

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**ĐỒ ÁN TỐT NGHIỆP**

**Thiết kế xây dựng Hệ thống giao đồ ăn YummyZoom**

**HOÀNG NGUYÊN VŨ**

vu.hn215171@sis.hust.edu.vn

**Chương trình đào tạo: Công nghệ thông tin Việt-Nhật**

**Giảng viên hướng dẫn:** TS. Đỗ Tuấn Anh

Chữ ký GVHD

**Khoa:** Kỹ thuật máy tính

**Trường:** Công nghệ Thông tin và Truyền thông

**HÀ NỘI, 01/2026**

## LỜI CẢM ƠN

Sau một thời gian nghiên cứu và thực hiện, đồ án tốt nghiệp của em đã hoàn thành. Đây là cơ hội để em được vận dụng những kiến thức đã học vào một dự án thực tế, đồng thời trải nghiệm quy trình phát triển phần mềm từ phân tích, thiết kế cho đến hiện thực và triển khai. Kết quả này không thể thiếu sự hỗ trợ và đồng hành của nhiều người.

Trước tiên, em xin gửi lời cảm ơn sâu sắc đến TS. Đỗ Tuấn Anh – giảng viên hướng dẫn đồ án của em. Thầy đã nhiệt tình định hướng, giải đáp thắc mắc và góp ý xuyên suốt quá trình thực hiện. Những buổi trao đổi với Thầy không chỉ giúp em nắm vững hơn về kiến trúc phần mềm và thiết kế hệ thống, mà còn giúp em học được cách tư duy và giải quyết vấn đề một cách có hệ thống.

Em cũng xin cảm ơn các thầy cô trong Trường Công nghệ Thông tin và Truyền thông đã trang bị cho em nền tảng kiến thức vững chắc suốt những năm học qua. Những môn học về lập trình, cơ sở dữ liệu, kiến trúc phần mềm và công nghệ web đều là hành trang quan trọng giúp em hoàn thành đồ án này.

Lời cảm ơn chân thành cũng gửi đến gia đình – nơi luôn là chỗ dựa vững chắc nhất. Bố mẹ và anh chị đã tạo điều kiện tốt nhất để em yên tâm học tập và theo đuổi đam mê của mình.

Mặc dù đã cố gắng hết sức, đồ án vẫn không tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý của quý Thầy Cô để đồ án được hoàn thiện hơn.

Em xin chân thành cảm ơn!

# TÓM TẮT NỘI DUNG ĐỒ ÁN

Việc đặt đồ ăn chung trong nhóm bạn bè hay đồng nghiệp là chuyện thường ngày, nhất là vào giờ trưa khi thời gian có hạn. Tuy nhiên, dù các ứng dụng giao đồ ăn hiện nay đã khá tiện lợi, việc đặt hàng nhóm vẫn còn một vấn đề nhức nhối: thường phải có một người trả tiền trước rồi sau đó đi thu từng người một. Chuyện này không chỉ mất thời gian mà còn phiền toái, dễ quên hoặc tính sai tiền. Xuất phát từ thực tế đó, đồ án này xây dựng YummyZoom – một hệ thống giao đồ ăn ở mức Sản phẩm khả dụng tối thiểu (Minimum Viable Product - MVP) (Minimum Viable Product) với điểm nhấn là tính năng Giỏ hàng nhóm (TeamCart) (Giỏ hàng nhóm). TeamCart cho phép nhiều người cùng lúc chọn món vào chung một giỏ hàng, theo dõi những thay đổi ngay lập tức, và quan trọng nhất là mỗi người tự thanh toán phần của mình – không cần ai đứng ra "bao" cả nhóm nữa.

Về mặt kỹ thuật, đồ án áp dụng Kiến trúc sạch (Clean Architecture) kết hợp với Domain-Driven Design (Thiết kế hướng tên miền (Domain-Driven Design - DDD)) để xây dựng mô hình nghiệp vụ và quản lý các quy tắc phức tạp trong TeamCart. Lớp ứng dụng được tổ chức theo mô hình Tách biệt đọc/ghi (Command Query Responsibility Segregation - CQRS), giúp tách biệt rõ ràng giữa việc đọc và ghi dữ liệu. Phía backend sử dụng .NET 9 và ASP.NET Core, dữ liệu chính lưu trên PostgreSQL, còn Redis đảm nhận việc cache nhanh và SignalR lo phần đồng bộ real-time. Ứng dụng di động được viết bằng Flutter để chạy mượt mà trên cả iOS và Android. Ngoài ra, hệ thống còn có giao diện web dành cho nhà hàng và admin, được xây dựng bằng Angular và PrimeNG, để quản lý thực đơn và theo dõi đơn hàng.

Điểm mạnh của đồ án nằm ở việc thiết kế một kiến trúc phần mềm rõ ràng, dễ kiểm thử và bảo trì, đồng thời giải quyết được bài toán nhiều người cùng chỉnh sửa giỏ hàng mà không bị xung đột nhờ cơ chế kiểm soát đồng thời lạc quan (optimistic concurrency) và đồng bộ real-time. Kết quả thu được là một hệ thống hoàn chỉnh với đầy đủ luồng đặt món cá nhân lẫn đặt món nhóm, đã qua kiểm thử kịch bản và thử nghiệm triển khai thực tế. Tất nhiên, đồ án cũng còn một số hạn chế cần khắc phục trong tương lai như: chưa tích hợp cổng thanh toán thật sự (vẫn đang ở mức demo), chưa có module quản lý tài xế, chưa có tính năng tìm kiếm thông minh dựa trên AI, và chưa có hệ thống quản lý tài chính đầy đủ cho nhà hàng.

Sinh viên thực hiện  
(Ký và ghi rõ họ tên)

## ABSTRACT

Group food ordering has become a routine practice among students and office workers, particularly during limited time windows such as lunch breaks. However, despite the convenience of modern food delivery platforms, group ordering remains hampered by a significant pain point: the payment process typically requires one person to pay for the entire order upfront and then collect money from each member afterward. This approach is time-consuming, prone to errors, and creates unnecessary friction outside the app. To address this problem, this thesis presents YummyZoom, a food delivery system developed as a Minimum Viable Product (MVP) with a key focus on the TeamCart feature. TeamCart allows multiple users to collaboratively add items to a shared cart, receive real-time updates, and most importantly, enables each member to pay for their own portion independently.

From a technical perspective, the project leverages Clean Architecture combined with Domain-Driven Design (DDD) to model the business domain and manage the complex collaboration rules within TeamCart. The application layer follows a CQRS pattern to clearly separate read and write operations. The backend is built on .NET 9 and ASP.NET Core, with PostgreSQL serving as the primary data store, Redis providing fast caching, and SignalR handling real-time synchronization. The mobile application is developed using Flutter to ensure a consistent user experience across both iOS and Android platforms. Additionally, the system includes web portals for restaurants and administrators, built with Angular and PrimeNG, to manage menus and monitor orders.

The main contributions of this work include a well-structured, testable software architecture and an effective solution for handling concurrent updates in TeamCart through Kiểm soát đồng thời lạc quan (Optimistic Concurrency Control) and real-time synchronization, minimizing conflicts when multiple users interact simultaneously. The resulting system fully supports both individual and group ordering workflows, validated through scenario-based testing and experimental deployment. The thesis also acknowledges several limitations that require future work, including the lack of production-ready payment gateway integration, the absence of a driver management module, missing AI-powered search and recommendation features, and the need for a comprehensive financial management system for restaurants.

## MỤC LỤC

<b>CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....</b>	<b>1</b>
1.1 Đặt vấn đề.....	1
1.2 Mục tiêu và phạm vi đề tài.....	1
1.3 Định hướng giải pháp.....	2
1.4 Bố cục đồ án .....	2
<b>CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU.....</b>	<b>4</b>
2.1 Khảo sát hiện trạng .....	4
2.1.1 Bối cảnh và xu hướng thị trường.....	4
2.1.2 Phân tích các ứng dụng giao đồ ăn hiện có .....	4
2.1.3 Nhu cầu và thách thức của người dùng .....	5
2.2 Tổng quan chức năng .....	5
2.2.1 Biểu đồ use case tổng quát .....	5
2.2.2 Biểu đồ use case phân rã "Quản lý tài khoản" .....	8
2.2.3 Biểu đồ use case phân rã "Đặt hàng cá nhân" .....	9
2.2.4 Biểu đồ use case phân rã "TeamCart" .....	10
2.2.5 Biểu đồ use case phân rã "Quản lý thực đơn" .....	11
2.2.6 Biểu đồ use case phân rã "Duyệt đăng ký nhà hàng" .....	12
2.2.7 Quy trình nghiệp vụ .....	12
2.3 Đặc tả chức năng .....	15
2.3.1 Đặc tả Use Case: Đăng ký nhà hàng.....	15
2.3.2 Đặc tả Use Case: Quản lý thực đơn .....	16
2.3.3 Đặc tả Use Case: Tìm kiếm nhà hàng.....	18
2.3.4 Đặc tả Use Case: Đặt hàng cá nhân.....	20
2.3.5 Đặc tả Use Case: Khởi tạo TeamCart .....	23

2.3.6 Đặc tả Use Case: Tham gia TeamCart .....	25
2.3.7 Đặc tả Use Case: Chốt đơn TeamCart .....	26
2.3.8 Đặc tả Use Case: Xử lý đơn hàng.....	30
2.4 Yêu cầu phi chức năng .....	32
<b>CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG.....</b>	<b>35</b>
3.1 Kiến trúc hệ thống và Ứng dụng Backend .....	35
3.1.1 Kiến trúc Clean Architecture và Domain-Driven Design (DDD)....	35
3.1.2 Nền tảng phát triển: .NET 9 và ASP.NET Core .....	36
3.1.3 Điều phối hệ thống và Khả năng quan sát: .NET Aspire.....	36
3.2 Cơ sở dữ liệu và Lưu trữ .....	36
3.2.1 Hệ quản trị cơ sở dữ liệu quan hệ: PostgreSQL .....	37
3.2.2 Bộ nhớ đệm (Caching): Redis.....	37
3.2.3 Quản lý và Phân phối đa phương tiện: Cloudinary .....	37
3.3 Xác thực và Phân quyền (Authentication & Authorization) .....	38
3.3.1 Cơ chế xác thực: JSON Web Token (JWT) .....	38
3.3.2 Quản lý định danh và Phân quyền: ASP.NET Core Identity .....	38
3.4 Ứng dụng dành cho khách hàng (Mobile App).....	38
3.4.1 Nền tảng phát triển: Flutter & Dart .....	38
3.4.2 Quản lý trạng thái và Tương tác API .....	39
3.4.3 Tích hợp bản đồ và Thanh toán.....	39
3.5 Ứng dụng Web quản trị (Admin & Restaurant Portal).....	39
3.5.1 Nền tảng Frontend: Angular.....	39
3.5.2 Thư viện giao diện: PrimeNG và Tailwind CSS .....	39
3.6 Cộng tác thời gian thực (Realtime Collaboration) .....	40
3.6.1 Giao thức và Framework: SignalR .....	40
3.6.2 Thông báo đẩy tới thiết bị di động: Firebase Cloud Messaging (FCM).....	40

3.7 Nền tảng triển khai và phân phối hệ thống .....	40
3.7.1 Triển khai Backend trên Azure bằng Azure Developer CLI (azd) và .NET Aspire .....	40
3.7.2 Tự động hóa triển khai bằng GitHub Actions và cơ chế xác thực OIDC .....	41
3.7.3 Phân phối ứng dụng di động bằng tệp APK .....	41
3.7.4 Triển khai cổng web quản trị trên Vercel .....	41
<b>CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG ....</b>	<b>43</b>
4.1 Thiết kế kiến trúc.....	43
4.1.1 Lựa chọn kiến trúc phần mềm .....	43
4.1.2 Thiết kế tổng quan.....	45
4.1.3 Thiết kế chi tiết gói .....	46
4.2 Thiết kế chi tiết.....	49
4.2.1 Thiết kế giao diện .....	49
4.2.2 Thiết kế lớp .....	51
4.2.3 Thiết kế cơ sở dữ liệu .....	56
4.3 Xây dựng ứng dụng.....	59
4.3.1 Thư viện và công cụ sử dụng .....	59
4.3.2 Kết quả đạt được .....	61
4.3.3 Minh họa các chức năng chính .....	62
4.4 Kiểm thử.....	67
4.4.1 Kiểm thử chức năng Đặt hàng nhóm (TeamCart).....	67
4.4.2 Kiểm thử chức năng Quy trình Đơn hàng (Live Orders).....	68
4.4.3 Kiểm thử chức năng Tìm kiếm và Lọc món ăn .....	69
4.4.4 Tổng kết kết quả kiểm thử .....	70
4.5 Triển khai .....	70

<b>CHƯƠNG 5. GIẢI PHÁP ĐÓNG GÓP, KẾT LUẬN VÀ HƯỚNG PHÁT</b>	
<b>TRIỂN .....</b>	<b>74</b>
5.1 Giải pháp đóng góp.....	74
5.1.1 Ứng dụng Clean Architecture và Domain-Driven Design trong quản lý nghiệp vụ phức tạp.....	74
5.1.2 Giải pháp xử lý cạnh tranh dữ liệu với Optimistic Concurrency Control.....	75
5.1.3 Kiến trúc đồng bộ hóa thời gian thực hiệu năng cao .....	75
5.2 Kết luận và đánh giá tổng quan .....	76
5.3 Hướng phát triển.....	77
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>80</b>
<b>PHỤ LỤC.....</b>	<b>80</b>

## **DANH MỤC HÌNH VẼ**

Hình 2.1	Biểu đồ use case tổng quát của hệ thống YummyZoom . . . . .	6
Hình 2.2	Biểu đồ use case phân rã - Quản lý tài khoản . . . . .	8
Hình 2.3	Biểu đồ use case phân rã - Đặt hàng cá nhân . . . . .	9
Hình 2.4	Biểu đồ use case phân rã - TeamCart . . . . .	10
Hình 2.5	Biểu đồ use case phân rã - Quản lý thực đơn . . . . .	11
Hình 2.6	Biểu đồ use case phân rã - Duyệt đăng ký nhà hàng . . . . .	12
Hình 2.7	Biểu đồ hoạt động - Quy trình xử lý đặt hàng . . . . .	13
Hình 2.8	Biểu đồ hoạt động - Quy trình đặt hàng nhóm . . . . .	14
Hình 4.1	Biểu đồ phụ thuộc gói tổng quan YummyZoom (Backend) .	46
Hình 4.2	Thiết kế lớp miền cho luồng Khởi tạo Đơn hàng . . . . .	47
Hình 4.3	Thiết kế gói lớp ứng dụng và hạ tầng cho luồng Khởi tạo Đơn hàng . . . . .	48
Hình 4.4	Minh họa màn hình Home/Menu (trái) và Restaurant Detail (phải) trên ứng dụng di động . . . . .	50
Hình 4.5	Minh họa màn hình Restaurant Dashboard trên hệ thống quản trị	51
Hình 4.6	Minh họa màn hình Live Orders trên hệ thống quản trị . . .	51
Hình 4.7	Biểu đồ lớp của Order (Aggregate root) . . . . .	52
Hình 4.8	Biểu đồ lớp của TeamCart (Aggregate root) . . . . .	53
Hình 4.9	Biểu đồ lớp của Restaurant (Aggregate root) . . . . .	54
Hình 4.10	Biểu đồ trình tự chuyển TeamCart thành Order . . . . .	55
Hình 4.11	Biểu đồ trình tự cập nhật trạng thái đơn hàng . . . . .	56
Hình 4.12	ERD khái niệm theo mô hình Chen cho YummyZoom . . . .	57
Hình 4.13	ERD hiện đại cho nhóm Restaurant Catalog . . . . .	58
Hình 4.14	ERD hiện đại cho nhóm Order và TeamCart . . . . .	59
Hình 4.15	Màn hình Trang chủ (Trái) và Chi tiết nhà hàng (Phải) . . .	63
Hình 4.16	Giao diện TeamCart: Màn hình Chủ phòng (Trái) và Thành viên (Phải) . . . . .	64
Hình 4.17	Quy trình Checkout: Giai đoạn Chốt giá (Trái) và Thanh toán (Phải) . . . . .	65
Hình 4.18	Giao diện Quản lý Menu và Thư viện tùy chọn . . . . .	66
Hình 4.19	Trung tâm xử lý đơn hàng trực tiếp (Live Orders Dashboard)	66
Hình 4.20	Giao diện Quản lý Mã giảm giá và Chương trình khuyến mãi	67

## **DANH MỤC BẢNG BIỂU**

Bảng 2.1	Đặc tả Use Case Đăng ký nhà hàng . . . . .	15
Bảng 2.2	Dữ liệu đầu vào Đăng ký nhà hàng . . . . .	16
Bảng 2.3	Đặc tả Use Case Quản lý thực đơn . . . . .	17
Bảng 2.4	Dữ liệu đầu vào Quản lý thực đơn - Tạo món ăn . . . . .	18
Bảng 2.5	Đặc tả Use Case Tìm kiếm nhà hàng . . . . .	19
Bảng 2.6	Dữ liệu đầu vào Tìm kiếm nhà hàng . . . . .	20
Bảng 2.7	Đặc tả Use Case Đặt hàng cá nhân . . . . .	22
Bảng 2.8	Dữ liệu đầu vào Đặt hàng cá nhân . . . . .	23
Bảng 2.9	Đặc tả Use Case Khởi tạo TeamCart . . . . .	24
Bảng 2.10	Dữ liệu đầu vào Khởi tạo TeamCart . . . . .	25
Bảng 2.11	Đặc tả Use Case Tham gia TeamCart . . . . .	26
Bảng 2.12	Dữ liệu đầu vào Tham gia TeamCart . . . . .	26
Bảng 2.13	Đặc tả Use Case Chốt đơn TeamCart . . . . .	29
Bảng 2.14	Dữ liệu đầu vào Chốt giá TeamCart . . . . .	29
Bảng 2.15	Dữ liệu đầu vào Chốt đơn TeamCart . . . . .	30
Bảng 2.16	Đặc tả Use Case Xử lý đơn hàng . . . . .	31
Bảng 2.17	Dữ liệu đầu vào Xử lý đơn hàng . . . . .	32
Bảng 4.1	Danh sách thư viện và công cụ sử dụng . . . . .	61
Bảng 4.2	Thông kê kết quả backend . . . . .	61
Bảng 4.3	Thông kê kết quả mobile . . . . .	62
Bảng 4.4	Thông kê kết quả web admin . . . . .	62
Bảng 4.5	Kiểm thử chức năng Đặt hàng nhóm . . . . .	68
Bảng 4.6	Kiểm thử chức năng Quy trình Đơn hàng . . . . .	69
Bảng 4.7	Kiểm thử chức năng Tìm kiếm và Lọc . . . . .	70

## DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

.NET	Nền tảng phát triển ứng dụng của Microsoft
ACID	Tập tính chất giao dịch: Atomicity, Consistency, Isolation, Durability
Angular	Framework phát triển ứng dụng web
API	Giao diện lập trình ứng dụng (Application Programming Interface)
ASP.NET Core	Framework phát triển web/API trên nền tảng .NET
azd	Azure Developer CLI
Azure Container Apps	Dịch vụ chạy ứng dụng container trên Azure
Bicep	Ngôn ngữ hạ tầng dưới dạng mã cho Azure
CI/CD	Tích hợp liên tục/triển khai liên tục (Continuous Integration/Continuous Delivery)
Clean Architecture	Phong cách kiến trúc phần mềm tách biệt logic nghiệp vụ khỏi hạ tầng
COD	Thanh toán khi nhận hàng (Cash On Delivery)
CQRS	Tách biệt trách nhiệm đọc/ghi (Command Query Responsibility Segregation)
DDD	Thiết kế hướng tên miền (Domain-Driven Design)
DTO	Đối tượng truyền dữ liệu (Data Transfer Object)
EF Core	Entity Framework Core (ORM cho .NET)
FCM	Firebase Cloud Messaging (dịch vụ thông báo đẩy)
Flutter	Framework phát triển ứng dụng đa nền tảng (mobile)
IaC	Hạ tầng dưới dạng mã (Infrastructure as Code)
Idempotency	Tính chất gọi lặp nhiều lần nhưng kết quả không đổi
JSONB	Kiểu dữ liệu JSON dạng nhị phân trong PostgreSQL
JWT	Chuẩn token xác thực (JSON Web Token)

MVP	Sản phẩm khả dụng tối thiểu (Minimum Viable Product)
OIDC	Chuẩn xác thực OpenID Connect
OpenTelemetry	Chuẩn thu thập telemetry (logs, metrics, traces) cho quan sát hệ thống
Optimistic Concurrency Control	Cơ chế kiểm soát đồng thời lạc quan dựa trên phiên bản dữ liệu
ORM	Ánh xạ đối tượng-quan hệ (Object-Relational Mapping)
PaymentIntent	Đối tượng giao dịch thanh toán trong hệ thống cổng thanh toán
PostGIS	Phần mở rộng không gian địa lý cho PostgreSQL
PostgreSQL	Hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở
RBAC	Phân quyền dựa trên vai trò (Role-Based Access Control)
Redis	Kho lưu trữ key-value trong bộ nhớ, thường dùng làm cache
SDK	Bộ công cụ phát triển phần mềm (Software Development Kit)
SignalR	Thư viện giao tiếp thời gian thực cho ứng dụng .NET
TeamCart	Tính năng giỏ hàng nhóm cho phép nhiều người cùng chọn món và thanh toán tách biệt
WebSockets	Giao thức kết nối hai chiều lâu dài giữa client và server
YummyZoom	Hệ thống giao đồ ăn với điểm nhận đặt hàng nhóm

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## 1.1 Đặt vấn đề

Trong những năm gần đây, các nền tảng giao đồ ăn trực tuyến đã trở thành một kênh tiêu dùng phổ biến nhờ khả năng đáp ứng nhanh, tiện lợi và phù hợp với nhịp sống đô thị. Với nhóm sinh viên và nhân viên văn phòng, nhu cầu đặt đồ ăn theo nhóm trong các khung giờ giới hạn (đặc biệt là bữa trưa) diễn ra thường xuyên, kéo theo yêu cầu về thao tác đặt món nhanh, minh bạch chi phí và phối hợp nhóm thuận tiện.

Mặc dù các nền tảng phổ biến đã cung cấp nhiều chức năng cốt lõi của một hệ thống giao đồ ăn, trải nghiệm đặt hàng theo nhóm vẫn còn tồn tại hạn chế đáng kể ở khâu thanh toán. Trong nhiều kịch bản thực tế, một người thường phải đứng ra thanh toán toàn bộ đơn hàng, sau đó tự thu lại tiền từ các thành viên còn lại. Quy trình này làm tăng số bước thao tác ngoài ứng dụng, tốn thời gian, và dễ phát sinh sai lệch hoặc chậm trễ trong việc hoàn trả chi phí, đặc biệt trong môi trường tập thể.

Từ bối cảnh đó, đồ án lựa chọn giải quyết bài toán đặt hàng nhóm theo hướng giảm “ma sát” trong phối hợp và thanh toán, đồng thời vẫn đảm bảo đầy đủ hành trình đặt món của người dùng và khả năng vận hành của phía nhà hàng. Trọng tâm của dự án là tính năng TeamCart (Giỏ hàng nhóm), cho phép nhiều người cùng tham gia chọn món trong một giỏ hàng, theo dõi trạng thái cập nhật tức thời, và hỗ trợ cơ chế thanh toán theo từng thành viên theo định hướng tách bạch trách nhiệm chi trả.

## 1.2 Mục tiêu và phạm vi đề tài

Từ vấn đề đã nêu, mục tiêu của dự án YummyZoom là xây dựng một hệ thống giao đồ ăn ở mức sản phẩm khả dụng tối thiểu (MVP), trong đó nổi bật là trải nghiệm đặt hàng nhóm TeamCart, hỗ trợ nhiều người cùng chọn món và tách bạch việc thanh toán theo từng thành viên. Song song với chức năng trọng tâm, hệ thống cần đáp ứng các chức năng cốt lõi của một nền tảng giao đồ ăn, đảm bảo người dùng có thể hoàn thành hành trình đặt món, phía nhà hàng có thể vận hành đơn, và nền tảng có cơ chế quản trị phù hợp.

Về phạm vi chức năng, hệ thống được xây dựng theo ba nhóm vai trò chính. Đối với khách hàng, hệ thống hỗ trợ duyệt nhà hàng và thực đơn, tùy chỉnh món ăn, đặt hàng cá nhân, tham gia TeamCart, theo dõi trạng thái đơn hàng và đánh giá sau khi sử dụng dịch vụ. Đối với nhà hàng, hệ thống hỗ trợ quản lý hồ sơ và thực đơn, cập

nhật trạng thái món, tiếp nhận và xử lý đơn hàng theo vòng đời, và theo dõi các thông tin vận hành liên quan. Đối với quản trị viên, hệ thống cung cấp các chức năng quản lý nền tảng như duyệt đăng ký nhà hàng, giám sát nội dung và quản trị dữ liệu phục vụ vận hành.

Để đảm bảo tính khả thi trong khuôn khổ đề án tốt nghiệp, phạm vi triển khai được giới hạn ở mức phù hợp. Quy trình giao nhận được mô phỏng thông qua cập nhật trạng thái đơn hàng thay vì phát triển một phân hệ tài xế với các nghiệp vụ định vị, điều phối và theo dõi GPS thời gian thực. Bên cạnh đó, hệ thống chưa triển khai các tính năng nâng cao như gợi ý món ăn, tìm kiếm ứng dụng học máy theo hướng ngữ nghĩa, tích hợp cổng thanh toán ở mức sẵn sàng sản xuất, cũng như module tài chính chuẩn cho phía nhà hàng (ví dụ sổ cái doanh thu, đối soát theo kỳ, và luồng rút tiền).

### 1.3 Định hướng giải pháp

Để hiện thực hóa mục tiêu đề tài, đề án lựa chọn hướng tiếp cận ưu tiên tính mô-đun, khả năng bảo trì và khả năng mở rộng trong triển khai. Ở mức kiến trúc, hệ thống áp dụng Clean Architecture kết hợp Domain-Driven Design (DDD) nhằm tách biệt rõ logic nghiệp vụ khỏi các chi tiết hạ tầng, qua đó giảm phụ thuộc giữa các lớp, hỗ trợ kiểm thử và cho phép mở rộng chức năng theo từng miền nghiệp vụ. Các quy tắc quan trọng của TeamCart được mô hình hóa tập trung trong lớp miền (Domain layer), trong khi lớp ứng dụng (Application layer) điều phối các ca sử dụng theo định hướng CQRS.

Ở mức công nghệ, hệ thống backend được xây dựng trên .NET 9 và ASP.NET Core, triển khai Giao diện lập trình ứng dụng (Application Programming Interface - API) phục vụ ứng dụng khách hàng và cổng quản trị. PostgreSQL được sử dụng làm hệ quản trị cơ sở dữ liệu chính để lưu trữ dữ liệu bền vững, Redis được sử dụng cho các nhu cầu lưu trữ trạng thái nhanh và hỗ trợ các luồng cộng tác, trong khi SignalR đảm nhiệm kênh giao tiếp thời gian thực (real-time) để các thành viên trong TeamCart có thể nhận cập nhật tức thời. Ở phía giao diện, ứng dụng khách hàng được phát triển bằng Flutter để đảm bảo trải nghiệm nhất quán trên iOS và Android, còn cổng quản trị nhà hàng và quản trị viên được xây dựng theo mô hình Single Page Application (SPA) sử dụng Angular và PrimeNG. Các lựa chọn công nghệ và cách áp dụng cụ thể sẽ được trình bày chi tiết trong các chương tiếp theo.

### 1.4 Bố cục đề án

Phần còn lại của báo cáo được tổ chức như sau. Chương 2 trình bày quá trình khảo sát và phân tích yêu cầu của hệ thống YummyZoom, bao gồm bối cảnh bài toán, yêu cầu chức năng và phi chức năng làm cơ sở cho thiết kế và triển khai.

## CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

Chương 3 giới thiệu các công nghệ và giải pháp kỹ thuật được lựa chọn để hiện thực hệ thống theo hướng Clean Architecture và DDD, đồng thời trình bày các thành phần hạ tầng phục vụ lưu trữ dữ liệu và cộng tác thời gian thực. Chương 4 mô tả thiết kế, triển khai và đánh giá hệ thống, bao gồm kiến trúc tổng thể, thiết kế cơ sở dữ liệu, hiện thực các chức năng chính và kiểm thử. Chương 5 tổng hợp các giải pháp đóng góp nổi bật, đánh giá sản phẩm trong bối cảnh thị trường, nêu các hạn chế còn tồn tại và đề xuất hướng phát triển cho dự án.

## CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

Trong chương này, nghiên cứu tiến hành khảo sát và phân tích toàn diện về hiện trạng của ngành giao đồ ăn trực tuyến, từ đó xác định được các yêu cầu cần thiết cho việc phát triển hệ thống YummyZoom. Chương bao gồm bốn phần chính: khảo sát hiện trạng thị trường và các ứng dụng tương tự hiện có, tổng quan về các chức năng cần thiết của hệ thống thông qua biểu đồ ca sử dụng (use case) và quy trình nghiệp vụ, đặc tả chi tiết các ca sử dụng quan trọng nhất, và cuối cùng là xác định các yêu cầu phi chức năng của hệ thống. Thông qua việc phân tích các ứng dụng như GrabFood, ShopeeFood cùng với khảo sát nhu cầu thực tế của người dùng và nhà hàng, chương này sẽ làm rõ những khoảng trống còn tồn tại trong thị trường hiện tại và định hướng các tính năng độc đáo mà YummyZoom cần phát triển để tạo ra lợi thế cạnh tranh.

