

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

## **ĐỒ ÁN TỐT NGHIỆP**

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

**HOÀNG NGUYỄN 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 kí GVHD**

**Khoa:** Kỹ thuật máy tính

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

**HÀ NỘI, 01/2026**

## LỜI CẢM ƠN

Lời cảm ơn của sinh viên (SV) tới người yêu, gia đình, bạn bè, thầy cô, và chính bản thân mình vì đã chăm chỉ và quyết tâm thực hiện ĐATN để đạt kết quả tốt nhất, nên viết phần cảm ơn ngắn gọn, tránh dùng các từ sáo rỗng, giới hạn trong khoảng 100-150 từ.

## TÓM TẮT NỘI DUNG ĐỀ ÁN

Sinh viên viết tóm tắt ĐATN của mình trong mục này, với 200 đến 350 từ. Theo trình tự, các nội dung tóm tắt cần có: (i) Giới thiệu vấn đề (tại sao có vấn đề đó, hiện tại được giải quyết chưa, có những hướng tiếp cận nào, các hướng này giải quyết như thế nào, hạn chế là gì), (ii) Hướng tiếp cận sinh viên lựa chọn là gì, vì sao chọn hướng đó, (iii) Tổng quan giải pháp của sinh viên theo hướng tiếp cận đã chọn, và (iv) Đóng góp chính của ĐATN là gì, kết quả đạt được sau cùng là gì. Sinh viên cần viết thành đoạn văn, không được viết ý hoặc gạch đầu dòng.

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

## **ABSTRACT**

Mục này khuyến khích sinh viên viết lại mục “Tóm tắt” đồ án tốt nghiệp ở trang trước bằng tiếng Anh. Phần này phải có đầy đủ các nội dung như trong phần tóm tắt bằng tiếng Việt. Sinh viên không nhất thiết phải trình bày mục này.

Nhưng nếu lựa chọn trình bày, sinh viên cần đảm bảo câu từ và ngữ pháp chuẩn tắc, nếu không sẽ có tác dụng ngược, gây phản cảm.

## 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.....	2
1.3 Định hướng giải pháp.....	3
1.4 Bố cục đồ án .....	5
<b>CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU.....</b>	<b>10</b>
2.1 Khảo sát hiện trạng .....	10
2.1.1 Bối cảnh và xu hướng thị trường .....	10
2.1.2 Phân tích các ứng dụng giao đồ ăn hiện có .....	11
2.1.3 Nhu cầu và thách thức của người dùng .....	13
2.2 Tổng quan chức năng .....	15
2.2.1 Biểu đồ use case tổng quát .....	16
2.2.2 Biểu đồ use case phân rã "Quản lý tài khoản" .....	18
2.2.3 Biểu đồ use case phân rã "Đặt hàng cá nhân" .....	19
2.2.4 Biểu đồ use case phân rã "TeamCart" .....	20
2.2.5 Biểu đồ use case phân rã "Quản lý thực đơn" .....	21
2.2.6 Biểu đồ use case phân rã "Duyệt đăng ký nhà hàng" .....	22
2.2.7 Quy trình nghiệp vụ .....	22
2.3 Đặc tả chức năng .....	25
2.3.1 Đặc tả Use Case: Đăng ký nhà hàng.....	25
2.3.2 Đặc tả Use Case: Duyệt đăng ký nhà hàng.....	27
2.3.3 Đặc tả Use Case: Quản lý thực đơn.....	29
2.3.4 Đặc tả Use Case: Tìm kiếm nhà hàng.....	31
2.3.5 Đặc tả Use Case: Đặt hàng cá nhân.....	33

2.3.6 Đặc tả Use Case: Khởi tạo TeamCart .....	36
2.3.7 Đặc tả Use Case: Tham gia TeamCart .....	38
2.3.8 Đặc tả Use Case: Chốt đơn TeamCart .....	40
2.3.9 Đặc tả Use Case: Xử lý đơn hàng.....	44
2.3.10 Đặc tả Use Case: Đánh giá nhà hàng .....	46
2.4 Yêu cầu phi chức năng .....	47
<b>CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG.....</b>	<b>50</b>
3.1 Kiến trúc hệ thống và Ứng dụng Backend .....	50
3.1.1 Kiến trúc Clean Architecture và Domain-Driven Design (DDD).....	50
3.1.2 Nền tảng phát triển: .NET 9 và ASP.NET Core .....	51
3.1.3 Điều phối hệ thống và Khả năng quan sát: .NET Aspire.....	52
3.2 Cơ sở dữ liệu và Lưu trữ.....	52
3.2.1 Hệ quản trị cơ sở dữ liệu quan hệ: PostgreSQL .....	52
3.2.2 Bộ nhớ đệm (Caching): Redis.....	54
3.2.3 Quản lý và Phân phối đa phương tiện: Cloudinary.....	54
3.3 Xác thực và Phân quyền (Authentication & Authorization).....	55
3.3.1 Cơ chế xác thực: JSON Web Token (JWT) .....	55
3.3.2 Quản lý định danh và Phân quyền: ASP.NET Core Identity .....	55
3.4 Ứng dụng dành cho khách hàng (Mobile App).....	56
3.4.1 Nền tảng phát triển: Flutter & Dart .....	56
3.4.2 Quản lý trạng thái và Tương tác API .....	57
3.4.3 Tích hợp bản đồ và Thanh toán.....	57
3.5 Ứng dụng Web quản trị (Admin & Restaurant Portal).....	58
3.5.1 Nền tảng Frontend: Angular.....	58
3.5.2 Thư viện giao diện: PrimeNG và Tailwind CSS .....	59

3.6 Cộng tác thời gian thực (Realtime Collaboration) .....	59
3.6.1 Giao thức và Framework: SignalR .....	59
3.6.2 Thông báo đẩy tới thiết bị di động: Firebase Cloud Messaging (FCM).....	60
<b>CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG ....</b>	<b>61</b>
4.1 Thiết kế kiến trúc.....	61
4.1.1 Lựa chọn kiến trúc phần mềm .....	61
4.1.2 Thiết kế tổng quan.....	64
4.1.3 Thiết kế chi tiết gói .....	65
4.2 Thiết kế chi tiết.....	70
4.2.1 Thiết kế giao diện .....	70
4.2.2 Thiết kế lớp .....	70
4.2.3 Thiết kế cơ sở dữ liệu .....	70
4.3 Xây dựng ứng dụng.....	71
4.3.1 Thư viện và công cụ sử dụng.....	71
4.3.2 Kết quả đạt được .....	71
4.3.3 Minh họa các chức năng chính .....	71
4.4 Kiểm thử.....	71
4.5 Triển khai .....	71
<b>CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT .....</b>	<b>72</b>
<b>CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>73</b>
6.1 Kết luận .....	73
6.2 Hướng phát triển.....	73
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>77</b>
<b>PHỤ LỤC.....</b>	<b>79</b>

<b>A. HƯỚNG DẪN VIẾT ĐỒ ÁN TỐT NGHIỆP .....</b>	<b>79</b>
A.1 Ngành học.....	80
A.2 Đánh dấu (bullet) và đánh số (numering) .....	80
A.3 Cách thêm bảng.....	81
A.4 Chèn hình ảnh .....	81
A.5 Tài liệu tham khảo .....	82
A.6 Cách viết phương trình và công thức toán học.....	82
A.7 Qui cách đóng quyển.....	82
<b>B. ĐẶC TẢ USE CASE.....</b>	<b>84</b>
B.1 Đặc tả use case “Thống kê tình hình mượn sách” .....	84
B.2 Đặc tả use case “Đăng ký làm thẻ mượn” .....	84

## DANH MỤC HÌNH VẼ

Hình 2.1	Biểu đồ use case tổng quát của hệ thống YummyZoom . . .	16
Hình 2.2	Biểu đồ use case phân rã - Quản lý tài khoản . . . . .	18
Hình 2.3	Biểu đồ use case phân rã - Đặt hàng cá nhân . . . . .	19
Hình 2.4	Biểu đồ use case phân rã - TeamCart . . . . .	20
Hình 2.5	Biểu đồ use case phân rã - Quản lý thực đơn . . . . .	21
Hình 2.6	Biểu đồ use case phân rã - Duyệt đăng ký nhà hàng . . . . .	22
Hình 2.7	Biểu đồ hoạt động - Quy trình xử lý đặt hàng . . . . .	23
Hình 2.8	Biểu đồ hoạt động - Quy trình đặt hàng nhóm . . . . .	24
Hình 4.1	Biểu đồ phụ thuộc gói tổng quan YummyZoom (Backend) .	65
Hình 4.2	Biểu đồ gói chi tiết phân hệ Restaurant (Create & Search) . .	67
Hình 4.3	Biểu đồ gói chi tiết phân hệ Order (Initiate & Status) . . . .	69
Hình A.1	Internet vạn vật . . . . .	81
Hình A.2	Quy cách đóng quyển đồ án . . . . .	83
Hình A.3	Quy cách đóng quyển đồ án . . . . .	83

## DANH MỤC BẢNG BIỂU

Bảng 2.1	Đặc tả Use Case Đăng ký nhà hàng . . . . .	25
Bảng 2.2	Dữ liệu đầu vào Đăng ký nhà hàng . . . . .	26
Bảng 2.3	Đặc tả Use Case Duyệt đăng ký nhà hàng . . . . .	27
Bảng 2.4	Dữ liệu đầu vào Duyệt đăng ký nhà hàng . . . . .	28
Bảng 2.5	Đặc tả Use Case Quản lý thực đơn . . . . .	30
Bảng 2.6	Dữ liệu đầu vào Quản lý thực đơn - Tạo món ăn . . . . .	30
Bảng 2.7	Đặc tả Use Case Tìm kiếm nhà hàng . . . . .	32
Bảng 2.8	Dữ liệu đầu vào Tìm kiếm nhà hàng . . . . .	32
Bảng 2.9	Đặc tả Use Case Đặt hàng cá nhân . . . . .	34
Bảng 2.10	Dữ liệu đầu vào Đặt hàng cá nhân . . . . .	35
Bảng 2.11	Đặc tả Use Case Khởi tạo TeamCart . . . . .	36
Bảng 2.12	Dữ liệu đầu vào Khởi tạo TeamCart . . . . .	37
Bảng 2.13	Đặc tả Use Case Tham gia TeamCart . . . . .	39
Bảng 2.14	Dữ liệu đầu vào Tham gia TeamCart . . . . .	39
Bảng 2.15	Đặc tả Use Case Chốt đơn TeamCart . . . . .	42
Bảng 2.16	Dữ liệu đầu vào Chốt giá TeamCart . . . . .	42
Bảng 2.17	Dữ liệu đầu vào Chốt đơn TeamCart . . . . .	43
Bảng 2.18	Đặc tả Use Case Xử lý đơn hàng . . . . .	45
Bảng 2.19	Dữ liệu đầu vào Xử lý đơn hàng . . . . .	45
Bảng 2.20	Đặc tả Use Case Đánh giá nhà hàng . . . . .	47
Bảng 2.21	Dữ liệu đầu vào Đánh giá nhà hàng . . . . .	47
Bảng 4.1	Danh sách thư viện và công cụ sử dụng . . . . .	71
Bảng A.1	Table to test captions and labels. . . . .	81

## DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

API	Giao diện lập trình ứng dụng (Application Programming Interface)
EUD	Phát triển ứng dụng người dùng cuối(End-User Development)
GWT	Công cụ lập trình Javascript bằng Java của Google (Google Web Toolkit)
HTML	Ngôn ngữ đánh dấu siêu văn bản (HyperText Markup Language)
IaaS	Dịch vụ hạ tầng

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## 1.1 Đặt vấn đề

Trong bối cảnh cuộc cách mạng công nghiệp 4.0, lĩnh vực công nghệ thực phẩm (FoodTech), đặc biệt là các nền tảng giao đồ ăn trực tuyến, đã trải qua sự phát triển mạnh mẽ và trở thành một phần không thể thiếu trong đời sống hiện đại. Theo báo cáo của Momentum Works, thị trường giao đồ ăn tại Việt Nam đã đạt mức tăng trưởng 26% trong năm 2024, cao nhất khu vực Đông Nam Á, với tổng giá trị giao dịch (GMV) tăng từ 1.4 tỷ USD vào năm 2023 lên 1.8 tỷ USD vào năm 2024. Sự tăng trưởng này được thúc đẩy bởi sự phổ biến của điện thoại thông minh, lối sống bận rộn của người dân đô thị và nhu cầu ngày càng cao về sự tiện lợi trong ăn uống. Kết quả khảo sát cho thấy có đến 30% người dùng lựa chọn hình thức đặt đồ ăn qua ứng dụng cho bữa trưa, chứng tỏ thói quen này đã trở nên phổ biến.

Hiện tại, thị trường giao đồ ăn trực tuyến tại Việt Nam đang bị thống trị bởi hai nền tảng lớn là GrabFood và ShopeeFood, tạo nên một thế song cực với thị phần lần lượt là 48% và 47%. Hai nền tảng này cạnh tranh khốc liệt thông qua các chiến dịch khuyến mãi lớn, xây dựng hệ sinh thái dịch vụ đa dạng và phát triển các chương trình khách hàng thân thiết. GrabFood tận dụng lợi thế từ hệ sinh thái "siêu ứng dụng" Grab với các dịch vụ đi chuyển, giao hàng và thanh toán điện tử, cùng với chương trình GrabRewards và gói thành viên GrabUnlimited. Trong khi đó, ShopeeFood khai thác hiệu quả mô hình tích hợp với nền tảng thương mại điện tử Shopee và văn hóa "săn sale" của người tiêu dùng Việt Nam thông qua các chương trình voucher và khuyến mãi liên tục.

Tuy nhiên, bên cạnh những thành công đạt được, các nền tảng hiện tại vẫn chưa giải quyết triệt để một vấn đề quan trọng đối với một nhóm người dùng cụ thể. Đó là nhóm sinh viên đại học và nhân viên văn phòng, những người thường xuyên có nhu cầu đặt đồ ăn theo nhóm trong giờ nghỉ trưa ngắn ngủi. Mặc dù cả GrabFood và ShopeeFood đều cung cấp tính năng đặt hàng nhóm (Group Order), quy trình thanh toán của họ vẫn tồn tại một "nỗi đau" đáng kể. Trong mô hình hiện tại, một người (chủ nhóm) phải đứng ra thanh toán toàn bộ đơn hàng, sau đó phải mất công thu lại tiền từ từng thành viên trong nhóm. Quy trình này không chỉ gây bất tiện và mất thời gian mà còn tạo ra tâm lý ngại ngùng, đặc biệt trong môi trường công sở. Hơn nữa, tình huống quên thu tiền hoặc thành viên quên trả tiền diễn ra thường xuyên, gây khó chịu và ảnh hưởng đến mối quan hệ trong nhóm.

