.1. Giao diện đăng nhập

Màn hình trang chủ:



Hình 3.2. Màn hình trang chủ

Màn hình quản lý đơn hàng:

A screenshot of a computer

Description automatically generated

Hình 3.3. Màn hình quản lý đơn hàng

Màn hình chi tiết quản lý đơn hàng tại một bàn:

A screenshot of a computer

Description automatically generated

Hình 3.4. Màn hình chi tiết quản lý đơn hàng tại một bàn

Màn hình chi tiết cập nhật đơn hàng

*A screenshot of a chat

Description automatically generated*

Hình 3.5. Màn hình chi tiết cập nhật đơn hàng

Màn hình quản lý bàn ăn:

*A qr code on a white background

Description automatically generated*

Hình 3.6. Màn hình quản lý bàn ăn

Màn hình quản lý món ăn:

*A screenshot of a computer

Description automatically generated*

Hình 3.7. Màn hình quản lý món ăn

Màn hình thống kê doanh thu:

*A screenshot of a graph

Description automatically generated*

Hình 3.8. Màn hình thống kê doanh thu

Màn hình quản lý nhân viên:

*A screenshot of a computer

Description automatically generated*

Hình 3.9. Màn hình quản lý nhân viên

* + 1. Giao diện phía khách hàng

Màn hình chi tiết menu món ăn:

A screenshot of a computer

Description automatically generated

Hình 3.10. Màn hình chi tiết menu món ăn

Màn hình đơn đặt hàng:

*A screenshot of a computer

Description automatically generated*

Hình 3.11. Màn hình đơn đặt hàng

# **CHƯƠNG 4: KIỂM THỬ PHẦN MỀM**

## **4.1. Kế hoạch kiểm thử**

### **4.1.1. Giới thiệu**

1. Mục đích

Tài liệu kế hoạch kiểm thử này đưa ra các mục đích sau:

- Xác định thông tin cơ bản về dự án và các thành phần chức năng được kiểm thử và không được kiểm thử.

- Liệt kê những yêu cầu cho việc kiểm thử (Test Requirements).

- Những chiến lược kiểm thử nên được sử dụng.

- Ước lượng những yêu cầu về tài nguyên và chi phí cho việc kiểm thử.

- Những tài liệu được lập sau khi hoàn thành việc kiểm thử.

1. Các định nghĩa và các từ viết tắt.

Các định nghĩa:

|  |  |
| --- | --- |
| **Thuật ngữ** | **Định nghĩa** |
| http://localhost:3000/ | Địa chỉ của trang web |
| URL | Cách gọi chung địa chỉ trang Web |
| Test Designer/ Tester | Người thiết kế kiểm thử/ Kiểm thử viên |

Bảng 4.1. Bảng định nghĩa và các từ viết tắt

Các từ viết tắt:

|  |  |
| --- | --- |
| **Từ viết tắt** | **Mô tả** |
| UC | Use case |
| TC | Test case |
| GUI | Giao diện người dùng |
| ST | System test |
| UAT | User acceptance test |
| TR | Test report |

1. Lịch trình công việc

|  |  |  |  |
| --- | --- | --- | --- |
| **Mốc công việc** | **Sản phẩm** | **Bắt đầu** | **Kết thúc** |
| Lập kế hoạch kiểm thử | Test plan | 12/12/2024 | 12/12/2024 |
| Xem lại các tài liệu | Test plan | 12/12/2024 | 12/12/2024 |
| Thiết kế các test case | Test case | 12/12/2024 | 12/12/2024 |
| Viết các test case | Test case | 12/12/2024 | 12/12/2024 |
| Xem lại các testcase | Test case | 12/12/2024 | 12/12/2024 |
| Thực thi các testcase | Test case | 12/12/2024 | 12/12/2024 |
| Ghi nhận và đánh giá kết quả kiểm thử | Test report | 13/12/2024 | 14/12/2024 |

Bảng 4.2. Bảng lịch trình công việc

1. Phạm vi kiểm thử

Các chức năng được kiểm thử:

* Đăng nhập: Kiểm tra chức năng đăng nhập vào hệ thống với tài khoản vừa tạo.
* Đổi mật khẩu: Kiểm tra chức năng thay đổi mật khẩu của tài khoản đã có.
* Thông báo realtime cho người dùng về trạng thái đơn đặt hàng.