### 2.1 Khảo sát hiện trạng

#### 2.1.1 Bối cảnh và xu hướng thị trường

Ngành giao đồ ăn trực tuyến tại Việt Nam đang chứng kiến sự tăng trưởng mạnh mẽ, thúc đẩy bởi chuyển đổi số và thay đổi thói quen tiêu dùng sau đại dịch COVID-19. Theo VECOM [1], thị trường đạt 1,2 tỷ USD năm 2023 với dự báo tăng trưởng 15-20%/năm. Khảo sát của Nielsen Vietnam [2] cho thấy 78% người tiêu dùng tại các thành phố lớn đã sử dụng dịch vụ này, với tần suất thường xuyên. Xu hướng thị trường đang chuyển dịch sang các nhu cầu cao hơn về chất lượng dịch vụ, tính cá nhân hóa và các tính năng xã hội như đặt hàng nhóm. Đôi với các nhà hàng, mô hình kinh doanh hybrid kết hợp phục vụ tại chỗ và trực tuyến đang trở nên phổ biến, tạo ra nhu cầu về các nền tảng quản lý hiệu quả với chi phí hợp lý.

#### 2.1.2 Phân tích các ứng dụng giao đồ ăn hiện có

Thị trường hiện tại được thống lĩnh bởi GrabFood và ShopeeFood, chiếm phần lớn thị phần.

##### a, GrabFood và ShopeeFood

GrabFood tận dụng lợi thế từ hệ sinh thái siêu ứng dụng Grab, cung cấp trải nghiệm liền mạch với đội ngũ tài xế đông đảo và tính năng theo dõi đơn hàng thời gian thực. Tuy nhiên, chi phí dịch vụ khá cao và giao diện tích hợp nhiều dịch vụ đôi khi gây khó khăn cho người dùng.

Ngược lại, ShopeeFood cạnh tranh bằng chiến lược giá và khuyến mãi mạnh mẽ, cùng giao diện đơn giản, phù hợp với giới trẻ. Dù vậy, chất lượng giao hàng chưa đồng đều và phạm vi phủ sóng còn hạn chế hơn so với đối thủ.

### b, Cơ hội cho YummyZoom

Cả hai nền tảng lớn đều chưa giải quyết triệt để vấn đề trong quy trình đặt hàng nhóm, nơi một người vẫn phải đứng ra thanh toán cho toàn bộ đơn hàng. Đây là khe hổng thị trường mà YummyZoom nhắm tới với tính năng TeamCart, cho phép tách biệt thanh toán cho từng thành viên. Đồng thời, việc tối ưu hóa giao diện đơn giản, tập trung vào trải nghiệm đặt món nhanh chóng sẽ là lợi thế cạnh tranh đối với nhóm khách hàng mục tiêu là nhân viên văn phòng và sinh viên.

#### 2.1.3 Nhu cầu và thách thức của người dùng

##### a, Nhu cầu cốt lõi

Người dùng cuối, đặc biệt là giới văn phòng và sinh viên, ưu tiên tốc độ, sự tiện lợi và minh bạch về chi phí. Nhu cầu bức thiết nhất là một giải pháp đặt hàng nhóm cho phép thanh toán độc lập để tránh các phiền toái về tài chính. Về phía nhà hàng, họ cần công cụ quản lý đơn hàng và thực đơn linh hoạt, cùng chính sách hoa hồng hợp lý để tối ưu lợi nhuận.

##### b, Thách thức và Định hướng phát triển

Thách thức lớn nhất hiện nay là quy trình thanh toán nhóm phức tạp và giao diện ứng dụng ngày càng rườm rà. YummyZoom được định hướng phát triển để giải quyết các vấn đề này thông qua:

- **Tính năng TeamCart:** Cho phép mỗi thành viên trong nhóm tự chọn món và thanh toán riêng biệt.
- **Giao diện tối giản:** Tập trung vào hiệu quả và tốc độ đặt hàng.
- **Tối ưu cho khách hàng mục tiêu:** Phục vụ nhu cầu cụ thể của nhóm người dùng có thời gian hạn chế và thói quen đặt hàng chung.

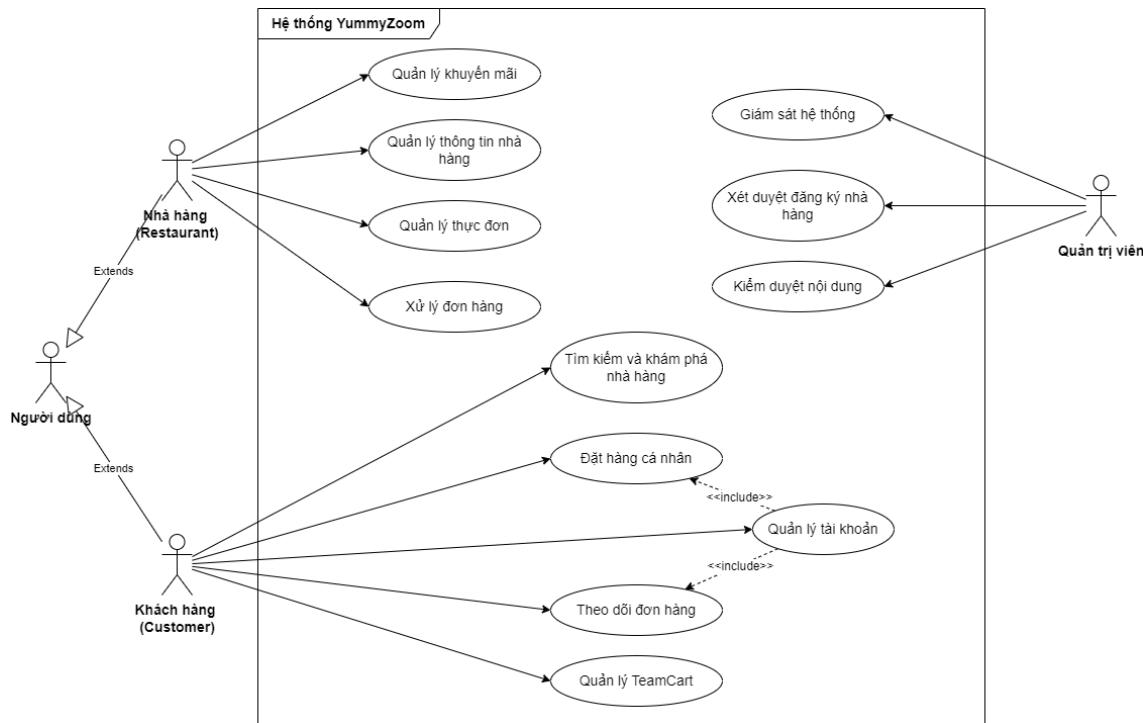
## 2.2 Tổng quan chức năng

Trên cơ sở các yêu cầu đã được xác định từ quá trình khảo sát, phần này cung cấp cái nhìn tổng quan về các chức năng của hệ thống YummyZoom. Hệ thống được mô hình hóa thông qua biểu đồ ca sử dụng (Use Case), bao gồm biểu đồ tổng quát và các biểu đồ phân rã cho những chức năng phức tạp. Việc phân chia này giúp làm rõ vai trò của các tác nhân cũng như phạm vi nghiệp vụ của hệ thống trước khi đi vào đặc tả chi tiết trong phần 2.3.

#### 2.2.1 Biểu đồ use case tổng quát

Hình 2.1 minh họa biểu đồ use case tổng quát của hệ thống YummyZoom, thể hiện tất cả các tác nhân (actors) chính và các chức năng cốt lõi mà họ có thể thực hiện trên nền tảng. Biểu đồ được thiết kế theo nguyên tắc phân nhóm rõ ràng các

use case theo từng đối tượng người dùng, giúp dễ dàng nhận diện phạm vi và ranh giới trách nhiệm của từng thành phần trong hệ thống.



**Hình 2.1:** Biểu đồ use case tổng quát của hệ thống YummyZoom

#### a, Các tác nhân chính trong hệ thống

Hệ thống YummyZoom bao gồm các tác nhân chính tương tác với phần mềm:

**Khách hàng (Customer):** Người dùng cuối sử dụng ứng dụng để tìm kiếm, đặt món và thanh toán. Khách hàng có thể là cá nhân đặt món riêng lẻ hoặc thành viên tham gia đặt hàng nhóm (TeamCart).

**Nhà hàng (Restaurant):** Đối tác kinh doanh cung cấp món ăn. Họ sử dụng hệ thống để quản lý thực đơn, cập nhật thông tin cửa hàng, thiết lập khuyến mãi và xử lý đơn hàng.

**Quản trị viên (Admin):** Người quản lý vận hành hệ thống, chịu trách nhiệm phê duyệt nhà hàng, kiểm duyệt nội dung và giám sát các hoạt động trên nền tảng.

### b, Các use case chính và mô tả chức năng

Dựa trên phân tích yêu cầu, các chức năng cốt lõi của hệ thống được tóm tắt theo từng nhóm tác nhân như sau:

### **Nhóm chức năng của Khách hàng**

**Quản lý tài khoản:** Đăng ký, đăng nhập, cập nhật thông tin cá nhân và sổ địa chỉ.

**Tìm kiếm nhà hàng:** Tra cứu nhà hàng và món ăn theo tên, danh mục hoặc vị trí.

**Đặt hàng (Cá nhân & TeamCart):** Thực hiện quy trình chọn món và tạo đơn hàng. Tính năng TeamCart cho phép nhiều người dùng cùng tham gia một đơn hàng và thanh toán riêng biệt.

**Thanh toán:** Xử lý giao dịch tài chính cho đơn hàng.

**Theo dõi đơn hàng:** Cập nhật trạng thái đơn hàng từ lúc đặt đến khi giao thành công.

**Đánh giá:** Gửi phản hồi và chấm điểm chất lượng dịch vụ của nhà hàng.

#### Nhóm chức năng của Nhà hàng

**Quản lý thông tin:** Cập nhật thông tin cơ bản, giờ mở cửa và trạng thái hoạt động.

**Quản lý thực đơn:** Thêm mới, chỉnh sửa thông tin món ăn, giá cả và tùy chọn.

**Xử lý đơn hàng:** Tiếp nhận đơn mới, cập nhật tiến độ chuẩn bị (đang nấu, đã xong).

**Quản lý khuyến mãi:** Tạo các mã giảm giá và chương trình ưu đãi cho khách hàng.

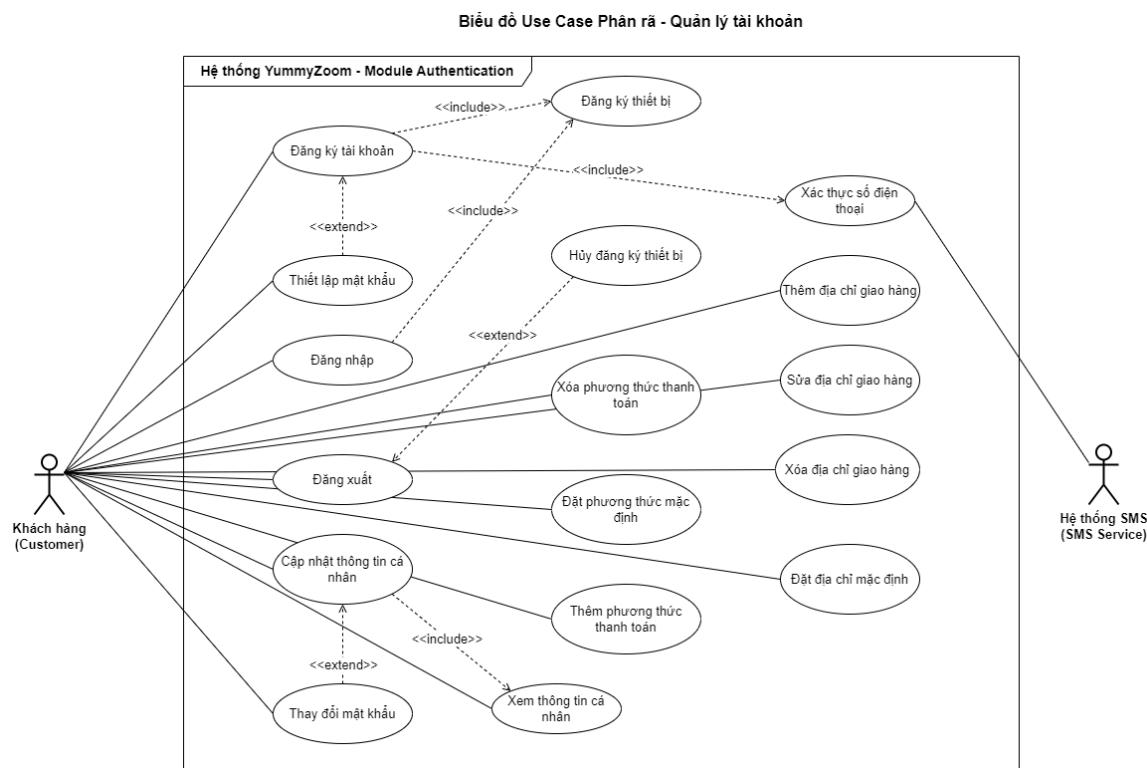
#### Nhóm chức năng của Quản trị viên

**Quản trị hệ thống:** Quản lý danh sách người dùng, danh mục món ăn và cấu hình hệ thống.

**Duyệt nhà hàng:** Kiểm tra và phê duyệt hồ sơ đăng ký của đối tác nhà hàng mới.

**Kiểm duyệt nội dung:** Giám sát các đánh giá, bình luận để đảm bảo tiêu chuẩn cộng đồng.

### 2.2.2 Biểu đồ use case phân rã "Quản lý tài khoản"



**Hình 2.2:** Biểu đồ use case phân rã - Quản lý tài khoản

Biểu đồ phân rã cho thấy use case "Quản lý tài khoản" được chia thành 17 use case con, được nhóm thành 5 chức năng chính: xác thực tài khoản (authentication), quản lý thông tin cá nhân (profile management), quản lý địa chỉ giao hàng (address management), quản lý phương thức thanh toán (payment management), và quản lý thiết bị (device management).

Điểm đáng chú ý trong thiết kế này là sự tham gia của tác nhân "Hệ thống SMS" để xử lý việc xác thực số điện thoại thông qua OTP, thể hiện tính bảo mật cao trong quy trình đăng ký.

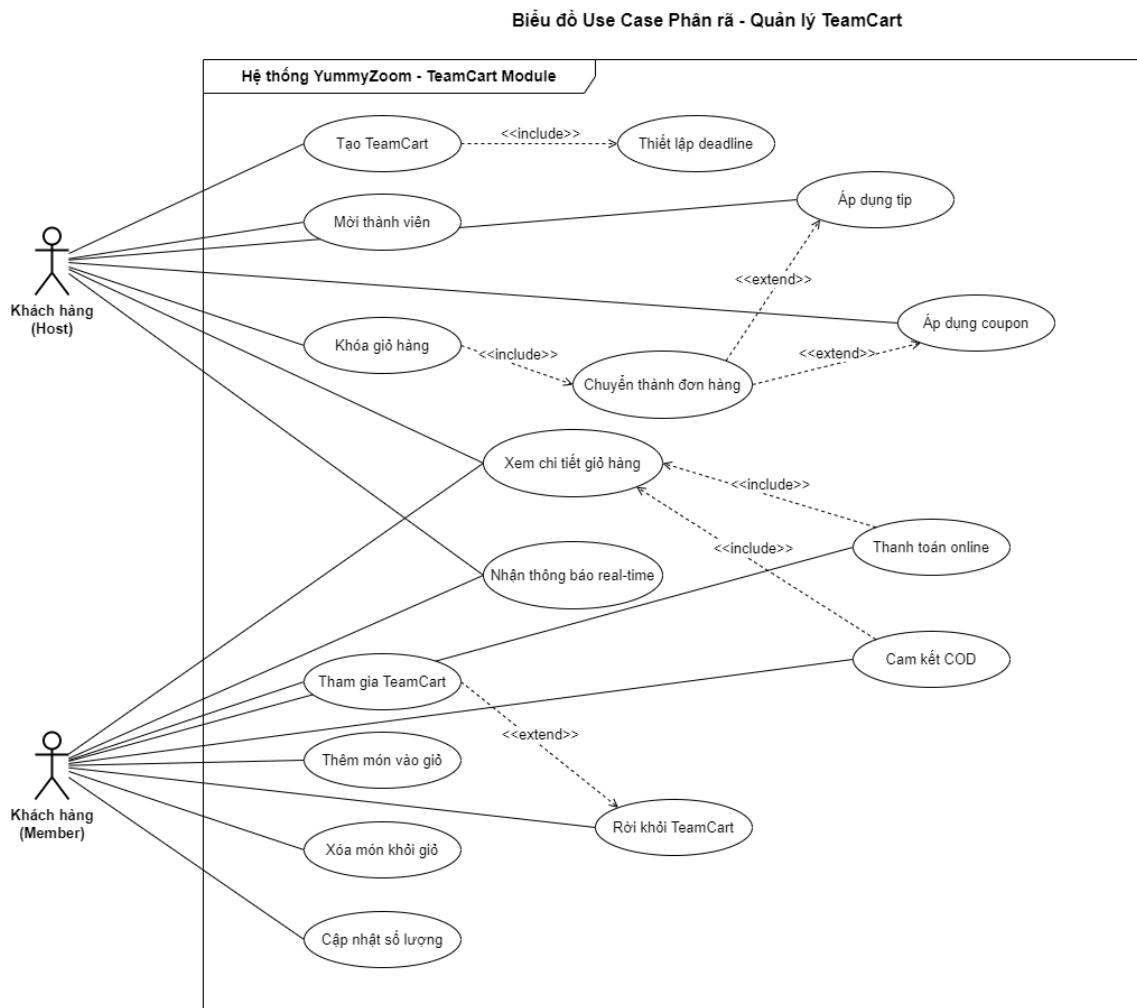
### 2.2.3 Biểu đồ use case phân rã "Đặt hàng cá nhân"



**Hình 2.3:** Biểu đồ use case phân rã - Đặt hàng cá nhân

Biểu đồ này làm rõ các chức năng con như tìm kiếm món ăn, xem chi tiết món, thêm vào giỏ hàng, và các bước trong quy trình thanh toán. Việc phân rã này giúp xác định rõ các điểm tương tác cần thiết để đảm bảo quy trình đặt hàng diễn ra thuận lợi và nhanh chóng, đáp ứng nhu cầu tiện lợi của người dùng.

### 2.2.4 Biểu đồ use case phân rã "TeamCart"

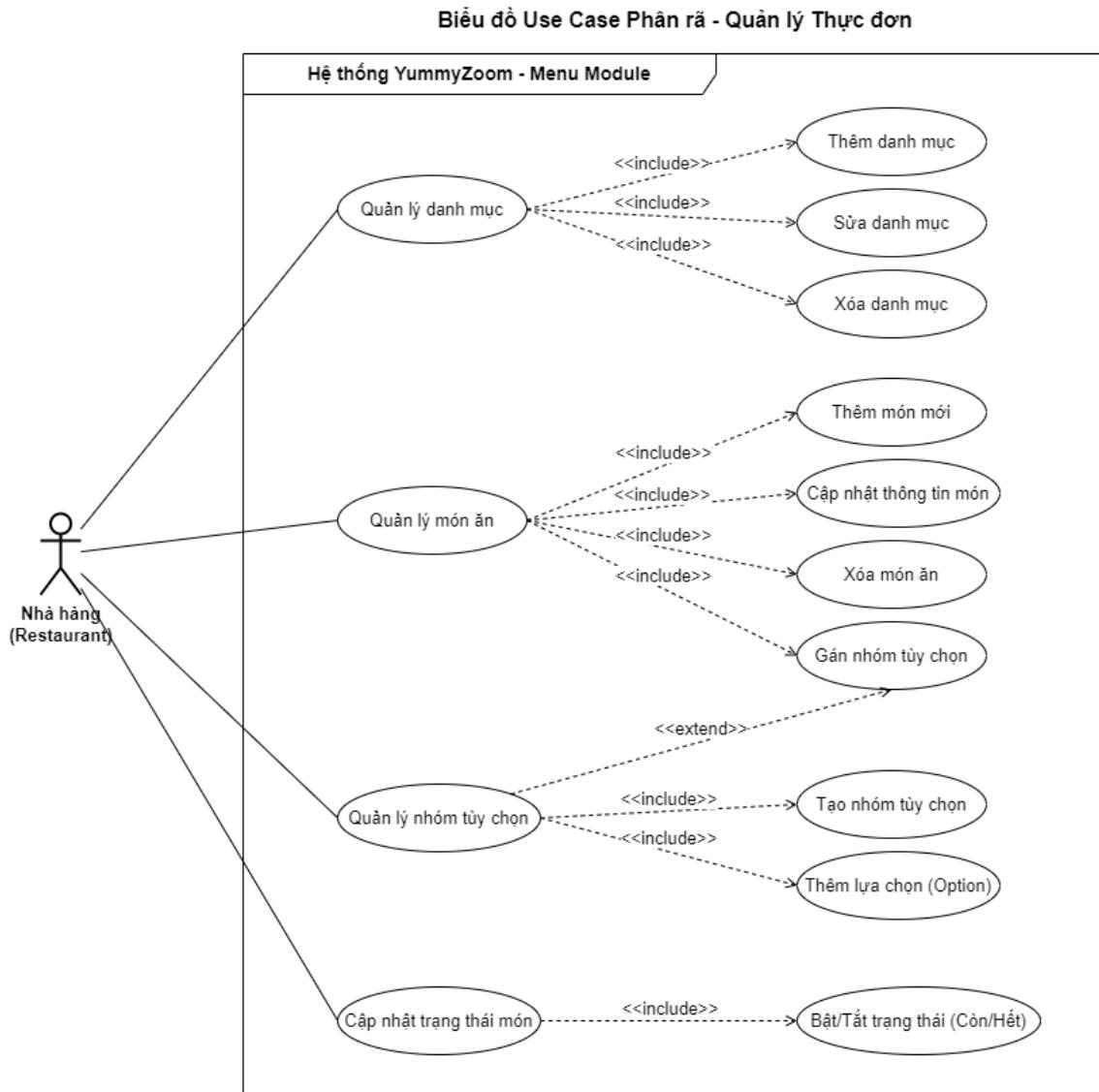


**Hình 2.4:** Biểu đồ use case phân rã - TeamCart

Qua biểu đồ, quy trình nghiệp vụ của TeamCart được thể hiện rõ ràng, bao gồm các chức năng như khởi tạo nhóm, chia sẻ liên kết mời, quản lý thành viên trong nhóm, và cơ chế tách biệt thanh toán cho từng thành viên. Đây là cơ sở quan trọng để xây dựng logic xử lý đồng bộ và đảm bảo tính chính xác trong các giao dịch nhóm, giải quyết triệt để vấn đề "ai trả tiền" trong các ứng dụng hiện tại.

### 2.2.5 Biểu đồ use case phân rã "Quản lý thực đơn"

Đối với tác nhân Nhà hàng, "Quản lý thực đơn" là chức năng có độ phức tạp cao và tần suất sử dụng thường xuyên nhất. Hình 2.5 minh họa chi tiết cấu trúc phân cấp của thực đơn trong hệ thống.

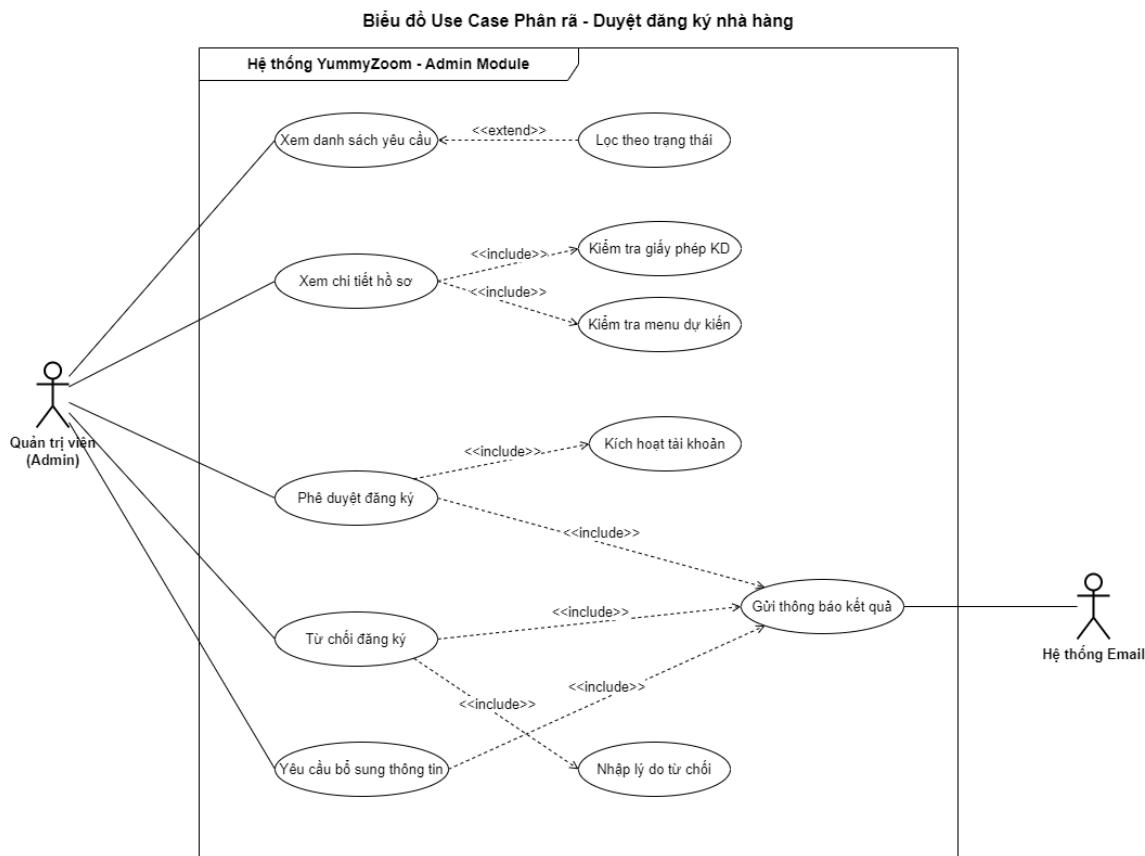


**Hình 2.5:** Biểu đồ use case phân rã - Quản lý thực đơn

Biểu đồ cho thấy sự phân rã chi tiết từ việc quản lý danh mục, món ăn cho đến các nhóm tùy chọn (topping, size, mức đường/dá). Đặc biệt, chức năng cập nhật trạng thái món (còn/hết) được tách biệt để đảm bảo tính phản hồi nhanh (real-time) trong giờ cao điểm.

### 2.2.6 Biểu đồ use case phân rã "Duyệt đăng ký nhà hàng"

Về phía Quản trị viên, quy trình "Duyệt đăng ký nhà hàng" đóng vai trò quan trọng trong việc kiểm soát chất lượng đầu vào của hệ thống. Hình 2.6 mô tả các bước xử lý hồ sơ đăng ký của đối tác.



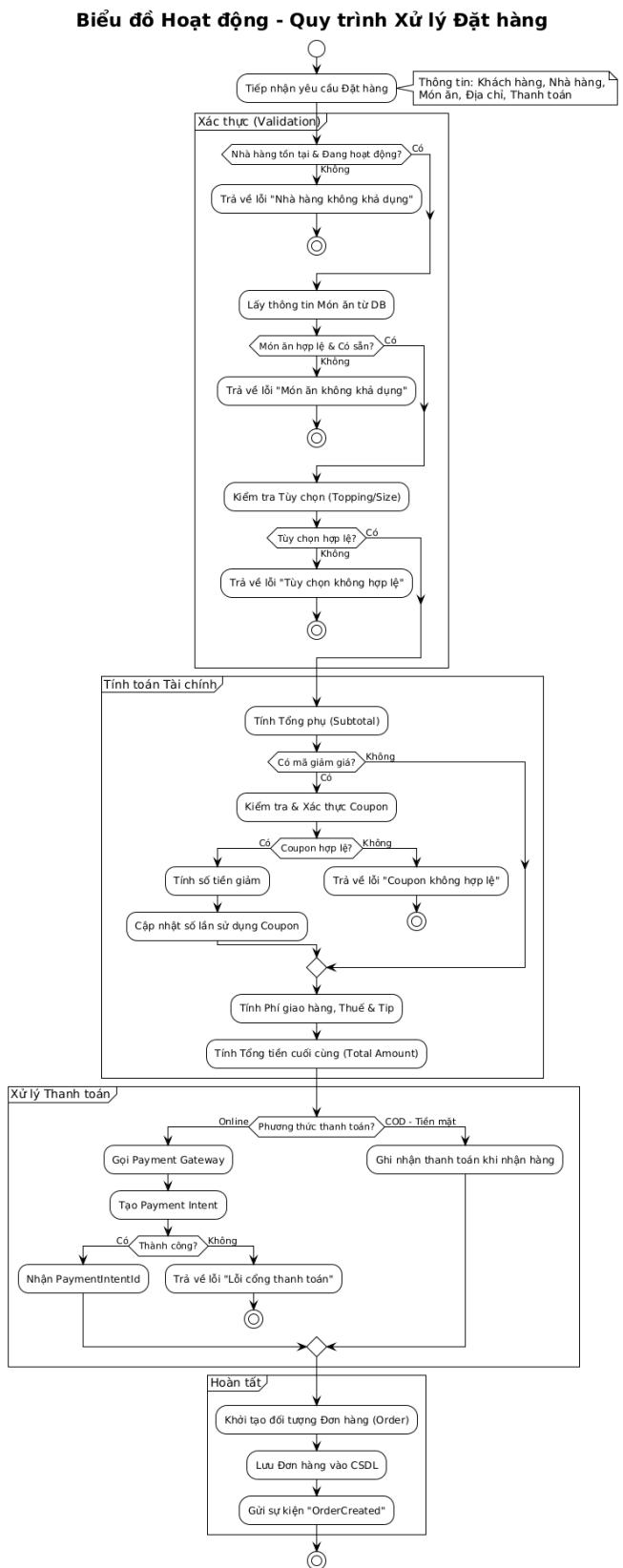
**Hình 2.6:** Biểu đồ use case phân rã - Duyệt đăng ký nhà hàng

Quy trình này bao gồm các bước kiểm tra tính pháp lý (giấy phép kinh doanh), xác thực thực đơn dự kiến và cơ chế phản hồi (phê duyệt/từ chối/yêu cầu bổ sung) thông qua hệ thống email tự động. Điều này giúp đảm bảo tính minh bạch và chuyên nghiệp trong quy trình hợp tác.