Vấn đề này có tầm quan trọng đáng kể vì nó ảnh hưởng trực tiếp đến trải nghiệm ăn uống tập thể, một nét văn hóa phổ biến tại Việt Nam. Việc giải quyết hiệu quả

bài toán này không chỉ cải thiện trải nghiệm người dùng mà còn mở ra cơ hội cho một nền tảng mới tiếp cận thị trường ngách với giá trị khác biệt rõ ràng. Với quy mô đông đảo của nhóm đối tượng sinh viên và nhân viên văn phòng tại các thành phố lớn, tiềm năng thị trường cho giải pháp này là đáng kể. Nếu được triển khai thành công, mô hình này còn có thể được áp dụng cho các tình huống đặt hàng nhóm khác như tiệc tại văn phòng, sự kiện nhóm hoặc các buổi họp mặt gia đình.

## **1.2 Mục tiêu và phạm vi đề tài**

Để xác định hướng đi phù hợp cho dự án YummyZoom, việc phân tích và đánh giá các nền tảng giao đồ ăn hàng đầu trên thị trường là điều kiện tiên quyết. GrabFood với thế mạnh là hệ sinh thái dịch vụ toàn diện, bao gồm di chuyển, giao hàng và thanh toán điện tử, đã xây dựng được lòng trung thành cao từ nhóm người dùng từ 35 tuổi trở lên thông qua các chương trình như GrabRewards và GrabUnlimited. ShopeeFood, với chiến lược tận dụng nền tảng thương mại điện tử Shopee, tập trung vào nhóm người dùng trẻ từ 16-24 tuổi bằng các chương trình khuyến mãi và voucher liên tục. Cả hai nền tảng đều cung cấp đầy đủ các tính năng cơ bản như tìm kiếm nhà hàng, đặt hàng, thanh toán đa dạng, theo dõi đơn hàng và đánh giá. Đặc biệt, cả hai đều có tính năng đặt hàng nhóm, cho phép nhiều người cùng tham gia vào một đơn hàng.

Tuy nhiên, qua quá trình phân tích, một hạn chế quan trọng đã được xác định. Tính năng đặt hàng nhóm hiện tại trên cả GrabFood và ShopeeFood đều yêu cầu chủ nhóm phải thanh toán toàn bộ đơn hàng, sau đó tự thu tiền từ các thành viên khác. Quy trình này tạo ra nhiều bất tiện: người chủ nhóm phải tạm ứng một khoản tiền lớn, mất thời gian để thu lại từng phần, và có nguy cơ không thu đủ tiền do các thành viên quên hoặc trì hoãn. Đối với nhóm sinh viên và nhân viên văn phòng, những người thường xuyên đặt đồ ăn chung trong giờ nghỉ trưa ngắn, vấn đề này càng trở nên bất tiện và gây mất thời gian quý báu. Hơn nữa, tình huống này còn có thể tạo ra những khoảnh khắc ngượng ngùng khi phải nhắc nhở đồng nghiệp về việc trả tiền, ảnh hưởng đến mối quan hệ trong nhóm.

Dựa trên phân tích trên, dự án YummyZoom được xác định với mục tiêu chính là xây dựng một ứng dụng giao đồ ăn tập trung vào việc giải quyết vấn đề đặt hàng nhóm thông qua cơ chế thanh toán phân tán. Để đạt được mục tiêu này, hệ thống sẽ được phát triển với các chức năng cốt lõi sau đây. Thứ nhất, các chức năng cơ bản đáp ứng tiêu chuẩn ngành bao gồm hệ thống quản lý tài khoản cho ba vai trò (khách hàng, nhà hàng, quản trị viên), tìm kiếm và khám phá nhà hàng với bộ lọc đa dạng, đặt hàng cá nhân với khả năng tùy chỉnh món ăn, quản lý giỏ hàng và áp dụng mã khuyến mãi, thanh toán mô phỏng thông qua chế độ thử nghiệm của Stripe, theo dõi trạng thái đơn hàng, và hệ thống đánh giá nhà hàng. Thứ hai, tính năng đột phá

tạo nên lợi thế cạnh tranh chính là TeamCart (Giỏ hàng nhóm), cho phép mỗi thành viên trong nhóm tự thêm món ăn mình muốn và tự thanh toán cho phần của riêng mình, kèm theo khả năng cập nhật thời gian thực để các thành viên có thể theo dõi tiến trình của nhóm. Thứ ba, các chức năng quản lý dành cho nhà hàng bao gồm đăng ký và quản lý hồ sơ, quản lý thực đơn và tính năng đánh dấu món hết hàng, tạo và quản lý các chương trình khuyến mãi, xử lý đơn hàng với thông báo thời gian thực, và xem đánh giá từ khách hàng. Cuối cùng, các chức năng quản trị bao gồm bảng điều khiển tổng quan về các chỉ số hoạt động của hệ thống, quản lý và phê duyệt đăng ký nhà hàng mới, và giám sát nội dung như đánh giá và chương trình khuyến mãi.

Để đảm bảo tính khả thi trong khuôn khổ đồ án tốt nghiệp, phạm vi dự án được xác định rõ ràng với các giới hạn hợp lý. Dự án sẽ tập trung phát triển ba vai trò chính là khách hàng, nhà hàng và quản trị viên, với đầy đủ luồng đặt hàng cá nhân và nhóm. Quá trình giao hàng sẽ được mô phỏng thông qua việc cập nhật các trạng thái đơn hàng thay vì triển khai hệ thống theo dõi tài xế thực tế. Các chức năng nằm ngoài phạm vi dự án bao gồm module ứng dụng dành cho tài xế với các nghiệp vụ phức tạp như đăng ký, định tuyến và quản lý thu nhập, hệ thống theo dõi vị trí GPS thời gian thực, các thuật toán gợi ý sử dụng trí tuệ nhân tạo, chương trình khách hàng thân thiết với tích điểm và đổi thưởng, và tích hợp các cổng thanh toán thực tế như VNPAY hay MoMo. Các quyết định giới hạn này được đưa ra nhằm tập trung nguồn lực vào việc hoàn thiện tính năng cốt lõi là TeamCart và đảm bảo chất lượng của các chức năng nền tảng, thay vì dàn trải cho quá nhiều tính năng phức tạp vượt quá khả năng thực hiện trong thời gian và nguồn lực giới hạn của một đồ án tốt nghiệp.

### 1.3 Định hướng giải pháp

Để giải quyết các vấn đề đã xác định ở phần 1.2, dự án YummyZoom được phát triển theo định hướng ứng dụng kiến trúc phần mềm hiện đại và các công nghệ phù hợp. Về mặt kiến trúc, dự án áp dụng Clean Architecture (Kiến trúc sạch) kết hợp với Domain-Driven Design (DDD - Thiết kế hướng miền) làm nền tảng. Clean Architecture được lựa chọn vì khả năng tách biệt logic nghiệp vụ khỏi các chi tiết kỹ thuật, giúp hệ thống dễ bảo trì, kiểm thử và mở rộng trong tương lai. Domain-Driven Design được áp dụng để mô hình hóa các quy tắc nghiệp vụ phức tạp của lĩnh vực giao đồ ăn thông qua các khái niệm như tập hợp (Aggregate), thực thể (Entity), đối tượng giá trị (Value Object) và sự kiện (Domain Event), đảm bảo tính nhất quán dữ liệu và phản ánh chính xác các quy trình thực tế.

Về công nghệ triển khai, dự án sử dụng .NET 9 và C# cho phát triển phần backend. .NET 9 được lựa chọn vì khả năng xử lý hiệu năng cao, hỗ trợ lập trình

bất đồng bộ (async/await) cần thiết cho các tính năng thời gian thực, và một hệ sinh thái phong phú với nhiều thư viện hỗ trợ. Entity Framework Core được sử dụng làm công cụ ánh xạ quan hệ đối tượng (Object-Relational Mapping - ORM), cho phép áp dụng phương pháp ưu tiên mã nguồn (code-first) phù hợp với DDD và tự động hóa việc quản lý lược đồ cơ sở dữ liệu thông qua migrations. Đặc biệt, để triển khai tính năng TeamCart với khả năng cập nhật thời gian thực, dự án tích hợp SignalR, một thư viện hỗ trợ giao tiếp hai chiều qua WebSocket, cho phép các thành viên trong nhóm nhìn thấy các thay đổi ngay lập tức khi có người thêm hoặc xóa món ăn.

Giải pháp tổng thể của YummyZoom được thiết kế theo kiến trúc phân lớp với các thành phần chính sau. Lớp miền (Domain layer) chứa các tập hợp (aggregate) cốt lõi như User, Restaurant, Order, Menu và TeamCart, đóng gói toàn bộ quy tắc nghiệp vụ và đảm bảo tính nhất quán của dữ liệu. Lớp ứng dụng (Application layer) triển khai các ca sử dụng thông qua Commands và Queries theo mô hình CQRS (Command Query), xử lý các yêu cầu từ người dùng và điều phối các đối tượng miền. Lớp hạ tầng (Infrastructure layer) cung cấp các triển khai cụ thể cho tầng lưu trữ bền vững sử dụng Entity Framework Core với PostgreSQL, quản lý định danh thông qua ASP.NET Identity, và giao tiếp thời gian thực sử dụng SignalR. Lớp giao diện (Web layer) triển khai các điểm cuối API theo chuẩn RESTful sử dụng ASP.NET Core, xử lý xác thực và phân quyền thông qua JWT tokens. Luồng hoạt động chính của hệ thống bắt đầu từ khách hàng duyệt nhà hàng và thực đơn, sau đó có thể chọn đặt hàng cá nhân hoặc tạo một TeamCart. Đối với TeamCart, chủ nhóm tạo giỏ hàng và chia sẻ liên kết, các thành viên tham gia và tự thêm món ăn, mỗi người tự thanh toán cho phần của mình, và khi tất cả đã hoàn tất, giỏ hàng nhóm được chuyển thành một đơn hàng duy nhất gửi đến nhà hàng. Nhà hàng nhận đơn hàng, xử lý và cập nhật trạng thái theo thời gian thực, trong khi quản trị viên giám sát toàn bộ hoạt động của hệ thống.

Đóng góp chính của đề án thể hiện ở ba khía cạnh quan trọng. Thứ nhất, dự án thành công trong việc áp dụng Clean Architecture kết hợp Domain-Driven Design vào một miền nghiệp vụ phức tạp như giao đồ ăn, với nhiều tác nhân tương tác và quy trình nghiệp vụ đa dạng. Thứ hai, giải pháp TeamCart với cơ chế thanh toán phân tán là một đột phá so với các ứng dụng hiện tại, giải quyết trực tiếp "nỗi đau" thực tế của người dùng khi đặt hàng nhóm. Cơ chế này không chỉ loại bỏ gánh nặng tài chính cho chủ nhóm mà còn tạo ra sự công bằng và minh bạch trong thanh toán. Thứ ba, việc triển khai tính năng cộng tác thời gian thực cho phép nhiều người cùng tương tác với một giỏ hàng chung một cách mượt mà, với các thay đổi được đồng bộ ngay lập tức giữa tất cả các thành viên. Về kết quả đạt được, dự án đã xây

dựng thành công một ứng dụng sản phẩm khả dụng tối thiểu (MVP) hoàn chỉnh với đầy đủ các tính năng cốt lõi, tính năng TeamCart hoạt động ổn định với khả năng cập nhật thời gian thực, kiến trúc hệ thống rõ ràng và dễ bảo trì, và hiệu năng đáp ứng yêu cầu với thời gian phản hồi dưới 2 giây cho hầu hết các thao tác.

### 1.4 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Chương 2 trình bày về khảo sát và phân tích yêu cầu hệ thống. Trong chương này, tôi thực hiện phân tích chi tiết các ứng dụng giao đồ ăn hàng đầu tại thị trường Việt Nam, đặc biệt là GrabFood và ShopeeFood, để xác định các tiêu chuẩn ngành và tìm ra cơ hội khác biệt hóa. Tiếp đến, tôi xây dựng biểu đồ ca sử dụng (use case) tổng quát và các biểu đồ ca sử dụng phân rã cho các chức năng chính, sau đó tiến hành đặc tả chi tiết cho các ca sử dụng quan trọng nhất của hệ thống. Đặc biệt, chương này tập trung vào việc mô tả chi tiết quy trình nghiệp vụ của tính năng TeamCart, làm rõ sự khác biệt so với tính năng đặt hàng nhóm truyền thống. Cuối chương, tôi trình bày các yêu cầu phi chức năng về hiệu năng (performance), khả năng sử dụng (usability), bảo mật (security) và khả năng mở rộng (scalability) mà hệ thống cần đáp ứng.

Chương 3, giới thiệu về các công nghệ và phương pháp được sử dụng trong dự án. Chương này bắt đầu với việc trình bày chi tiết về Clean Architecture và các nguyên lý thiết kế của nó, sau đó giải thích cách áp dụng Domain-Driven Design để mô hình hóa miền giao đồ ăn với các khái niệm như tập hợp (Aggregate), thực thể (Entity) và đối tượng giá trị (Value Object). Tiếp theo, tôi phân tích các công nghệ cụ thể bao gồm .NET 9 với ASP.NET Core cho phát triển phần backend, Entity Framework Core cho truy cập dữ liệu với phương pháp ưu tiên mã nguồn (code-first), SignalR cho giao tiếp thời gian thực, và ASP.NET Identity kết hợp JWT cho xác thực và phân quyền. Mỗi công nghệ được phân tích về lý do lựa chọn, các lựa chọn thay thế có thể xem xét, và cách thức áp dụng cụ thể vào YummyZoom.

Chương 4 trình bày chi tiết về thiết kế, triển khai và đánh giá hệ thống. Đầu tiên, tôi mô tả kiến trúc tổng thể của hệ thống theo Clean Architecture với bốn lớp chính (Domain, Application, Infrastructure và Web) và các mối quan hệ phụ thuộc giữa chúng. Tiếp theo, tôi trình bày thiết kế chi tiết cho các tập hợp miền (domain aggregates) quan trọng nhất như User, Restaurant, Order, Menu và đặc biệt là TeamCart với các thực thể (entity) và đối tượng giá trị (value object) bên trong. Phần thiết kế cơ sở dữ liệu trình bày lược đồ cơ sở dữ liệu với các bảng, quan hệ và ràng buộc được sinh ra từ mô hình miền thông qua Entity Framework Core migrations. Tôi cũng trình bày thiết kế các điểm cuối API theo chuẩn RESTful,

cách tổ chức bộ điều khiển (controllers) và các tầng trung gian (middleware) được sử dụng. Phần cuối chương trình bày về quá trình xây dựng ứng dụng với các công cụ và thư viện được sử dụng, cũng như kết quả kiểm thử thông qua kiểm thử đơn vị (unit tests) cho lớp miền, kiểm thử tích hợp (integration tests) cho lớp hạ tầng và kiểm thử chức năng (functional tests) cho lớp ứng dụng.