1. Tài nguyên kiểm thử
2. Phần cứng

Máy tính cá nhân có kết nối mạng Internet.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CPU** | **RAM** | **SSD** |  | **Architecture** |
| Intel Core i5, 2.3 GHz | 16 GB | 512 GB |  | 64 bits |

Bảng 4.3. Bảng cấu hình phần cứng

1. Phần mềm.

|  |  |  |
| --- | --- | --- |
| **Tên phần mềm** | **Phiên bản** | **Loại** |
| Google Chrome | 19.0.1084.56 | Trình duyệt web |
| Microsoft Windows 11 | 10 | Hệ điều hành |

Bảng 4.4. Bảng thông tin phần mềm

1. Công cụ kiểm thử.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hoạt động** | **Công cụ** | **Nhà cung cấp** | **Phiên bản** |
| Quản lý Test Case | Microsoft Office Excel | Microsoft | 2021 |
| Quản lý phiên bản | Microsoft Office Word | Microsoft | 2021 |
| Theo dõi lỗi | Microsoft Office Excel | Microsoft | 2021 |

Bảng 4.5. Bảng công cụ kiểm thử

1. Nhân sự

|  |  |
| --- | --- |
| **Thành viên** | **Vai trò** |
| Lê Văn An | Test Manager / Test Designer / Tester |

Bảng 4.6. Bảng nhân sự

1. Chiến lược kiểm thử
2. Các giai đoạn kiểm thử.

Kiểm thử ở mức hệ thống (ST) và kiểm thử chấp nhận (UAT):

- Dùng kiểu kiểm thử tự động để kiểm thử từng chức năng.

- Việc kiểm thử chỉ bắt đầu khi đã hoàn thiện bộ test case để kiểm thử chức năng.

- Thiết kế test case theo phương pháp phân vùng tương đương.

- Chỉ thực hiện kiểm thử hồi quy, không thực hiện kiểm thử lại.

1. Loại kiểm thử

Kiểm thử hộp đen

|  |  |
| --- | --- |
| Mục đích kiểm tra | Đảm bảo rằng tất cả các chức năng được kiểm tra hoạt động đúng theo đặc tả yêu cầu, không phụ thuộc vào cách thức mà phần mềm được triển khai. |
| Kỹ thuật | Thực thi tất cả các trường hợp có thể có cho mỗi nhóm chức năng, sử dụng dữ liệu hợp lệ và không hợp lệ để xác định:  - Kết quả mong đợi khi dữ liệu hợp lệ được sử dụng.  - Cảnh báo phù hợp hiện ra khi dữ liệu không hợp lệ được sử dụng. |
| Tiêu chuẩn dừng | - Tất cả các test case đã được thiết kế đều được thực thi.  - Tất cả các lỗi tìm thấy đều được ghi nhận lý do rõ ràng để có thể giúp cho lập trình viên khắc phục. |
| Chịu trách nhiệm kiểm thử | Test Designer / Tester |
| Cách kiểm thử | Kiểm thử tự động bằng công cụ Selenium, tuần tự theo các bước được định nghĩa trong test case |
| Xử lý ngoại lệ | Liệt kê tất cả các vấn đề liên quan phát sinh trong quá trình thực thi kiểm thử. |

Bảng 4.7. Bảng loại kiểm thử hộp đen

1. Điều kiện chấp nhận kiểm thử.

* Tỉ lệ test case đạt(passed): 100%
* Tỉ lệ test case không đạt (failed): 0%
* Hệ thống vẫn chạy ổn định trên các trình duyệt web khác nhau (Google Chrome, Microsoft Edge)

1. Phân loại lỗi

|  |  |
| --- | --- |
| Mức độ nghiêm trọng | Đặc tả lỗi |
| High | - Không thể đăng nhập  - Không đặt khám được kể cả bác sĩ và phòng khám.  - Không đổi mật khẩu được. |
| Medium | - Đăng nhập sai không báo lỗi, không trả về kết quả  - Thay đổi thông tin hoặc thêm hồ sơ bệnh nhân không được. |
| Low | - Tìm kiếm trung tâm tiêm chủng sai.  - Tìm kiếm bài viết sai. |

Bảng 4.8. Bảng loại phân loại lỗi

### **4.1.2. Thực hiện kiểm thử**

1. Kiểm thử chức năng đăng nhập
2. Giao diện chức năng:

A screenshot of a login form

Description automatically generated

Hình 4.1. Giao diện Đăng nhập*.*

1. Mô tả giao diện và các điều kiện ràng buộc:

* Tiến hành nhập email và mật khẩu rồi nhấn nút “Đăng nhập”.
* Điều kiện ràng buộc: Không được để trống email và mật khẩu.