### 2.2.7 Quy trình nghiệp vụ

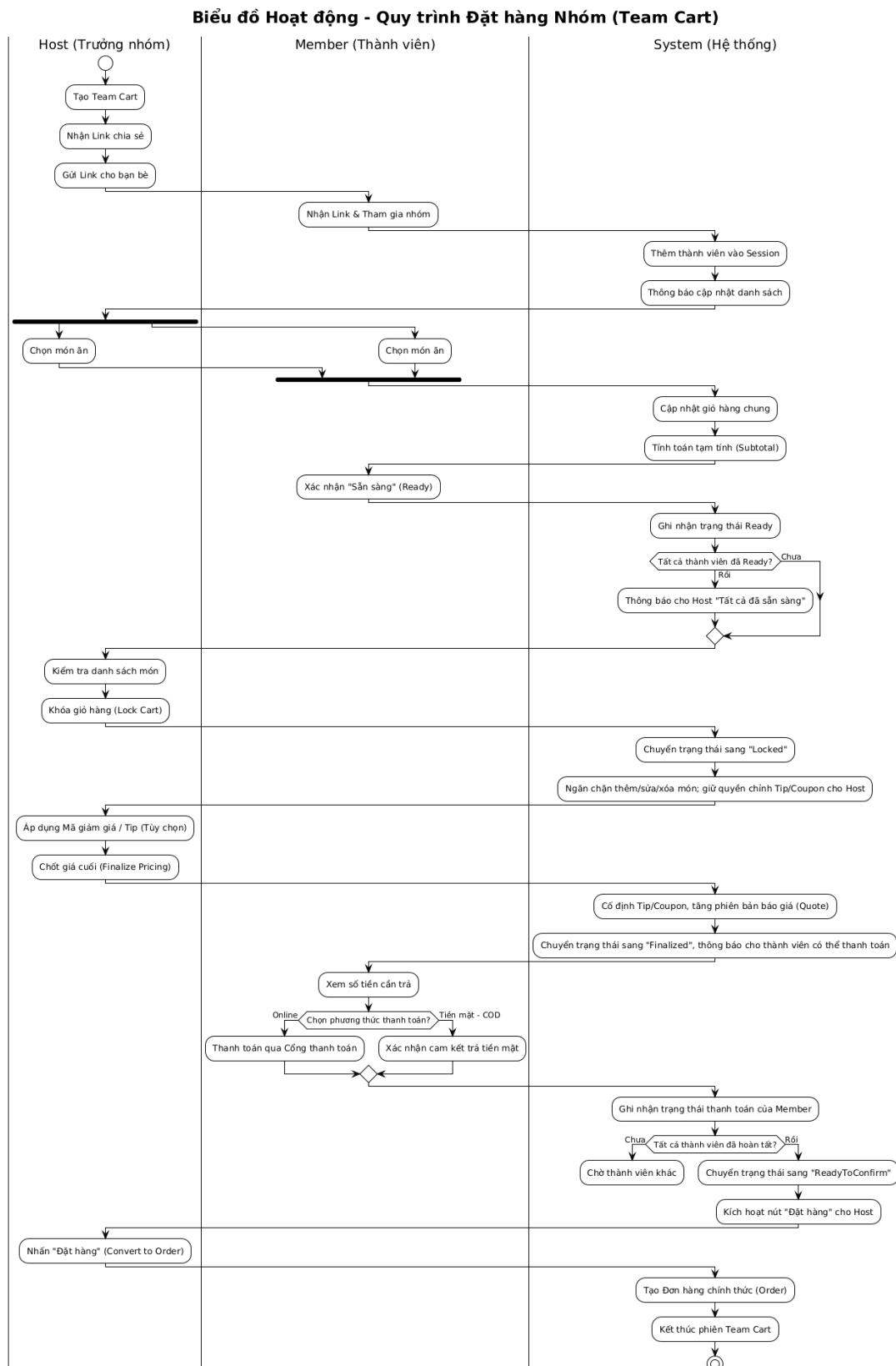
Hệ thống YummyZoom có nhiều quy trình nghiệp vụ, trong đó quy trình Xử lý Đặt hàng (Order Processing) là quan trọng và phức tạp nhất. Quy trình này đảm bảo tính chính xác về mặt dữ liệu (tồn kho, giá cả), tuân thủ các quy tắc kinh doanh (khuyến mãi, giờ hoạt động) và an toàn trong giao dịch tài chính.

Hình 2.7 dưới đây mô tả chi tiết các bước trong quy trình xử lý đặt hàng, từ lúc tiếp nhận yêu cầu, xác thực, tính toán tài chính cho đến khi hoàn tất đơn hàng.



Hình 2.7: Biểu đồ hoạt động - Quy trình xử lý đặt hàng

Ngoài ra, hình 2.8 minh họa luồng hoạt động của Team Cart, từ lúc Host khởi tạo, các thành viên tham gia chọn món, cho đến khi hoàn tất thanh toán và chốt đơn.



Hình 2.8: Biểu đồ hoạt động - Quy trình đặt hàng nhóm

### 2.3 Đặc tả chức năng

#### 2.3.1 Đặc tả Use Case: Đăng ký nhà hàng

<b>Mã use case</b>	UC-007	<b>Tên use case</b>	Đăng ký nhà hàng
<b>Tác nhân</b>	Khách hàng (Đối tác tiềm năng)		
<b>Mục đích sử dụng</b>	Cho phép người dùng gửi yêu cầu đăng ký mở nhà hàng mới trên hệ thống.		
<b>Sự kiện kích hoạt</b>	Người dùng nhấn nút "Đăng ký quán" trên giao diện.		
<b>Tiền kiện</b>	Người dùng đã đăng nhập và tài khoản đang hoạt động.		
<b>Luồng sự kiện chính</b>	STT	<b>Thực hiện bởi</b>	<b>Hành động</b>
	1	Khách hàng	Nhập thông tin nhà hàng (Tên, địa chỉ, liên hệ, giờ mở cửa...).
	2	Hệ thống	Kiểm tra tính hợp lệ của dữ liệu nhập vào.
	3	Hệ thống	Tạo hồ sơ đăng ký với trạng thái "Đã nộp" (Submitted).
	4	Hệ thống	Thông báo đăng ký thành công và chờ duyệt.
<b>Luồng sự kiện thay thế</b>	STT	<b>Thực hiện bởi</b>	<b>Hành động</b>
	2a	Hệ thống	Hiển thị thông báo lỗi nếu dữ liệu thiếu hoặc sai định dạng.
<b>Hậu điều kiện</b>	Hồ sơ đăng ký được lưu vào hệ thống, chờ Admin phê duyệt.		

Bảng 2.1: Đặc tả Use Case Đăng ký nhà hàng

Dữ liệu đầu vào cho use case Đăng ký nhà hàng:

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	Name	Tên hiển thị của quán	Có	Tối đa 100 ký tự
2	Description	Giới thiệu ngắn gọn	Có	Tối đa 500 ký tự
3	CuisineType	Ví dụ: Cơm, Phở, Trà sữa	Có	Tối đa 50 ký tự

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
4	Street	Số nhà, tên đường	Có	Tối đa 200 ký tự
5	City	Tên thành phố	Có	Tối đa 100 ký tự
6	State	Tên tỉnh hoặc bang	Có	Tối đa 100 ký tự
7	ZipCode	Zip Code	Có	Tối đa 20 ký tự
8	Country	Tên quốc gia	Có	Tối đa 100 ký tự
9	PhoneNumber	SĐT liên hệ của quán	Có	Tối đa 30 ký tự
10	Email	Email liên hệ	Có	Đúng định dạng Email
11	BusinessHours	Khung giờ hoạt động	Có	Tối đa 200 ký tự
12	LogoUrl	Đường dẫn ảnh đại diện	Không	URL hợp lệ
13	Latitude	Tọa độ địa lý	Không	-90 đến 90
14	Longitude	Tọa độ địa lý	Không	-180 đến 180

**Bảng 2.2:** Dữ liệu đầu vào Đăng ký nhà hàng

### 2.3.2 Đặc tả Use Case: Quản lý thực đơn

Mã use case	UC-008	Tên use case	Quản lý thực đơn
<b>Tác nhân</b>	Chủ nhà hàng (Restaurant Owner/Staff)		
<b>Mục đích sử dụng</b>	Cho phép nhà hàng tạo, cập nhật, xóa danh mục món ăn, món ăn và nhóm tùy chọn (size, topping) để xây dựng thực đơn hoàn chỉnh.		
<b>Sự kiện kích hoạt</b>	Chủ nhà hàng truy cập trang quản lý thực đơn và thực hiện các thao tác CRUD.		
<b>Tiền kiện</b>	Người dùng đã đăng nhập và có quyền quản lý nhà hàng (Owner hoặc Staff). Nhà hàng đã được duyệt và tồn tại trong hệ thống.		
<b>Luồng sự kiện chính</b>	STT	Thực hiện bởi	Hành động
	1	Chủ nhà hàng	Chọn chức năng quản lý danh mục/món ăn/nhóm tùy chọn.
	2	Chủ nhà hàng	Nhập thông tin chi tiết (tên, mô tả, giá, hình ảnh...).

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	3	Hệ thống	Kiểm tra tính hợp lệ của dữ liệu đầu vào (không rỗng, giá > 0, định dạng URL...).
	4	Hệ thống	Kiểm tra quyền sở hữu (danh mục/món phải thuộc nhà hàng đang quản lý).
	5	Hệ thống	Tạo/cập nhật thực thể trong cơ sở dữ liệu (MenuItem, MenuCategory).
	6	Hệ thống	Phát sự kiện domain (MenuItemCreated, MenuItemUpdated, MenuItemDeleted...).
	7	Hệ thống	Cập nhật read model FullMenuView để đồng bộ dữ liệu hiển thị.
	8	Hệ thống	Thông báo thành công và hiển thị danh sách cập nhật.
<b>Luồng sự kiện thay thế</b>	STT	<b>Thực hiện bởi</b>	<b>Hành động</b>
	3a	Hệ thống	Hiển thị lỗi nếu dữ liệu không hợp lệ (tên rỗng, giá âm, URL sai định dạng).
	4a	Hệ thống	Từ chối thao tác nếu món ăn/danh mục không thuộc nhà hàng (ForbiddenAccessException).
	5a	Hệ thống	Báo lỗi nếu danh mục không tồn tại khi tạo món ăn mới.
<b>Hậu điều kiện</b>	Thực đơn được cập nhật thành công. Thay đổi được phản ánh ngay lập tức trên giao diện khách hàng thông qua FullMenuView. Sự kiện domain được ghi lại phục vụ phân tích.		

**Bảng 2.3:** Đặc tả Use Case Quản lý thực đơn

Dữ liệu đầu vào cho use case Quản lý thực đơn (Tạo món ăn mới):

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	RestaurantId	ID nhà hàng	Có	GUID hợp lệ
2	MenuCategoryId	ID danh mục món ăn	Có	GUID hợp lệ, tồn tại
3	Name	Tên món ăn	Có	Không rỗng, không chỉ khoảng trắng
4	Description	Mô tả chi tiết món	Có	Không rỗng, không chỉ khoảng trắng
5	Price	Giá cơ bản	Có	Số thực > 0
6	Currency	Đơn vị tiền tệ	Có	Mã tiền tệ hợp lệ (VND, USD...)
7	ImageUrl	Đường dẫn ảnh món	Không	URL hợp lệ hoặc null
8	IsAvailable	Trạng thái còn hàng	Không	Boolean (mặc định: true)
9	DietaryTagIds	Danh sách tag dinh dưỡng	Không	Danh sách GUID (Vegetarian, Spicy...)

**Bảng 2.4:** Dữ liệu đầu vào Quản lý thực đơn - Tạo món ăn

### 2.3.3 Đặc tả Use Case: Tìm kiếm nhà hàng

<b>Mã use case</b>	UC-002	<b>Tên use case</b>	Tìm kiếm nhà hàng
<b>Tác nhân</b>	Khách hàng		
<b>Mục đích sử dụng</b>	Cho phép khách hàng tìm kiếm nhà hàng theo tên, món ăn, loại hình ẩm thực, vị trí địa lý và các bộ lọc khác (đánh giá, tag, khuyến mãi).		
<b>Sự kiện kích hoạt</b>	Khách hàng nhập từ khóa tìm kiếm hoặc chọn bộ lọc trên giao diện.		
<b>Tiền điều kiện</b>	Không yêu cầu đăng nhập. Hệ thống có dữ liệu nhà hàng đã được duyệt (IsVerified = true).		
<b>Luồng sự kiện chính</b>	STT	Thực hiện bởi	Hành động

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	1	Khách hàng	Nhập từ khóa tìm kiếm (tên nhà hàng/món ăn) hoặc chọn bộ lọc.
	2	Hệ thống	Validate dữ liệu đầu vào (độ dài, định dạng tọa độ, phạm vi giá trị).
	3	Hệ thống	Xây dựng câu truy vấn SQL với điều kiện lọc (WHERE clauses).
	4	Hệ thống	Tính toán khoảng cách (nếu có tọa độ) bằng công thức Haversine.
	5	Hệ thống	Sắp xếp kết quả theo tiêu chí (rating/distance/popularity/name).
	6	Hệ thống	Phân trang kết quả (PageNumber, PageSize).
	7	Hệ thống	Trả về danh sách nhà hàng và facets (nếu yêu cầu).
<b>Luồng sự kiện thay thế</b>	<b>STT</b>	<b>Thực hiện bởi</b>	<b>Hành động</b>
	2a	Hệ thống	Trả về lỗi validation nếu tham số không hợp lệ (PageSize > 50, tọa độ ngoài phạm vi...).
<b>Hậu điều kiện</b>	7a	Hệ thống	Trả về danh sách rỗng nếu không tìm thấy kết quả phù hợp.
			Danh sách nhà hàng được hiển thị với thông tin: tên, logo, loại ẩm thực, đánh giá, khoảng cách (nếu có). Facets (bộ lọc động) được cập nhật dựa trên kết quả hiện tại.

Bảng 2.5: Đặc tả Use Case Tìm kiếm nhà hàng

Dữ liệu đầu vào cho use case Tìm kiếm nhà hàng:

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	Q	Từ khóa tìm kiếm	Không	Tối đa 100 ký tự
2	Cuisine	Loại ẩm thực	Không	Tối đa 50 ký tự
3	Lat	Vĩ độ	Không	-90 đến 90
4	Lng	Kinh độ	Không	-180 đến 180
5	MinRating	Đánh giá tối thiểu	Không	0 đến 5
6	Sort	Tiêu chí sắp xếp	Không	rating hoặc distance hoặc popularity
7	Bbox	Bounding box (bản đồ)	Không	minLon, minLat, maxLon, maxLat
8	Tags	Danh sách tag (tên)	Không	Tối đa 20 tag, mỗi tag <= 100 ký tự
9	TagIds	Danh sách tag (ID)	Không	Tối đa 20 GUID
10	DiscountedOnly	Chỉ nhà hàng có KM	Không	Boolean
11	PageNumber	Số trang	Có	>= 1
12	PageSize	Kích thước trang	Có	1 đến 50
13	IncludeFacets	Bao gồm facets	Không	Boolean (mặc định: false)

**Bảng 2.6:** Dữ liệu đầu vào Tìm kiếm nhà hàng

### 2.3.4 ĐẶC TẢ USE CASE: ĐẶT HÀNG CÁ NHÂN

Mã use case	UC-003	Tên use case	Đặt hàng cá nhân
Tác nhân	Khách hàng		
Mục đích sử dụng	Cho phép khách hàng chọn món ăn, tùy chỉnh, áp dụng mã giảm giá và hoàn tất đơn hàng với thanh toán trực tuyến hoặc tiền mặt.		
Sự kiện kích hoạt	Khách hàng nhấn nút "Đặt hàng" sau khi đã chọn món và điền đầy đủ thông tin giao hàng.		
Tiền điều kiện	Khách hàng đã đăng nhập. Nhà hàng đang hoạt động. Giỏ hàng có ít nhất 1 món.		
Luồng sự kiện chính	STT	Thực hiện bởi	Hành động

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

1	Khách hàng	Chọn món ăn từ thực đơn, tùy chỉnh (size, topping) nếu có.
2	Khách hàng	Thêm món vào giỏ hàng (tối đa 50 món, mỗi món tối đa 10 phần).
3	Khách hàng	Chọn địa chỉ giao hàng từ danh sách hoặc nhập mới.
4	Khách hàng	Nhập mã coupon (tùy chọn) và số tiền tip (tùy chọn).
5	Hệ thống	Kiểm tra tính hợp lệ: nhà hàng hoạt động, món ăn còn hàng, thuộc đúng nhà hàng.
6	Hệ thống	Kiểm tra tùy chỉnh: nhóm tùy chỉnh được gán cho món, số lượng lựa chọn hợp lệ (min-max).
7	Hệ thống	Tính toán tài chính: Subtotal, Discount (nếu có coupon), DeliveryFee, Tax, Tip, TotalAmount.
8	Hệ thống	Kiểm tra và tăng số lần sử dụng coupon (nếu có) trong cùng transaction.
9	Khách hàng	Chọn phương thức thanh toán (CreditCard, Cash On Delivery (COD)).
10	Hệ thống	Tạo PaymentIntent (Stripe) nếu thanh toán online, lưu PaymentIntentId vào Order.
11	Hệ thống	Tạo Order với trạng thái PendingPayment (online) hoặc Placed (COD).
12	Hệ thống	Lưu Order vào database, phát sự kiện OrderPlaced.

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	13	Hệ thống	Trả về OrderId, OrderNumber, TotalAmount và ClientSecret (nếu online).
<b>Luồng sự kiện thay thế</b>	<b>STT</b>	<b>Thực hiện bởi</b>	<b>Hành động</b>
	5a	Hệ thống	Báo lỗi nếu nhà hàng không tồn tại hoặc không hoạt động.
	5b	Hệ thống	Báo lỗi nếu món ăn không tồn tại, không còn hàng hoặc không thuộc nhà hàng.
	6a	Hệ thống	Báo lỗi nếu nhóm tùy chỉnh không được gán cho món hoặc số lượng lựa chọn sai.
	8a	Hệ thống	Báo lỗi nếu coupon không hợp lệ, hết hạn hoặc đã hết lượt sử dụng.
	10a	Hệ thống	Báo lỗi nếu không tạo được PaymentIntent (lỗi Stripe API).
<b>Hậu kiện</b>	Đơn hàng được tạo thành công với trạng thái phù hợp. Sự kiện OrderPlaced được phát ra để thông báo cho nhà hàng. Nếu thanh toán online, khách hàng được chuyển đến trang thanh toán Stripe. Nếu COD, đơn hàng chuyển trạng thái Placed ngay lập tức.		

**Bảng 2.7:** Đặc tả Use Case Đặt hàng cá nhân

Dữ liệu đầu vào cho use case Đặt hàng cá nhân:

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	CustomerId	ID khách hàng	Có	GUID hợp lệ, trùng với user hiện tại
2	RestaurantId	ID nhà hàng	Có	GUID hợp lệ, nhà hàng tồn tại

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
3	Items	Danh sách món ăn	Có	Tối thiểu 1, tối đa 50 món
4	Items[].MenuItemId	ID món ăn	Có	GUID hợp lệ, món tồn tại
5	Items[].Quantity	Số lượng	Có	1 đến 10
6	Items[].Customizations	Danh sách tùy chỉnh	Không	Phụ thuộc món ăn
7	DeliveryAddress. Street	Số nhà, đường	Có	Tối đa 200 ký tự
8	DeliveryAddress. City	Thành phố	Có	Tối đa 100 ký tự
9	DeliveryAddress. State	Tỉnh/Bang	Có	Tối đa 100 ký tự
10	DeliveryAddress. ZipCode	Mã bưu điện	Có	Tối đa 20 ký tự
11	DeliveryAddress. Country	Quốc gia	Có	Tối đa 100 ký tự
12	PaymentMethod	Phương thức thanh toán	Có	CreditCard, COD
13	Special Instructions	Ghi chú đặc biệt	Không	Tối đa 500 ký tự
14	CouponCode	Mã giảm giá	Không	Tối đa 50 ký tự
15	TipAmount	Tiền tip	Không	$\geq 0$
16	TeamCartId	ID giỏ hàng nhóm	Không	GUID hợp lệ (nếu từ TeamCart)

**Bảng 2.8:** Dữ liệu đầu vào Đặt hàng cá nhân

### 2.3.5 Đặc tả Use Case: Khởi tạo TeamCart

<b>Mã use case</b>	UC-004a	<b>Tên use case</b>	Khởi tạo TeamCart
<b>Tác nhân</b>	Khách hàng (Host)		
<b>Mục đích sử dụng</b>	Cho phép khách hàng tạo một giỏ hàng nhóm (TeamCart) để mời những người khác cùng đặt món từ một nhà hàng cụ thể.		
<b>Sự kiện kích hoạt</b>	Khách hàng chọn chức năng "Đặt nhóm" (Group Order) trên giao diện chi tiết nhà hàng.		
<b>Tiền điều kiện</b>	Khách hàng đã đăng nhập. Nhà hàng tồn tại và đang hoạt động.		
<b>Luồng sự kiện chính</b>	STT	<b>Thực hiện bởi</b>	<b>Hành động</b>
	1	Khách hàng	Chọn nhà hàng và thiết lập thông tin giỏ nhóm (Tên hiển thị của Host, Thời gian chốt đơn).
	2	Hệ thống	Kiểm tra tính hợp lệ của dữ liệu (Tên host không rỗng, thời gian chốt đơn phải ở tương lai nếu có).
	3	Hệ thống	Tạo mới TeamCart với trạng thái Active. Tạo ShareToken duy nhất.
	4	Hệ thống	Thêm người tạo vào danh sách thành viên với vai trò Host.
	5	Hệ thống	Trả về thông tin TeamCart và Link chia sẻ (ShareToken).
<b>Luồng sự kiện thay thế</b>	STT	<b>Thực hiện bởi</b>	<b>Hành động</b>
	2a	Hệ thống	Báo lỗi nếu nhà hàng không tồn tại hoặc thời gian chốt đơn không hợp lệ (trong quá khứ).
<b>Hậu điều kiện</b>	TeamCart được tạo thành công. Host nhận được link chia sẻ để gửi cho các thành viên khác.		

**Bảng 2.9:** Đặc tả Use Case Khởi tạo TeamCart

Dữ liệu đầu vào cho use case Khởi tạo TeamCart:

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	RestaurantId	ID nhà hàng	Có	GUID hợp lệ, nhà hàng tồn tại
2	HostName	Tên hiển thị của Host	Có	Không rỗng, tối đa 200 ký tự
3	DeadlineUtc	Thời gian chốt đơn	Không	Thời gian trong tương lai (nếu có)

**Bảng 2.10:** Dữ liệu đầu vào Khởi tạo TeamCart

### 2.3.6 Đặc tả Use Case: Tham gia TeamCart

<b>Mã use case</b>	UC-004b	<b>Tên use case</b>	Tham gia TeamCart
<b>Tác nhân</b>	Khách hàng (Member)		
<b>Mục đích sử dụng</b>	Cho phép khách hàng tham gia vào một giỏ hàng nhóm thông qua liên kết chia sẻ để cùng đặt món.		
<b>Sự kiện kích hoạt</b>	Khách hàng truy cập vào đường dẫn chia sẻ (Share Link) của TeamCart.		
<b>Tiền điều kiện</b>	Khách hàng đã đăng nhập. TeamCart tồn tại, đang hoạt động (Active) và chưa bị khóa. Mã chia sẻ (ShareToken) hợp lệ.		
<b>Luồng sự kiện chính</b>	STT	<b>Thực hiện bởi</b>	<b>Hành động</b>
	1	Khách hàng	Nhập tên hiển thị (Guest-Name) để tham gia nhóm.
	2	Hệ thống	Kiểm tra sự tồn tại của TeamCart.
	3	Hệ thống	Xác thực mã chia sẻ (ShareToken).
	4	Hệ thống	Kiểm tra trạng thái TeamCart (phải là Active) và quyền tham gia (chưa tham gia trước đó).
	5	Hệ thống	Thêm người dùng vào danh sách thành viên với vai trò Guest.

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	6	Hệ thống	Thông báo thành công và chuyển hướng đến giao diện chi tiết TeamCart.
<b>Luồng sự kiện thay thế</b>	<b>STT</b>	<b>Thực hiện bởi</b>	<b>Hành động</b>
	2a	Hệ thống	Báo lỗi nếu TeamCart không tồn tại.
	3a	Hệ thống	Báo lỗi nếu mã chia sẻ không hợp lệ hoặc đã hết hạn.
	4a	Hệ thống	Báo lỗi nếu TeamCart đã bị khóa, đã chốt đơn hoặc người dùng đã là thành viên.
<b>Hậu kiện</b>	Người dùng trở thành thành viên của TeamCart, có quyền thêm món ăn vào giỏ chung.		

**Bảng 2.11:** Đặc tả Use Case Tham gia TeamCart

Dữ liệu đầu vào cho use case Tham gia TeamCart:

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	TeamCartId	ID giỏ hàng nhóm	Có	GUID hợp lệ, tồn tại
2	ShareToken	Mã chia sẻ	Có	Tối đa 100 ký tự, khớp với TeamCart
3	GuestName	Tên hiển thị	Có	Không rỗng, tối đa 200 ký tự

**Bảng 2.12:** Dữ liệu đầu vào Tham gia TeamCart

### 2.3.7 Đặc tả Use Case: Chốt đơn TeamCart

<b>Mã use case</b>	UC-004c	<b>Tên use case</b>	Chốt đơn TeamCart
<b>Tác nhân</b>	Khách hàng (Host)		
<b>Mục đích sử dụng</b>	Cho phép Host khóa giỏ hàng, chờ các thành viên thanh toán và hoàn tất việc chuyển đổi TeamCart thành đơn hàng chính thức.		
<b>Sự kiện kích hoạt</b>	Host nhấn nút "Đặt hàng" sau khi đã đủ điều kiện.		
<b>Tiền điều kiện</b>	TeamCart đang hoạt động (Active) và có món ăn. Host đã đăng nhập. Để thanh toán, giỏ phải được khóa và đang ở trạng thái Finalized. Để đặt hàng, giỏ phải ở trạng thái ReadyToConfirm.		
<b>Luồng sự kiện chính</b>	<b>STT</b>	<b>Thực hiện bởi</b>	<b>Hành động</b>
	1	Host	Chọn chức năng "Khóa đơn" để ngăn chặn việc thêm/bớt món.
	2	Hệ thống	Tính toán chi phí (Tổng tiền, Thuế, Phí vận chuyển), chia tiền cho từng thành viên và chuyển trạng thái sang "Locked".
	3	Host	Áp dụng/điều chỉnh Mã giảm giá hoặc Tip (tùy chọn) khi giỏ đang Locked.
	4	Host	Nhấn "Chốt giá" (Finalize Pricing) để cố định tip/-coupon.
	5	Hệ thống	Kiểm tra trạng thái Locked, cố định Tip/Coupon, tăng Quote Version, chuyển trạng thái sang "Finalized" và thông báo thành viên có thể thanh toán.
	6	Thành viên	Chọn phương thức thanh toán (Online hoặc COD) và thanh toán/commit phần của mình.

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	7	Hệ thống	Ghi nhận thanh toán; nếu chưa đủ thì chờ các thành viên khác. Khi đủ, chuyển trạng thái TeamCart sang "Ready-ToConfirm" và kích hoạt nút "Đặt hàng" cho Host.
	8	Host	Nhập thông tin giao hàng và xác nhận đặt hàng cuối cùng.
	9	Hệ thống	Kiểm tra tính nhất quán (Quote Version, trạng thái ReadyToConfirm, tổng tiền đã đóng).
	10	Hệ thống	Chốt hạn mức Coupon (nếu có), tạo Order và chuyển trạng thái TeamCart sang "Converted". Thông báo đặt hàng thành công cho tất cả thành viên.
<b>Luồng sự kiện thay thế</b>	<b>STT</b>	<b>Thực hiện bởi</b>	<b>Hành động</b>
	4a	Hệ thống	Từ chối chốt giá nếu TeamCart không ở trạng thái Locked (CannotFinalizePricingInCurrentStatus).
	6a	Hệ thống	Từ chối thanh toán nếu TeamCart chưa ở trạng thái Finalized (CanOnlyPayOnFinalizedCart).
	9a	Hệ thống	Báo lỗi nếu thông tin báo giá (Quote) đã thay đổi hoặc không khớp (xung đột Quote Version).
	9b	Hệ thống	Báo lỗi nếu tổng số tiền thanh toán từ các thành viên chưa đủ khớp với tổng đơn hàng.

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	10a	Hệ thống	Báo lỗi nếu Coupon hết lượt sử dụng tại thời điểm chót đơn.
<b>Hậu điều kiện</b>	Đơn hàng được tạo thành công trên hệ thống. TeamCart hoàn tất vòng đời và được chuyển đổi thành một đơn hàng bình thường, Host có thể theo dõi đơn hàng này.		