Chương 5 tập trung vào các giải pháp kỹ thuật và đóng góp nổi bật của đồ án. Tôi phân tích chi tiết cách áp dụng Clean Architecture vào miền phức tạp của giao đồ ăn, làm rõ cách tách biệt logic nghiệp vụ khỏi các vấn đề hạ tầng và lợi ích của việc này trong kiểm thử và bảo trì. Tiếp theo, tôi trình bày giải pháp xử lý thông báo thời gian thực sử dụng SignalR, bao gồm cách thiết kế các trung tâm (hubs), quản lý kết nối và phát thông điệp đến các máy khách. Phần quan trọng nhất là phân tích chi tiết thiết kế và triển khai của tính năng TeamCart, từ mô hình miền với tập hợp TeamCart, các quy tắc nghiệp vụ đảm bảo tính nhất quán, cơ chế khóa khi chuyển đổi trạng thái, đến việc xử lý thanh toán phân tán và đồng bộ hóa thời gian thực giữa các thành viên. Mỗi giải pháp được so sánh với cách tiếp cận của các ứng dụng hiện có để làm rõ tính đột phá và hiệu quả của phương pháp được áp dụng.

Chương 6 tổng kết lại toàn bộ công việc đã thực hiện và các kết quả đạt được. Tôi đánh giá những thành công của dự án trong việc xây dựng một ứng dụng sản phẩm khả dụng tối thiểu hoàn chỉnh với tính năng TeamCart hoạt động ổn định và hiệu quả. Đồng thời, tôi cũng thẳng thắn chỉ ra những hạn chế còn tồn tại như việc chưa triển khai ứng dụng di động gốc (native mobile app), chưa có tích hợp học máy (machine learning) và AI cho gợi ý cá nhân hóa, và việc mô phỏng quá trình giao hàng thay vì tích hợp với hệ thống tài xế thực tế. Cuối cùng, tôi đề xuất các hướng phát triển trong tương lai bao gồm phát triển ứng dụng di động cho iOS và Android, tích hợp các cổng thanh toán thực tế phổ biến tại Việt Nam như MoMo và VNPAY, xây dựng chương trình khách hàng thân thiết với tích điểm và đổi thưởng, áp dụng học máy để gợi ý món ăn và nhà hàng phù hợp, và triển khai kiến trúc microservices để tăng khả năng mở rộng của hệ thống khi số lượng người dùng tăng lên. [Sửa sau]

## PHẦN HƯỚNG DẪN TỪ MẪU BÁO CÁO

Lưu ý: **Trước khi viết ĐATN, sinh viên cần đọc kỹ hướng dẫn và quy định chi tiết** về cách viết ĐATN trong Phụ lục A. Sinh viên tuân theo mẫu tài liệu này để viết báo cáo đồ án tốt nghiệp, vì tài liệu này đã được căn chỉnh, chỉnh sửa theo đúng chuẩn báo cáo kỹ thuật đồ án tốt nghiệp (ISO 7144:1986). Sinh viên viết trực tiếp vào file này, chỉ chỉnh sửa nội dung, và không viết trên file mới.

**Khi đóng quyển ĐATN**, sinh viên cần lưu ý tuân thủ hướng dẫn ở phụ lục A.9

**SV cần đặc biệt lưu ý cách hành văn.** Mỗi đoạn văn không được quá dài và cần có ý tứ rõ ràng, bao gồm duy nhất một ý chính và các ý phân tích hỗ trợ để làm rõ hơn ý chính. Các câu văn trong đoạn phải đầy đủ chủ ngữ vị ngữ, cùng hướng đến chủ đề chung. Câu sau phải liên kết với câu trước, đoạn sau liên kết với đoạn trước. Trong văn phong khoa học, sinh viên không được dùng từ trong văn nói, không dùng các từ phóng đại, thái quá, các từ thiếu khách quan, thiên về cảm xúc, về quan điểm cá nhân như “tuyệt vời”, “cực hay”, “cực kỳ hữu ích”, v.v. Các câu văn cần được tối ưu hóa, đảm bảo rất khó để thể thêm hoặc bớt đi được dù chỉ một từ. Cách diễn đạt cần ngắn gọn, súc tích, không dài dòng.

Mẫu ĐATN này được thiết kế phù hợp nhất với đa số các đề tài xây dựng phần mềm ứng dụng. Với các dạng đề tài khác (giải pháp, nghiên cứu, phần mềm đặc thù, v.v.), sinh viên dựa trên cấu trúc và hướng dẫn của báo cáo này để đề xuất và trao đổi với giáo viên hướng dẫn để thiết kế khung báo cáo đồ án cho phù hợp. Sinh viên lưu ý **trong mọi trường hợp, SV luôn phải sử dụng định dạng báo cáo này, và phải đọc kỹ toàn bộ các hướng dẫn từ đầu tới cuối.** Các hướng dẫn không chỉ áp dụng riêng cho đề tài ứng dụng, mà còn phù hợp với các dạng đề tài khác. Ngoài ra, trong mẫu ĐATN này đã được tích hợp một số hướng dẫn dành riêng cho đề tài nghiên cứu.

Chương 1 có độ dài từ 3 đến 6 trang với các nội dung sau đây

### Hướng dẫn viết phần 1.1 - Đặt vấn đề

Khi đặt vấn đề, sinh viên cần làm nổi bật mức độ cấp thiết, tầm quan trọng và/hoặc quy mô của bài toán của mình.

Gợi ý cách trình bày cho sinh viên: Xuất phát từ tình hình thực tế gì, dẫn đến vấn đề hoặc bài toán gì. Vấn đề hoặc bài toán đó, nếu được giải quyết, đem lại lợi ích gì, cho những ai, còn có thể được áp dụng vào các lĩnh vực khác nữa không. Sinh viên cần lưu ý phần này chỉ trình bày vấn đề, tuyệt đối không trình bày giải pháp.

### Hướng dẫn viết phần 1.2 - Mục tiêu và phạm vi đề tài

Sinh viên trước tiên cần trình bày tổng quan các kết quả của các nghiên cứu hiện nay cho bài toán giới thiệu ở phần 1.1 (đối với đề tài nghiên cứu), hoặc về các sản phẩm hiện tại/về nhu cầu của người dùng (đối với đề tài ứng dụng). Tiếp đến, sinh viên tiến hành so sánh và đánh giá tổng quan các sản phẩm/nghiên cứu này.

Dựa trên các phân tích và đánh giá ở trên, sinh viên khái quát lại các hạn chế hiện tại đang gặp phải. Trên cơ sở đó, sinh viên sẽ hướng tới giải quyết vấn đề cụ thể gì, khắc phục hạn chế gì, phát triển phần mềm **có các chức năng chính gì**, tạo nên đột phá gì, v.v.

Trong phần này, sinh viên lưu ý chỉ trình bày tổng quan, không đi vào chi tiết của vấn đề hoặc giải pháp. Nội dung chi tiết sẽ được trình bày trong các chương tiếp theo, đặc biệt là trong Chương 5.

### Hướng dẫn viết phần 1.3 - Định hướng giải pháp

Từ việc xác định rõ nhiệm vụ cần giải quyết ở phần 1.2, sinh viên đề xuất định hướng giải pháp của mình theo trình tự sau: (i) Sinh viên trước tiên trình bày sẽ giải quyết vấn đề theo định hướng, phương pháp, thuật toán, kỹ thuật, hay công nghệ nào; Tiếp theo, (ii) sinh viên mô tả ngắn gọn giải pháp của mình là gì (khi đi theo định hướng/phương pháp nêu trên); và sau cùng, (iii) sinh viên trình bày đóng góp chính của đề án là gì, kết quả đạt được là gì.

Sinh viên lưu ý không giải thích hoặc phân tích chi tiết công nghệ/thuật toán trong phần này. Sinh viên chỉ cần nêu tên định hướng công nghệ/thuật toán, mô tả ngắn gọn trong một đến hai câu và giải thích nhanh lý do lựa chọn.

### Hướng dẫn viết phần 1.4 - Bố cục đề án

Phần còn lại của báo cáo đề án tốt nghiệp này được tổ chức như sau.

Chương 2 trình bày về v.v.

Trong Chương 3, em/tôi giới thiệu về v.v.

**Chú ý:** Sinh viên cần viết mô tả thành đoạn văn đầy đủ về nội dung chương. Tuyệt đối không viết ý hay gạch đầu dòng. Chương 1 không cần mô tả trong phần này.

Ví dụ tham khảo mô tả chương trong phần bố cục đề án tốt nghiệp: Chương \*\*\* trình bày đóng góp chính của đề án, đó là một nền tảng ABC cho phép khai phá và tích hợp nhiều nguồn dữ liệu, trong đó mỗi nguồn dữ liệu lại có định dạng đặc thù riêng. Nền tảng ABC được phát triển dựa trên khái niệm DEF, là các module ngữ nghĩa trợ giúp người dùng tìm kiếm, tích hợp và hiển thị trực quan dữ liệu theo mô

hình cộng tác và mô hình phân tán.

**Chú ý:** Trong phần nội dung chính, mỗi chương của đồ án nên có phần Tổng quan và Kết chương. Hai phần này đều có định dạng văn bản “Normal”, sinh viên không cần tạo định dạng riêng, ví dụ như không in đậm/in nghiêng, không đóng khung, v.v.

Trong phần Tổng quan của chương N, sinh viên nên có sự liên kết với chương N-1 rồi trình bày sơ qua lý do có mặt của chương N và sự cần thiết của chương này trong đồ án. Sau đó giới thiệu những vấn đề sẽ trình bày trong chương này là gì, trong các đề mục lớn nào.

Ví dụ về phần Tổng quan: Chương 3 đã thảo luận về nguồn gốc ra đời, cơ sở lý thuyết và các nhiệm vụ chính của bài toán tích hợp dữ liệu. Chương 4 này sẽ trình bày chi tiết các công cụ tích hợp dữ liệu theo hướng tiếp cận “mashup”. Với mục đích và phạm vi của đề tài, sáu nhóm công cụ tích hợp dữ liệu chính được trình bày bao gồm: (i) nhóm công cụ ABC trong phần 4.1, (ii) nhóm công cụ DEF trong phần 4.2, nhóm công cụ GHK trong phần 4.3, v.v.

Trong phần Kết chương, sinh viên đưa ra một số kết luận quan trọng của chương. Những vấn đề mở ra trong Tổng quan cần được tóm tắt lại nội dung và cách giải quyết/thực hiện như thế nào. Sinh viên lưu ý không viết Kết chương giống hệt Tổng quan. Sau khi đọc phần Kết chương, người đọc sẽ nắm được sơ bộ nội dung và giải pháp cho các vấn đề đã trình bày trong chương. Trong Kết chương, Sinh viên nên có thêm câu liên kết tới chương tiếp theo.

Ví dụ về phần Kết chương: Chương này đã phân tích chi tiết sáu nhóm công cụ tích hợp dữ liệu. Nhóm công cụ ABC và DEF thích hợp với những bài toán tích hợp dữ liệu phạm vi nhỏ. Trong khi đó, nhóm công cụ GHK lại chứng tỏ thế mạnh của mình với những bài toán cần độ chính xác cao, v.v. Từ kết quả nghiên cứu và phân tích về sáu nhóm công cụ tích hợp dữ liệu này, tôi đã thực hiện phát triển phần mềm tự động bóc tách và tích hợp dữ liệu sử dụng nhóm công cụ GHK. Phần này được trình bày trong chương tiếp theo – Chương 5.

## 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

Trong bối cảnh chuyển đổi số mạnh mẽ và sự thay đổi thói quen tiêu dùng của người dân Việt Nam, ngành giao đồ ăn trực tuyến đã trải qua một giai đoạn tăng trưởng ấn tượng trong những năm gần đây. Theo báo cáo của Hiệp hội Thương mại điện tử Việt Nam (VECOM), thị trường giao đồ ăn trực tuyến tại Việt Nam đạt giá trị khoảng 1,2 tỷ USD vào năm 2023 và dự kiến sẽ tăng trưởng với tốc độ trung bình 15-20% mỗi năm trong giai đoạn 2024-2028. Sự bùng nổ này được thúc đẩy bởi nhiều yếu tố quan trọng, bao gồm sự phát triển của công nghệ di động, tăng cường khả năng tiếp cận internet băng rộng, và đặc biệt là sự thay đổi mạnh mẽ trong hành vi tiêu dùng sau đại dịch COVID-19.

Đại dịch COVID-19 đã tạo ra một bước ngoặt quan trọng trong ngành dịch vụ ăn uống, khi các biện pháp giãn cách xã hội và phong tỏa đã buộc người tiêu dùng phải chuyển sang sử dụng các dịch vụ giao đồ ăn trực tuyến như một giải pháp thay thế cho việc dùng bữa tại nhà hàng. Điều đáng chú ý là sau khi các biện pháp hạn chế được nới lỏng, thói quen sử dụng dịch vụ giao đồ ăn trực tuyến vẫn được duy trì và tiếp tục phát triển, cho thấy sự chuyển đổi căn bản trong hành vi tiêu dùng. Theo khảo sát của công ty nghiên cứu thị trường Nielsen Vietnam, khoảng 78% người tiêu dùng tại các thành phố lớn như Hà Nội và TP.HCM đã sử dụng dịch vụ giao đồ ăn trực tuyến ít nhất một lần trong năm 2023, trong đó có 45% sử dụng thường xuyên với tần suất từ 2-3 lần mỗi tuần.

Xu hướng phát triển của thị trường giao đồ ăn trực tuyến không chỉ thể hiện qua con số tăng trưởng về doanh thu mà còn qua sự đa dạng hóa trong nhu cầu của

người tiêu dùng. Khách hàng ngày càng đòi hỏi cao hơn về chất lượng dịch vụ, tính tiện lợi, tốc độ giao hàng, và đặc biệt là trải nghiệm người dùng trên ứng dụng di động. Đồng thời, nhu cầu về tính cá nhân hóa và các tính năng xã hội hóa như đặt hàng nhóm, chia sẻ trải nghiệm ẩm thực cũng đang trở thành các yếu tố quan trọng trong việc lựa chọn nền tảng giao đồ ăn. Điều này tạo ra cơ hội lớn cho các ứng dụng mới có khả năng đáp ứng những nhu cầu chưa được thỏa mãn một cách đầy đủ bởi các nền tảng hiện có.

Từ góc độ doanh nghiệp, ngành nhà hàng và dịch vụ ăn uống cũng đã chứng kiến sự thay đổi mạnh mẽ trong mô hình kinh doanh. Nhiều nhà hàng truyền thống đã phải chuyển đổi sang mô hình hybrid, kết hợp giữa phục vụ tại chỗ và giao hàng trực tuyến để duy trì hoạt động kinh doanh. Đặc biệt, các nhà hàng quy mô vừa và nhỏ đang tìm kiếm những nền tảng giao đồ ăn có chính sách hoa hồng hợp lý và cung cấp các công cụ quản lý hiệu quả để tối ưu hóa hoạt động kinh doanh của mình. Điều này không chỉ tạo ra nhu cầu về các giải pháp công nghệ mới mà còn mở ra cơ hội cho các ứng dụng có thể hỗ trợ tốt hơn cho cộng đồng nhà hàng địa phương.