Sử dụng phương pháp phân hoạch tương đương thiết lập các ca kiểm thử:

|  |  |  |
| --- | --- | --- |
| Điều kiện đầu vào | Vùng hợp lệ | Vùng không hợp lệ |
| Email | Email hợp lệ VD: email@domain | Email không hợp lệ |
|  |
| Mật khẩu | Bất kì ký tự nào | Không có ký tự nào |  |
| Nhập không đúng mật khẩu hiện tại |  |

Bảng 4.9. Bảng điều kiện ràng buộc cho Đăng nhập

1. Danh sách các Test case:

A screenshot of a computer

Description automatically generated

Hình 4.2. Danh sách Test case Đăng nhập

1. Thực hiện kiểm thử tự động bằng Selenium.

Công việc kiểm thử tự động được thực hiện bằng cách sử dụng Selenium IDE. Ở đây ta có kịch bản kiểm thử với các test case được test tự động.

A screenshot of a computer

Description automatically generated

Hình 4.3. Kết quả kiểm thử Đăng nhập bằng Selenium

1. Kiểm thử chức năng đổi mật khẩu
2. Giao diện chức năng:

A screenshot of a chat

Description automatically generatedHình 4.4. Giao diện Đổi mật khẩu.

1. Mô tả giao diện và các điều kiện ràng buộc:

* Tiến hành nhập mật khẩu cũ và mật khẩu mới rồi nhấn nút “Lưu thông tin”.
* Điều kiện ràng buộc: Không được để trống 3 trường nêu trên.

Sử dụng phương pháp phân hoạch tương đương thiết lập các ca kiểm thử:

|  |  |  |
| --- | --- | --- |
| Điều kiện đầu vào | Vùng hợp lệ | Vùng không hợp lệ |
| Mật khẩu cũ | Đủ 6 ký tự | Không đủ 6 ký tự |
|
| Mật khẩu mới | Đủ 6 ký tự | Không đủ 6 ký tự |
| Không khớp trường nhập lại mật khẩu |
| Nhập lại mật khẩu mới | Đủ 6 ký tự | Không đủ 6 ký tự |
| Không khớp trường mật khẩu |

Bảng 4.10. Bảng điều kiện ràng buộc cho Đổi mật khẩu

1. Danh sách các Test case:

A screenshot of a computer

Description automatically generated

Hình 4.5. Danh sách Test case Đổi mật khẩu

1. Thực hiện kiểm thử tự động bằng Selenium:

Công việc kiểm thử tự động được thực hiện bằng cách sử dụng Selenium IDE. Ở đây ta có kịch bản kiểm thử với các test case được test tự động.

A screenshot of a computer

Description automatically generated

Hình 4.6. Kết quả kiểm thử Đổi mật khẩu bằng Selenium

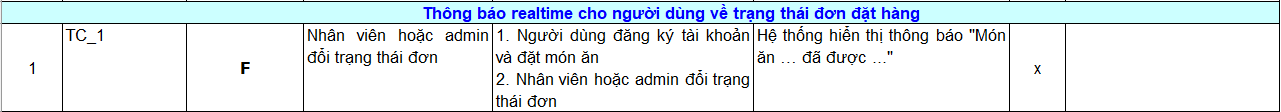
1. Kiểm thử thông báo realtime cho người dùng về trạng thái đơn đặt hàng.
2. Giao diện chức năng

A screenshot of a menu

Description automatically generated

Hình 4.7. Giao diện. thử thông báo realtime cho người dùng về trạng thái đơn đặt hàng

1. Danh sách các Test case:



Hình 4.8. Danh sách Test case Kiểm thử thông báo realtime cho người dùng về trạng thái đơn đặt hàng

1. Thực hiện kiểm thử tự động bằng Selenium:

Công việc kiểm thử tự động được thực hiện bằng cách sử dụng Selenium IDE. Ở đây ta có kịch bản kiểm thử với các test case được test tự động.