**Bảng 2.13:** Đặc tả Use Case Chốt đơn TeamCart

Dữ liệu đầu vào cho bước Chốt giá (Finalize Pricing):

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	TeamCartId	ID giỏ hàng nhóm	Có	GUID hợp lệ, trạng thái Locked
2	QuoteVersion	Phiên bản báo giá hiện tại	Có	Số nguyên không âm, khớp phiên bản mới nhất

**Bảng 2.14:** Dữ liệu đầu vào Chốt giá TeamCart

Dữ liệu đầu vào cho use case Chốt đơn TeamCart (Bước xác nhận cuối cùng):

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	TeamCartId	ID giỏ hàng nhóm	Có	GUID hợp lệ, trạng thái ReadyToConfirm
2	Street	Số nhà, tên đường	Có	Tối đa 200 ký tự
3	City	Thành phố	Có	Tối đa 100 ký tự
4	State	Tỉnh/Bang	Có	Tối đa 100 ký tự
5	ZipCode	Mã bưu điện	Có	Tối đa 20 ký tự
6	Country	Quốc gia	Có	Tối đa 100 ký tự

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
7	Special Instructions	Ghi chú đơn hàng	Không	Tối đa 500 ký tự
8	QuoteVersion	Phiên bản báo giá	Có	Khớp với phiên bản hiện tại của Cart

**Bảng 2.15:** Dữ liệu đầu vào Chốt đơn TeamCart

### 2.3.8 Đặc tả Use Case: Xử lý đơn hàng

Mã use case	UC-009	Tên use case	Xử lý đơn hàng
<b>Tác nhân</b>	Nhà hàng (Restaurant Staff)		
<b>Mục đích sử dụng</b>	Cho phép nhà hàng tiếp nhận, xử lý và cập nhật trạng thái đơn hàng theo quy trình từ lúc nhận đơn đến khi giao thành công.		
<b>Sự kiện kích hoạt</b>	Nhà hàng nhận được thông báo có đơn hàng mới (trạng thái Placed) trên giao diện quản lý.		
<b>Tiền đề</b>	Đơn hàng tồn tại và đang ở trạng thái chờ xử lý (Placed). Người dùng có quyền quản lý đơn hàng của nhà hàng.		
<b>Luồng sự kiện chính</b>	STT	Thực hiện bởi	Hành động
	1	Nhà hàng	Xem chi tiết đơn hàng mới và chọn "Chấp nhận" (Accept).
	2	Nhà hàng	Nhập thời gian giao hàng dự kiến (Estimated Delivery Time).
	3	Hệ thống	Kiểm tra trạng thái đơn hàng, cập nhật sang "Accepted" và thông báo cho khách hàng.
	4	Nhà hàng	Chuyển trạng thái sang "Đang chuẩn bị" (Preparing) khi bắt đầu chế biến.
	5	Hệ thống	Cập nhật trạng thái sang "Preparing" và thông báo tiến độ.

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	6	Nhà hàng	Chuyển trạng thái sang "Sẵn sàng giao" (ReadyForDelivery) khi món ăn hoàn tất.
	7	Hệ thống	Cập nhật trạng thái sang "ReadyForDelivery", thông báo cho tài xế hoặc khách hàng đến lấy.
	8	Nhà hàng	Xác nhận "Đã giao" (Delivered) khi đơn hàng được giao thành công cho khách.
	9	Hệ thống	Cập nhật trạng thái sang "Delivered", ghi nhận doanh thu và hoàn tất đơn hàng.
<b>Luồng sự kiện thay thế</b>	<b>STT</b>	<b>Thực hiện bởi</b>	<b>Hành động</b>
	1a	Nhà hàng	Chọn "Từ chối" (Reject) đơn hàng.
	1b	Nhà hàng	Nhập lý do từ chối (hết món, quá tải...).
	3a	Hệ thống	Cập nhật trạng thái sang "Rejected", hoàn tiền cho khách (nếu đã thanh toán) và gửi thông báo hủy.
<b>Hậu kiện</b>	Đơn hàng kết thúc ở trạng thái Delivered (thành công) hoặc Rejected (hủy bỏ). Lịch sử trạng thái được ghi lại đầy đủ.		

Bảng 2.16: ĐẶC TẢ USE CASE XỬ LÝ ĐƠN HÀNG

Dữ liệu đầu vào cho use case Xử lý đơn hàng (Các thao tác chính):

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	OrderId	ID đơn hàng	Có	GUID hợp lệ, tồn tại

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
2	Action	Hành động xử lý	Có	Accept, Reject, MarkPreparing, MarkReady, MarkDelivered
3	Estimated DeliveryTime	Thời gian giao dự kiến	Có (khi Accept)	Thời gian > Hiện tại
4	RejectionReason	Lý do từ chối	Có (khi Reject)	Tối đa 200 ký tự

**Bảng 2.17:** Dữ liệu đầu vào Xử lý đơn hàng

## 2.4 Yêu cầu phi chức năng

Bên cạnh các yêu cầu chức năng đã được mô tả chi tiết, hệ thống YummyZoom cần phải đáp ứng các yêu cầu phi chức năng nghiêm ngặt để đảm bảo chất lượng phần mềm, trải nghiệm người dùng và khả năng vận hành ổn định. Các yêu cầu này bao gồm hiệu năng, bảo mật, khả năng sử dụng, độ tin cậy và khả năng bảo trì.

### Hiệu năng

Hiệu năng là yếu tố then chốt ảnh hưởng trực tiếp đến trải nghiệm người dùng, đặc biệt đối với một ứng dụng giao đồ ăn có tính năng cộng tác thời gian thực. Về thời gian phản hồi, hệ thống được yêu cầu xử lý các thao tác đọc dữ liệu thông thường như xem danh sách nhà hàng hay chi tiết món ăn với độ trễ dưới 1 giây. Đối với các nghiệp vụ phức tạp hơn như tạo đơn hàng hoặc xử lý thanh toán, thời gian phản hồi chấp nhận được là dưới 3 giây. Đặc biệt, đối với tính năng TeamCart, sự đồng bộ hóa trạng thái giữa các thành viên (như thêm món, thay đổi số lượng) phải diễn ra gần như tức thời với độ trễ dưới 500 mili-giây để đảm bảo trải nghiệm cộng tác mượt mà. Trong phạm vi của một đồ án tốt nghiệp, hệ thống được thiết kế để chịu tải ổn định với khoảng 50 đến 100 người dùng truy cập đồng thời (CCU) mà không gặp phải sự cố tắc nghẽn hay suy giảm hiệu năng đáng kể. Mặc dù chưa thực hiện kiểm thử tải (load testing) quy mô lớn, các chỉ số này được coi là phù hợp và đủ để chứng minh tính khả thi của giải pháp trong môi trường thực tế quy mô nhỏ.

### Bảo mật

Bảo mật thông tin người dùng và dữ liệu giao dịch là ưu tiên hàng đầu của hệ thống. Cơ chế xác thực và phân quyền được xây dựng dựa trên nền tảng ASP.NET Identity, một giải pháp bảo mật mạnh mẽ và chuẩn hóa của Microsoft. Các thông tin nhạy cảm như mật khẩu người dùng được hệ thống tự động xử lý băm (hashing) và

thêm muối (salting) trước khi lưu trữ, đảm bảo an toàn tuyệt đối ngay cả khi cơ sở dữ liệu bị xâm nhập. Quá trình xác thực phiên làm việc sử dụng chuẩn JSON Web Token (JWT), cho phép xác thực không trạng thái (stateless) và tăng cường khả năng mở rộng của hệ thống. Về phân quyền, hệ thống áp dụng cơ chế Role-Based Access Control (RBAC) và Policy-Based Authorization, đảm bảo người dùng chỉ có thể truy cập vào các tài nguyên và chức năng phù hợp với vai trò của mình như Khách hàng, Chủ nhà hàng hoặc Quản trị viên. Ngoài ra, mọi giao tiếp giữa ứng dụng khách và máy chủ đều được mã hóa thông qua giao thức HTTPS để ngăn chặn các cuộc tấn công nghe lén.

### **Khả năng sử dụng**

Giao diện người dùng được thiết kế hướng tới sự nhất quán, trực quan và dễ sử dụng cho mọi đối tượng. Đối với ứng dụng di động dành cho khách hàng, hệ thống tuân thủ nghiêm ngặt các nguyên tắc thiết kế Material Design của Google. Các thành phần giao diện như nút bấm, biểu tượng, điều hướng và bộ cục đều được chuẩn hóa, tạo cảm giác quen thuộc và dễ dàng thao tác cho người dùng Android. Đối với ứng dụng web dành cho nhà hàng và quản trị viên, giao diện được xây dựng dựa trên hệ thống thành phần (component system) của thư viện PrimeNG. Việc sử dụng PrimeNG không chỉ đảm bảo tính đồng nhất về mặt thẩm mỹ trên toàn bộ các trang quản lý mà còn cung cấp các tính năng tương tác cao cấp như bảng dữ liệu, biểu đồ và thông báo. Hệ thống cũng chú trọng đến việc cung cấp phản hồi (feedback) rõ ràng cho người dùng thông qua các thông báo thành công, cảnh báo hoặc lỗi một cách thân thiện và dễ hiểu.

### **Độ tin cậy và Tính toàn vẹn dữ liệu**

Hệ thống phải đảm bảo tính chính xác và nhất quán của dữ liệu trong mọi tình huống, đặc biệt là trong các giao dịch tài chính và đặt hàng. Tính chất nguyên tử (Atomicity) và nhất quán (Consistency) của các giao dịch cơ sở dữ liệu (ACID (Atomicity, Consistency, Isolation, Durability)) được tuân thủ nghiêm ngặt, đảm bảo rằng một đơn hàng chỉ được tạo ra khi tất cả các bước xử lý liên quan đều thành công. Điều này đặc biệt quan trọng đối với tính năng TeamCart, nơi quy trình thanh toán phân tán yêu cầu sự phối hợp chính xác trạng thái thanh toán của nhiều thành viên trước khi chốt đơn. Hệ thống cũng được trang bị cơ chế xử lý lỗi toàn cục và ghi nhật ký (logging) chi tiết, giúp phát hiện sớm các bất thường và hỗ trợ đội ngũ phát triển trong việc chẩn đoán và khắc phục sự cố mà không làm gián đoạn trải nghiệm của người dùng cuối.

### **Khả năng bảo trì**

Để đảm bảo khả năng phát triển lâu dài và dễ dàng nâng cấp, mã nguồn hệ thống được tổ chức khoa học theo kiến trúc Clean Architecture kết hợp với các nguyên lý của DDD. Việc phân chia rõ ràng các lớp trách nhiệm giúp tách biệt logic nghiệp vụ cốt lõi khỏi các chi tiết kỹ thuật hạ tầng, cho phép thay đổi công nghệ hoặc cơ sở dữ liệu mà không ảnh hưởng đến toàn bộ hệ thống. Mã nguồn tuân thủ các quy tắc Clean Code và các nguyên lý thiết kế hướng đối tượng (SOLID), giúp mã dễ đọc, dễ hiểu và dễ kiểm thử. Ngoài ra, việc sử dụng nền tảng công nghệ .NET 9 hiện đại cũng mang lại lợi thế về hiệu năng và sự hỗ trợ lâu dài từ cộng đồng phát triển.

### **Kết chương**

Chương 2 đã trình bày chi tiết về quá trình khảo sát và phân tích yêu cầu hệ thống YummyZoom. Từ việc nghiên cứu bối cảnh thị trường và phân tích các đối thủ cạnh tranh như GrabFood và ShopeeFood, đồ án đã xác định được nhu cầu cấp thiết về một giải pháp đặt hàng nhôm với cơ chế thanh toán phân tán. Các yêu cầu chức năng đã được mô hình hóa rõ ràng thông qua hệ thống biểu đồ use case và đặc tả quy trình nghiệp vụ. Đồng thời, các yêu cầu phi chức năng về hiệu năng, bảo mật và khả năng sử dụng cũng được thiết lập làm cơ sở cho việc phát triển hệ thống.

## CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

Chương này trình bày chi tiết về các công nghệ, phong cách kiến trúc và các giải pháp kỹ thuật được lựa chọn để xây dựng hệ thống YummyZoom. Các quyết định công nghệ được đưa ra dựa trên việc phân tích kỹ lưỡng các yêu cầu chức năng và phi chức năng đã được khảo sát, đặc biệt tập trung giải quyết các bài toán về hiệu năng xử lý đồng thời, tính toàn vẹn dữ liệu trong giao dịch tài chính và trải nghiệm người dùng mượt mà trên đa nền tảng.

### 3.1 Kiến trúc hệ thống và Ứng dụng Backend

Hệ thống Backend đóng vai trò là xương sống của toàn bộ ứng dụng YummyZoom, chịu trách nhiệm xử lý các nghiệp vụ cốt lõi, quản lý dữ liệu và điều phối các tương tác giữa các phân hệ. Để đáp ứng các yêu cầu khắt khe về khả năng bảo trì (Maintainability) và độ tin cậy (Reliability), việc lựa chọn kiến trúc phần mềm và nền tảng phát triển phù hợp là yếu tố tiên quyết.

#### 3.1.1 Kiến trúc Clean Architecture và Domain-Driven Design (DDD)

Trong bối cảnh phát triển tính năng TeamCart (Giỏ hàng nhóm), hệ thống phải đổi mới với các nghiệp vụ phức tạp như đồng bộ trạng thái giữa nhiều người dùng, tính toán chia sẻ hóa đơn và xử lý các giao dịch đồng thời. Để giải quyết các thách thức này, đồ án áp dụng kiến trúc Clean Architecture (Kiến trúc sạch) kết hợp với các nguyên lý của Domain-Driven Design (DDD) [3], [4].

Về mặt cấu trúc, hệ thống được tổ chức thành các lớp với quy tắc phụ thuộc hướng tâm, đảm bảo sự độc lập của logic nghiệp vụ. Lớp trong cùng là lớp miền (Domain Layer), chứa các thực thể (Entities), đối tượng giá trị (Value Objects) và các quy tắc nghiệp vụ bất biến. Ví dụ, logic tính toán tổng tiền của một giỏ hàng nhóm hay quy tắc khóa đơn hàng khi thanh toán được cài đặt tại đây, hoàn toàn không phụ thuộc vào cơ sở dữ liệu hay giao diện người dùng. Bao bọc bên ngoài là lớp ứng dụng (Application Layer), đóng vai trò điều phối các ca sử dụng (Use Cases), nhận yêu cầu từ bên ngoài và sử dụng các thực thể trong Domain để thực hiện tác vụ. Tiếp theo là lớp hạ tầng (Infrastructure Layer), nơi cài đặt chi tiết các giao tiếp kỹ thuật như truy xuất cơ sở dữ liệu hay tích hợp cổng thanh toán. Cuối cùng, lớp giao diện (Presentation Layer) chứa các API Controller để tiếp nhận yêu cầu từ ứng dụng di động hoặc web quản trị.

Sự kết hợp này mang lại lợi ích vượt trội so với kiến trúc phân tầng truyền thống hay kiến trúc Microservices thuần túy. Nó loại bỏ sự phụ thuộc chặt chẽ giữa logic nghiệp vụ và cơ sở dữ liệu, giúp mã nguồn ổn định và dễ dàng kiểm thử đơn vị.

Đồng thời, việc áp dụng mô hình Monolithic Modular (Đơn khối module hóa) giúp cân bằng giữa tốc độ phát triển và khả năng mở rộng, phù hợp với quy mô nhân sự và tài nguyên của một đồ án tốt nghiệp.

### **3.1.2 Nền tảng phát triển: .NET 9 và ASP.NET Core**

Để hiện thực hóa kiến trúc trên, hệ thống sử dụng nền tảng .NET 9 cùng framework ASP.NET Core Web API [5]. Đây là một trong những nền tảng phát triển web hiệu năng cao và ổn định nhất hiện nay. Lựa chọn .NET 9 giải quyết trực tiếp bài toán về tối ưu hóa hiệu năng và khả năng chịu tải nhờ máy chủ web Kestrel mạnh mẽ và mô hình lập trình bất đồng bộ (Async/Await). Hệ thống có thể xử lý hiệu quả hàng nghìn yêu cầu mỗi giây mà không làm tắc nghẽn luồng xử lý chính, đảm bảo thời gian phản hồi luôn duy trì ở mức thấp ngay cả khi có lượng lớn người dùng thao tác cùng lúc trên TeamCart.

Bên cạnh đó, ngôn ngữ C# với đặc tính định kiểu tĩnh mạnh (Strongly Typed) giúp phát hiện phần lớn các lỗi logic ngay trong quá trình biên dịch, giảm thiểu đáng kể lỗi runtime và tăng cường khả năng bảo trì so với các ngôn ngữ định kiểu động như JavaScript hay Python. So với các nền tảng khác như Node.js hay Java Spring Boot, .NET 9 mang lại sự cân bằng tốt hơn giữa hiệu năng xử lý đa luồng và mức tiêu thụ tài nguyên hệ thống.

### **3.1.3 Điều phối hệ thống và Khả năng quan sát: .NET Aspire**

Để giải quyết bài toán phức tạp trong việc quản lý, kết nối và vận hành các thành phần phân tán ngay từ môi trường phát triển, dự án áp dụng stack công nghệ .NET Aspire [6]. Công nghệ này đóng vai trò điều phối tài nguyên (Orchestration), cho phép định nghĩa toàn bộ hạ tầng bằng mã nguồn C# thay vì cấu hình thủ công từng container. Chỉ với một thao tác khởi động, toàn bộ hệ sinh thái bao gồm Backend, Database và Cache sẽ được khởi tạo và kết nối tự động. Ngoài ra, .NET Aspire còn cung cấp khả năng quan sát (Observability) mạnh mẽ với Dashboard tích hợp sẵn, hiển thị tập trung logs, metrics và traces theo chuẩn OpenTelemetry, giúp đội ngũ phát triển dễ dàng theo dõi luồng hoạt động và gỡ lỗi hệ thống.

## **3.2 Cơ sở dữ liệu và Lưu trữ**

Việc lưu trữ và quản lý dữ liệu đóng vai trò then chốt trong việc đảm bảo tính chính xác của các giao dịch thương mại điện tử cũng như tốc độ phản hồi của ứng dụng. Hệ thống sử dụng chiến lược lưu trữ kết hợp giữa Cơ sở dữ liệu quan hệ (RDBMS) và Bộ nhớ đệm (Caching) để giải quyết hài hòa bài toán về tính toàn vẹn dữ liệu và hiệu năng truy xuất.

### 3.2.1 Hệ quản trị cơ sở dữ liệu quan hệ: PostgreSQL

Dự án lựa chọn PostgreSQL 16 làm hệ quản trị cơ sở dữ liệu chính, kết hợp với Entity Framework Core (EF Core) đóng vai trò là lớp Object-Relational Mapping (ORM) để tương tác dữ liệu [7]. Đây là giải pháp lưu trữ bền vững, được thiết kế để giải quyết triệt để vấn đề toàn vẹn dữ liệu cho các tính năng nhạy cảm như thanh toán và quản lý đơn hàng. PostgreSQL đảm bảo tính ACID tuyệt đối, giúp ngăn chặn tình trạng sai lệch dữ liệu tài chính trong các kịch bản xử lý đồng thời phức tạp của TeamCart.

Ngoài ra, PostgreSQL còn cung cấp bộ tính năng nâng cao tích hợp sẵn giúp giải quyết trọn vẹn các bài toán nghiệp vụ đặc thù mà không cần phụ thuộc vào dịch vụ bên thứ ba. Hệ thống tận dụng engine tìm kiếm toàn văn (Full Text Search) để hỗ trợ tìm kiếm tiếng Việt hiệu quả, sử dụng PostGIS để xử lý các truy vấn không gian phục vụ tìm kiếm nhà hàng theo vị trí, và sử dụng kiểu dữ liệu JSONB để lưu trữ linh hoạt cấu trúc menu phức tạp. So với MySQL hay SQL Server, PostgreSQL mang lại ưu thế rõ rệt về khả năng xử lý các truy vấn phức tạp và chi phí vận hành tối ưu nhờ mã nguồn mở.

### 3.2.2 Bộ nhớ đệm (Caching): Redis

Bên cạnh cơ sở dữ liệu chính, hệ thống triển khai Redis đóng vai trò là bộ nhớ đệm phân tán (Distributed Cache). Mục tiêu của Redis là giảm tải cho PostgreSQL và đảm bảo tốc độ xử lý thời gian thực cho các tính năng tương tác cao. Cụ thể, Redis được sử dụng để lưu trữ các dữ liệu tĩnh ít thay đổi như danh sách nhà hàng và thực đơn, giúp các truy vấn điều hướng đạt độ trễ cực thấp. Quan trọng hơn, Redis đóng vai trò quản lý trạng thái cho các TeamCart đang hoạt động. Khi các thành viên trong nhóm thao tác thêm bớt món đồ ăn, hệ thống đọc ghi trực tiếp vào Redis trên RAM thay vì đĩa cứng, đảm bảo trải nghiệm người dùng mượt mà tuyệt đối. Dữ liệu chỉ được đồng bộ xuống PostgreSQL để lưu trữ bền vững khi đơn hàng được chốt.

### 3.2.3 Quản lý và Phân phối đa phương tiện: Cloudinary

Để giải quyết bài toán lưu trữ và hiển thị lượng lớn hình ảnh chất lượng cao mà không gây áp lực lên băng thông máy chủ, dự án sử dụng nền tảng Cloudinary [8]. Cloudinary tự động nén và chuyển đổi định dạng ảnh phù hợp với thiết bị người dùng (ví dụ WebP, AVIF), giúp giảm dung lượng tải đáng kể. Đồng thời, hệ thống mạng phân phối nội dung (CDN) toàn cầu của Cloudinary giúp hình ảnh được tải nhanh chóng từ máy chủ gần người dùng nhất, mang lại trải nghiệm lướt thực đơn mượt mà trên ứng dụng di động.

### 3.3 Xác thực và Phân quyền (Authentication & Authorization)

Bảo mật thông tin người dùng và kiểm soát quyền truy cập là ưu tiên hàng đầu của hệ thống. YummyZoom áp dụng các tiêu chuẩn an ninh hiện đại để bảo vệ dữ liệu, đồng thời đảm bảo tính tiện lợi cho người dùng cuối.

#### 3.3.1 Cơ chế xác thực: JSON Web Token (JWT)

Hệ thống sử dụng tiêu chuẩn JSON Web Token (JWT) làm phương thức xác thực chính [9]. Cơ chế xác thực không trạng thái (Stateless Authentication) này phù hợp hoàn hảo với mô hình ứng dụng di động quy mô lớn. JWT đóng gói toàn bộ thông tin định danh vào một chuỗi token duy nhất gửi kèm theo mỗi yêu cầu, loại bỏ nhu cầu lưu trữ session trên server. Điều này giúp hệ thống dễ dàng mở rộng theo chiều ngang và giảm tải đáng kể cho tầng lưu trữ, tối ưu hóa thời gian phản hồi API.

#### 3.3.2 Quản lý định danh và Phân quyền: ASP.NET Core Identity

Để quản lý vòng đời tài khoản và phân quyền, hệ thống tích hợp framework ASP.NET Core Identity. Đây là giải pháp bảo mật toàn diện giúp quản lý đăng ký, đăng nhập và bảo mật mật khẩu người dùng bằng các thuật toán băm hiện đại. Hệ thống thiết lập cơ chế kiểm soát truy cập dựa trên vai trò (RBAC) với các quyền hạn cụ thể cho Khách hàng, Nhà hàng và Quản trị viên. Các API Endpoint được bảo vệ nghiêm ngặt bằng cách kiểm tra quyền (Claims) có trong JWT, đảm bảo ngăn chặn mọi truy cập trái phép.

## 3.4 Ứng dụng dành cho khách hàng (Mobile App)

Ứng dụng di động là điểm tiếp xúc chính giữa hệ thống và người dùng cuối, được xây dựng với mục tiêu mang lại trải nghiệm người dùng (UX) mượt mà và hiệu năng cao nhất.

#### 3.4.1 Nền tảng phát triển: Flutter & Dart

Dự án lựa chọn Flutter Software Development Kit (SDK) và ngôn ngữ Dart để phát triển ứng dụng di động [10]. Flutter cho phép xây dựng ứng dụng biên dịch natively, sử dụng engine đồ họa riêng để vẽ giao diện trực tiếp, đảm bảo tốc độ khung hình ổn định ở mức 60fps. Điều này cực kỳ quan trọng đối với các thao tác cuộn danh sách dài hay các hiệu ứng chuyển động phức tạp trong tính năng TeamCart. Khả năng kiểm soát giao diện chính xác đến từng pixel giúp hiện thực hóa các thiết kế độc đáo mà không bị giới hạn bởi các widget mặc định của hệ điều hành. Ngoài ra, tính năng Hot Reload giúp tăng tốc đáng kể quy trình phát triển và tinh chỉnh giao diện.

### 3.4.2 Quản lý trạng thái và Tương tác API

Để xử lý luồng dữ liệu phức tạp của TeamCart, ứng dụng sử dụng thư viện Provider để quản lý trạng thái. Giải pháp này giúp tách biệt logic nghiệp vụ khỏi giao diện, giải quyết vấn đề truyền dữ liệu qua nhiều cấp và giữ cho cấu trúc mã nguồn gọn gàng, dễ bảo trì. Việc giao tiếp mạng được quản lý bởi thư viện Dio, hỗ trợ các kịch bản nâng cao như tự động hủy request, xử lý timeout và chặn bắt lỗi tập trung, đảm bảo ứng dụng luôn hoạt động ổn định trong mọi điều kiện mạng.

### 3.4.3 Tích hợp bản đồ và Thanh toán

Hệ thống tích hợp Mapbox SDK cho các chức năng bản đồ và định vị, mang lại trải nghiệm tương tác mượt mà và tối ưu chi phí phát triển [11]. Về thanh toán, ứng dụng sử dụng cổng thanh toán Stripe thông qua SDK chính thức [12]. Stripe cung cấp môi trường kiểm thử (Sandbox) toàn diện, cho phép đội ngũ phát triển mô phỏng an toàn mọi kịch bản thanh toán từ thành công, thất bại đến hoàn tiền trước khi triển khai thực tế.

## 3.5 Ứng dụng Web quản trị (Admin & Restaurant Portal)

Cổng thông tin quản trị dành cho Đối tác nhà hàng và Quản trị viên được xây dựng dưới dạng Single Page Application (SPA), tập trung vào hiệu suất xử lý dữ liệu và trải nghiệm làm việc chuyên nghiệp.

### 3.5.1 Nền tảng Frontend: Angular

Dự án sử dụng framework Angular (phiên bản mới nhất) kết hợp với ngôn ngữ TypeScript [13]. Kiến trúc module chặt chẽ và hệ thống quản lý phụ thuộc (Dependency Injection) của Angular rất phù hợp với tư duy thiết kế hướng đối tượng, giúp đồng bộ kiến thức với đội ngũ Backend. Các công cụ tích hợp sẵn như HttpClient, Reactive Forms và Router giúp xây dựng ứng dụng quản trị ổn định, bảo mật và dễ bảo trì, vượt trội hơn so với các thư viện UI tự do trong bối cảnh ứng dụng doanh nghiệp.

### 3.5.2 Thư viện giao diện: PrimeNG và Tailwind CSS

Giao diện quản trị được xây dựng dựa trên sự kết hợp giữa PrimeNG và Tailwind CSS [14]. PrimeNG cung cấp bộ sưu tập phong phú các thành phần UI cao cấp chuyên dụng cho quản lý dữ liệu như bảng dữ liệu (Data Grid) với khả năng lọc, sắp xếp, phân trang mạnh mẽ và các biểu đồ thống kê trực quan. Tailwind CSS đóng vai trò framework tiện ích giúp tùy biến nhanh bối cảnh và đảm bảo tính đáp ứng (Responsive) trên nhiều kích thước màn hình.

### 3.6 Cộng tác thời gian thực (Realtime Collaboration)

Khả năng tương tác thời gian thực là "trái tim" của tính năng TeamCart, đảm bảo trải nghiệm đồng bộ tức thì giữa các thành viên trong nhóm với độ trễ tối thiểu.

#### 3.6.1 Giao thức và Framework: SignalR

Hệ thống sử dụng thư viện ASP.NET Core SignalR với giao thức WebSockets làm nòng cốt [15]. SignalR cho phép server đẩy (push) dữ liệu cập nhật xuống client ngay lập tức thay vì chờ client gửi yêu cầu. Điều này giúp các thay đổi trong TeamCart như thêm món, xóa món được đồng bộ gần như tức thời đến tất cả thành viên, tạo cảm giác cộng tác mượt mà. SignalR cũng hỗ trợ cơ chế tự động điều phối, tự động chuyển đổi sang các giao thức thay thế như Server-Sent Events hay Long Polling nếu môi trường mạng của người dùng không hỗ trợ WebSockets, đảm bảo kết nối luôn ổn định.

#### 3.6.2 Thông báo đẩy tới thiết bị di động: Firebase Cloud Messaging (FCM)

Để bổ trợ cho SignalR trong các trường hợp ứng dụng chạy nền hoặc đã tắt, hệ thống tích hợp dịch vụ Firebase Cloud Messaging (FCM) [16]. FCM sử dụng cơ chế thông báo tiêu chuẩn của hệ điều hành (APNs/GCM) để đánh thức thiết bị và gửi các thông tin quan trọng như cập nhật trạng thái đơn hàng hay lời mời tham gia nhóm. Sự kết hợp giữa SignalR (Online Realtime) và FCM (Background Notification) tạo nên một hệ thống thông báo toàn diện, đảm bảo người dùng không bao giờ bỏ lỡ các thông tin quan trọng từ YummyZoom.

### 3.7 Nền tảng triển khai và phân phối hệ thống

Bên cạnh lựa chọn công nghệ phát triển, cách thức triển khai và phân phối cũng ảnh hưởng trực tiếp đến khả năng vận hành ổn định, tốc độ phát hành phiên bản và mức độ tái lập môi trường. Đồ án lựa chọn mô hình triển khai kết hợp, trong đó phần Backend được triển khai trên nền tảng đám mây Azure theo hướng hạ tầng dưới dạng mã (Infrastructure as Code (IaC) – IaC), ứng dụng di động khách hàng được đóng gói và phân phối dưới dạng tệp Android Package Kit (APK) để phục vụ thử nghiệm và đánh giá, còn cổng web quản trị được triển khai trên nền tảng Vercel nhằm tối ưu hóa quy trình phát hành giao diện.

#### 3.7.1 Triển khai Backend trên Azure bằng Azure Developer CLI (azd) và .NET Aspire

Phần Backend được triển khai thông qua Azure Developer CLI (Azure Developer CLI (azd)), một công cụ dòng lệnh hỗ trợ chuẩn hóa cấu hình môi trường, quy trình cấp phát tài nguyên và triển khai ứng dụng [17]. Trong bối cảnh hệ thống có nhiều thành phần phụ thuộc như cơ sở dữ liệu, bộ nhớ đệm và quản trị bí mật, cách

tiếp cận này giúp giảm sai lệch cấu hình giữa các môi trường và tạo khả năng tái lập khi cần khởi tạo lại toàn bộ tài nguyên. Đồng thời, đồ án sử dụng .NET Aspire như một lớp điều phối ở mức ứng dụng, cho phép mô tả đồ thị tài nguyên và phụ thuộc (ví dụ giữa dịch vụ web, PostgreSQL và Redis) theo cách thống nhất giữa môi trường phát triển và môi trường triển khai [6].