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

Để hiểu rõ bối cảnh cạnh tranh và xác định được các cơ hội phát triển cho YummyZoom, nghiên cứu tiến hành phân tích chi tiết hai ứng dụng giao đồ ăn đang thống lĩnh thị trường Việt Nam hiện nay: GrabFood và ShopeeFood. Theo báo cáo của Momentum Works năm 2024, hai nền tảng này chiếm giữ gần như toàn bộ thị phần với GrabFood dẫn đầu ở mức 48% và ShopeeFood bám sát với 47%, tạo nên một thế song cực trong ngành giao đồ ăn trực tuyến tại Việt Nam.

#### **a, GrabFood - Sức mạnh từ hệ sinh thái siêu ứng dụng**

GrabFood không đơn thuần là một ứng dụng giao đồ ăn độc lập mà là một thành phần quan trọng trong hệ sinh thái siêu ứng dụng Grab. Lợi thế cạnh tranh cốt lõi của GrabFood xuất phát từ khả năng tận dụng tối đa các dịch vụ khác trong hệ sinh thái này, bao gồm dịch vụ di chuyển (GrabBike, GrabCar), giao hàng (GrabExpress), và hệ thống thanh toán tích hợp (ví điện tử Moca). Sự liên kết chặt chẽ này cho phép Grab có được một lượng khách hàng khổng lồ có sẵn, đội ngũ tài xế đông đảo, và đặc biệt là một kho dữ liệu người dùng phong phú để tối ưu hóa trải nghiệm dịch vụ.

Từ góc độ tính năng, GrabFood excel trong việc cung cấp trải nghiệm theo dõi đơn hàng theo thời gian thực với khả năng hiển thị vị trí tài xế trên bản đồ một cách chính xác. Hệ thống gợi ý món ăn của GrabFood được đánh giá cao nhờ việc áp dụng các thuật toán học máy để phân tích hành vi người dùng và đưa ra các đề xuất

cá nhân hóa. Đồng thời, GrabFood cũng có một hệ thống đánh giá và phản hồi khá hoàn chỉnh, cho phép người dùng đánh giá cả nhà hàng và tài xế, từ đó tạo ra một cơ chế kiểm soát chất lượng dịch vụ hiệu quả.

Tuy nhiên, GrabFood cũng tồn tại những điểm yếu đáng chú ý. Chi phí sử dụng dịch vụ của GrabFood thường cao hơn so với các đối thủ, với phí giao hàng và phí dịch vụ có thể lên đến 15-20% giá trị đơn hàng. Giao diện người dùng của ứng dụng Grab, do phải tích hợp nhiều dịch vụ, đôi khi trở nên phức tạp và khó điều hướng, đặc biệt đối với người dùng lớn tuổi hoặc ít am hiểu công nghệ. Về tính năng đặt hàng nhóm, mặc dù GrabFood có hỗ trợ chức năng này, nhưng quy trình vẫn yêu cầu một người làm chủ nhóm phải đứng ra thanh toán toàn bộ đơn hàng, sau đó tự thu lại tiền từ các thành viên khác.

### **b, ShopeeFood - Thống lĩnh bằng văn hóa sẵn khuyến mãi**

ShopeeFood, với tiền thân là Now.vn - một trong những nền tảng giao đồ ăn tiên phong tại Việt Nam, đã được tích hợp sâu vào hệ sinh thái thương mại điện tử Shopee sau khi SEA Group mua lại. Điểm mạnh nổi bật nhất của ShopeeFood nằm ở chiến lược marketing aggressive với vô số chương trình khuyến mãi, voucher giảm giá và ưu đãi miễn phí giao hàng. Nền tảng này rất thành công trong việc khai thác tâm lý "săn sale" đặc trưng của người tiêu dùng Việt Nam, đặc biệt là thông qua việc tận dụng các sự kiện mua sắm lớn của Shopee như 11/11, 12/12 để thu hút người dùng.

Về mặt tính năng, ShopeeFood có giao diện tương đối đơn giản và trực quan, phù hợp với đối tượng người dùng trẻ tuổi. Hệ thống tìm kiếm và lọc món ăn được thiết kế khá hiệu quả, cho phép người dùng dễ dàng tìm thấy các món ăn theo sở thích và ngân sách. ShopeeFood cũng tích hợp tốt với ví điện tử ShopeePay, tạo ra một trải nghiệm thanh toán mượt mà và thường đi kèm với các ưu đãi hoàn tiền. Đặc biệt, nền tảng này có điểm mạnh trong việc hỗ trợ các cửa hàng nhỏ lẻ, quán ăn vặt với chính sách hoa hồng cạnh tranh.

Tuy nhiên, ShopeeFood cũng đối mặt với một số thách thức đáng kể. Chất lượng dịch vụ giao hàng không ổn định là một vấn đề thường xuyên được người dùng phàn nàn, với tình trạng giao hàng chậm trễ và thái độ phục vụ của tài xế chưa đồng đều. Số lượng nhà hàng đối tác và phạm vi phủ sóng của ShopeeFood vẫn còn hạn chế hơn so với GrabFood, đặc biệt tại các khu vực ngoại thành. Về tính năng đặt hàng nhóm, ShopeeFood cũng gặp phải vấn đề tương tự như GrabFood khi yêu cầu một người phải đứng ra thanh toán cho toàn bộ nhóm.

### **c, Phân tích so sánh và nhận định tổng quan**

Qua việc phân tích chi tiết hai nền tảng dẫn đầu thị trường, có thể thấy rằng cả GrabFood và ShopeeFood đều đã xây dựng được những lợi thế cạnh tranh riêng biệt nhưng cũng tồn tại những khoảng trống có thể khai thác. GrabFood thành công nhờ hệ sinh thái tích hợp và chất lượng dịch vụ ổn định, trong khi ShopeeFood chiến thắng bằng chiến lược giá cả cạnh tranh và chương trình khuyến mãi hấp dẫn. Tuy nhiên, cả hai nền tảng đều chưa giải quyết được hoàn toàn vấn đề trong quy trình đặt hàng nhóm, nơi mà một người phải gánh vác trách nhiệm tài chính cho cả nhóm.

Đây chính là cơ hội mà YummyZoom có thể tận dụng thông qua tính năng TeamCart độc đáo, cho phép mỗi thành viên trong nhóm tự thanh toán phần của mình. Bên cạnh đó, cả hai nền tảng hiện tại đều có giao diện khá phức tạp với nhiều tính năng không cần thiết, tạo ra cơ hội cho một ứng dụng tập trung vào sự đơn giản và trải nghiệm người dùng tối ưu. Đối với nhóm khách hàng mục tiêu là sinh viên và nhân viên văn phòng - những người có nhu cầu đặt hàng nhóm cao và thời gian nghỉ trưa hạn chế, việc có một giải pháp đơn giản, nhanh chóng và giải quyết được vấn đề thanh toán nhóm sẽ mang lại giá trị thực tiễn đáng kể.

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

Dựa trên việc phân tích thị trường và các ứng dụng hiện có, có thể nhận diện được những nhu cầu cốt lõi và thách thức chưa được giải quyết triệt để trong lĩnh vực giao đồ ăn trực tuyến. Việc hiểu rõ các nhu cầu này không chỉ giúp định hướng phát triển sản phẩm mà còn tạo cơ sở để YummyZoom xây dựng được lợi thế cạnh tranh bền vững.

##### **a, Nhu cầu cốt lõi của người dùng cuối**

Người dùng cuối, đặc biệt là nhóm sinh viên và nhân viên văn phòng, có những nhu cầu rất cụ thể khi sử dụng dịch vụ giao đồ ăn trực tuyến. Nhu cầu quan trọng nhất là tính tiện lợi và tốc độ trong việc đặt hàng, đặc biệt trong những khoảng thời gian nghỉ trưa ngắn ngủi. Họ mong muốn có thể tìm kiếm, lựa chọn và đặt hàng một cách nhanh chóng mà không phải trải qua quá nhiều bước phức tạp. Đồng thời, sự đa dạng trong lựa chọn món ăn và nhà hàng cũng là yếu tố quan trọng, giúp người dùng có thể thỏa mãn các sở thích ẩm thực khác nhau và tránh cảm giác nhàm chán.

Tính minh bạch về giá cả, phí dịch vụ và thời gian giao hàng là một nhu cầu không thể thiếu khác. Người dùng muốn biết chính xác họ sẽ phải trả bao nhiêu tiền và món ăn sẽ đến khi nào, từ đó có thể lập kế hoạch thời gian và ngân sách một cách hợp lý. Đặc biệt quan trọng là nhu cầu về trải nghiệm đặt hàng nhóm không rườm rà, khi một nhóm bạn bè hoặc đồng nghiệp muốn cùng đặt món từ một nhà

hàng. Hiện tại, việc phải có một người đứng ra làm chủ nhóm và thanh toán toàn bộ, sau đó thu lại tiền từ các thành viên khác, tạo ra không ít phiền phức và ngại ngùng trong các mối quan hệ xã hội.

Cuối cùng, nhu cầu về thanh toán linh hoạt và an toàn cũng rất được quan tâm. Người dùng muốn có nhiều lựa chọn về phương thức thanh toán và đảm bảo rằng thông tin tài chính của họ được bảo vệ một cách tối ưu. Đặc biệt với thế hệ trẻ, việc thanh toán điện tử nhanh chóng và thuận tiện thường được ưu tiên hơn so với thanh toán tiền mặt.

### **b, Nhu cầu của nhà hàng đối tác**

Từ góc độ nhà hàng, đặc biệt là các cơ sở kinh doanh quy mô vừa và nhỏ, họ cần có công cụ quản lý đơn hàng hiệu quả và dễ sử dụng. Nhà hàng muốn có thể nhận thông báo đơn hàng mới một cách kịp thời, xem chi tiết đơn hàng rõ ràng, và cập nhật trạng thái chuẩn bị món ăn một cách thuận tiện. Việc quản lý thực đơn cũng cần phải linh hoạt, cho phép nhà hàng dễ dàng thêm món mới, cập nhật giá cả, và đặt trạng thái "hết hàng" khi cần thiết.

Hệ thống khuyến mãi linh hoạt là một nhu cầu quan trọng khác của nhà hàng. Họ muốn có thể tự tạo và quản lý các chương trình giảm giá, coupon theo ý muốn để thu hút khách hàng mà không phụ thuộc hoàn toàn vào các chương trình khuyến mãi của nền tảng. Điều này giúp nhà hàng chủ động hơn trong việc điều chỉnh chiến lược kinh doanh và cạnh tranh trên thị trường.

Chi phí hoa hồng hợp lý cũng là mối quan tâm lớn của các nhà hàng, đặc biệt là những cơ sở nhỏ lẻ với biên lợi nhuận không cao. Họ mong muốn có thể tiếp cận được khách hàng mới thông qua nền tảng trực tuyến mà không phải gánh chịu chi phí quá lớn. Cuối cùng, nhà hàng cũng cần được hỗ trợ kỹ thuật kịp thời khi gặp vấn đề trong quá trình sử dụng hệ thống.

### **c, Những thách thức chưa được giải quyết tốt**

Mặc dù thị trường đã có những ứng dụng thành công, vẫn tồn tại một số thách thức chưa được giải quyết một cách tối ưu. Thách thức lớn nhất và rõ ràng nhất là quy trình đặt hàng nhóm phức tạp với vấn đề thanh toán tập trung. Việc một người phải đứng ra thanh toán cho cả nhóm không chỉ tạo ra gánh nặng tài chính tạm thời mà còn gây ra những tình huống khó xử trong các mối quan hệ xã hội, đặc biệt khi có thành viên quên hoặc chậm trả tiền.

Giao diện ứng dụng quá phức tạp với nhiều tính năng không cần thiết là một thách thức khác. Các ứng dụng hiện tại thường cố gắng tích hợp quá nhiều chức năng, dẫn đến việc người dùng, đặc biệt là những người ít am hiểu công nghệ, cảm

thấy khó khăn trong việc điều hướng và sử dụng. Điều này đặc biệt trở nên rõ ràng khi người dùng có thời gian hạn chế, như trong giờ nghỉ trưa, và muốn đặt hàng một cách nhanh chóng.

Thiếu giải pháp tối ưu cho nhóm khách hàng trẻ, đặc biệt là sinh viên và nhân viên văn phòng, cũng là một khoảng trống đáng chú ý. Nhóm người dùng này có thói quen và nhu cầu rất cụ thể: họ thường đặt hàng theo nhóm, có ngân sách hạn chế, ưa thích sự đơn giản và tốc độ, nhưng lại chưa có ứng dụng nào thực sự được thiết kế riêng để phục vụ những đặc thù này.

### **d, Định hướng phát triển cho YummyZoom**

Dựa trên việc phân tích các nhu cầu và thách thức trên, YummyZoom được định hướng phát triển theo những nguyên tắc rõ ràng. Trước hết, ứng dụng sẽ tập trung vào việc tạo ra một trải nghiệm đơn giản và trực quan, loại bỏ những tính năng phức tạp không cần thiết để người dùng có thể đặt hàng một cách nhanh chóng và hiệu quả nhất. Giao diện được thiết kế với triết lý "ít hơn là nhiều hơn", ưu tiên tính dễ sử dụng và tốc độ thao tác.

Giải quyết triệt để vấn đề đặt hàng nhóm thông qua tính năng TeamCart là định hướng cốt lõi của YummyZoom. Thay vì yêu cầu một người thanh toán cho cả nhóm, mỗi thành viên sẽ tự chịu trách nhiệm thanh toán phần món ăn của mình, từ đó loại bỏ hoàn toàn những phiền phức và ngại ngùng trong các mối quan hệ xã hội. Tính năng này không chỉ là một cải tiến kỹ thuật mà còn mang lại giá trị thực tiễn cao cho người dùng.