A screenshot of a computer

Description automatically generated

Hình 4.9. Kết quả kiểm thử Thông báo realtime bằng Selenium

## **4.2. Kết quả kiểm thử**

* Không phát hiện lỗi trên website ở các tính năng có khả năng bị ảnh hưởng sau khi thêm các tính năng mới, tỷ lệ pass 100%.
* Quá trình kiểm thử hồi quy diễn ra suôn sẻ và không gặp khó khăn đáng kể.
* Các test case đều chạy ổn định trên công cụ Selenium.

A screenshot of a test report

Description automatically generated

Hình 4.10. Test report

## **4.3. Kết luận**

Dựa trên kết quả kiểm thử của các chức năng cần thiết và các chức năng mới trên trang web, ta có một vài lời kết luận:

* Đánh giá về tính ổn định: Trang web có cho thấy tính ổn định tốt trong quá trình kiểm thử hồi quy. Không có các lỗi nghiêm trọng được phát hiện, như các lỗi giao diện người dùng (UI) cơ bản hay lỗi hoạt động của các tính năng chính.
* Độ phủ kiểm thử: Tỷ lệ phủ của các ca kiểm thử đối với các chức năng cần thiết và các chức năng mới là đủ để đảm bảo chất lượng. Các kịch bản kiểm thử đã bao quát được các trường hợp thông thường cũng như trường hợp biên của từng tính năng.
* Hiệu suất và thời gian phản hồi: Trang web đáp ứng nhanh và không có vấn đề đáng kể về hiệu suất trong quá trình kiểm thử. Thời gian phản hồi của các tương tác người dùng là hợp lý và không gây ngạt ngào.
* Cải tiến tiềm năng: Dựa trên những phản hồi từ kiểm thử, có một số đề xuất về cải tiến có thể áp dụng để cải thiện trải nghiệm người dùng hoặc tối ưu hóa hiệu suất. Tuy nhiên, không có vấn đề cấp bách cần phải giải quyết ngay lập tức.

# **KẾT LUẬN**

## **Kết quả đạt được.**

Dự án Phát triển ứng dụng web đặt đồ ăn cho nhà hàng sử dụng Next.js và .NET Core đã được hoàn thành với các chức năng chính: người dùng có thể duyệt thực đơn, đặt món, và nhận thông báo trạng thái đơn hàng,... Ứng dụng được xây dựng dựa trên Clean Architecture, đảm bảo cấu trúc phần mềm rõ ràng, dễ bảo trì và mở rộng. Công nghệ Next.js mang lại hiệu suất cao, SEO tối ưu, trong khi .NET Core đảm bảo tính bảo mật và ổn định cho API. Hệ thống cũng hỗ trợ quản lý nhà hàng như theo dõi đơn hàng, quản lý thực đơn và khách hàng. Quá trình phát triển đã tuân thủ các nguyên tắc kiểm thử và đảm bảo chất lượng, đáp ứng nhu cầu thực tế.

## **Hướng phát triển.**

Trong tương lai, ứng dụng có thể được mở rộng với các tính năng nâng cao như gợi ý món ăn dựa trên sở thích, tích hợp AI để tối ưu hóa quản lý nhà hàng, và triển khai ứng dụng di động nhằm nâng cao trải nghiệm người dùng. Đồng thời, hệ thống có thể mở rộng để hỗ trợ đa nhà hàng trên cùng nền tảng, tích hợp ví điện tử hoặc cổng thanh toán quốc tế, và tối ưu hóa hiệu suất với công nghệ serverless nhằm đáp ứng số lượng lớn người dùng. Việc tăng cường bảo mật và tuân thủ các tiêu chuẩn an toàn dữ liệu cũng sẽ là ưu tiên hàng đầu để đảm bảo hệ thống hoạt động ổn định và bảo vệ thông tin người dùng.

# **TÀI LIỆU THAM KHẢO**

[1] Thạc Bình Cường – Nguyễn Đức Mận. (2023). *Kiểm thử và đảm bảo chất lượng phần mềm*. ĐH Bách Khoa Hà Nội.

[2] Microsoft. (2023). *ASP.NET Core Documentation*. https://learn.microsoft.com/en-us/aspnet/core/

[3] Dan Abramov & Next.js Team. (2023). *Next.js Documentation*. <https://nextjs.org/docs>*.*

[4] Robert C. Martin. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.*

[5] Mark Seemann. (2011). Dependency Injection in .NET. Manning Publications.