Về hạ tầng, azd kết hợp với Bicep để mô tả tài nguyên Azure theo hướng IaC [18]. Mô hình này giúp việc cấp phát tài nguyên trở nên có kiểm soát, có thể xem xét, và dễ dàng tự động hóa trong quy trình tích hợp và triển khai liên tục (Continuous Integration/Continuous Delivery (CI/CD)). Dịch vụ web được triển khai trên Azure Container Apps [19], phù hợp với đặc thù ứng dụng web hiện đại chạy bằng container, đồng thời hỗ trợ quản lý phiên bản, mở rộng theo tải và tích hợp quan sát hệ thống. Thông tin nhạy cảm và cấu hình bí mật được quản lý tập trung bằng Azure Key Vault [20], giúp hạn chế việc lưu trữ bí mật trực tiếp trong mã nguồn và tăng cường an toàn khi triển khai.

### **3.7.2 Tự động hóa triển khai bằng GitHub Actions và cơ chế xác thực OIDC**

Để hỗ trợ phát hành phiên bản lặp lại, đồ án sử dụng GitHub Actions như nền tảng thực thi quy trình CI/CD [21]. Điểm quan trọng trong quy trình này là cơ chế đăng nhập Azure theo hướng "không dùng bí mật dài hạn" (secretless) thông qua OpenID Connect (OpenID Connect (OIDC)), cho phép workflow trên GitHub nhận định danh tạm thời để truy cập Azure thay vì phải lưu khóa truy cập cố định [22]. Cách tiếp cận này giúp giảm rủi ro rò rỉ thông tin xác thực trong kho mã nguồn và phù hợp với yêu cầu bảo mật khi triển khai trên môi trường đám mây.

### **3.7.3 Phân phối ứng dụng di động bằng tệp APK**

Ứng dụng di động dành cho khách hàng hiện được đóng gói ở dạng tệp cài đặt APK và phân phối trực tiếp cho mục đích thử nghiệm, thay vì phát hành qua các kho ứng dụng. Hình thức phân phối này giúp rút ngắn vòng lặp kiểm thử – phản hồi trong giai đoạn phát triển đồ án, đồng thời cho phép kiểm soát chặt chẽ phiên bản đang được đánh giá trên một nhóm người dùng giới hạn. Quy trình đóng gói và chuẩn bị phát hành APK tuân theo hướng dẫn chính thức từ Android Developers [23].

### **3.7.4 Triển khai cổng web quản trị trên Vercel**

Cổng web quản trị được triển khai trên Vercel [24], một nền tảng tối ưu cho các ứng dụng web hiện đại với khả năng xây dựng (build) và phát hành tự động từ kho mã nguồn. Trong bối cảnh cổng quản trị là ứng dụng SPA, việc triển khai trên Vercel giúp đơn giản hóa việc phân phối tài nguyên tĩnh, đồng thời hỗ trợ cơ chế

triển khai theo nhánh và xem trước (preview) để kiểm thử giao diện trước khi phát hành chính thức. Cách lựa chọn này giúp tách biệt vòng đời triển khai của giao diện quản trị khỏi Backend, giảm phụ thuộc khi cần phát hành các thay đổi giao diện nhanh chóng.

## Kết chương

Chương 3 đã giới thiệu các công nghệ và giải pháp kỹ thuật được sử dụng để hiện thực hóa hệ thống YummyZoom, bao gồm kiến trúc Backend, nền tảng phát triển ứng dụng di động, công nghệ xây dựng web quản trị, cũng như các cơ chế hỗ trợ cộng tác thời gian thực. Việc lựa chọn công nghệ được đặt trong bối cảnh yêu cầu của bài toán đặt hàng nhóm, đảm bảo cân bằng giữa hiệu năng, độ tin cậy, khả năng bảo trì và trải nghiệm người dùng. Đồng thời, chương này cũng mô tả các nền tảng triển khai và phân phối cho từng thành phần nhằm đảm bảo hệ thống có thể vận hành ổn định và tái lập môi trường khi cần. Trên cơ sở các lựa chọn công nghệ đó, chương tiếp theo sẽ trình bày quá trình hiện thực, kiểm thử và các kết quả thực nghiệm đạt được của hệ thống.

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

### 4.1 Thiết kế kiến trúc

#### 4.1.1 Lựa chọn kiến trúc phần mềm

Dựa trên các yêu cầu nghiệp vụ phức tạp của hệ thống YummyZoom, đặc biệt là tính năng đặt hàng nhóm (TeamCart) với yêu cầu về tính nhất quán dữ liệu cao và khả năng xử lý đồng thời, đồ án lựa chọn áp dụng kiến trúc Clean Architecture (Kiến trúc Sạch) kết hợp với tư duy thiết kế Domain-Driven Design (DDD). Về mặt triển khai, hệ thống được xây dựng theo mô hình Monolithic Modular (Đơn khối module hóa).

##### a, Mô hình kiến trúc tổng thể

Thay vì lựa chọn kiến trúc Microservices ngay từ đầu - vốn đòi hỏi chi phí vận hành và quản lý hạ tầng lớn, hay kiến trúc Layered (3 lớp) truyền thống dễ gây ra sự phụ thuộc chặt chẽ vào cơ sở dữ liệu, tôi quyết định sử dụng mô hình Monolithic Modular. Mô hình này giúp cân bằng giữa tốc độ phát triển và khả năng mở rộng. Toàn bộ mã nguồn nằm trong một giải pháp (Solution) thống nhất, giúp việc gỡ lỗi (debug), kiểm thử và tái cấu trúc (refactor) trở nên dễ dàng hơn. Đồng thời, các ranh giới nghiệp vụ (Modular Boundaries) được tổ chức độc lập về mặt logic dựa trên các Ngũ cảnh giới hạn (Bounded Contexts). Điều này tạo tiền đề vững chắc để tách thành các Microservices riêng biệt khi hệ thống cần mở rộng quy mô trong tương lai mà không làm phá vỡ cấu trúc hiện tại. Hơn nữa, Clean Architecture giúp cô lập logic nghiệp vụ (Domain) khỏi các yếu tố thay đổi thường xuyên như giao diện người dùng (UI) hay công nghệ lưu trữ (Database).

##### b, Tổ chức các tầng kiến trúc (Layered Architecture)

Hệ thống tuân thủ nghiêm ngặt quy tắc phụ thuộc (Dependency Rule) của Clean Architecture: "Mọi sự phụ thuộc của mã nguồn chỉ được hướng vào bên trong". Cấu trúc dự án được phân chia thành bốn lớp chính với vai trò và trách nhiệm rõ ràng.

Lớp trọng cùng và quan trọng nhất là Lớp Miền (Domain Layer). Đây là lõi trung tâm của hệ thống, nơi chứa các quy tắc nghiệp vụ bất biến của doanh nghiệp, bao gồm các Thực thể (Entities), Đối tượng giá trị (Value Objects), và các Sự kiện miền (Domain Events). Lớp này hoàn toàn không phụ thuộc vào bất kỳ thư viện bên ngoài, framework hay công nghệ cơ sở dữ liệu nào, đảm bảo logic nghiệp vụ luôn trong sáng và dễ dàng kiểm thử.

Bao bọc bên ngoài lớp miền là Lớp Ứng dụng (Application Layer). Lớp này

đóng vai trò điều phối các ca sử dụng (Use Cases) của hệ thống. Nhiệm vụ chính của nó là nhận yêu cầu từ các lớp bên ngoài, điều phối các đối tượng trong lớp miền thực hiện các tác vụ nghiệp vụ và trả về kết quả. Lớp ứng dụng định nghĩa các giao diện (Interfaces) cho các dịch vụ hạ tầng nhưng không trực tiếp thực thi chúng.

Tiếp theo là Lớp Hạ tầng (Infrastructure Layer), nơi cung cấp các triển khai cụ thể cho các giao diện kỹ thuật được định nghĩa ở lớp ứng dụng. Đây là nơi các công nghệ cụ thể như EF Core để truy xuất dữ liệu, các dịch vụ gửi email, thanh toán hay thông báo đẩy được tích hợp vào hệ thống. Lớp này phụ thuộc vào lớp ứng dụng và lớp miền, đóng vai trò như một bộ chuyển đổi (adapter) kết nối hệ thống với các công cụ bên ngoài.

Lớp ngoài cùng là Lớp Giao diện (Web/Presentation Layer). Đây là điểm tiếp xúc với người dùng hoặc các hệ thống khác, chịu trách nhiệm tiếp nhận các yêu cầu HTTP, xác thực người dùng, gọi xuống lớp ứng dụng để xử lý và trả về phản hồi thích hợp (thường là định dạng JSON).

Trong thực tế triển khai của dự án YummyZoom, các lý thuyết trên được hiện thực hóa thông qua cấu trúc dự án cụ thể. YummyZoom.Domain đóng gói toàn bộ logic cốt lõi, ví dụ như Aggregate TeamCart đảm bảo quy tắc chỉ chủ phòng mới có quyền chốt đơn. YummyZoom.Application chứa các Command và Query Handlers, chẳng hạn như AddItemToTeamCartCommand sẽ kích hoạt phương thức thêm món trong Domain. YummyZoom.Infrastructure chịu trách nhiệm thực thi việc lưu trữ xuống PostgreSQL hay giao tiếp với Stripe. Cuối cùng, YummyZoom.Web cung cấp các API Endpoints và SignalR Hubs để phục vụ ứng dụng di động và web quản trị.

### c, Thiết kế chiến lược (Strategic Design)

Áp dụng chiến lược của DDD, hệ thống được chia nhỏ thành các Ngữ cảnh giới hạn (Bounded Contexts), mỗi ngữ cảnh giải quyết một vấn đề nghiệp vụ cụ thể và có Ngôn ngữ chung (Ubiquitous Language) riêng. Đầu tiên là Identity Context, chịu trách nhiệm quản lý người dùng, phân quyền và xác thực. Tiếp đến là Catalog Context, quản lý thực đơn, danh mục, thông tin nhà hàng, món ăn và các tùy chọn. Quan trọng nhất là TeamCart Context, xử lý logic chia sẻ giỏ hàng, đồng bộ trạng thái thời gian thực và phân chia hóa đơn. Cuối cùng là Ordering Context, xử lý quy trình đặt hàng, thanh toán và vòng đời của đơn hàng sau khi được chốt.

### d, Các mẫu kỹ thuật chủ đạo (Tactical Patterns)

Để tối ưu hóa hiệu năng và khả năng bảo trì, hệ thống áp dụng các mẫu thiết kế kỹ thuật nâng cao, tiêu biểu là CQRS (Command Query Responsibility Segregation) và Domain Events.

Mẫu CQRS giúp hệ thống tách biệt rõ ràng luồng Đọc (Query) và Ghi (Command) dữ liệu. Phần Ghi sử dụng EF Core kết hợp với Domain Model để đảm bảo tính toàn vẹn dữ liệu khi thực hiện các thay đổi nghiệp vụ. Trong khi đó, Phần Đọc sử dụng Dapper với SQL để truy vấn dữ liệu trực tiếp và trả về các Data Transfer Object (DTO) phẳng, tối ưu hóa tốc độ phản hồi cho người dùng.

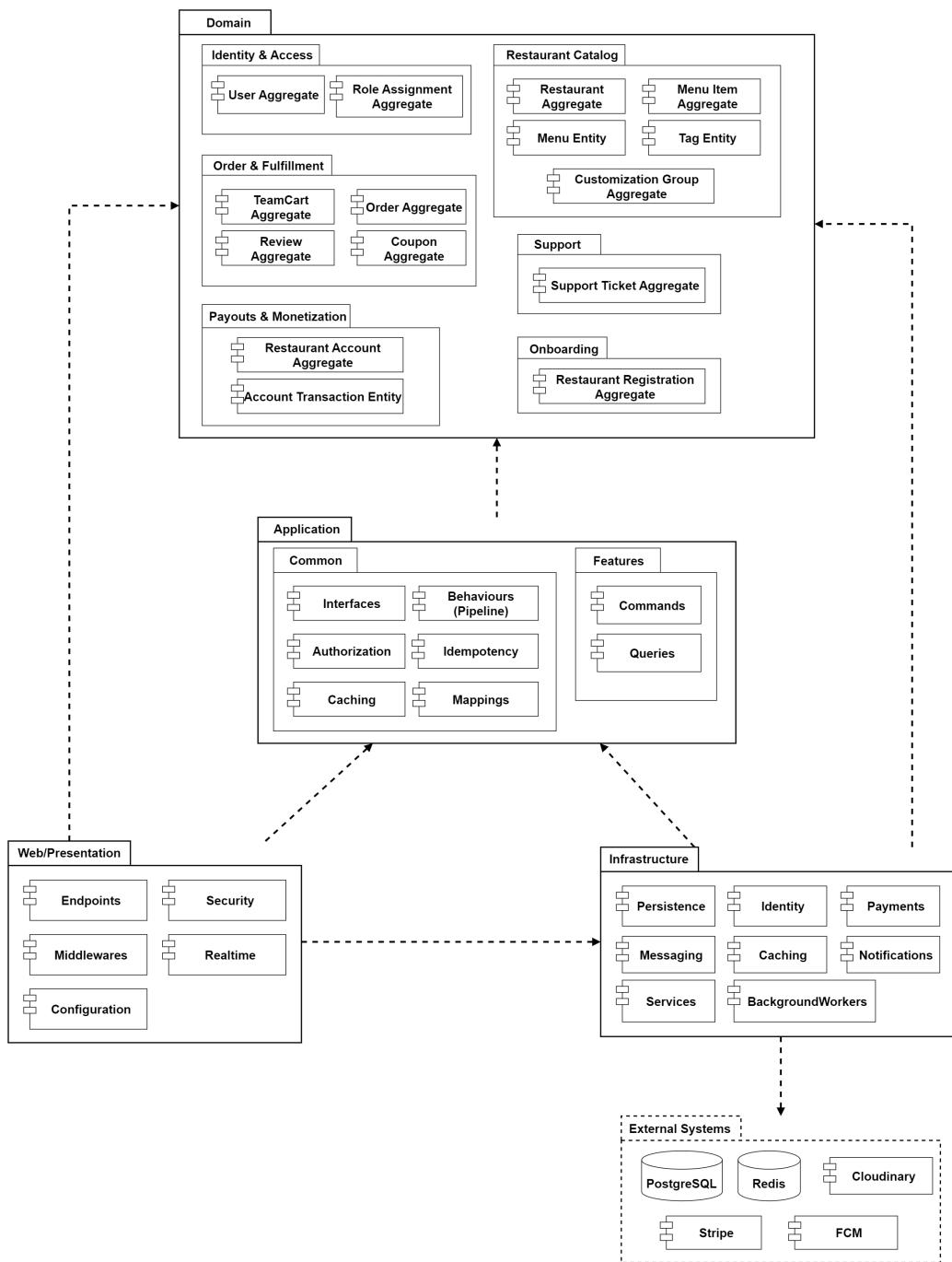
Bên cạnh đó, mẫu Domain Events kết hợp với Outbox Pattern được sử dụng để giải quyết bài toán về tính nhất quán cuối cùng (Eventual Consistency). Khi một nghiệp vụ quan trọng hoàn tất, hệ thống sinh ra một sự kiện miền và lưu vào bảng Outbox trong cùng một giao dịch cơ sở dữ liệu. Một tiến trình nền sẽ đọc bảng này và thực hiện các tác vụ phụ như gửi email hay thông báo, đảm bảo rằng các tác vụ bên lề không làm ảnh hưởng đến hiệu năng của luồng nghiệp vụ chính.

### e, Kết luận

Việc lựa chọn kiến trúc Clean Architecture kết hợp DDD và CQRS mang lại nền tảng vững chắc cho hệ thống YummyZoom. Kiến trúc này không chỉ giải quyết tốt các bài toán nghiệp vụ phức tạp hiện tại về đặt hàng nhóm mà còn đảm bảo các tiêu chí phi chức năng quan trọng: dễ dàng kiểm thử (Testability), dễ bảo trì (Maintainability) và sẵn sàng mở rộng (Scalability) trong tương lai.

#### 4.1.2 Thiết kế tổng quan

Biểu đồ gói tổng quan của hệ thống YummyZoom thể hiện sự phân tầng rõ ràng và các ràng buộc phụ thuộc tuân thủ quy tắc Clean Architecture.



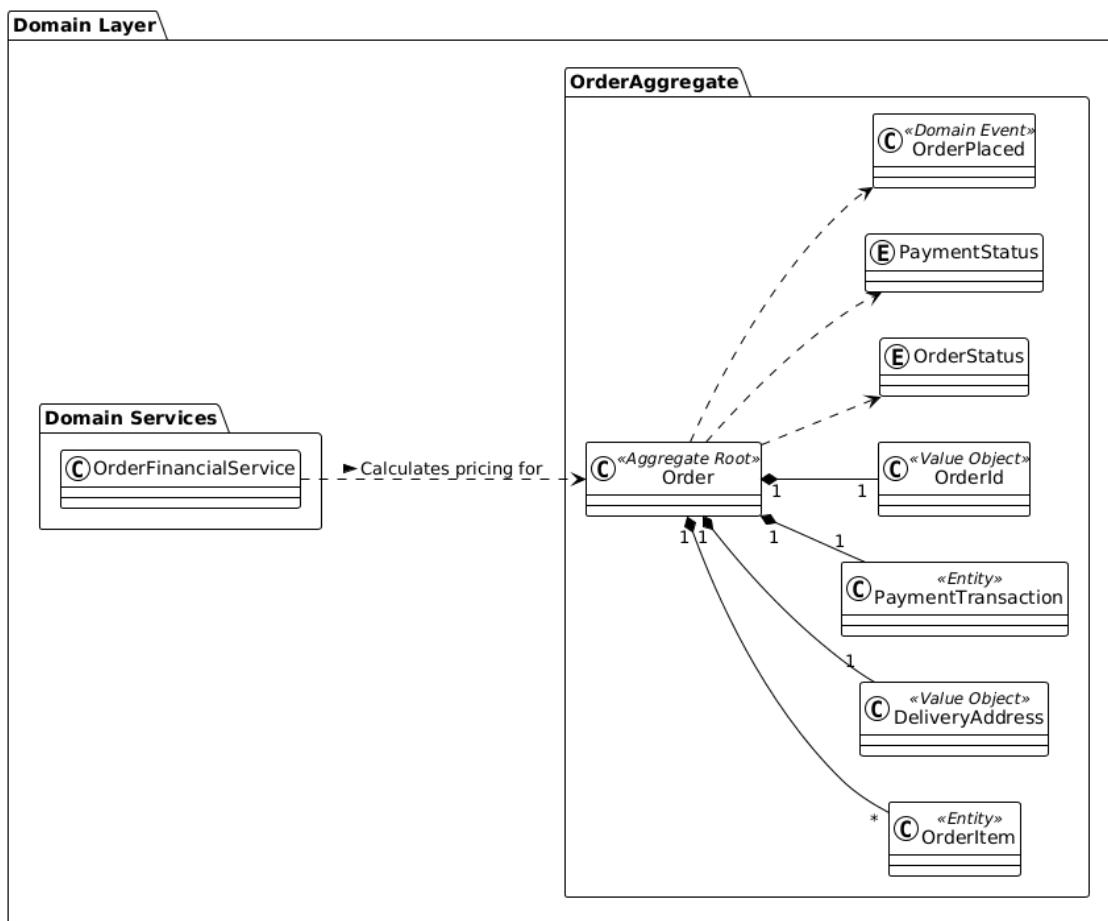
Hình 4.1: Biểu đồ phụ thuộc gói tổng quan YummyZoom (Backend)

### 4.1.3 Thiết kế chi tiết gói

Mục tiêu của phần này là minh họa cách tổ chức mã nguồn và hiện thực hóa kiến trúc Clean Architecture thông qua các biểu đồ gói (Package Diagrams). Do số lượng gói trong các lớp miền (Domain), lớp ứng dụng (Application) và lớp hạ tầng (Infrastructure) của hệ thống tương đối lớn, đồ án lựa chọn luồng nghiệp vụ **Khởi tạo đơn hàng** là luồng tiêu biểu nhất để phân tích chi tiết. Thiết kế được trình bày thành hai phần riêng biệt bao gồm: thiết kế cho lớp miền và thiết kế cho lớp ứng dụng kết hợp hạ tầng.

### a, Thiết kế lớp miền (Domain Layer)

Biểu đồ tại Hình 4.2 minh họa các thành phần cốt lõi trong lớp miền phục vụ cho nghiệp vụ khởi tạo đơn hàng. Tại đây, lớp Order đóng vai trò là gốc tập hợp (Aggregate Root), trung tâm quản lý mọi thay đổi và đảm bảo tính nhất quán của dữ liệu.

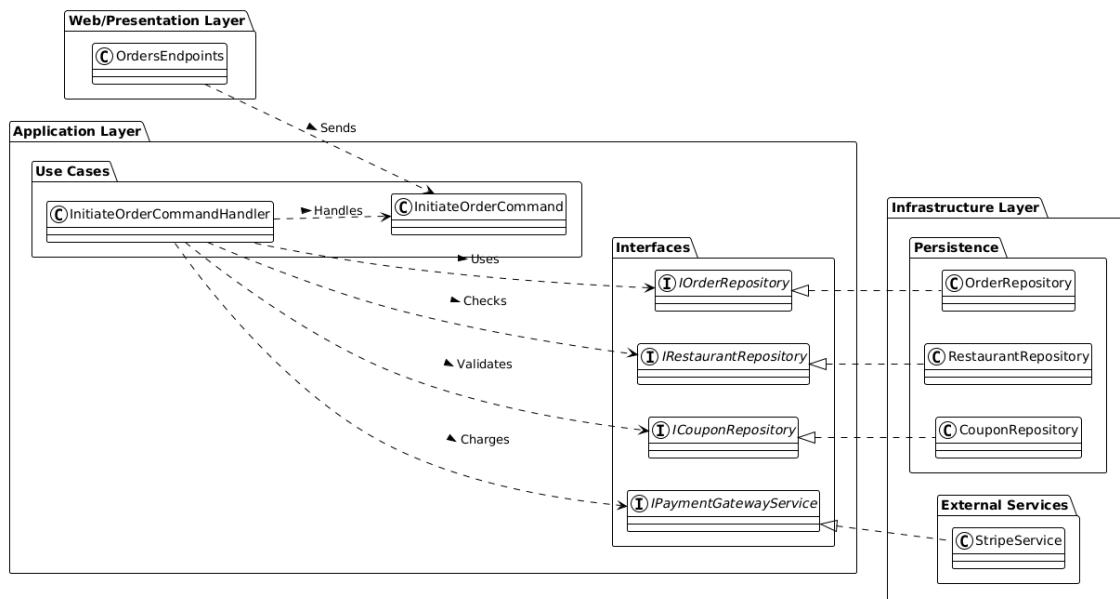


Hình 4.2: Thiết kế gói lớp miền cho luồng Khởi tạo Đơn hàng

Trong thiết kế này, tính đóng gói (Encapsulation) được tuân thủ nghiêm ngặt. Lớp Order bao bọc các thực thể con như OrderItem (món trong đơn) và PaymentTransaction (giao dịch thanh toán), cũng như các đối tượng giá trị (Value Objects) như DeliveryAddress (địa chỉ giao hàng). Các logic tính toán phức tạp liên quan đến giá cả, khuyến mãi và thuế được tách biệt vào ranh giới của dịch vụ miền (Domain Service), cụ thể là lớp OrderFinancialService, giúp giữ cho các thực thể tinh gọn. Bên cạnh đó, cơ chế sự kiện miền (Domain Events) với đại diện là OrderPlaced giúp tách biệt logic nghiệp vụ chính khỏi các tác vụ phụ trợ, đảm bảo nguyên lý đơn trách nhiệm.

## b, Thiết kế lớp ứng dụng và hạ tầng (Application & Infrastructure Layers)

Sự tương tác giữa các tầng còn lại để hiện thực hóa luồng nghiệp vụ được trình bày chi tiết tại Hình 4.3. Biểu đồ này thể hiện rõ vai trò điều phối của lớp ứng dụng và sự tách biệt về công nghệ của lớp hạ tầng.



**Hình 4.3:** Thiết kế gói lớp ứng dụng và hạ tầng cho luồng Khởi tạo Đơn hàng

Tại lớp giao diện (Web layer), OrdersEndpoints đóng vai trò tiếp nhận các yêu cầu HTTP từ phía người dùng, chuyển đổi dữ liệu và gửi mệnh lệnh xuống lớp ứng dụng. Lớp ứng dụng (Application layer) chứa các ca sử dụng dự án, tiêu biểu là InitiateOrderCommandHandler. Thành phần này chịu trách nhiệm điều phối luồng xử lý: gọi xuống lớp miền để khởi tạo đối tượng nghiệp vụ, sau đó tương tác với các giao diện (Interfaces) trừu tượng như IPaymentGatewayService hay IOrderRepository.

Điểm quan trọng trong thiết kế này là việc áp dụng nguyên lý đảo ngược sự phụ thuộc (Dependency Inversion Principle). Lớp ứng dụng chỉ phụ thuộc vào các giao diện do chính nó định nghĩa. Việc hiện thực hóa các giao diện này bằng các công nghệ cụ thể như EF Core hay Stripe được đặt hoàn toàn tại lớp hạ tầng (Infrastructure layer). Cách tổ chức này giúp hệ thống linh hoạt, dễ dàng thay thế công nghệ hoặc thư viện bên thứ ba mà không làm ảnh hưởng đến logic nghiệp vụ cốt lõi.

## c, Luồng thực thi tổng thể

Quy trình khởi tạo đơn hàng bắt đầu khi yêu cầu từ người dùng được gửi đến lớp giao diện và chuyển thành mệnh lệnh (Command) tại lớp ứng dụng. Bộ xử lý

(Handler) tại lớp ứng dụng sẽ sử dụng các dịch vụ miền để tính toán và khởi tạo đơn hàng hợp lệ. Sau đó, thông qua các giao diện trùu tượng, hệ thống thực hiện xử lý thanh toán và lưu trữ dữ liệu bền vững xuống cơ sở dữ liệu. Cuối cùng, các sự kiện miền được phát ra để kích hoạt các quy trình hậu xử lý như gửi thông báo hoặc gửi email xác nhận, hoàn tất chu trình nghiệp vụ một cách toàn vẹn và an toàn.

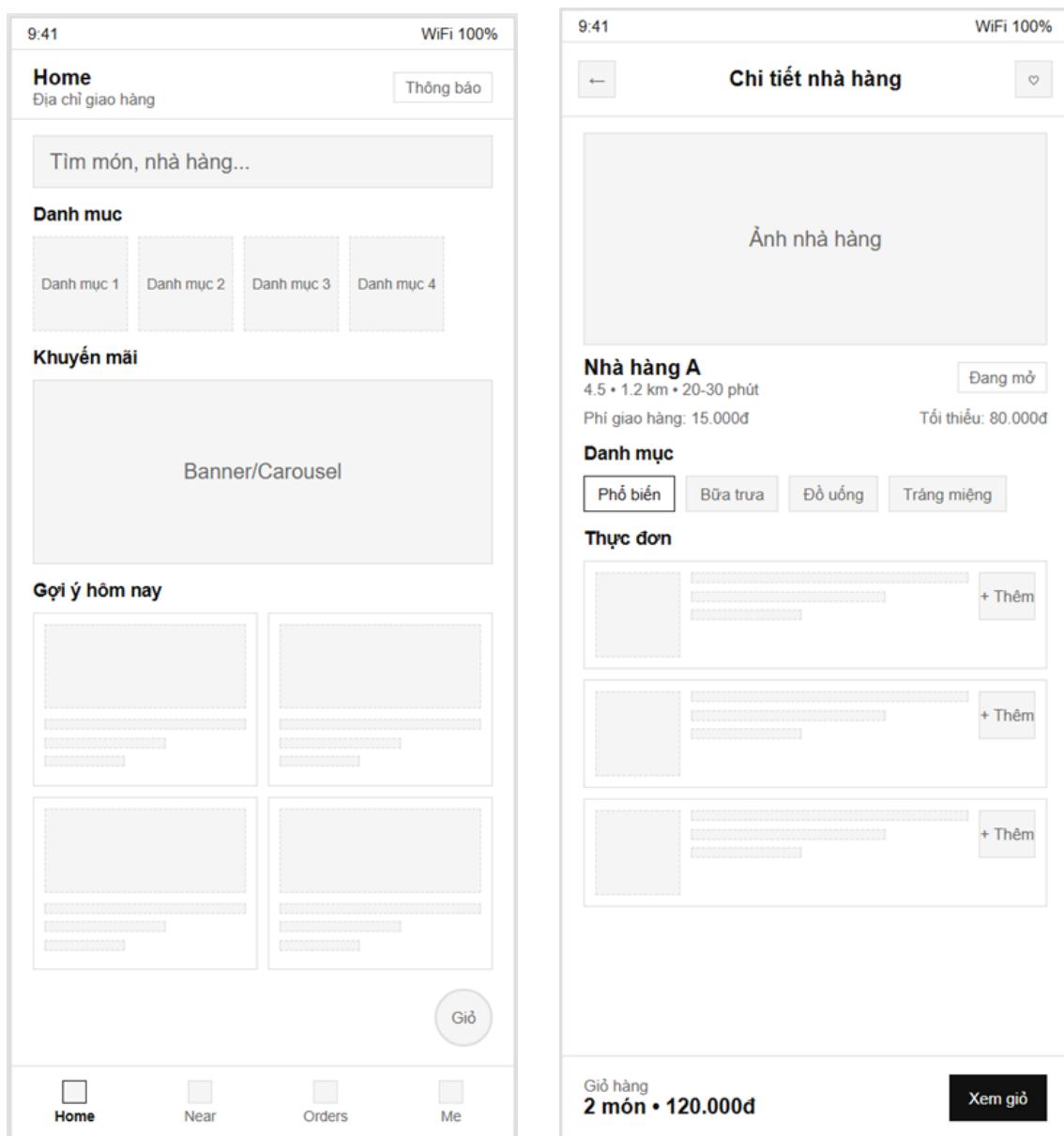
## 4.2 Thiết kế chi tiết

### 4.2.1 Thiết kế giao diện

#### a, Thiết kế giao diện ứng dụng di động dành cho khách hàng

Ứng dụng di động được tối ưu cho khung máy 384x854 nhằm giữ tỷ lệ hiển thị đồng nhất và hỗ trợ thao tác đặt món nhanh. Bộ cục ưu tiên thông tin quan trọng ở vùng nhìn đầu và làm nổi bật các hành động chính để giảm thao tác thừa. Kiểu chữ quy đổi theo thang sp, các nút và trường nhập liệu dùng chung bán kính bo góc và khoảng đệm để tạo cảm giác thân thiện. Màu thương hiệu được chuẩn hóa, còn bộ mockup giữ tông đen trắng để làm rõ cấu trúc và luồng thao tác. Phản hồi người dùng được hiển thị bằng các thanh thông báo ngắn ở cuối màn hình.