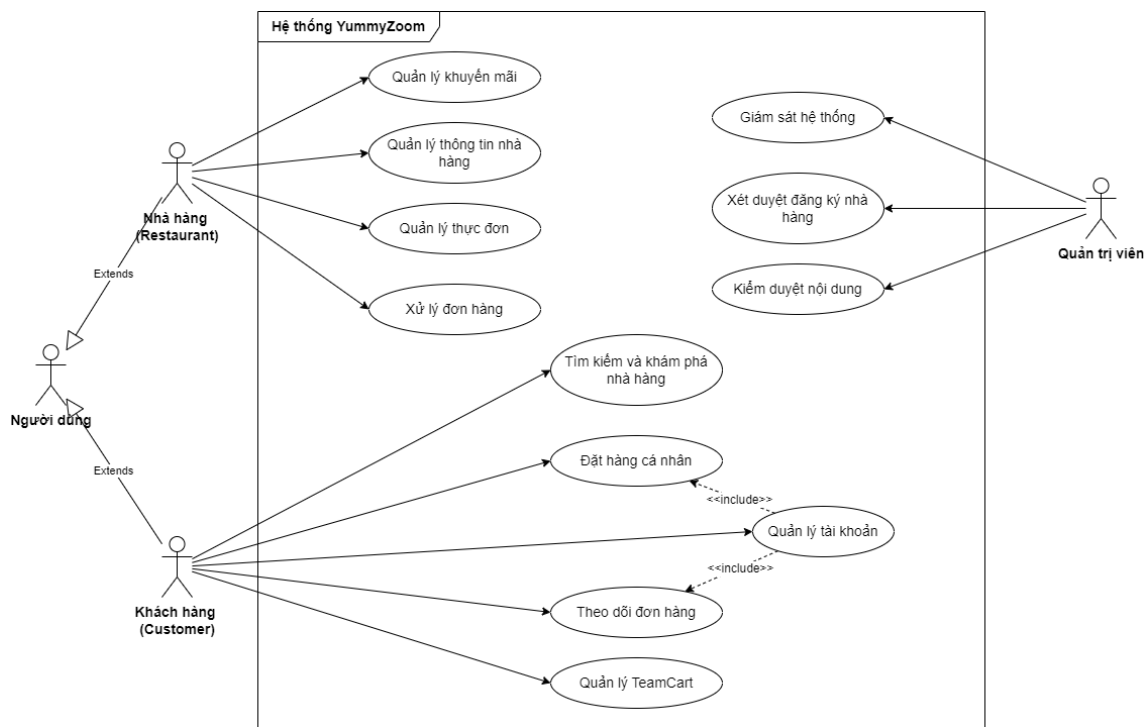
Cuối cùng, YummyZoom được định hướng để phục vụ đặc thù của sinh viên và nhân viên văn phòng - những người có nhu cầu đặt hàng nhóm cao, thời gian hạn chế, và ưa thích sự tiện lợi. Từ việc tối ưu hóa quy trình đặt hàng cho đến việc cung cấp các tùy chọn thanh toán phù hợp với thói quen của thế hệ trẻ, tất cả đều được thiết kế với sự hiểu biết sâu sắc về nhu cầu và hành vi của nhóm khách hàng mục tiêu này.

## **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 số 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.

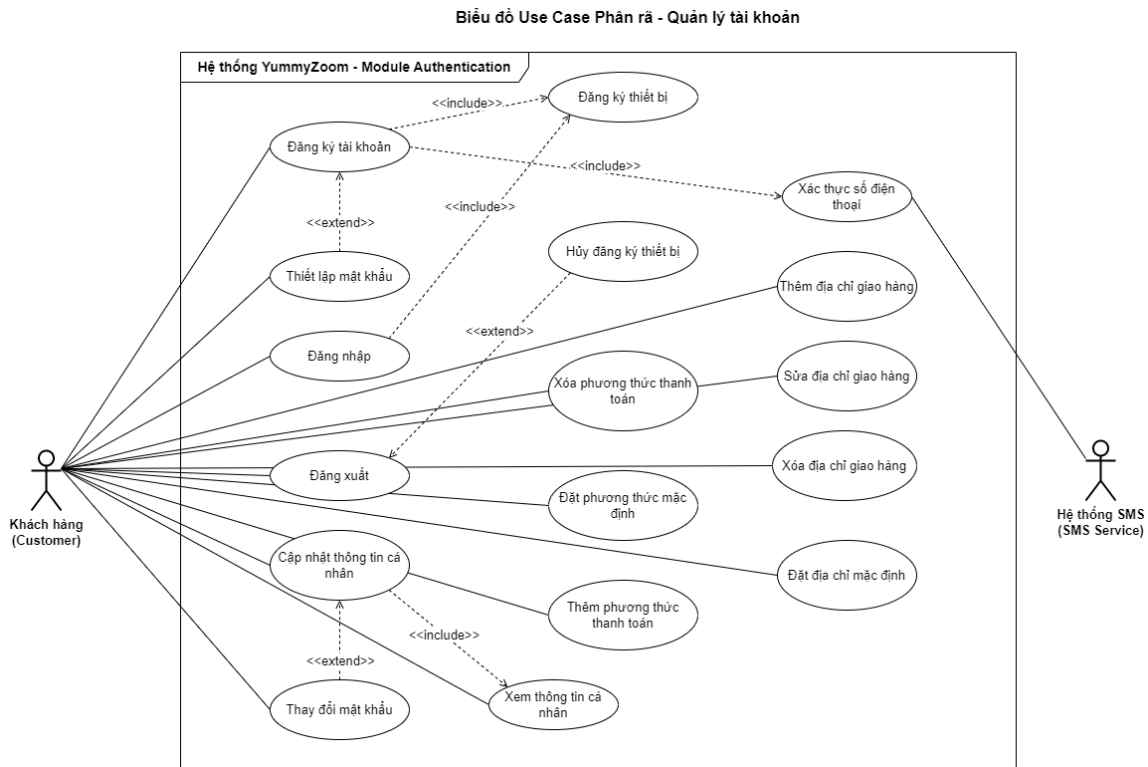
### **Nhóm chức năng của Quản trị viên**

**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"

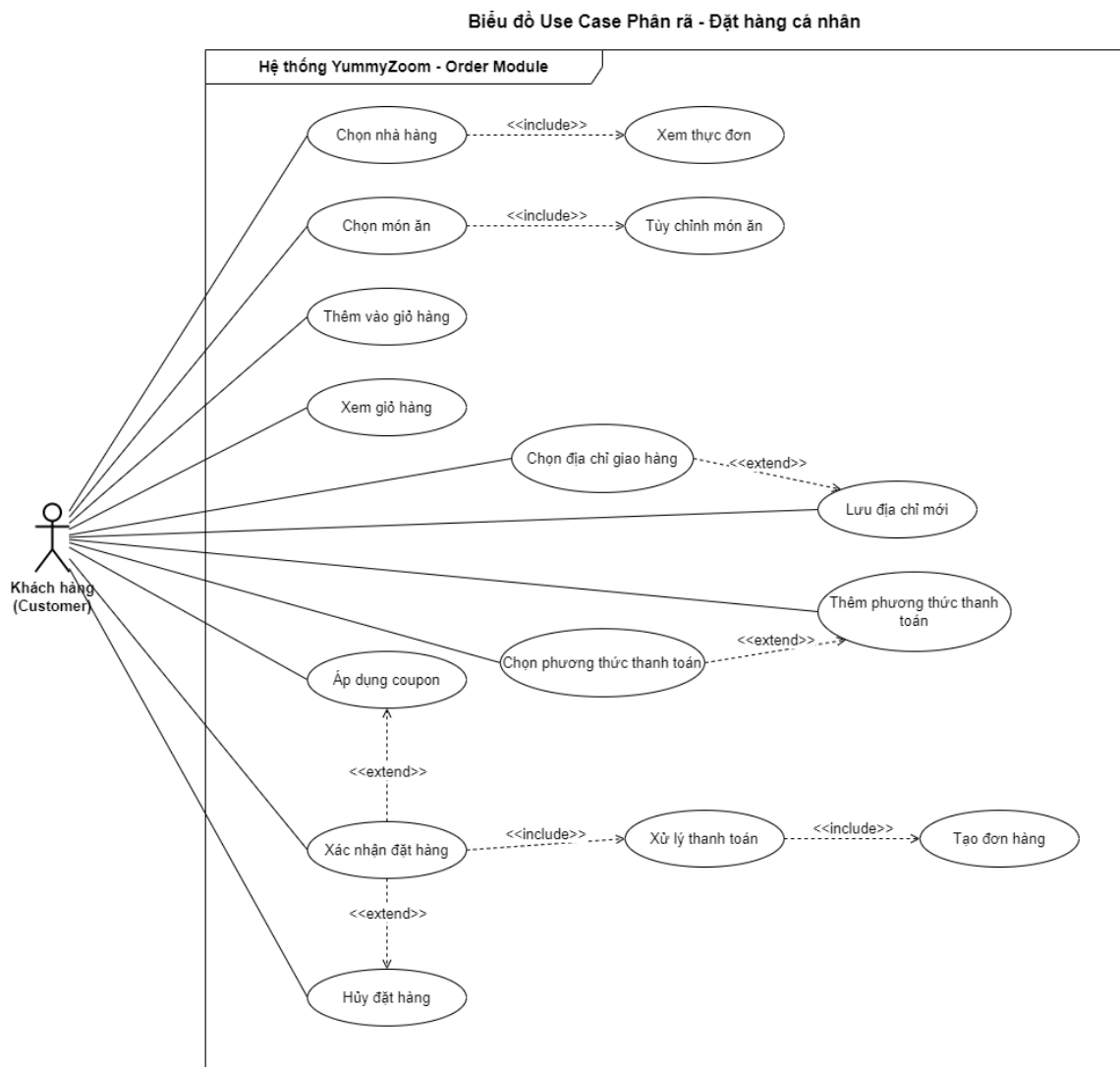


**Hình 2.2:** Biểu đồ use case phân rã - Quản lý tài khoản

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.

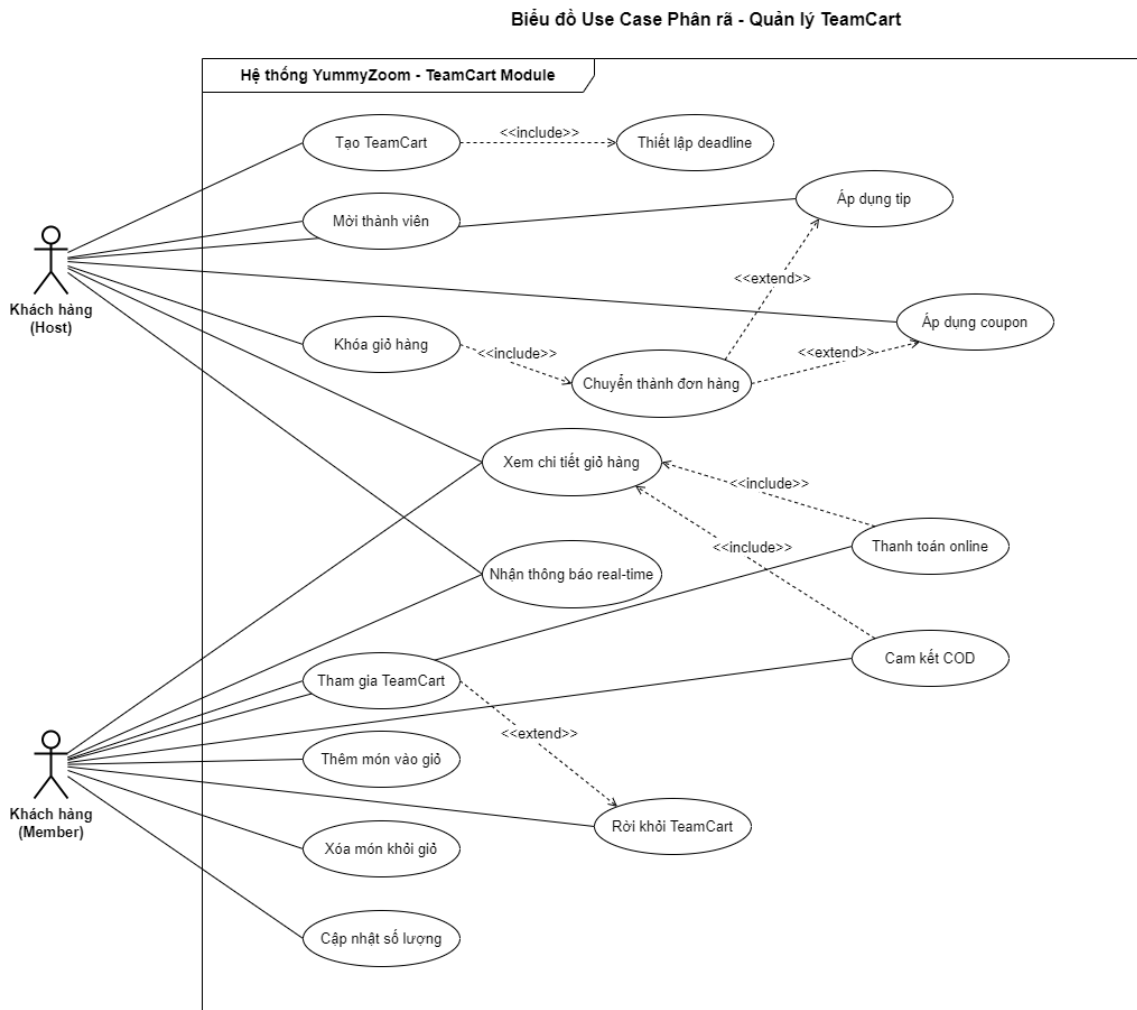
### 2.2.3 Biểu đồ use case phân rã "Đặt hàng cá nhân"



**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"

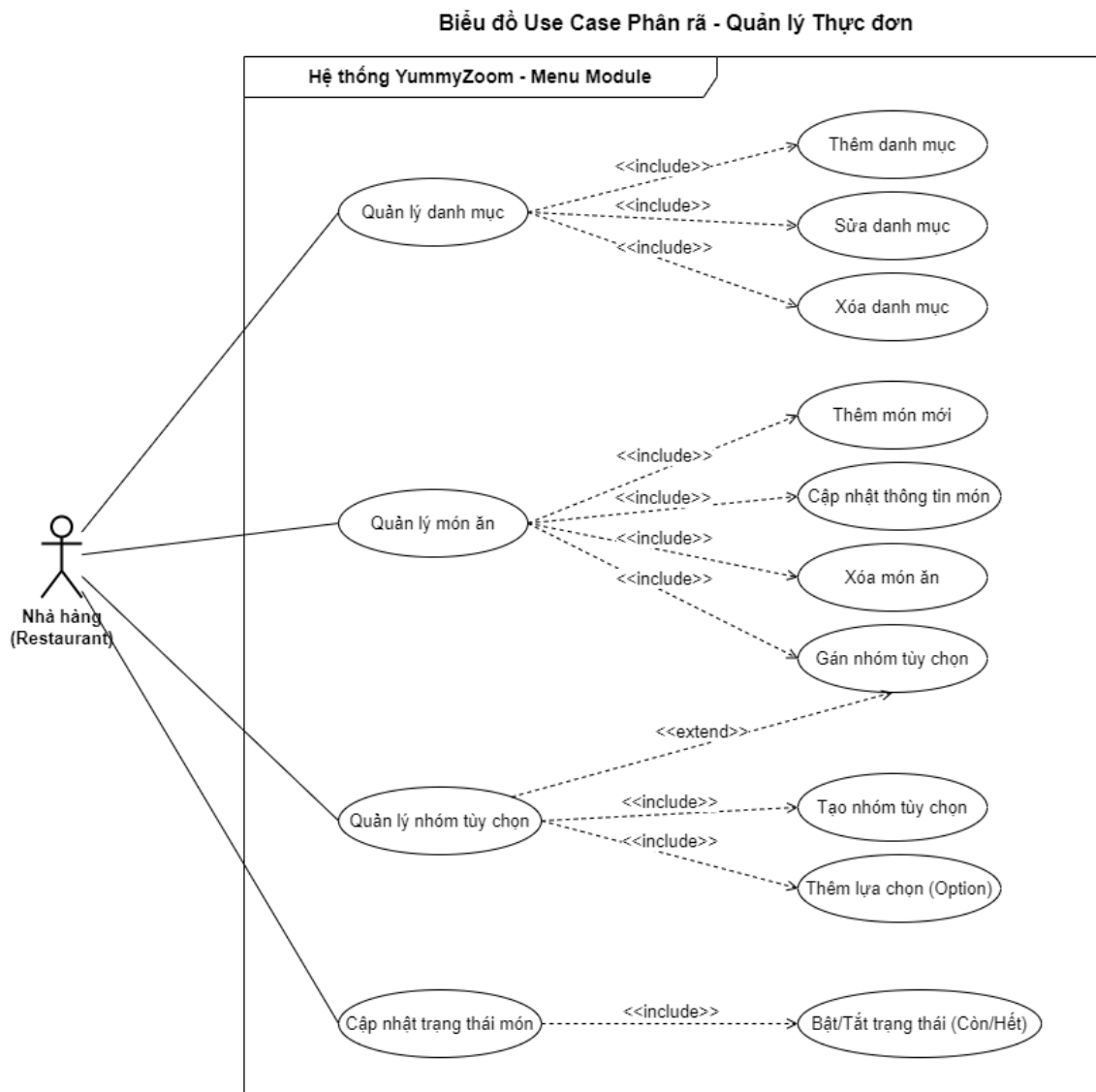


**Hình 2.4:** Biểu đồ use case phân rã - TeamCart

Thông 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.

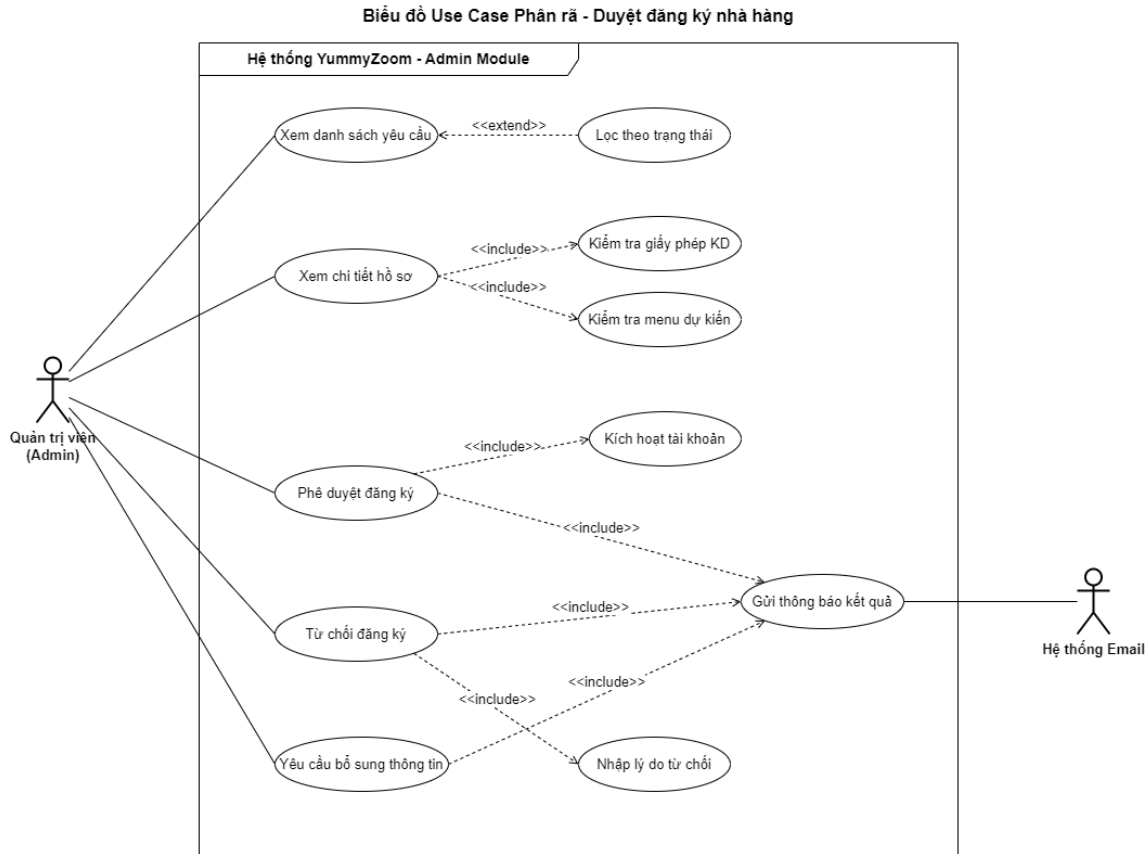


**Hình 2.5:** Biểu đồ use case phân rã - Quản lý thực đơn

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/đá). 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

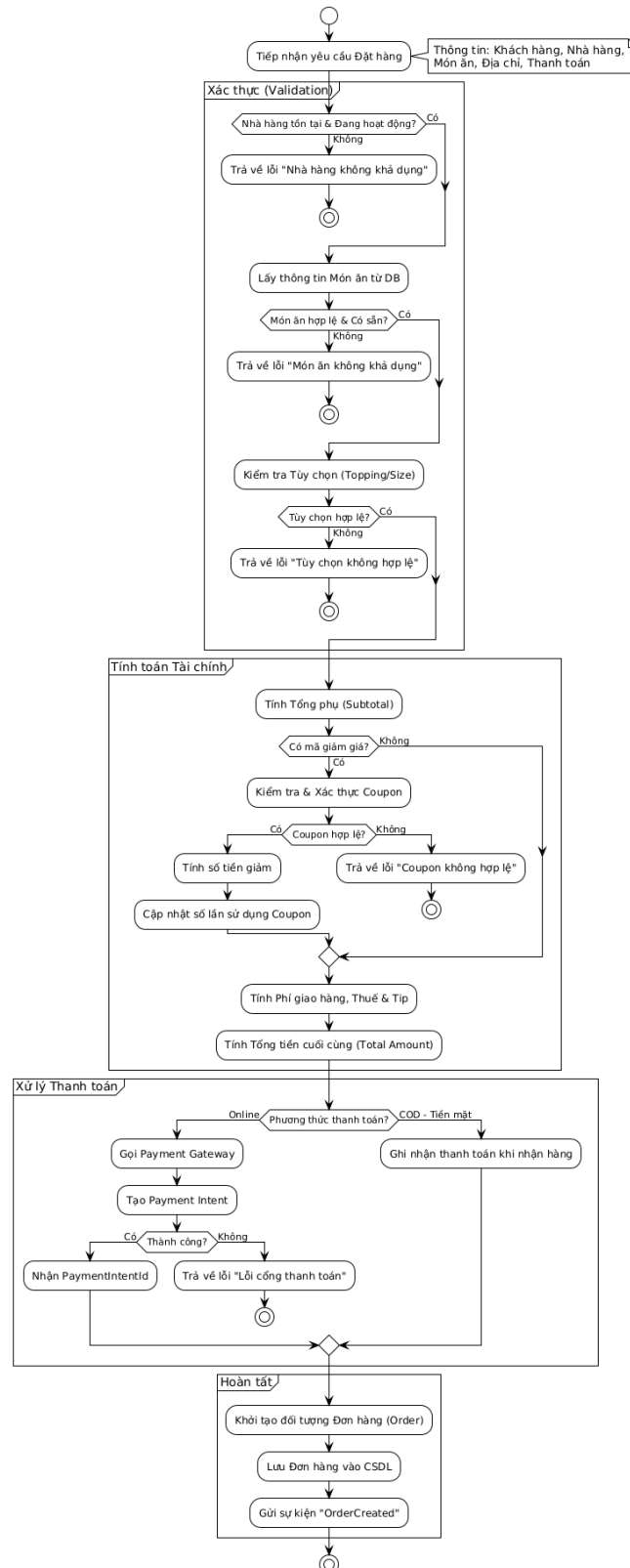
Quy trình này bao gồm các bước kiểm tra tính pháp lý (giấy phép kinh doanh), xác thực thực đơn dự kiến và cơ chế phản hồi (phê duyệt/từ chối/yêu cầu bổ sung) thông qua hệ thống email tự động. Điều này giúp đảm bảo tính minh bạch và chuyên nghiệp trong quy trình hợp tác.

### 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.

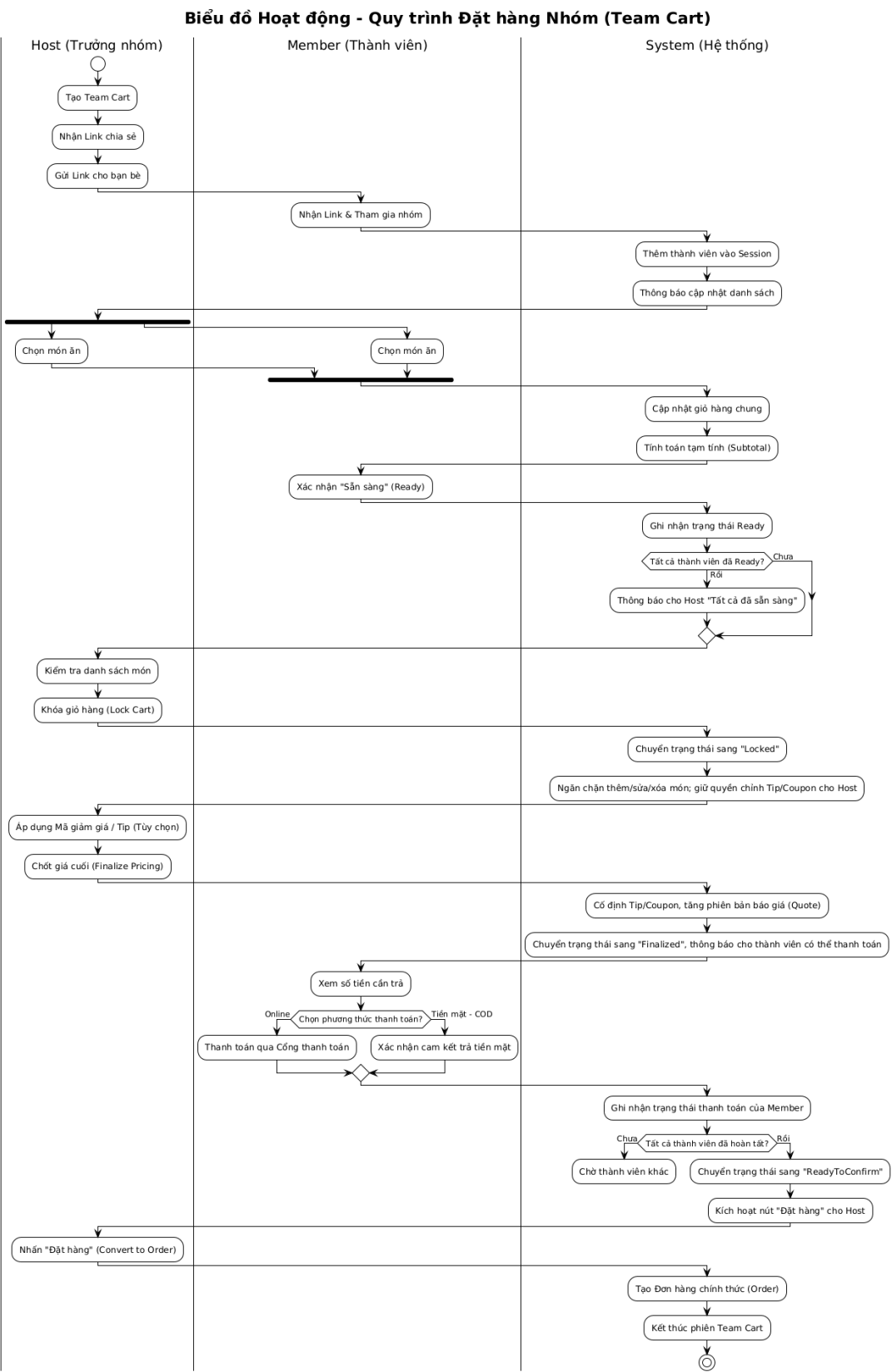
Hình 2.7 dưới đây mô tả chi tiết các bước trong quy trình xử lý đặt hàng, từ lúc tiếp nhận yêu cầu, xác thực, tính toán tài chính cho đến khi hoàn tất đơn hàng.

**Biểu đồ Hoạt động - Quy trình Xử lý Đặt 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 điều kiện</b>	Người dùng đã đăng nhập và tài khoản đang hoạt động.		
<b>Luồng sự kiện chính</b>	<b>STT</b>	<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>	<b>STT</b>	<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
4	Street	Số nhà, tên đường	Có	Tối đa 200 ký tự
5	City	Tên thành phố	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
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: Duyệt đăng ký nhà hàng

<b>Mã use case</b>	UC-012	<b>Tên use case</b>	Duyệt đăng ký nhà hàng
<b>Tác nhân</b>	Quản trị viên (Admin)		
<b>Mục đích sử dụng</b>	Cho phép Admin xem xét và phê duyệt yêu cầu đăng ký nhà hàng, đồng thời khởi tạo nhà hàng mới trên hệ thống.		
<b>Sự kiện kích hoạt</b>	Admin nhấn nút "Duyệt" (Approve) trên chi tiết hồ sơ đăng ký.		
<b>Tiền điều kiện</b>	Admin đã đăng nhập. Hồ sơ đăng ký tồn tại và đang ở trạng thái chờ duyệt (Submitted/UnderReview).		
<b>Luồng sự kiện chính</b>	<b>STT</b>	<b>Thực hiện bởi</b>	<b>Hành động</b>
	1	Admin	Nhập ghi chú (tùy chọn) và xác nhận duyệt.
	2	Hệ thống	Tạo mới thực thể Nhà hàng (Restaurant) từ thông tin đăng ký.
	3	Hệ thống	Gán quyền Chủ sở hữu (Owner) cho tài khoản người đăng ký.
	4	Hệ thống	Cập nhật trạng thái hồ sơ thành "Đã duyệt" (Approved).
	5	Hệ thống	Gửi thông báo thành công cho Admin và người đăng ký.
<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 quá trình tạo nhà hàng hoặc gán quyền thất bại.
<b>Hậu điều kiện</b>	Nhà hàng mới được tạo và hiển thị trên hệ thống. Người đăng ký có quyền quản lý nhà hàng đó.		

**Bảng 2.3:** Đặc tả Use Case Duyệt đăng ký nhà hàng

Dữ liệu đầu vào cho use case Duyệt đăng ký nhà hàng:

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	RegistrationId	ID của hồ sơ đăng ký	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	Note	Ghi chú nội bộ của Admin	Không	Tối đa 500 ký tự

**Bảng 2.4:** Dữ liệu đầu vào Duyệt đăng ký nhà hàng

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

<b>Mã use case</b>	UC-008	<b>Tên use case</b>	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 điều 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>	<b>STT</b>	<b>Thực hiện bởi</b>	<b>Hành động</b>
	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...).
	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, Menu-Category).
	6	Hệ thống	Phát sự kiện domain (MenuItemCreated, MenuItemUpdated, MenuItemDeleted...).
	7	Hệ thống	Cập nhật read model Full-MenuView 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>	<b>STT</b>	<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).

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	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 FullMenu-View. Sự kiện domain được ghi lại phục vụ phân tích.		