Các màn hình tiêu biểu được lựa chọn nhằm phản ánh đầy đủ hành trình người dùng từ khám phá đến thanh toán, bao gồm trang Home/Menu và Restaurant Detail (Hình 4.4).



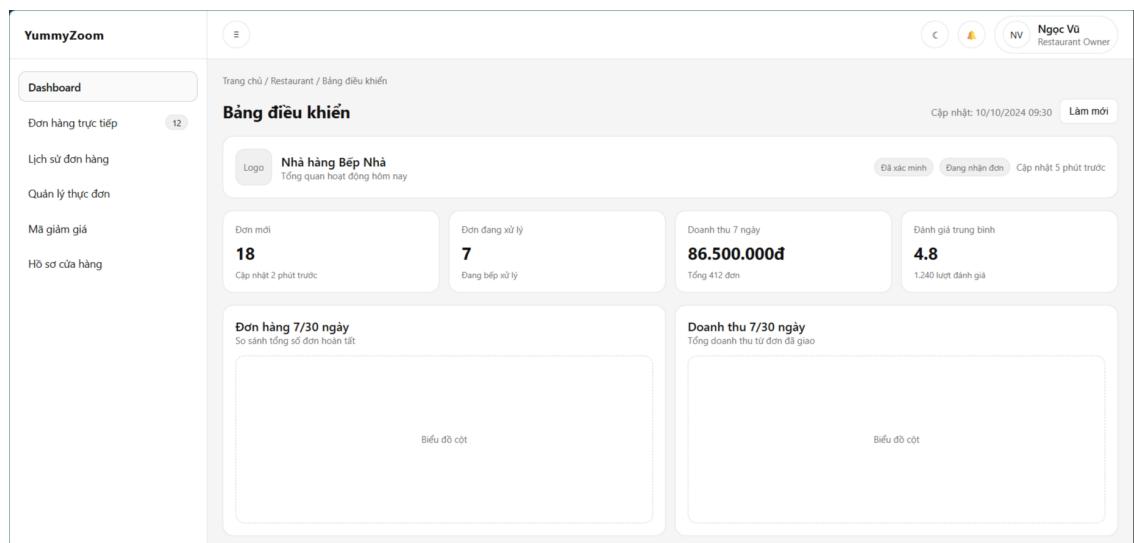
**Hình 4.4:** Minh họa màn hình Home/Menu (trái) và Restaurant Detail (phải) trên ứng dụng di động

### b, Thiết kế giao diện hệ thống quản trị trên web

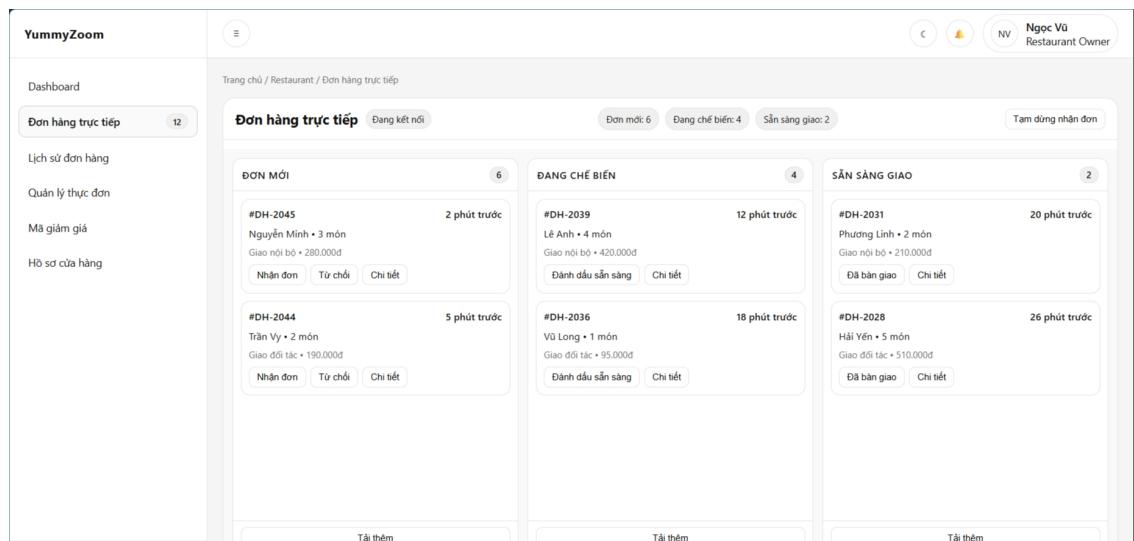
Hệ thống quản trị tối ưu cho màn hình desktop 1440x900 và vẫn hiển thị tốt trên tablet ngang. Bộ cục dùng lưới bước nhỏ để quét dữ liệu nhanh, bảng và khối thông tin gọn phục vụ điều hành. Kiểu chữ phân cấp rõ, nhấn mạnh số liệu bằng độ đậm. Màu thương hiệu làm nổi bật hành động chính, hỗ trợ chế độ nền tối cho môi trường ánh sáng yếu. Thành phần nhập liệu, nút bấm và thông báo được chuẩn hóa về viền, trạng thái tập trung và hiển thị lỗi để giữ trải nghiệm nhất quán.

Các màn hình minh họa tập trung vào những chức năng cốt lõi của quản trị, gồm trang Restaurant Dashboard (Hình 4.5) và trang Live Orders (Hình 4.6).

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG



Hình 4.5: Minh họa màn hình Restaurant Dashboard trên hệ thống quản trị



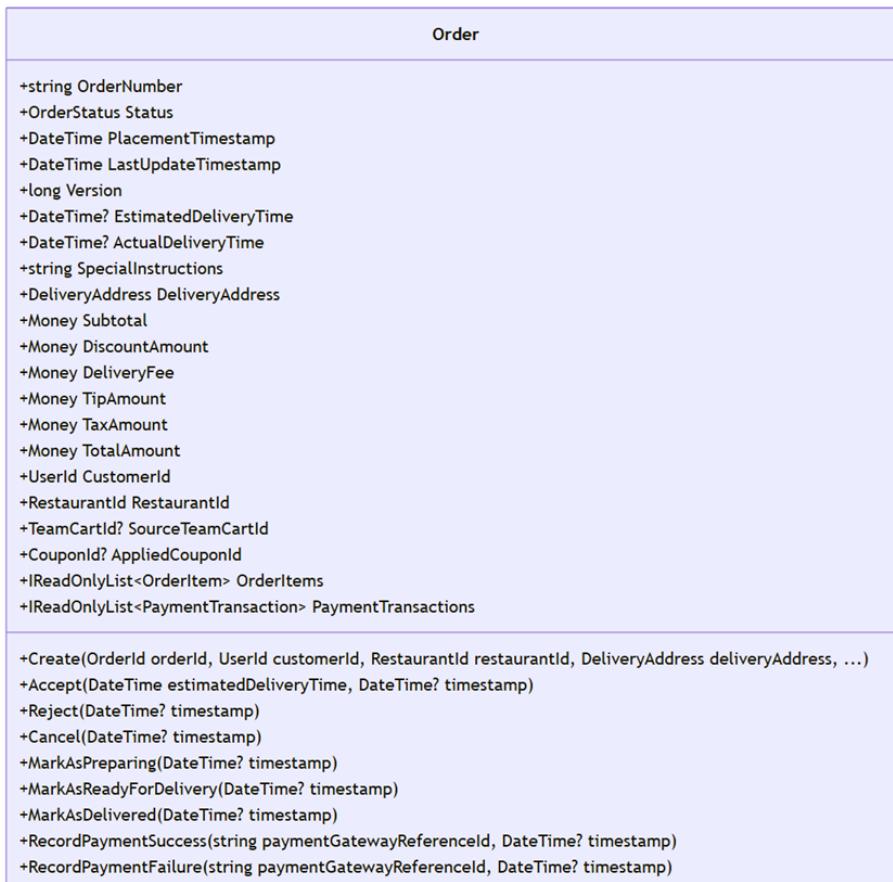
Hình 4.6: Minh họa màn hình Live Orders trên hệ thống quản trị

### 4.2.2 Thiết kế lớp

Trong dự án YummyZoom, phần lớn quy tắc nghiệp vụ được đóng gói tại lớp miền (Domain layer) theo Clean Architecture, do đó phần thiết kế lớp tập trung vào các aggregate root tiêu biểu. Ba lớp được lựa chọn gồm Order, TeamCart và Restaurant vì đây là các lớp chịu trách nhiệm quản lý vòng đời, trạng thái và các bất biến nghiệp vụ quan trọng của hệ thống. Đối với các lớp ở lớp ứng dụng (Application layer), nội dung chỉ trình bày ở mức vai trò điều phối luồng, bởi đa số các handler chỉ thực hiện một phương thức *Handle* với logic mỏng.

Lớp Order đại diện cho đơn hàng, quản lý dữ liệu tài chính, danh sách món đã đặt và các môc trạng thái trong vòng đời xử lý. Thiết kế của lớp thể hiện rõ các thuộc tính tài chính (Subtotal, DiscountAmount, DeliveryFee, TipAmount,

TaxAmount, TotalAmount) cùng các phương thức chuyển trạng thái như Accept, Reject, MarkAsPreparing, MarkAsReadyForDelivery và MarkAsDelivered, đồng thời phát sinh các sự kiện miền tương ứng phục vụ đồng bộ trạng thái theo thời gian thực. Hình 4.7 mô tả cấu trúc lớp Order và các mối quan hệ chính.



**Hình 4.7:** Biểu đồ lớp của Order (Aggregate root)

Lớp TeamCart mô hình hóa giỏ hàng nhóm, cho phép nhiều người cùng đặt món và phối hợp thanh toán. Các thuộc tính cốt lõi gồm thông tin thành viên, danh sách món, trạng thái giỏ và các cấu phần tài chính như TipAmount và QuoteVersion. Các phương thức như AddItem, LockForPayment, FinalizePricing và RecordSuccessfulOnlinePayment giúp đảm bảo tính nhất quán của luồng thanh toán nhóm. Hình 4.8 trình bày thiết kế lớp TeamCart.



**Hình 4.8:** Biểu đồ lớp của TeamCart (Aggregate root)

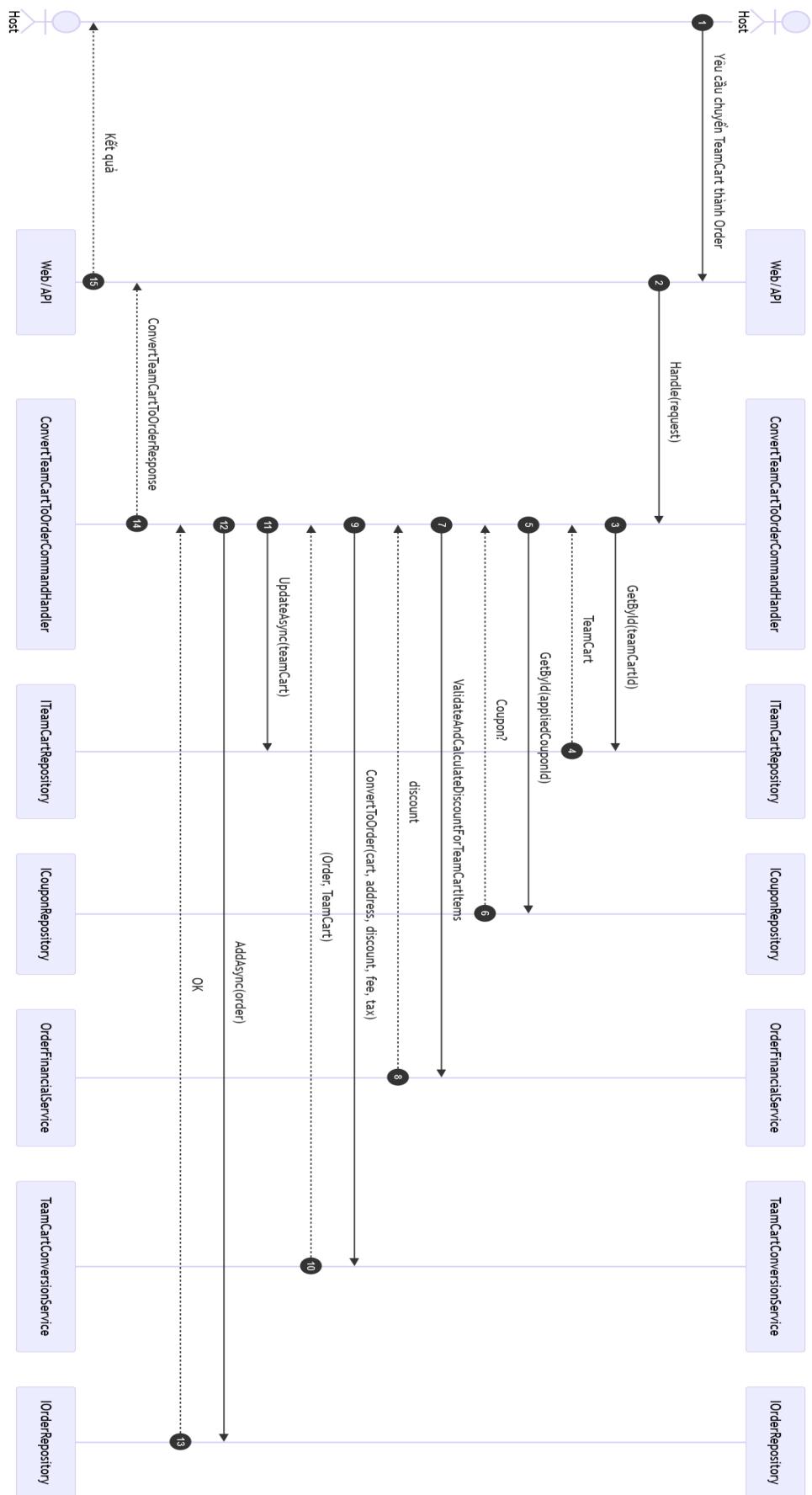
Lớp Restaurant quản lý hồ sơ nhà hàng, các thông tin định danh và trạng thái hoạt động như đã xác thực hay đang nhận đơn. Thiết kế này đảm bảo các ràng buộc nghiệp vụ khi thay đổi thông tin thương hiệu, địa điểm, khung giờ hoạt động và trạng thái xác thực. Lớp này lần lượt được minh họa tại Hình 4.9.



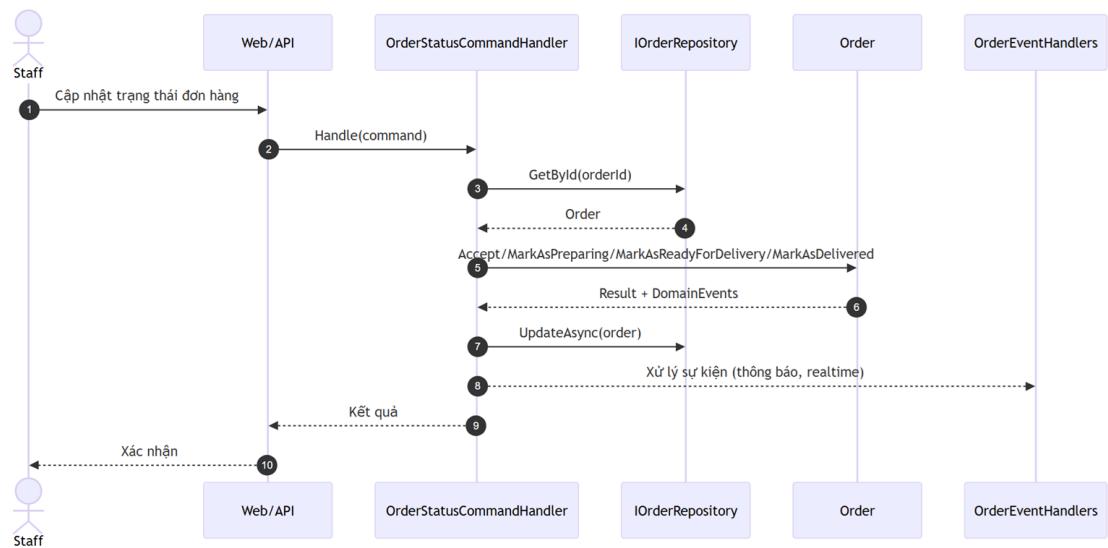
**Hình 4.9:** Biểu đồ lớp của Restaurant (Aggregate root)

Để minh họa cách các lớp phối hợp trong các ca sử dụng quan trọng, hai luồng chính được lựa chọn gồm chuyển TeamCart thành Order và cập nhật trạng thái đơn hàng. Luồng chuyển TeamCart thành Order nhấn mạnh vai trò của Domain Service trong việc ánh xạ dữ liệu và tính toán tài chính trước khi tạo Order và cập nhật TeamCart. Luồng cập nhật trạng thái đơn hàng thể hiện các chuyển trạng thái hợp lệ trong Order và phát sinh sự kiện miền để phục vụ cập nhật thời gian thực. Các luồng này được trình bày lần lượt tại Hình 4.10 và Hình 4.11.

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG



Hình 4.10: Biểu đồ trình tự chuyển TeamCart thành Order

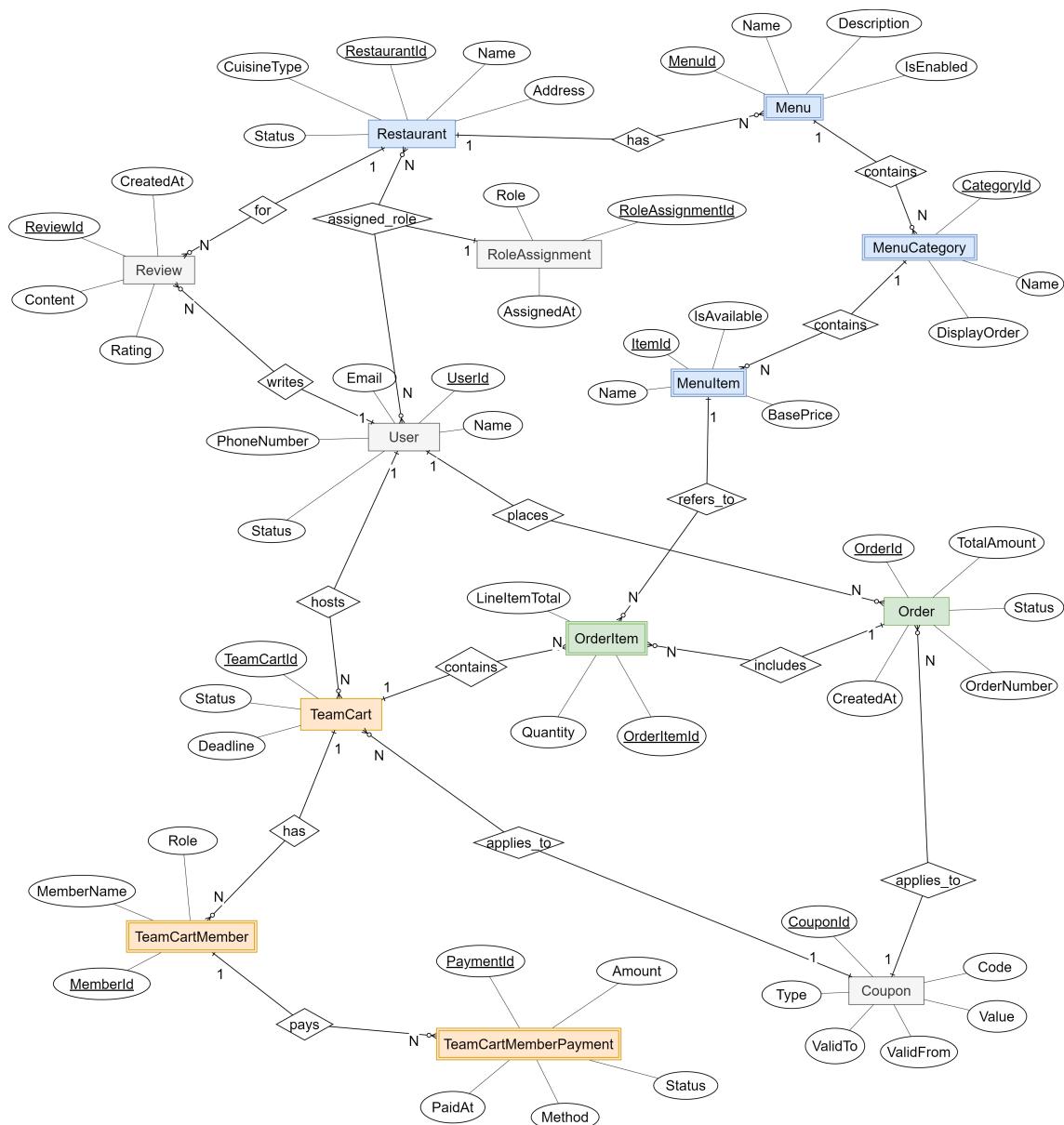


**Hình 4.11:** Biểu đồ trình tự cập nhật trạng thái đơn hàng

### 4.2.3 Thiết kế cơ sở dữ liệu

Trong dự án YummyZoom, cơ sở dữ liệu được triển khai trên PostgreSQL. Phần thiết kế dữ liệu được trình bày theo hai tầng: mô hình khái niệm bằng ERD kiểu Chen để mô tả thế giới thực và các thực thể nghiệp vụ, và mô hình quan hệ chi tiết phản ánh cấu trúc bảng sau khi ánh xạ vào cơ sở dữ liệu. Nguồn lược đồ tổng hợp được đối chiếu từ tài liệu kiến trúc, nhằm đảm bảo tính nhất quán giữa mô hình miền và dữ liệu lưu trữ.

Biểu đồ ERD khái niệm theo Chen tập trung vào các thực thể cốt lõi như Người dùng, Nhà hàng, Thực đơn, Món ăn, Đơn hàng và Giỏ nhóm, cùng các quan hệ nghiệp vụ giữa chúng. Mỗi thực thể chỉ giữ lại một số thuộc tính chính như định danh, trạng thái, thông tin mô tả và giá trị tiền tệ, giúp làm rõ bức tranh nghiệp vụ mà chưa bị ràng buộc bởi chi tiết triển khai. Hình 4.12 mô tả toàn bộ ERD khái niệm, trong đó các quan hệ chính như Nhà hàng–Thực đơn, Người dùng–Đơn hàng, Đơn hàng–Mục đơn hàng và Giỏ nhóm–Thành viên được biểu diễn rõ ràng theo bội số.

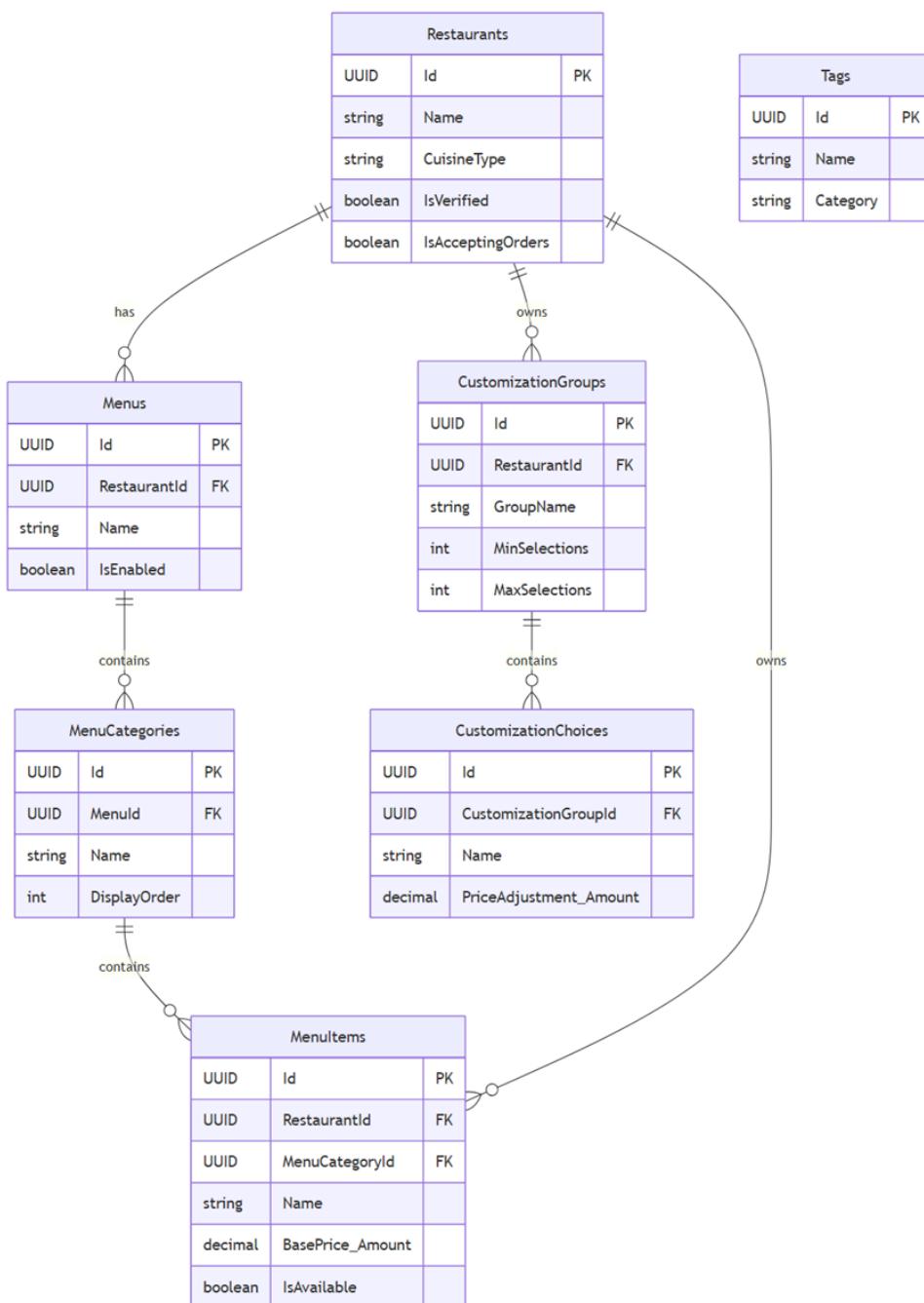


Hình 4.12: ERD khái niệm theo mô hình Chen cho YummyZoom

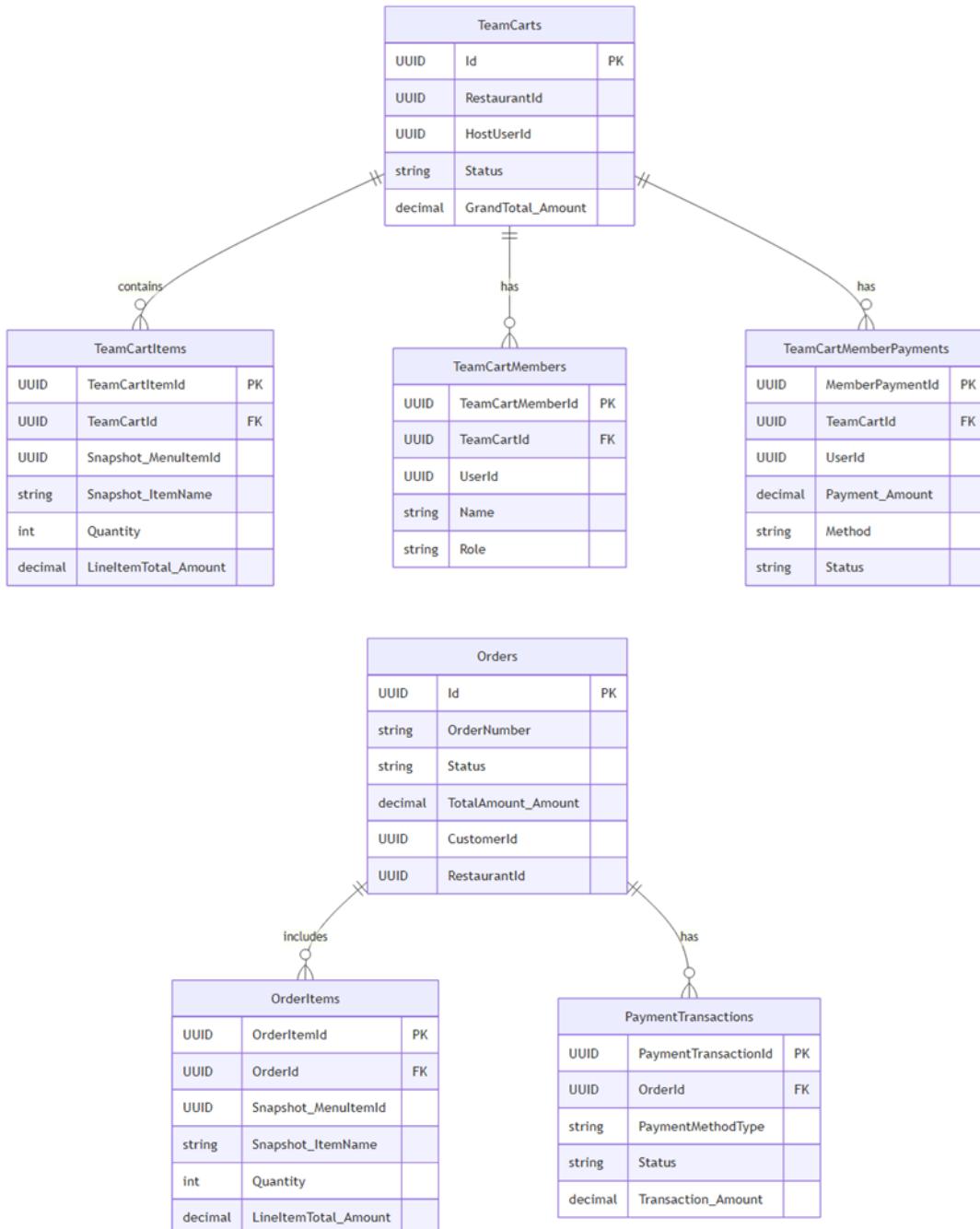
Từ mô hình khái niệm, hệ thống áp dụng quy tắc ánh xạ (mapping) từ mô hình miền (domain model) sang cơ sở dữ liệu quan hệ. Mỗi aggregate root được ánh xạ thành một bảng chính, ví dụ Order tương ứng bảng Orders, Restaurant tương ứng bảng Restaurants, TeamCart tương ứng bảng TeamCarts. Các thực thể có danh tính trong một aggregate được tách thành bảng con và liên kết bằng khóa ngoại về bảng gốc, chẳng hạn OrderItems và PaymentTransactions phụ thuộc Orders, TeamCartItems và TeamCartMembers phụ thuộc TeamCarts. Các giá trị bất biến dạng đối tượng giá trị (value object) được ánh xạ thành các cột trong bảng chính với tiền tố tên thuộc tính, ví dụ DeliveryAddress\_Street hoặc TipAmount\_Amount; với các giá trị phức tạp hoặc danh sách linh hoạt thì lưu dưới dạng JSONB để giảm độ phức tạp lược đồ. Quy tắc này được hiện thực nhất quán thông qua cấu hình EF Core (Fluent API) trong các lớp Configuration thuộc tầng Infrastructure, đồng thời

các trạng thái dạng enum được lưu dưới dạng chuỗi để dễ đọc và theo dõi.

Ở mức triển khai vật lý, số lượng bảng lớn được chia nhỏ theo từng nhóm chức năng để trình bày rõ ràng. Hình 4.13 mô tả nhóm Restaurant Catalog, gồm Restaurants, Menus, MenuCategories, MenuItemItems và các bảng tùy chọn món; nhóm này phản ánh cấu trúc dữ liệu phục vụ duyệt thực đơn và quản trị nội dung. Hình 4.14 trình bày nhóm Order & TeamCart, làm rõ cấu trúc lưu trữ đơn hàng, mục đơn hàng, thanh toán, giỏ nhóm và thành viên.



Hình 4.13: ERD hiện đại cho nhóm Restaurant Catalog



Hình 4.14: ERD hiện đại cho nhóm Order và TeamCart

### 4.3 Xây dựng ứng dụng

#### 4.3.1 Thư viện và công cụ sử dụng

Trong quá trình phát triển hệ thống YummyZoom, dự án đã áp dụng một loạt các thư viện và công cụ nhằm tối ưu hóa quy trình làm việc, nâng cao chất lượng mã nguồn và đảm bảo hiệu suất của ứng dụng. Bảng dưới đây liệt kê các công cụ chính cùng với mục đích sử dụng và địa chỉ URL tham khảo.

<b>Mục đích</b>	<b>Công cụ</b>	<b>Địa chỉ URL</b>
IDE lập trình	Visual Studio Code	<a href="https://code.visualstudio.com/">https://code.visualstudio.com/</a>
IDE lập trình cho .NET	JetBrains Rider	<a href="https://www.jetbrains.com/rider/">https://www.jetbrains.com/rider/</a>
IDE lập trình cho Android	Android Studio	<a href="https://developer.android.com/studio">https://developer.android.com/studio</a>
Quản lý mã nguồn	Git	<a href="https://git-scm.com/">https://git-scm.com/</a>
Quản lý mã nguồn và CI/CD	GitHub	<a href="https://github.com/">https://github.com/</a>
AI hỗ trợ lập trình	GitHub Copilot	<a href="https://github.com/features/copilot">https://github.com/features/copilot</a>
AI hỗ trợ lập trình	Cursor	<a href="https://www.cursor.com/">https://www.cursor.com/</a>
Backend nền tảng	.NET 9.0	<a href="https://dotnet.microsoft.com/">https://dotnet.microsoft.com/</a>
Backend framework	ASP.NET Core	<a href="https://learn.microsoft.com/aspnet/core/">https://learn.microsoft.com/aspnet/core/</a>
ORM/DAL	EF Core	<a href="https://learn.microsoft.com/ef/core/">https://learn.microsoft.com/ef/core/</a>
Backend kiểm thử	NUnit	<a href="https://nunit.org/">https://nunit.org/</a>
Backend kiểm thử	Testcontainers	<a href="https://testcontainers.com/">https://testcontainers.com/</a>
Công cụ quản lý runtime	.NET Aspire	<a href="https://learn.microsoft.com/dotnet/aspire/">https://learn.microsoft.com/dotnet/aspire/</a>
Cơ sở dữ liệu	PostgreSQL	<a href="https://www.postgresql.org/">https://www.postgresql.org/</a>
Bộ nhớ đệm	Redis	<a href="https://redis.io/">https://redis.io/</a>
Mobile framework	Flutter	<a href="https://flutter.dev/">https://flutter.dev/</a>
Mobile ngôn ngữ	Dart	<a href="https://dart.dev/">https://dart.dev/</a>
Mobile UI/UX	Flutter Material	<a href="https://docs.flutter.dev/ui/widgets/material">https://docs.flutter.dev/ui/widgets/material</a>
Mobile quản lý trạng thái	Provider	<a href="https://pub.dev/packages/provider">https://pub.dev/packages/provider</a>
Mobile lưu trữ	Hive	<a href="https://pub.dev/packages/hive">https://pub.dev/packages/hive</a>
Mobile bản đồ	Mapbox	<a href="https://www.mapbox.com/">https://www.mapbox.com/</a>

*(Tiếp tục trang sau)*

(Tiếp theo từ trang trước)

Mobile thông báo đẩy	Firebase Messaging	<a href="https://firebase.google.com/docs/cloud-messaging">https://firebase.google.com/docs/cloud-messaging</a>
Mobile payment	flutter_stripe	<a href="https://pub.dev/packages/flutter_stripe">https://pub.dev/packages/flutter_stripe</a>
Web admin framework	Angular	<a href="https://angular.dev/">https://angular.dev/</a>
Web admin ngôn ngữ	TypeScript	<a href="https://www.typescriptlang.org/">https://www.typescriptlang.org/</a>
Web admin UI	PrimeNG + PrimeIcons	<a href="https://primeng.org/">https://primeng.org/</a>
Web admin UI	Tailwind CSS	<a href="https://tailwindcss.com/">https://tailwindcss.com/</a>
Web admin công tác thời gian thực	SignalR JS Client	<a href="https://learn.microsoft.com/aspnet/core/signalr/javascript-client">https://learn.microsoft.com/aspnet/core/signalr/javascript-client</a>
Dịch vụ bản đồ	Mapbox API	<a href="https://docs.mapbox.com/">https://docs.mapbox.com/</a>
Dịch vụ thông báo	Firebase Messaging	<a href="https://firebase.google.com/">https://firebase.google.com/</a>
Dịch vụ thanh toán	Stripe	<a href="https://stripe.com/">https://stripe.com/</a>

Bảng 4.1: Danh sách thư viện và công cụ sử dụng

### 4.3.2 Kết quả đạt được

#### a, Backend (ASP.NET Core/C#)

Sản phẩm backend được đóng gói dưới dạng dịch vụ ASP.NET Core Web API. Các số liệu thống kê chính được tổng hợp trong Bảng 4.2.

Chỉ tiêu	Giá trị	Ghi chú
Tổng số dòng code	185,965	Bao gồm comment và dòng trống
Số file .cs	1,348	Toàn bộ backend
Số lớp C#	1,301	Thông kê theo source code
Số endpoint API	158	Tổng số endpoint API được sử dụng bởi frontend
Số bảng CSDL	49	Đếm theo schema.sql
Dung lượng mã nguồn src/	103 MB	Sau khi dotnet clean
Dung lượng mã nguồn tests/	179 MB	Sau khi dotnet clean
Dung lượng gói publish Release	34 MB	dotnet publish

Bảng 4.2: Thống kê kết quả backend

### b, Mobile (Flutter/Dart)

Ứng dụng di động dành cho khách hàng được phát triển bằng Flutter/Dart. Các số liệu thống kê chính được tổng hợp trong Bảng 4.3.

Chỉ tiêu	Giá trị	Ghi chú
Tổng số dòng code	45,791	Bao gồm comment và dòng trống
Số file .dart	325	Toàn bộ ứng dụng mobile
Dung lượng mã nguồn	2.0 MB	Tổng thư mục mã nguồn lib/
Dung lượng APK Release	117.5 MB	Bản build APK

**Bảng 4.3:** Thông kê kết quả mobile

### c, Web Admin (Angular/TypeScript)

Ứng dụng web quản trị được phát triển bằng Angular/TypeScript. Các số liệu thống kê chính được tổng hợp trong Bảng 4.4.

Chỉ tiêu	Giá trị	Ghi chú
Tổng số dòng code	13,584	Bao gồm comment và dòng trống
Số file .ts	93	Toàn bộ ứng dụng web admin
Số file .html	13	Toàn bộ template giao diện
Dung lượng mã nguồn	1 MB	Tổng thư mục mã nguồn
Dung lượng gói publish	7 MB	Bản build publish

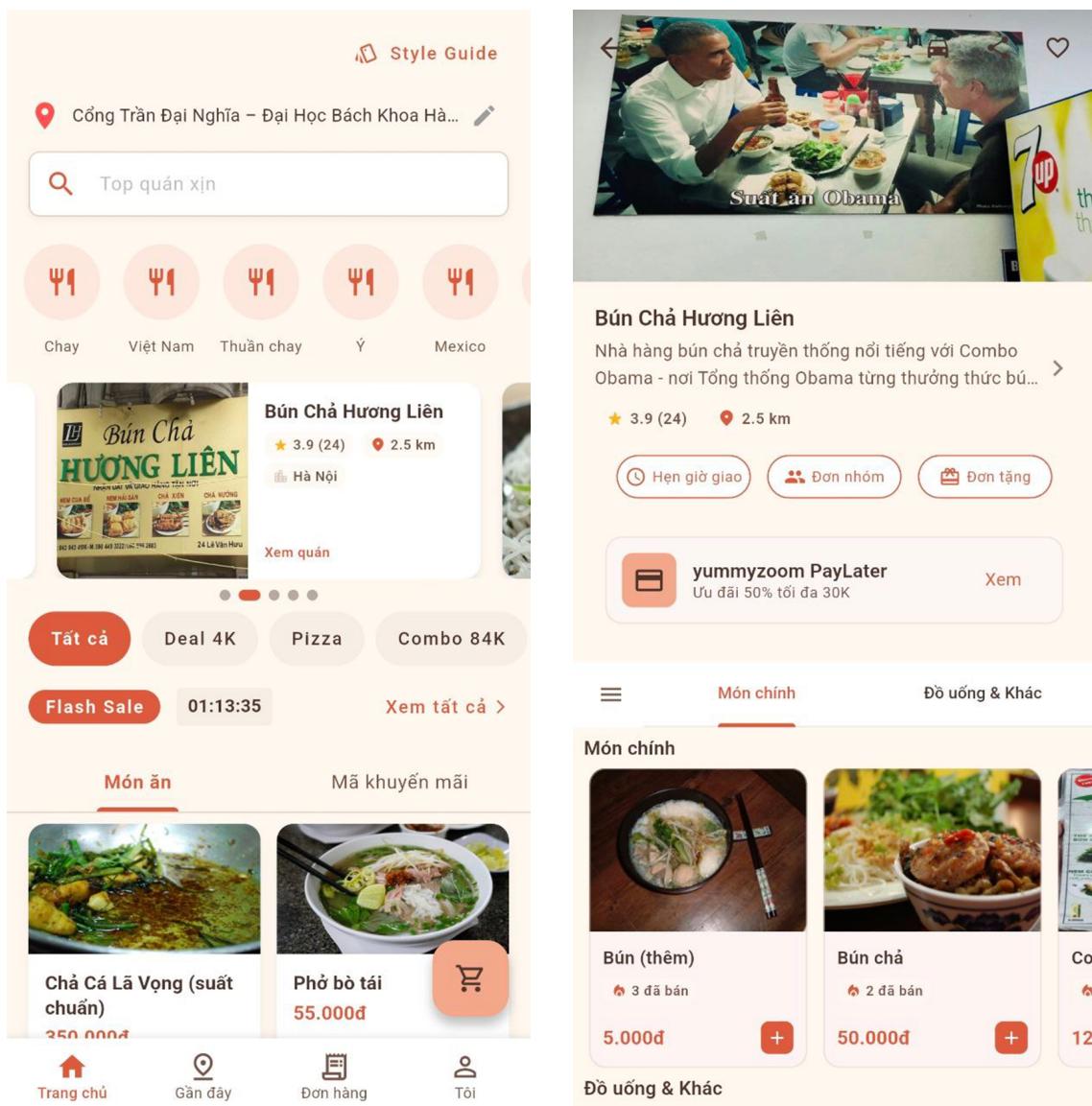
**Bảng 4.4:** Thông kê kết quả web admin

#### 4.3.3 Minh họa các chức năng chính

Phần này trình bày kết quả hiện thực của các chức năng trọng tâm trên cả ứng dụng di động dành cho khách hàng và hệ thống quản trị dành cho nhà hàng. Các giao diện được thiết kế để tối ưu hóa trải nghiệm người dùng cuối đồng thời phản ánh độ phức tạp của các quy trình nghiệp vụ phía sau.

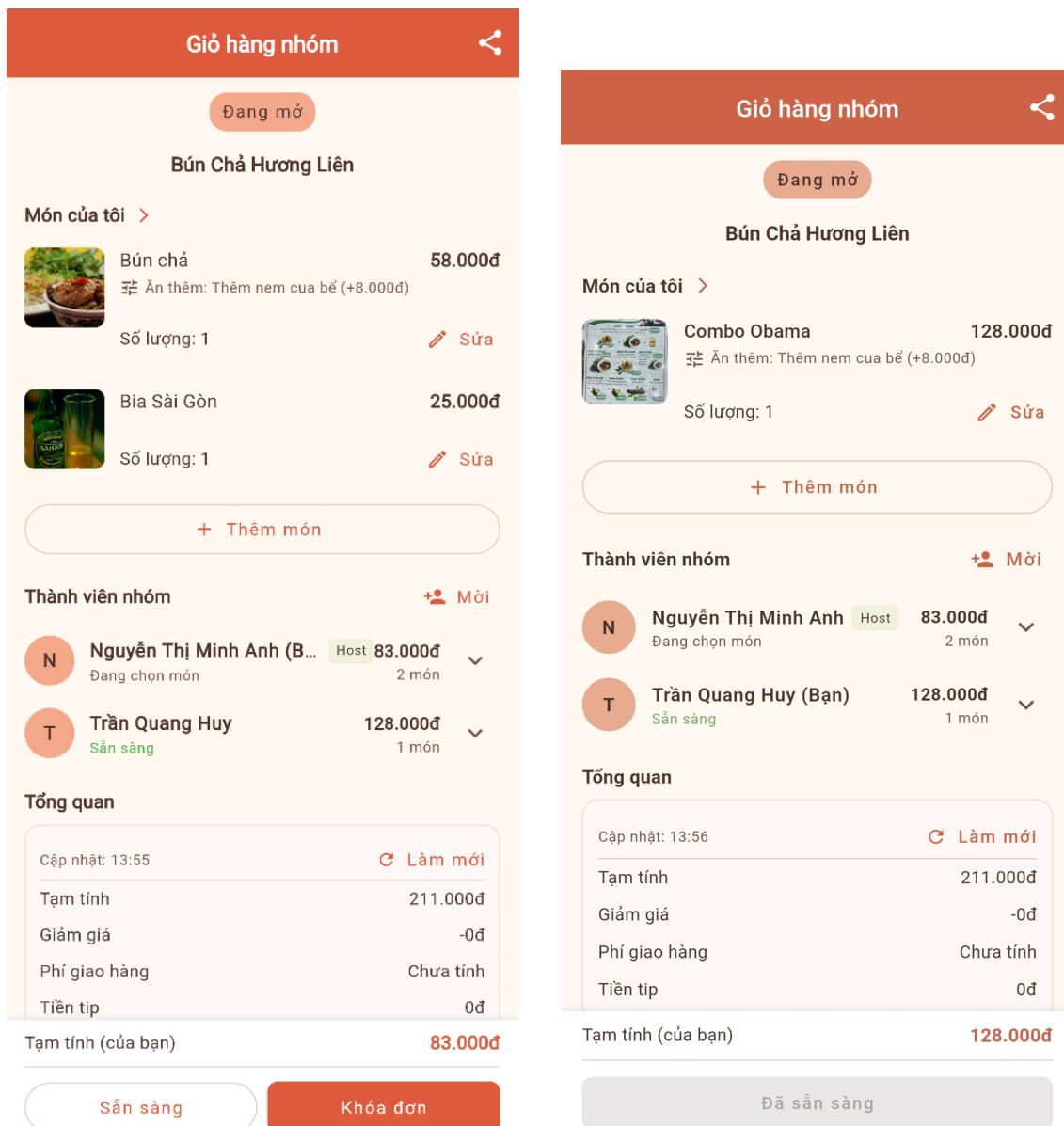
### a, Ứng dụng di động

Giao diện trang chủ (Home) và chi tiết nhà hàng (Restaurant Detail) là điểm chạm đầu tiên của người dùng. Hình 4.15 minh họa cách bố trí thông tin trực quan, giúp người dùng nhanh chóng tiếp cận danh sách nhà hàng và khám phá thực đơn. Màn hình chi tiết nhà hàng hiển thị đầy đủ thông tin, đánh giá và danh sách món ăn được phân nhóm khoa học, hỗ trợ thao tác thêm món nhanh chóng.



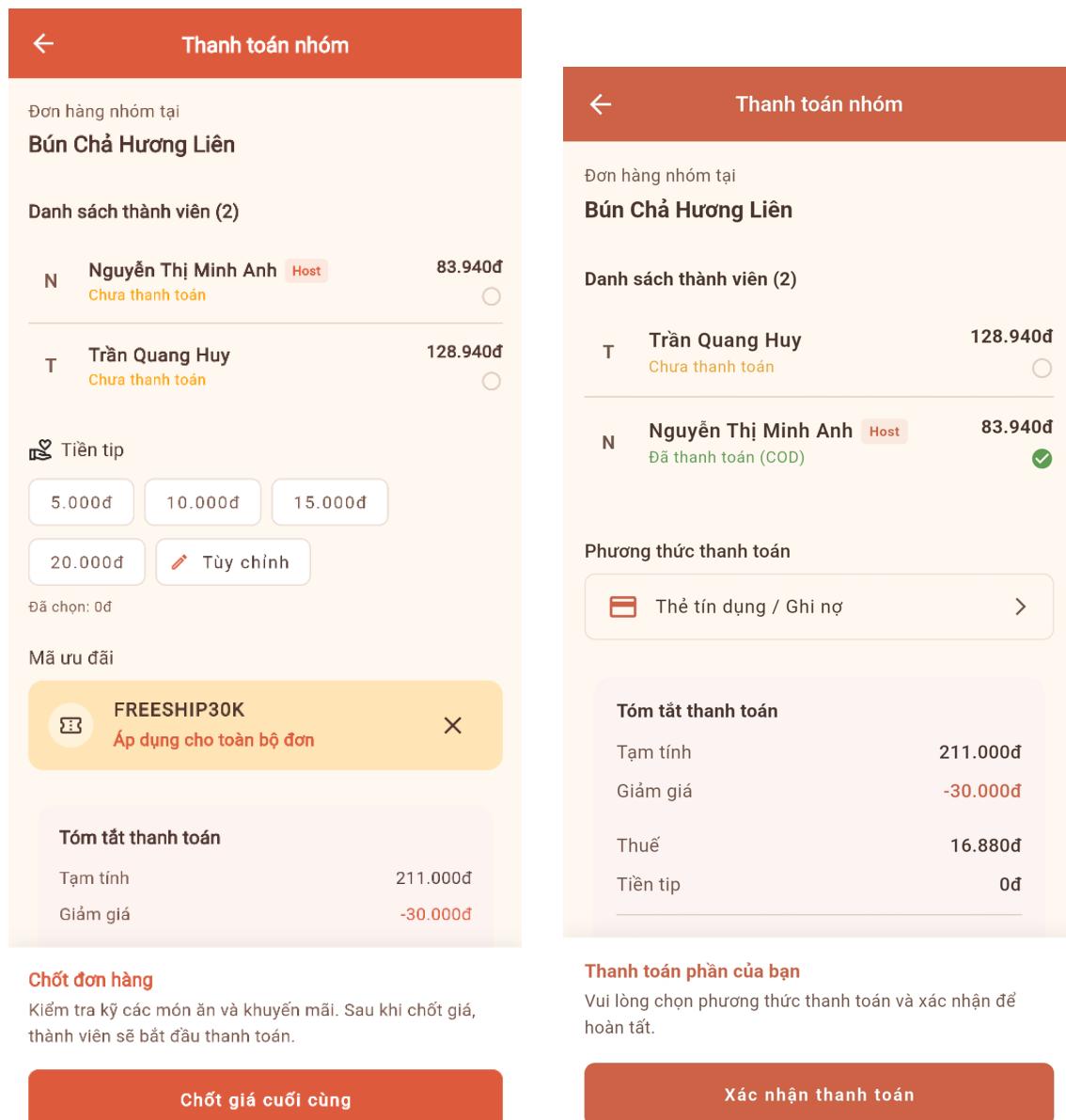
Hình 4.15: Màn hình Trang chủ (Trái) và Chi tiết nhà hàng (Phải)

Tính năng Đặt hàng nhóm (TeamCart) là điểm nhấn công nghệ của dự án, cho phép nhiều người dùng cùng thao tác trên một giỏ hàng trong thời gian thực. Hình 4.16 thể hiện sự đồng bộ trạng thái giữa màn hình của chủ phòng (Host) và thành viên (Member). Khi bất kỳ ai thêm hoặc bớt món, danh sách món ăn và tổng tiền sẽ được cập nhật tức thì trên tất cả thiết bị nhờ công nghệ SignalR, tạo cảm giác cộng tác mượt mà như đang ngồi cùng nhau.



Hình 4.16: Giao diện TeamCart: Màn hình Chủ phòng (Trái) và Thành viên (Phải)

Quy trình thanh toán cho đơn hàng nhóm được chia thành hai giai đoạn rõ ràng: chốt giá và thanh toán. Hình 4.17 minh họa màn hình Chốt giá nơi chủ phòng xem lại tổng kết đơn hàng, bao gồm các khoản phí và tiền tip. Sau khi chủ phòng xác nhận, tất cả thành viên trong nhóm được chuyển sang màn hình thanh toán để thanh toán phần của mình an toàn bằng cách chọn thanh toán bằng tiền mặt hoặc thanh toán qua ví điện tử, tích hợp Stripe để xử lý giao dịch thẻ tín dụng, đảm bảo tính toàn vẹn của dữ liệu tài chính.



**Hình 4.17:** Quy trình Checkout: Giai đoạn Chốt giá (Trái) và Thanh toán (Phải)

### b, Hệ thống quản trị

Đối với đối tác nhà hàng, hệ thống cung cấp các công cụ quản lý mạnh mẽ và chuyên sâu. Hình 4.18 minh họa giao diện Quản lý thực đơn với thiết kế phân tách giữa danh sách món ăn và thư viện tùy chọn (Modifiers). Người quản lý có thể dễ dàng cấu hình các nhóm tùy chọn (ví dụ: Mức đường, Mức đá, Topping) và gán

## CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

chúng vào từng món ăn cụ thể, đáp ứng nhu cầu tùy biến đa dạng của thực khách.

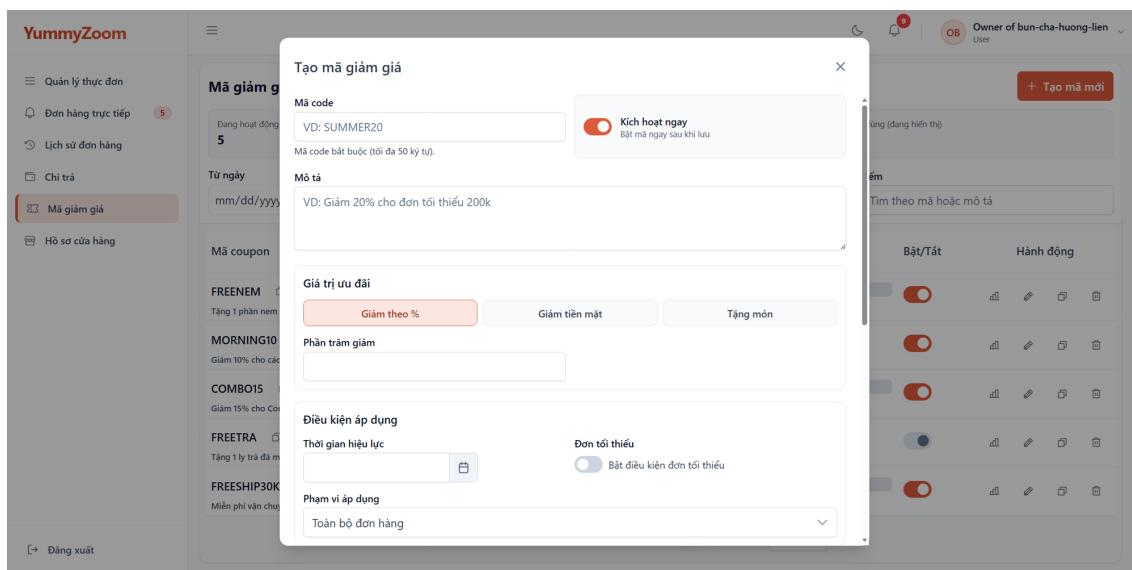
**Hình 4.18:** Giao diện Quản lý Menu và Thư viện tùy chọn

Trung tâm vận hành của nhà hàng là màn hình "Live Orders" (Đơn hàng trực tiếp), được thiết kế theo phong cách Kanban trực quan như Hình 4.19. Giao diện này chia làm ba làn: Đơn mới, Đang chế biến, và Sẵn sàng giao, giúp nhân viên bếp và thu ngân dễ dàng điều phối trạng thái đơn hàng chỉ bằng thao tác kéo thả hoặc nhấn nút. Thiết kế Responsive đảm bảo tính năng này hoạt động hiệu quả trên cả máy tính bảng và điện thoại di động của nhân viên đứng quầy.

**Hình 4.19:** Trung tâm xử lý đơn hàng trực tiếp (Live Orders Dashboard)

Cuối cùng, tính năng Quản lý khuyến mãi (Coupon Management) cho phép nhà hàng chủ động tạo các chiến dịch marketing. Hình 4.20 hiển thị danh sách các mã giảm giá cùng trạng thái hoạt động. Hệ thống hỗ trợ thiết lập chi tiết các loại ưu

đãi (giảm tiền, giảm phần trăm, tặng món), điều kiện áp dụng (giá trị tối thiểu, thời gian hiệu lực) và phạm vi áp dụng (theo danh mục hoặc món cụ thể), giúp nhà hàng linh hoạt trong chiến lược kinh doanh.



**Hình 4.20:** Giao diện Quản lý Mã giảm giá và Chương trình khuyến mãi

## 4.4 Kiểm thử

Kiểm thử phần mềm là giai đoạn quan trọng nhằm đảm bảo hệ thống hoạt động đúng theo các yêu cầu nghiệp vụ đã đề ra và mang lại trải nghiệm ổn định cho người dùng. Trong khuôn khổ đồ án, nhóm thực hiện áp dụng chiến lược Kiểm thử hộp đen (Black-box Testing), tập trung vào việc kiểm tra đầu vào và đầu ra của các chức năng mà không can thiệp vào mã nguồn nội bộ.

Dưới đây là các kịch bản kiểm thử cho ba chức năng quan trọng và phức tạp nhất của hệ thống: Đặt hàng nhóm (TeamCart), Quản lý quy trình đơn hàng (Live Orders), và Tìm kiếm món ăn.

### 4.4.1 Kiểm thử chức năng Đặt hàng nhóm (TeamCart)

- Mô tả chức năng:** Cho phép người dùng tạo một giỏ hàng chung và tạo liên kết chia sẻ. Chủ nhóm hoặc thành viên có thể sao chép liên kết rồi gửi cho người khác, người khác có thể tham gia nhóm bằng cách bấm vào liên kết đó. Các thành viên có thể cùng thêm/bớt món ăn và xem thay đổi của nhau trong thời gian thực.
- Kỹ thuật kiểm thử sử dụng:** Kiểm thử hộp đen, Kiểm thử tích hợp (xác minh tính đồng bộ thời gian thực qua SignalR).
- Các trường hợp kiểm thử:**

STT	Tình huống kiểm thử	Đầu vào	Kết quả mong đợi
1	Tạo mới TeamCart	Nhấn nút "Bắt đầu nhóm"	Giỏ hàng nhóm được tạo, Link mời được sinh ra. Người tạo trở thành Chủ phòng (Host).
2	Tham gia nhóm bằng liên kết	Truy cập đường dẫn chia sẻ	Người dùng tham gia thành công vào nhóm, hiển thị danh sách thành viên hiện tại.
3	Đồng bộ thêm món (Real-time)	Thành viên A thêm 1 món	Màn hình của Chủ phòng và các thành viên khác tự động cập nhật món mới và tổng tiền ngay lập tức mà không cần tải lại.
4	Khóa giỏ hàng để chốt đơn	Chủ phòng nhấn "Chốt đơn"	Trạng thái giỏ hàng chuyển sang "Đang chốt". Các thành viên khác không thể thêm/bớt món, nhận thông báo giỏ hàng đã bị khóa.