**Bảng 2.5:** 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 2.6:** Dữ liệu đầu vào Quản lý thực đơn - Tạo món ăn

### 2.3.4 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>	<b>STT</b>	<b>Thực hiện bởi</b>	<b>Hành động</b>
	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...).

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	7a	Hệ thống	Trả về danh sách rỗng nếu không tìm thấy kết quả phù hợp.
<b>Hậu điều kiện</b>	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 2.7:** 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 2.8:** Dữ liệu đầu vào Tìm kiếm nhà hàng

### 2.3.5 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
	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, PayPal, ApplePay, GooglePay, COD).

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	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.
	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 điều 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 2.9:** 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
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, PayPal, ApplePay, GooglePay, 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 2.10:** Dữ liệu đầu vào Đặt hàng cá nhân

### 2.3.6 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>	<b>STT</b>	<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>	<b>STT</b>	<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 2.11:** 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 2.12:** Dữ liệu đầu vào Khởi tạo TeamCart

### 2.3.7 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>	<b>STT</b>	<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.
	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 điều kiện</b>	Người dùng trở thành thành viên của TeamCart, có quyền thêm món ăn vào giỏ chung.		

**Bảng 2.13:** 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 Team-Cart
3	GuestName	Tên hiển thị	Có	Không rỗng, tối đa 200 ký tự

**Bảng 2.14:** Dữ liệu đầu vào Tham gia TeamCart

### 2.3.8 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 chốt 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 2.15:** 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 2.16:** 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 2.17:** Dữ liệu đầu vào Chốt đơn TeamCart

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

Mã use case	UC-009	Tên use case	Xử lý đơn hàng
Tác nhân	Nhà hàng (Restaurant Staff)		
Mục đích sử dụng	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.		
Sự kiện kích hoạt	Nhà hàng nhận được thông báo có đơn hàng mới (trạng thái Placed) trên giao diện quản lý.		
Tiền điều kiện	Đơn hàng tồn tại và đang ở trạng thái chờ xử lý (Placed). Người dùng có quyền quản lý đơn hàng của nhà hàng.		
Luồng sự kiện chính	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.
	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.

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

	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 điều 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 2.18:** 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
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 2.19:** Dữ liệu đầu vào Xử lý đơn hàng

**2.3.10 Đặc tả Use Case: Đánh giá nhà hàng**

<b>Mã use case</b>	UC-006	<b>Tên use case</b>	Đánh giá nhà hàng
<b>Tác nhân</b>	Khách hàng		
<b>Mục đích sử dụng</b>	Cho phép khách hàng đánh giá chất lượng dịch vụ và món ăn của nhà hàng sau khi hoàn tất đơn hàng.		
<b>Sự kiện kích hoạt</b>	Khách hàng chọn chức năng "Đánh giá" trên lịch sử đơn hàng đã giao thành công.		
<b>Tiền điều kiện</b>	Khách hàng đã đăng nhập. Đơn hàng đã hoàn tất (Delivered). Khách hàng chưa từng đánh giá nhà hàng này trước đó (hoặc hệ thống cho phép đánh giá lại).		
<b>Luồng sự kiện chính</b>	<b>STT</b>	<b>Thực hiện bởi</b>	<b>Hành động</b>
	1	Khách hàng	Chọn mức đánh giá (số sao từ 1 đến 5) và nhập nội dung bình luận (tùy chọn).
	2	Hệ thống	Kiểm tra tính hợp lệ của dữ liệu (số sao trong phạm vi, độ dài bình luận).
	3	Hệ thống	Kiểm tra điều kiện nghiệp vụ: Đơn hàng phải thuộc về khách hàng và đã giao thành công.
	4	Hệ thống	Kiểm tra trùng lặp: Đảm bảo khách hàng chưa đánh giá đơn hàng này (hoặc nhà hàng này) trước đó.
	5	Hệ thống	Lưu đánh giá vào hệ thống, cập nhật điểm đánh giá trung bình của nhà hàng và hiển thị thông báo thành công.
<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 số sao không hợp lệ hoặc nội dung quá dài.
	3a	Hệ thống	Báo lỗi nếu đơn hàng chưa hoàn tất hoặc không tồn tại.
	4a	Hệ thống	Báo lỗi nếu khách hàng đã đánh giá trước đó (ReviewAlreadyExists).

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

<b>Hậu kiện</b>	<b>điều kiện</b>	Đánh giá được hiển thị công khai trên trang chi tiết nhà hàng. Điểm số trung bình của nhà hàng được tính toán lại.
-----------------	------------------	---

**Bảng 2.20:** Đặc tả Use Case Đánh giá nhà hàng

Dữ liệu đầu vào cho use case Đánh giá nhà hàng:

STT	Trường dữ liệu	Mô tả	Bắt buộc?	Điều kiện hợp lệ
1	OrderId	ID đơn hàng	Có	GUID hợp lệ, đã giao
2	RestaurantId	ID nhà hàng	Có	GUID hợp lệ
3	Rating	Điểm đánh giá	Có	Số nguyên từ 1 đến 5
4	Title	Tiêu đề đánh giá	Không	Tối đa 100 ký tự
5	Comment	Nội dung chi tiết	Không	Tối đa 1000 ký tự

**Bảng 2.21:** Dữ liệu đầu vào Đánh giá nhà 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 phân quyền dựa trên vai trò (Role-Based Access Control - RBAC) và chính sách (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 bố 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) 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 Domain-Driven Design (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 nhóm 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, nền tảng và 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 ở Chương 2, đặc biệt là các vấn đề về hiệu năng, tính toàn vẹn dữ liệu, khả năng bảo trì và trải nghiệm người dùng.

### 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 khách hàng và nhà hàng. Để đáp ứng các yêu cầu khắt khe về **Khả năng bảo trì** (Maintainability) và **Độ tin cậy** (Reliability) được đặt ra trong Mục 2.4, 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 đối mặt 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) do Robert C. Martin đề xuất, kết hợp với các nguyên lý của **Domain-Driven Design** (Thiết kế hướng tên miền) [1], [2].

Về mặt cấu trúc, hệ thống được tổ chức thành các lớp với quy tắc phụ thuộc hướng tâm (Dependency Rule), đảm bảo sự độc lập của logic nghiệp vụ:

1. **Lớp miền (Domain Layer):** Đây là lõi của hệ thống, chứa các thực thể (Entities), đối tượng giá trị (Value Objects) và các quy tắc nghiệp vụ (Business Rules) 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.
2. **Lớp ứng dụng (Application Layer):** Đóng vai trò điều phối các ca sử dụng (Use Cases). Lớp này nhận yêu cầu từ bên ngoài, sử dụng các thực thể trong Domain để thực hiện tác vụ và trả về kết quả.
3. **Lớp hạ tầng (Infrastructure Layer):** Cài đặt chi tiết các giao tiếp kỹ thuật như truy xuất cơ sở dữ liệu, gửi email, tích hợp cổng thanh toán.
4. **Lớp giao diện (Presentation/Web Layer):** Nơi chứa các API Controller để tiếp nhận yêu cầu HTTP từ ứng dụng di động hoặc web quản trị.

Sự kết hợp này mang lại lợi ích vượt trội so với các kiến trúc khác:

- **So với Kiến trúc phân tầng truyền thống (Layered Architecture):** Kiến trúc truyền thống thường dẫn đến sự phụ thuộc chặt chẽ giữa logic nghiệp vụ và cơ sở dữ liệu (ví dụ: thay đổi cấu trúc bảng làm hỏng logic code). Clean Architecture loại bỏ vấn đề này bằng cách đảo ngược sự phụ thuộc (Dependency Inversion), giúp logic nghiệp vụ ổn định và dễ dàng kiểm thử đơn vị (Unit Test) mà không cần kết nối database thực tế.
- **So với Kiến trúc Microservices:** Mặc dù Microservices mang lại khả năng mở rộng tốt, nhưng mô hình này đòi hỏi hạ tầng vận hành phức tạp (DevOps, Kubernetes) và chi phí giao tiếp giữa các dịch vụ lớn. Với quy mô nhân sự của một đồ án tốt nghiệp và mục tiêu phục vụ 50-100 người dùng đồng thời, việc áp dụng Clean Architecture dưới dạng Monolithic Modular (đơn khối module hóa) là giải pháp tối ưu, cân bằng giữa tốc độ phát triển và chất lượng mã nguồn, đồng thời dễ dàng tách module thành microservice khi cần mở rộng sau này.

### 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** [3]. 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 hai bài toán lớn được đặt ra ở phần khảo sát:

- **Tối ưu hóa hiệu năng và khả năng chịu tải:** Web server Kestrel của ASP.NET Core được tối ưu hóa cực tốt cho việc xử lý hàng nghìn yêu cầu mỗi giây. Kết hợp với mô hình lập trình bất đồng bộ (Async/Await), hệ thống có thể xử lý hiệu quả các tác vụ I/O (như đọc ghi dữ liệu, gọi API thanh toán) mà không làm tắc nghẽn luồng xử lý chính (Thread Blocking). Điều này đảm bảo thời gian phản hồi API luôn duy trì ở mức thấp (dưới 500ms) ngay cả khi có 50-100 người dùng thao tác cùng lúc trên TeamCart.
- **An toàn và dễ bảo trì:** Ngôn ngữ C# là ngôn ngữ định kiểu tĩnh mạnh (Strongly Typed), giúp phát hiện phần lớn các lỗi logic ngay trong quá trình biên dịch (Compile time), giảm thiểu đáng kể lỗi runtime so với các ngôn ngữ định kiểu động như JavaScript/Python.

Khi so sánh với các nền tảng phổ biến khác:

- **So với Node.js (Express/NestJS):** Node.js sử dụng mô hình đơn luồng (Single-thread event loop), rất tốt cho I/O nhưng có thể bị suy giảm hiệu năng nếu

gặp các tác vụ tính toán phức tạp (như tổng hợp báo cáo doanh thu). .NET 9 với khả năng đa luồng (Multi-threading) quản lý tốt cả hai loại tác vụ này.

- **So với Java (Spring Boot):** Mặc dù Spring Boot rất mạnh mẽ, nhưng hệ sinh thái Java thường tiêu tốn nhiều tài nguyên bộ nhớ (RAM) hơn và có thời gian khởi động (Startup time) lâu hơn so với .NET Core hiện đại. Với yêu cầu triển khai linh hoạt và tiết kiệm tài nguyên trên môi trường Cloud hoặc VPS sinh viên, .NET 9 là lựa chọn kinh tế và hiệu quả hơn.

### 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 (Web API, Worker Service, Database, Redis) ngay từ môi trường phát triển (Local Development), dự án áp dụng stack công nghệ **.NET Aspire** [4].

Vai trò của .NET Aspire trong hệ thống YummyZoom:

- **Điều phối tài nguyên (Orchestration):** Thay vì phải cấu hình thủ công từng container Docker hoặc chạy nhiều cửa sổ dòng lệnh rời rạc, .NET Aspire cho phép định nghĩa toàn bộ hạ tầng (Infrastructure as Code) bằng C#. Chỉ với một nút nhấn "Start", toàn bộ hệ sinh thái bao gồm Backend, Database PostgreSQL container, và Redis container sẽ được khởi tạo và kết nối tự động thông qua cơ chế Service Discovery.
- **Khả năng quan sát (Observability):** Tích hợp sẵn Dashboard tập trung hiển thị Logs (Nhật ký), Metrics (Thông số hiệu năng) và Traces (Truy vết) theo chuẩn OpenTelemetry. Điều này giúp đội ngũ phát triển dễ dàng nhìn thấy luồng đi của một request từ khi vào API đến khi truy xuất Database, giúp việc gỡ lỗi (Debugging) và tối ưu hiệu năng trở nên trực quan hơn bao giờ hết.

## 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. Để giải quyết các yêu cầu về **Tính toàn vẹn dữ liệu** (Data Integrity) và **Hiệu năng truy xuất** (Performance) đã được đề ra tại Mục 2.4, hệ thống sử dụng chiến lược lưu trữ kết hợp giữa Cơ sở dữ liệu quan hệ (RDBMS) và Bộ nhớ đệm (Caching).

### 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** đóng vai trò là lớp ORM (Object-Relational Mapping) để tương tác dữ liệu từ mã nguồn .NET [5]. Đây là giải pháp lưu trữ bền vững, được

thiết kế để giải quyết triệt để vấn đề toàn vẹn dữ liệu cho tính năng nhạy cảm như thanh toán và quản lý đơn hàng.

Các lý do chính cho quyết định này bao gồm:

- **Đảm bảo tính ACID tuyệt đối:** Đối với ứng dụng giao đồ ăn, đặc biệt là tính năng TeamCart nơi nhiều người cùng thao tác trên một đơn hàng, việc đảm bảo tính Nguyên tử (Atomicity), Nhất quán (Consistency), Cô lập (Isolation) và Bền vững (Durability) là bắt buộc. PostgreSQL nổi tiếng với cơ chế quản lý giao dịch (Transaction) chặt chẽ, giúp ngăn chặn triệt để tình trạng sai lệch dữ liệu tài chính (ví dụ: trừ tiền nhưng không tạo đơn hàng).
- **Bộ tính năng nâng cao tích hợp sẵn:** Điểm vượt trội của PostgreSQL nằm ở hệ sinh thái các tiện ích mở rộng giúp giải quyết trọn vẹn các bài toán nghiệp vụ đặc thù mà không cần phụ thuộc vào dịch vụ bên thứ ba:
  - *Tìm kiếm toàn văn (Full Text Search):* Hệ thống tận dụng engine tìm kiếm tích hợp sẵn để thực hiện chức năng tìm kiếm món ăn và nhà hàng. Với khả năng đánh chỉ mục (indexing) văn bản và hỗ trợ tách từ, giải pháp này đáp ứng tốt nhu cầu tìm kiếm tiếng Việt, giúp tiết kiệm tài nguyên so với việc triển khai các Search Engine rời như Elasticsearch.
  - *Xử lý dữ liệu địa lý (PostGIS):* Đây là tính năng quan trọng để hỗ trợ tìm kiếm theo vị trí. Hệ thống sử dụng PostGIS để thực hiện các truy vấn không gian như "tìm nhà hàng trong bán kính 3km" hay tính toán chính xác khoảng cách đường đi để xác định phí giao hàng, được thực hiện trực tiếp tại tầng cơ sở dữ liệu với tốc độ xử lý rất cao.
  - *Kiểu dữ liệu JSONB:* Cấu trúc Menu với nhiều nhóm Tùy chọn/Topping phức tạp được lưu trữ linh hoạt dưới dạng JSONB. Điều này cho phép hệ thống truy vấn nhanh trên các thuộc tính động mà không bị ràng buộc bởi schema cứng nhắc, kết hợp ưu điểm của NoSQL ngay trong lòng CSDL quan hệ.