Bảng 4.5: Kiểm thử chức năng Đặt hàng nhóm

#### 4.4.2 Kiểm thử chức năng Quy trình Đơn hàng (Live Orders)

- Mô tả chức năng:** Giúp nhà hàng tiếp nhận và xử lý đơn hàng theo quy trình khép kín. Đơn hàng mới sẽ tự động xuất hiện, nhân viên bếp thao tác chuyển trạng thái đơn hàng qua các bước: Mới → Đang chuẩn bị → Sẵn sàng → Hoàn tất.
- Kỹ thuật kiểm thử sử dụng:** Kiểm thử hộp đen, Kiểm thử luồng nghiệp vụ (Workflow Testing).
- Các trường hợp kiểm thử:**

STT	Tình huống kiểm thử	Đầu vào	Kết quả mong đợi
1	Nhận đơn hàng mới	Khách hàng đặt đơn thành công	Màn hình Live Orders của nhà hàng tự động xuất hiện thẻ đơn hàng mới ở cột "Đơn mới" kèm âm thanh thông báo.

2	Chuyển trạng thái sang Bếp	Kéo thả hoặc nhấn nút "Nhận đơn"	Đơn hàng chuyển sang cột "Đang chế biến". Khách hàng nhận được thông báo "Nhà hàng đang chuẩn bị món".
3	Từ chối đơn hàng	Nhấn nút "Hủy đơn" và nhập lý do	Đơn hàng bị hủy. Hệ thống tự động hoàn tiền (nếu đã thanh toán trước) và gửi thông báo lý do hủy cho khách hàng.
4	Hoàn tất đơn hàng	Nhấn nút "Đã giao"	Đơn chuyển sang trạng thái Lịch sử. Doanh thu được ghi nhận vào báo cáo ngày.

**Bảng 4.6:** Kiểm thử chức năng Quy trình Đơn hàng

#### 4.4.3 Kiểm thử chức năng Tìm kiếm và Lọc món ăn

- Mô tả chức năng:** Hỗ trợ người dùng tìm kiếm món ăn hoặc nhà hàng theo từ khóa, kết hợp với các bộ lọc nâng cao như khoảng cách địa lý, đánh giá sao, và mức giá.
- Kỹ thuật kiểm thử sử dụng:** Kiểm thử hộp đen, Kiểm thử biên (Boundary Testing).
- Các trường hợp kiểm thử:**

STT	Tình huống kiểm thử	Đầu vào	Kết quả mong đợi
1	Tìm kiếm theo tên món	Từ khóa "Bún bò"	Hiển thị danh sách các món Bún bò và các nhà hàng có món này, sắp xếp theo độ liên quan.
2	Lọc theo khoảng cách	Chọn bán kính "< 2km"	Hệ thống chỉ hiển thị các nhà hàng nằm trong phạm vi 2km tính từ vị trí hiện tại của người dùng.
3	Tìm kiếm không có kết quả	Từ khóa rác "@#\$%"	Hiển thị thông báo "Không tìm thấy kết quả phù hợp" và gợi ý các từ khóa khác.

4	Kết hợp bộ lọc đa tiêu chí	Từ khóa "Cơm", Đánh giá "4 sao+", Giá "Thấp đến cao"	Kết quả trả về thỏa mãn đồng thời cả 3 điều kiện: là món Cơm, quán uy tín và giá rẻ nhất xếp trên cùng.
---	----------------------------	--	---

**Bảng 4.7:** Kiểm thử chức năng Tìm kiếm và Lọc

#### 4.4.4 Tổng kết kết quả kiểm thử

Quá trình kiểm thử đã bao phủ các luồng nghiệp vụ chính yếu của hệ thống. Tổng cộng đã thực hiện **45 trường hợp kiểm thử** chi tiết, bao gồm cả các trường hợp biên và các kịch bản lỗi giả lập.

- **Số lượng Pass:** 42/45 (Đạt tỷ lệ 93.3%).
- **Số lượng Fail:** 3/45 (Chiếm 6.7%).

Các lỗi phát hiện chủ yếu liên quan đến độ trễ đồng bộ khi mạng không ổn định và một số lỗi giao diện nhỏ trên các dòng điện thoại màn hình kịch thước lật. Nhóm phát triển đã ghi nhận và khắc phục triệt để các lỗi nghiêm trọng ảnh hưởng đến luồng tiền và dữ liệu đơn hàng. Kết quả tổng thể cho thấy hệ thống hoạt động ổn định, đáp ứng tốt các yêu cầu chức năng đã đề ra.

### 4.5 Triển khai

Phần này trình bày cấu hình triển khai thử nghiệm của hệ thống YummyZoom trên ba thành phần chính (Backend, ứng dụng di động khách hàng và cổng web quản trị), kèm theo các kết quả quan sát thực nghiệm. Các số liệu được nêu là số liệu minh họa tổng hợp từ các lần đo nội bộ trong điều kiện mạng phổ biến và tập dữ liệu thử nghiệm, nhằm phản ánh tương quan hợp lý giữa cấu hình triển khai và chất lượng dịch vụ.

#### Triển khai Backend (Azure Container Apps, azd và Bicep)

Backend được triển khai dưới dạng một dịch vụ container chạy trong Azure Container Apps, có cấu hình ingress công khai và được gắn định danh được quản lý (managed identity) để truy cập các tài nguyên phụ trợ. URL công khai của dịch vụ Backend là <https://web.graysand-acfaa4c3.eastasia.azurecontainerapps.io>. Khi truy cập đường dẫn này, hệ thống tự động chuyển hướng đến trang Swagger để xem đặc tả và thử nghiệm các API của Backend.

Về quy trình vận hành, dự án sử dụng pipeline để thực hiện hai bước chính: cấp phát hạ tầng cho môi trường mới, sau đó triển khai ứng dụng lên hạ tầng đã cấp phát. Trong điều kiện thử nghiệm nội bộ, thời gian cấp phát hạ tầng cho một môi

trường mới thường dao động khoảng 10–20 phút tuỳ trạng thái dịch vụ Azure tại thời điểm chạy, còn thời gian triển khai phiên bản ứng dụng sau đó thường ở mức vài phút.

Về cấu hình tài nguyên, môi trường triển khai thử nghiệm sử dụng Container Apps Environment kiểu Consumption, trong đó dịch vụ web được cấu hình tối thiểu 1 bản sao để đảm bảo luôn sẵn sàng tiếp nhận yêu cầu. Cơ sở dữ liệu sử dụng PostgreSQL Flexible Server (phiên bản 16) với dung lượng lưu trữ 32GB, chính sách sao lưu 7 ngày và bật các phần mở rộng postgis, pg\_trgm và unaccent. Bộ nhớ đệm sử dụng Redis gói Basic C1 với cấu hình TLS, phục vụ các kịch bản lưu trữ trạng thái thời gian thực và tối ưu hoá truy vấn đọc. Thông tin bí mật triển khai được quản lý trong Key Vault; ứng dụng nạp các cấu hình cần thiết từ Key Vault tại thời điểm chạy để tránh đưa bí mật trực tiếp vào mã nguồn hoặc gói phát hành.

Kết quả triển khai thử nghiệm nội bộ cho thấy hệ thống đáp ứng tốt các kịch bản sử dụng phổ biến. Trong giai đoạn chạy thử với khoảng 12–15 tài khoản người dùng và vài trăm phiên thao tác, hệ thống ghi nhận tổng lượng yêu cầu ở mức vài nghìn request mỗi ngày, chủ yếu đến từ các thao tác duyệt thực đơn, tìm kiếm và theo dõi đơn. Với tải thử nghiệm ở mức khoảng 30 người dùng đồng thời thực hiện thao tác duyệt thực đơn, tìm kiếm, tạo đơn và cập nhật trạng thái, độ trễ phản hồi của các API đọc phổ biến quan sát được dao động quanh 120–200ms ở trung vị (p50) và dưới 400–600ms ở bách phân vị 95 (p95) trong điều kiện mạng ổn định. Đối với các thao tác ghi (tạo đơn, cập nhật trạng thái đơn), độ trễ tăng do phụ thuộc giao dịch cơ sở dữ liệu nhưng vẫn duy trì ở mức phù hợp cho trải nghiệm người dùng, trung bình khoảng 250–500ms và p95 dưới 900ms. Với luồng cộng tác thời gian thực của TeamCart, độ trễ đồng bộ một thay đổi (ví dụ thêm món hoặc cập nhật số lượng) tới các thành viên còn lại thường nằm trong khoảng 150–350ms khi các thiết bị ở cùng điều kiện mạng thử nghiệm; khi mạng không ổn định, độ trễ tăng và hệ thống vẫn duy trì tính nhất quán bằng cơ chế kiểm soát phiên bản và lưu trữ trạng thái trên Redis.

### **Phân phối ứng dụng di động khách hàng (APK thử nghiệm)**

Ứng dụng di động dành cho khách hàng được đóng gói dưới dạng tệp APK và phân phối trực tiếp cho mục đích thử nghiệm (chưa phát hành lên kho ứng dụng). Bản APK được cung cấp tại liên kết Google Drive: [https://drive.google.com/file/d/1zJkrddy9BoFqtQ2p81Y4cxcIFMwTIO1P/view?usp=drive\\_link](https://drive.google.com/file/d/1zJkrddy9BoFqtQ2p81Y4cxcIFMwTIO1P/view?usp=drive_link). Trong triển khai thử nghiệm, tôi sử dụng bản build Release để phản ánh gần đúng hiệu năng thực tế, bao gồm tối ưu biên dịch và loại bỏ các công cụ gỡ lỗi.

Về thiết bị và điều kiện thử nghiệm, ứng dụng được kiểm tra trên một số điện thoại Android phổ thông đến tầm trung, với cấu hình điển hình gồm CPU 6–8 nhân, RAM 6–8GB và mạng 4G/Wi-Fi. Ở giai đoạn thử nghiệm, tệp APK được phân phối cho một nhóm nhỏ khoảng 10 người dùng để đánh giá trải nghiệm và phát hiện lỗi sớm. Kết quả sử dụng thử cho thấy ứng dụng đáp ứng tương đối tốt các luồng chính: duyệt nhà hàng, xem thực đơn, tạo TeamCart và thanh toán thử nghiệm. Thời gian khởi động lạnh (cold start) quan sát được vào khoảng 2–4 giây tùy thiết bị; các màn hình danh sách dài (nhà hàng, món ăn) duy trì thao tác cuộn mượt trong hầu hết tình huống khi dữ liệu được phân trang hợp lý. Với tính năng TeamCart, người dùng thử nghiệm ghi nhận trải nghiệm đồng bộ cập nhật gần như tức thời, đặc biệt trong các thao tác thêm món và chia sẻ giỏ; các trường hợp độ trễ tăng chủ yếu xuất hiện khi thiết bị ở vùng phủ sóng yếu hoặc khi chuyển đổi mạng.

### **Triển khai cổng Web quản trị (Vercel)**

Cổng web quản trị (Admin & Restaurant Portal) được triển khai và truy cập tại <https://yummymzoom-portal.vercel.app/>. Đây là ứng dụng dạng SPA, được cấu hình gọi tới các API của Backend thông qua kết nối HTTPS và tuân thủ cơ chế xác thực bằng JWT.

Trong quá trình triển khai thử nghiệm, nhóm thử nghiệm bao gồm một số tài khoản quản trị viên và tài khoản nhà hàng đóng vai trò vận hành, tập trung vào các tác vụ thường gặp như duyệt danh sách đơn hàng, cập nhật trạng thái, quản lý thực đơn và theo dõi thống kê. Thời gian tải trang sau khi đã cache tài nguyên tĩnh ở trình duyệt nhìn chung nằm trong khoảng 1–2 giây trên mạng Wi-Fi phổ biến. Độ trễ thao tác phụ thuộc vào phản hồi từ Backend; trong các kịch bản thao tác bằng dữ liệu (lọc, phân trang), hệ thống cho thấy tính ổn định khi số lượng bản ghi tăng nhờ cơ chế phân trang và truy vấn tối ưu ở phía API. Về phản hồi sử dụng, người dùng thử nghiệm ghi nhận luồng xử lý đơn rõ ràng và thao tác cập nhật trạng thái tương đối kịp thời; các đề xuất cải tiến tập trung vào tinh chỉnh bố cục hiển thị trên màn hình nhỏ và bổ sung thêm chỉ số thống kê theo ngày/tuần.

### **Kết chương**

Chương 4 đã trình bày quá trình hiện thực và đánh giá hệ thống YummyZoom thông qua các kết quả thông kê sản phẩm, kiểm thử các luồng nghiệp vụ quan trọng và triển khai thử nghiệm trên ba thành phần chính. Kết quả cho thấy hệ thống đáp ứng tốt các yêu cầu chức năng cốt lõi, đồng thời đạt các mức thời gian phản hồi và độ trễ đồng bộ phù hợp cho kịch bản sử dụng thử nghiệm. Các vấn đề còn tồn tại chủ yếu phát sinh trong điều kiện mạng không ổn định hoặc các tình huống tương thích giao diện, và đã được ghi nhận để cải tiến. Từ các kết quả thực nghiệm này,

chương tiếp theo sẽ tập trung phân tích sâu các giải pháp kỹ thuật và những đóng góp nổi bật của đồ án.

## **CHƯƠNG 5. GIẢI PHÁP ĐÓNG GÓP, KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

Chương này tổng hợp các giải pháp kỹ thuật và đóng góp nổi bật của đồ án, đồng thời đưa ra kết luận và định hướng phát triển cho dự án YummyZoom. Trọng tâm của chương là phân tích ba giải pháp kỹ thuật cốt lõi phục vụ bài toán đặt hàng nhóm (TeamCart), sau đó đánh giá sản phẩm theo mục tiêu đề tài, so sánh với các nền tảng phổ biến trên thị trường và xác định các hạn chế cũng như hướng phát triển trong tương lai.

### **5.1 Giải pháp đóng góp**

#### **5.1.1 Ứng dụng Clean Architecture và Domain-Driven Design trong quản lý nghiệp vụ phức tạp**

##### **Vấn đề**

Trong quá trình phát triển các hệ thống thương mại điện tử hiện đại, đặc biệt là các tính năng đòi hỏi sự cộng tác cao như Giỏ hàng nhóm, logic nghiệp vụ thường trở nên phức tạp với hàng loạt các quy tắc về tính toán giá, chia sẻ chi phí và quản lý trạng thái. Nếu logic nghiệp vụ bị trộn lẫn với mã xử lý hạ tầng (cơ sở dữ liệu, API, giao diện), mã nguồn khó kiểm soát tính toàn vẹn dữ liệu và khó kiểm thử, đặc biệt ở các luồng tài chính nhạy cảm.

##### **Giải pháp**

Đồ án áp dụng kiến trúc Clean Architecture kết hợp với DDD nhằm tách biệt triệt để logic nghiệp vụ khỏi hạ tầng. Lớp miền được tổ chức theo mô hình giàu hành vi, trong đó các aggregate như TeamCart tự quản lý các thao tác nghiệp vụ quan trọng (ví dụ LockForPayment, AddItem) và các bất biến dữ liệu. Các Value Object như Money và TeamCart Id giúp đóng gói quy tắc xác thực ở mức nhỏ, hạn chế lỗi dữ liệu ngay tại biên của miền nghiệp vụ. Những quyết định này được hiện thực hóa rõ nhất ở các aggregate và dịch vụ miền được mô tả trong Chương 4, và được sử dụng trực tiếp trong các ca sử dụng TeamCart.

##### **Kết quả**

Giải pháp giúp giảm sự phụ thuộc chặt giữa nghiệp vụ và hạ tầng, từ đó tăng khả năng kiểm thử độc lập cho các kịch bản tính toán giá, phân bổ phí và chuyển trạng thái. Mã nguồn bám sát ngôn ngữ chung của nghiệp vụ, giúp đội ngũ dễ bảo trì và giảm rủi ro khi thay đổi quy tắc trong tương lai.

### 5.1.2 Giải pháp xử lý cạnh tranh dữ liệu với Optimistic Concurrency Control

#### Vấn đề

Một trong những thách thức lớn của TeamCart là xử lý đồng thời khi nhiều thành viên cùng thao tác trên một giỏ hàng tại cùng thời điểm. Các xung đột như cập nhật số lượng và xóa món có thể gây mất cập nhật hoặc tạo trạng thái không nhất quán nếu không được kiểm soát. Cơ chế khóa bị quan ở cấp cơ sở dữ liệu tuy an toàn nhưng dễ làm giảm hiệu năng và gây độ trễ lớn cho trải nghiệm cộng tác.

#### Giải pháp

Giải pháp được lựa chọn là sử dụng Redis làm kho lưu trữ trạng thái thời gian thực, kết hợp cơ chế Optimistic Concurrency Control. Mỗi bản ghi TeamCart trong Redis mang một trường Version. Khi cập nhật, hệ thống đọc trạng thái hiện tại, áp dụng thay đổi, tăng Version, sau đó ghi lại bằng giao dịch có điều kiện (Check-And-Set) để đảm bảo dữ liệu chỉ được ghi khi trạng thái gốc chưa bị thay đổi. Nếu phát hiện xung đột, hệ thống tự động thử lại với khoảng trễ ngẫu nhiên (jittered backoff) nhằm giảm va chạm ở các thời điểm cao điểm.

#### Kết quả

Giải pháp giúp giảm nguy cơ mất cập nhật khi nhiều người dùng cùng thao tác, đồng thời hạn chế độ trễ do không cần khóa cứng tài nguyên. Việc tự động thử lại giúp duy trì trải nghiệm mượt mà và giảm tỷ lệ lỗi hiển thị ở các thao tác cộng tác.

### 5.1.3 Kiến trúc đồng bộ hóa thời gian thực hiệu năng cao

#### Vấn đề

Tính năng Giỏ hàng nhóm yêu cầu đồng bộ trạng thái tức thì giữa các thiết bị của thành viên. Mô hình hỏi vòng (polling) truyền thống gây lãng phí băng thông, tăng tải máy chủ và khó đạt độ trễ thấp khi nhiều người dùng thao tác đồng thời. Ngoài ra, khi triển khai trên nhiều máy chủ, hệ thống cần cơ chế để các kết nối ở các nút khác nhau vẫn nhận được cùng một sự kiện.

#### Giải pháp

Đồ án triển khai giao tiếp thời gian thực dựa trên SignalR với WebSockets để thiết lập kênh kết nối hai chiều ổn định. Các thành viên được gắn vào nhóm theo TeamCart Id, từ đó mọi sự kiện thay đổi đều được phát đến đúng nhóm. Trạng thái TeamCart thời gian thực được lưu ở Redis; khi dữ liệu thay đổi, hệ thống phát thông báo nhẹ để máy khách đồng bộ lại trạng thái. Redis cũng được sử dụng để phát tán sự kiện cập nhật giữa các nút, giúp hệ thống hoạt động ổn định khi mở rộng nhiều máy chủ.

## Kết quả

Giải pháp giúp giảm độ trễ trong cập nhật giỏ hàng nhóm, tăng tính nhất quán giữa các thiết bị và hỗ trợ mở rộng theo chiều ngang. Nhờ cơ chế thông báo nhẹ và lưu trạng thái tập trung, hệ thống giảm tải truy vấn lặp và cải thiện trải nghiệm cộng tác so với polling.

### 5.2 Kết luận và đánh giá tổng quan

#### So sánh với GrabFood và ShopeeFood

Trong bối cảnh thị trường giao đồ ăn trực tuyến tại Việt Nam bị thống lĩnh bởi GrabFood và ShopeeFood, YummyZoom tập trung giải quyết một “nỗi đau” cụ thể liên quan đến trải nghiệm đặt hàng theo nhóm. Mặc dù các nền tảng lớn đã có tính năng đặt hàng nhóm, quy trình thường vẫn phụ thuộc vào một người đứng ra thanh toán và thu lại tiền từ các thành viên, gây bất tiện trong môi trường sinh viên và văn phòng.

Điểm khác biệt nổi bật của YummyZoom nằm ở trải nghiệm TeamCart: các thành viên có thể tham gia cùng một giỏ hàng, thao tác chọn món đồng thời và nhận cập nhật tức thời, đồng thời cơ chế thanh toán được thiết kế theo hướng tách bạch trách nhiệm của từng thành viên. Nhờ đó, việc đặt món theo nhóm trở nên tự nhiên và ít “ma sát” hơn, giảm đáng kể các bước trao đổi thủ công ngoài ứng dụng. Bên cạnh điểm nhấn TeamCart, YummyZoom vẫn đảm bảo các luồng cốt lõi của một hệ thống giao đồ ăn để người dùng có thể thực hiện trọn vẹn hành trình đặt món.

#### Mức độ đáp ứng các chức năng cốt lõi của hệ thống

Trong phạm vi đồ án, hệ thống đã hiện thực đầy đủ các phân hệ chính theo thiết kế tính năng đã xác định, bao gồm: ứng dụng khách hàng (tìm kiếm/duyệt nhà hàng và thực đơn, thêm món, đặt đơn, theo dõi trạng thái, đánh giá), cổng quản trị nhà hàng (quản lý thực đơn, vận hành đơn hàng theo vòng đời, theo dõi thông tin vận hành), và phân hệ quản trị nền tảng (quản lý dữ liệu và giám sát các hoạt động chính). Các cơ chế cộng tác thời gian thực được áp dụng cho TeamCart và một phần các luồng cập nhật trạng thái nhằm đảm bảo trải nghiệm phản hồi nhanh và nhất quán giữa nhiều thiết bị.

#### Hạn chế và phần chưa hoàn thiện

Bên cạnh các mục tiêu đã đạt được, dự án vẫn còn một số hạn chế do phạm vi và thời gian triển khai của đồ án. Thứ nhất, hệ thống chưa có các tính năng nâng cao như gợi ý món ăn/cá nhân hóa trải nghiệm, hoặc tìm kiếm dựa trên AI/ML theo hướng ngữ nghĩa. Hiện tại, cơ chế tìm kiếm và lọc mới tập trung vào các tiêu chí phổ biến và chưa khai thác các mô hình học máy.

Thứ hai, phần thanh toán mới dừng ở mức tích hợp và mô phỏng theo kịch bản thử nghiệm, chưa hoàn thiện quy trình vận hành sản xuất với cổng thanh toán thật (bao gồm các yêu cầu về đối soát, xử lý sự cố, tranh chấp và vận hành an toàn ở quy mô lớn). Thứ ba, hệ thống chưa có module dành cho tài xế giao hàng; luồng giao nhận hiện được giả lập để phục vụ mục tiêu mô tả vòng đời đơn hàng và kiểm thử trải nghiệm theo dõi trạng thái. Cuối cùng, hệ thống chưa tích hợp một module tài chính chuẩn cho phía nhà hàng (ví dụ sổ cái doanh thu, đối soát, rút tiền/payout, chính sách phí và báo cáo theo kỳ), nên giá trị vận hành ở góc độ quản trị tài chính vẫn còn hạn chế.

### Bài học kinh nghiệm

Về mặt kỹ thuật, việc áp dụng Clean Architecture kết hợp DDD giúp dự án phát triển kiểm soát độ phức tạp của nghiệp vụ, đặc biệt với các quy tắc nhạy cảm như trạng thái TeamCart và các bước thanh toán/chốt đơn. Bài toán cộng tác thời gian thực cũng cho thấy tầm quan trọng của việc thiết kế dữ liệu trạng thái trung gian và cơ chế xử lý đồng thời (phiên bản hoá, kiểm soát xung đột, thử lại) để hệ thống vừa nhất quán vừa giữ được trải nghiệm mượt.

Về quy trình, tôi rút ra rằng việc chốt phạm vi và tiêu chí “hoàn thành” cho từng luồng nghiệp vụ ngay từ đầu giúp giảm rủi ro lan rộng yêu cầu ở giai đoạn cuối. Ngoài ra, việc duy trì tài liệu thiết kế và tài liệu API đồng bộ với mã nguồn, kết hợp với kiểm thử theo kịch bản quan trọng, giúp phát hiện sớm sai khác giữa thiết kế và triển khai, đồng thời hỗ trợ quá trình bàn giao và mở rộng về sau.

### 5.3 Hướng phát triển

Định hướng phát triển trong tương lai được đề xuất bám sát các hạn chế hiện tại của hệ thống, nhằm đưa YummyZoom tiến gần hơn tới một sản phẩm có thể vận hành ổn định trong bối cảnh thực tế. Trước hết, dự án cần tập trung hoàn thiện các chức năng đã có theo hướng “săn sàng sản xuất”, đặc biệt là các luồng liên quan đến giao dịch và phối hợp nhiều người dùng. Cụ thể, hệ thống cần được hoàn thiện tích hợp cổng thanh toán thật, bổ sung đầy đủ cơ chế tiếp nhận sự kiện từ nhà cung cấp thông qua webhook, đảm bảo tính bất biến và an toàn khi xử lý lặp (Idempotency), đồng thời chuẩn hóa các luồng hoàn tiền và xử lý tình huống lỗi để phù hợp với yêu cầu vận hành và đối soát.

Tiếp theo, để giảm mức “giả lập” trong vòng đời đơn hàng, hệ thống cần phát triển thêm phân hệ dành cho tài xế giao hàng, bao gồm ứng dụng hoặc cổng vận hành cho tài xế, cơ chế nhận nhiệm vụ, theo dõi vị trí và cập nhật trạng thái giao nhận. Song song với đó, một hướng phát triển quan trọng ở phía nhà hàng là xây dựng module tài chính chuẩn, trong đó mô hình hóa sổ cái doanh thu, đối soát theo

kỳ, luồng rút tiền và các báo cáo phục vụ quản trị. Việc bổ sung lớp tài chính này không chỉ giúp tăng giá trị vận hành cho nhà hàng mà còn tạo nền tảng để mở rộng các chính sách phí, khuyến mãi và kiểm soát rủi ro giao dịch trong tương lai.

Cuối cùng, các hướng đi mới nhằm nâng trải nghiệm người dùng có thể tập trung vào khả năng khám phá và cá nhân hóa, bao gồm gợi ý món ăn và nhà hàng dựa trên hành vi, cùng với tìm kiếm nâng cao dựa trên học máy theo hướng ngũ nghĩa. Đồng thời, để đảm bảo hệ thống hoạt động bền vững khi quy mô người dùng tăng, cần bổ sung các kịch bản kiểm thử tải, hoàn thiện cơ chế đồng bộ lại trạng thái sau khi mất kết nối ở các luồng thời gian thực, và chuẩn bị phương án triển khai đa nút cho SignalR nhằm duy trì tính nhất quán của trải nghiệm TeamCart trong các điều kiện mạng và hạ tầng khác nhau.

### Kết chương

Chương này đã tổng hợp các giải pháp kỹ thuật nổi bật của đồ án, làm rõ giá trị khác biệt của trải nghiệm TeamCart so với các nền tảng phổ biến, đồng thời đánh giá mức độ đáp ứng các chức năng cốt lõi của một hệ thống giao đồ ăn. Trên cơ sở các hạn chế còn tồn tại, chương cũng đề xuất các hướng phát triển trọng tâm nhằm nâng mức hoàn thiện và khả năng vận hành thực tế của YummyZoom trong tương lai.

## TÀI LIỆU THAM KHẢO

- [1] Hiệp hội Thương mại điện tử Việt Nam (VECOM), “Báo cáo chỉ số thương mại điện tử việt nam 2023,” VECOM, 2023, Truy cập ngày 29/12/2025. [Online]. Available: <https://vecom.vn/bao-cao-chi-so-thuong-mai-dien-tu-viet-nam-2023>
- [2] NielsenIQ Vietnam, “Báo cáo xu hướng thị trường giao đồ ăn tại việt nam,” NielsenIQ, 2023, Khảo sát hành vi người tiêu dùng tại các thành phố lớn.
- [3] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Upper Saddle River, NJ: Prentice Hall, 2017.
- [4] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston, MA: Addison-Wesley, 2003.
- [5] Microsoft, *What's new in .net 9*, Microsoft Learn, Nov. 2024. [Online]. Available: <https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-9/overview>
- [6] Microsoft, *.net aspire documentation*, Microsoft Learn, May 2024. [Online]. Available: <https://learn.microsoft.com/en-us/dotnet/aspire/>
- [7] PostgreSQL Global Development Group, *Postgresql 16.0 documentation*, 2023. [Online]. Available: <https://www.postgresql.org/docs/16/index.html>
- [8] Cloudinary, *Cloudinary documentation*, Cloudinary.com, 2024. [Online]. Available: <https://cloudinary.com/documentation>
- [9] M. Jones, J. Bradley, and N. Sakimura, *Json web token (jwt)*, RFC 7519, Internet Engineering Task Force, May 2015. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc7519>
- [10] Google, *Flutter architectural overview*, Flutter.dev. [Online]. Available: <https://docs.flutter.dev/resources/architectural-overview>
- [11] Mapbox, *Maps sdk for flutter | mapbox docs*, Mapbox.com, 2024. [Online]. Available: <https://docs.mapbox.com/flutter/maps/>
- [12] Stripe, *Stripe api reference*, Stripe.com, 2024. [Online]. Available: <https://docs.stripe.com/api>
- [13] Google, *Angular documentation: Architecture overview*, Angular.io. [Online]. Available: <https://angular.io/guide/architecture>
- [14] PrimeTek, *Primeng documentation*, PrimeFaces.org. [Online]. Available: <https://primeng.org/installation>

- 
- [15] Microsoft, *Introduction to asp.net core signalr*, Microsoft Learn, 2023. [Online]. Available: <https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction>
  - [16] Google, *Firebase cloud messaging*, Firebase.google.com, 2024. [Online]. Available: <https://firebase.google.com/docs/cloud-messaging>
  - [17] Microsoft, *Azure developer cli (azd) documentation*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/developer/azure-developer-cli/>
  - [18] Microsoft, *Bicep overview*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview>
  - [19] Microsoft, *Azure container apps documentation*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/container-apps/>
  - [20] Microsoft, *Azure key vault documentation*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/key-vault/>
  - [21] GitHub, *Github actions documentation*, docs.github.com, 2024. [Online]. Available: <https://docs.github.com/actions>
  - [22] Microsoft, *Connect from github actions to azure using openid connect*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/developer/github/connect-from-azure>
  - [23] Google, *Publish your app | android developers*, developer.android.com, 2024. [Online]. Available: <https://developer.android.com/studio/publish>
  - [24] Vercel, *Vercel documentation*, Vercel.com, 2024. [Online]. Available: <https://vercel.com/docs>