So sánh với các giải pháp thay thế:

- **So với MySQL:** Mặc dù MySQL có tốc độ đọc nhanh ở các tác vụ đơn giản, PostgreSQL vượt trội hơn ở khả năng xử lý các truy vấn phức tạp (Complex Queries) cần thiết cho các báo cáo doanh thu. Quan trọng hơn, engine của PostgreSQL hoạt động ổn định hơn trong các kịch bản ghi dữ liệu cường độ cao (High concurrency write) vào giờ cao điểm.
- **So với Microsoft SQL Server:** SQL Server tương thích tốt với .NET nhưng chi phí bản quyền là rào cản lớn khi triển khai thực tế. PostgreSQL là mã

nguồn mở (Open Source), giúp tiết kiệm chi phí vận hành mà vẫn cung cấp đầy đủ các tính năng doanh nghiệp, đồng thời chạy tối ưu trên môi trường Linux giúp giảm chi phí hạ tầng server.

### 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 không chỉ là giảm tải cho PostgreSQL mà còn đảm bảo tốc độ xử lý thời gian thực cho các tính năng tương tác cao.

Cụ thể, Redis được ứng dụng trong hai kịch bản trọng yếu:

1. **Lưu trữ dữ liệu tĩnh (Read-heavy):** Các thông tin ít thay đổi như Danh sách nhà hàng và Thực đơn (Menu) được lưu trong cache. Điều này giúp các truy vấn điều hướng cơ bản đạt độ trễ cực thấp (sub-millisecond) mà không cần truy cập Database.
2. **Quản lý trạng thái TeamCart (Write-heavy):** Đây là ứng dụng quan trọng nhất của Redis trong hệ thống. Đối với các Giỏ hàng nhóm đang hoạt động (Active TeamCarts), toàn bộ dữ liệu hiển thị (View Model) được lưu trữ trực tiếp trên RAM của Redis. Khi các thành viên trong nhóm thao tác thêm/bớt món đồ ăn, hệ thống đọc/ghi trực tiếp vào Redis thay vì Database. Cơ chế này loại bỏ hoàn toàn độ trễ I/O đĩa cứng, đảm bảo trải nghiệm người dùng mượt mà ngay cả khi cart có lưu lượng hoạt động cao. Dữ liệu chỉ được đồng bộ xuống PostgreSQL để lưu trữ bền vững (Persistence) khi đơn hàng được chốt hoặc kết thúc phiên.

Việc lựa chọn Redis thay vì sử dụng bộ nhớ cục bộ (In-Memory Cache) của server cho phép hệ thống dễ dàng mở rộng nhiều server (Scale-out) mà vẫn đảm bảo tính nhất quán dữ liệu giữa các phiên làm việc.

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

Đặc thù của ứng dụng giao đồ ăn là yêu cầu lưu trữ và hiển thị lượng lớn hình ảnh chất lượng cao (Avatar người dùng, Ảnh món ăn, Banner nhà hàng). Việc lưu trữ trực tiếp các file này trên Server hoặc trong Database sẽ gây áp lực lớn lên băng thông và dung lượng lưu trữ của hệ thống máy chủ.

Giải pháp được lựa chọn là nền tảng quản lý media dựa trên đám mây **Cloudinary** (SaaS) [6], với các ưu điểm kỹ thuật:

- **Tối ưu hóa tự động:** Cloudinary tự động nén và chuyển đổi định dạng ảnh phù hợp với thiết bị của người dùng (ví dụ: tự động chuyển sang định dạng WebP hoặc AVIF giúp giảm 30-50% dung lượng tải so với JPEG/PNG thông thường mà không giảm chất lượng hiển thị).

- **Mạng phân phối nội dung (CDN):** Hình ảnh được phân phối từ hệ thống Server toàn cầu của Cloudinary thay vì tải trực tiếp từ Backend của ứng dụng. Điều này giúp giảm đáng kể độ trễ hiển thị ảnh trên Mobile App, mang lại trải nghiệm lướt thực đơn mượt mà (Mục 2.1.3).
- **Quản lý tập trung:** Cung cấp API mạnh mẽ để upload và xóa ảnh an toàn từ Backend, tách biệt hoàn toàn tầng lưu trữ file (File Storage Layer) khỏi tầng xử lý nghiệp vụ.

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

Bảo mật thông tin người dùng và kiểm soát quyền truy cập là yêu cầu phi chức năng quan trọng hàng đầu (Mục 2.4). 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)** (RFC 7519) làm phương thức xác thực chính cho toàn bộ các giao tiếp giữa Client (Mobile App, Web Portal) và Server [7].

Cơ chế hoạt động và lợi ích:

- **Xác thực không trạng thái (Stateless Authentication):** Đây là yếu tố then chốt cho một ứng dụng di động có lượng người dùng lớn. Khác với xác thực dựa trên Session truyền thống (lưu trạng thái đăng nhập trên Server), JWT đóng gói toàn bộ thông tin định danh vào một chuỗi token duy nhất và gửi kèm theo mỗi request HTTP. Server chỉ cần xác thực chữ ký (Signature) của token mà không cần truy vấn lại cơ sở dữ liệu hay bộ nhớ đệm session.
- **Khả năng mở rộng (Scalability):** Nhờ tính chất stateless, hệ thống có thể dễ dàng mở rộng theo chiều ngang (thêm nhiều server xử lý) mà không gặp phải vấn đề đồng bộ session giữa các server (Sticky Session).
- **Hiệu năng:** Giảm tải đáng kể cho hệ thống lưu trữ vì không cần tra cứu session ID cho mỗi yêu cầu người dùng, giúp 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 được Microsoft thiết kế sẵn, giúp đội ngũ phát triển tập trung vào logic nghiệp vụ thay vì phải tự xây dựng lại các module an ninh nhạy cảm.

Các chức năng chính được sử dụng:

- **Quản lý người dùng và mật khẩu:** Identity cung cấp sẵn các API an toàn để đăng ký, đăng nhập. Mọi mật khẩu người dùng đều được băm (hashing) bằng các thuật toán hiện đại (như PBKDF2) kết hợp với Salt ngẫu nhiên trước khi lưu vào PostgreSQL, đảm bảo an toàn ngay cả khi dữ liệu bị rò rỉ.
- **Kiểm soát truy cập dựa trên vai trò (RBAC):** Hệ thống định nghĩa các vai trò (Roles) cụ thể như:
  - *Customer:* Người dùng đặt món, chỉ có quyền truy cập dữ liệu cá nhân.
  - *Restaurant Owner/Staff:* Đối tác nhà hàng, có quyền quản lý thực đơn và xem báo cáo doanh thu của quán mình sở hữu.
  - *Admin:* Quản trị viên hệ thống, có toàn quyền quản lý danh mục và tài khoản hệ thống.

Identity bảo vệ các API Endpoint bằng cách kiểm tra Role (Claims) có trong JWT, từ chối ngay lập tức các 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 (Sinh viên, nhân viên văn phòng). Do đó, việc xây dựng một trải nghiệm người dùng (UX) mượt mà, phản hồi nhanh và đồng nhất trên các thiết bị khác nhau là yêu cầu bắt buộc để đáp ứng tiêu chí đã đề ra tại Mục 2.1.3.

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

Dự án lựa chọn **Flutter SDK** sử dụng ngôn ngữ **Dart** để phát triển ứng dụng di động cho khách hàng [8]. Đây là bộ công cụ UI của Google cho phép xây dựng ứng dụng biên dịch natively (native-compiled) cho di động, web và máy tính để bàn từ một mã nguồn duy nhất.

Các lý do chính cho quyết định công nghệ này:

- **Hiệu năng Native:** Khác với các framework lai (hybrid) dựa trên WebView, Flutter sử dụng engine đồ họa riêng (Skia/Impeller) để vẽ trực tiếp lên màn hình, đảm bảo tốc độ khung hình ổn định ở mức 60fps (thậm chí 120fps). Điều này cực kỳ quan trọng đối với các thao tác cuộn danh sách món ăn dài hay các hiệu ứng chuyển động phức tạp trong giao diện đặt món.
- **Kiểm soát giao diện tuyệt đối (Pixel Perfect):** Tính năng *TeamCart* yêu cầu một giao diện chia sẻ trạng thái phức tạp và tùy biến cao (ví dụ: avatar người dùng bay vào giỏ hàng, cập nhật số lượng realtime). Flutter cho phép kiểm soát từng pixel trên màn hình, giúp hiện thực hóa chính xác các thiết kế giao

diện độc đáo mà không bị phụ thuộc vào các widget mặc định của hệ điều hành.

- **Tăng tốc độ phát triển:** Tính năng *Hot Reload* của Flutter giúp đội ngũ phát triển xem ngay kết quả thay đổi mã nguồn mà không cần biên dịch lại từ đầu, rút ngắn đáng kể thời gian tinh chỉnh giao diện (UI Polish).

So sánh với các lựa chọn khác:

- **So với React Native:** React Native phụ thuộc vào cầu nối (Bridge) JavaScript để giao tiếp với các module Native, có thể gây nghẽn cổ chai về hiệu năng khi xử lý nhiều tác vụ phức tạp đồng thời. Flutter loại bỏ cầu nối này nhờ biên dịch trước (AOT Compilation), mang lại hiệu năng ổn định hơn.
- **So với Phát triển Native thuần túy (Kotlin/Swift):** Việc phát triển riêng biệt hai ứng dụng Android và iOS đòi hỏi gấp đôi nguồn lực nhân sự và thời gian bảo trì. Flutter giúp tiết kiệm chi phí này nhờ khả năng chia sẻ hơn 90% mã nguồn giữa các nền tảng.

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

Để xử lý luồng dữ liệu phức tạp, đặc biệt là sự đồng bộ hóa liên tục của Giỏ hàng nhóm, ứng dụng áp dụng mô hình quản lý trạng thái (State Management) chặt chẽ:

- **Quản lý trạng thái với Provider:** Ứng dụng lựa chọn sử dụng thư viện **Provider** để quản lý trạng thái. Giải pháp này được đánh giá là phù hợp nhất với quy mô hiện tại của dự án, chưa quá lớn nhưng vẫn đòi hỏi sự tổ chức code bài bản. Provider mang lại hiệu quả vượt trội so với việc sử dụng biến trạng thái thông thường (`setState`) nhờ khả năng tách biệt logic nghiệp vụ khỏi giao diện (UI) và giải quyết triệt để vấn đề truyền dữ liệu qua nhiều cấp (prop drilling), đồng thời giữ cho cấu trúc dự án gọn nhẹ, tránh sự phức tạp không cần thiết của các mô hình như BLoC.
- **Giao tiếp mạng với Dio:** Thư viện Dio được sử dụng để quản lý các kịch bản gọi API nâng cao như tự động hủy request (Cancellation), xử lý timeout và chặn bắt tập trung (Interceptors), đảm bảo ứng dụng luôn phản hồi chính xác tình trạng kết nối mạng cho người dùng.

### 3.4.3 Tích hợp bản đồ và Thanh toán

- **Bản đồ số:** Hệ thống tích hợp **Mapbox SDK** để thực hiện các chức năng định vị và bản đồ. Mapbox được ưu tiên lựa chọn nhờ sự hỗ trợ mạnh mẽ cho nền tảng Flutter giúp việc tích hợp mượt mà, đồng thời cung cấp hạn mức sử dụng miễn phí lớn cho nhà phát triển, giúp tối ưu chi phí xây dựng dự án.

- **Cổng thanh toán:** Ứng dụng tích hợp cổng thanh toán **Stripe** thông qua SDK chính thức. Stripe giúp đơn giản hóa quá trình kết nối với Backend, đồng thời cung cấp môi trường Sandbox và Test Mode (Chế độ kiểm thử) toàn diện. Điều này cho phép đội ngũ phát triển mô phỏng các kịch bản thanh toán thành công, thất bại, hoặc hoàn tiền một cách dễ dàng và an toàn trước khi triển khai thực tế.

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

Đáp ứng nhu cầu quản lý của **Đối tác nhà hàng** đã được xác định trong Mục 2.1.3, hệ thống cung cấp một cổng thông tin quản trị tập trung. Đây là nơi các chủ nhà hàng thực hiện các thao tác quản lý thực đơn, theo dõi đơn hàng và xem báo cáo doanh thu.

#### 3.5.1 Nền tảng Frontend: Angular

Dự án lựa chọn **Angular** (phiên bản 21 mới nhất) kết hợp với ngôn ngữ **TypeScript** để xây dựng ứng dụng quản trị với công nghệ Single Page Application (SPA) [9].

Các lý do chính cho quyết định này:

- **Cấu trúc Module chặt chẽ:** Khác với các thư viện tự do như ReactJS, Angular cung cấp một khung làm việc (Framework) hoàn chỉnh với kiến trúc hướng module (NgModules) và quản lý phụ thuộc (Dependency Injection) rõ ràng. Điều này rất phù hợp với tư duy lập trình hướng đối tượng (OOP) được sử dụng ở Backend, giúp đội ngũ phát triển dễ dàng đồng bộ kiến thức giữa hai nền tảng.
- **Tích hợp toàn diện:** Angular đi kèm sẵn với các công cụ mạnh mẽ như HttpClient (giao tiếp API), Reactive Forms (xử lý biểu mẫu phức tạp) và Router (điều hướng). Điều này giúp giảm thiểu việc phụ thuộc vào các thư viện bên thứ ba rời rạc, đảm bảo tính ổn định và bảo mật cho ứng dụng quản trị doanh nghiệp.

So sánh với các lựa chọn khác:

- **So với ReactJS:** React chỉ là một thư viện UI, đòi hỏi lập trình viên phải tự lựa chọn và cấu hình thêm nhiều thư viện hỗ trợ (State management, Routing, Form...). Điều này tạo ra sự linh hoạt nhưng cũng làm tăng độ phức tạp khi cần chuẩn hóa cấu trúc dự án.
- **So với VueJS:** Mặc dù VueJS dễ học, nhưng cộng đồng doanh nghiệp (Enterprise) thường ưu tiên Angular hơn cho các hệ thống quản trị lớn nhờ tính chặt chẽ và khả năng bảo trì mã nguồn tốt hơn về lâu dài.

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

Để tối ưu hóa thời gian phát triển giao diện nhưng vẫn đảm bảo tính thẩm mỹ và chuyên nghiệp, dự án sử dụng thư viện **PrimeNG** kết hợp với **Tailwind CSS** [10].

- **PrimeNG:** Cung cấp bộ sưu tập khổng lồ các thành phần UI cao cấp (UI Components) dành riêng cho các ứng dụng quản trị dữ liệu, đặc biệt là các bảng dữ liệu (Data Grid) với khả năng phân trang, lọc, sắp xếp (Sorting/Filtering) và các biểu đồ thống kê (Charts) trực quan.
- **Tailwind CSS:** Framework CSS dạng tiện ích (Utility-first) giúp tùy biến nhanh giao diện, bố cục (Layout) và Responsive mà không cần viết quá nhiều mã CSS thủ công.

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

Đây là trái tim của tính năng **TeamCart**, đáp ứng yêu cầu kỹ thuật khắt khe về độ trễ thấp (dưới 500ms) được đặt ra tại Mục 2.4 để đảm bảo trải nghiệm đồng bộ mượt mà giữa các thành viên.

#### 3.6.1 Giao thức và Framework: SignalR

Hệ thống sử dụng thư viện **ASP.NET Core SignalR**, tận dụng giao thức **WebSockets** làm phương thức vận chuyển chính (Transport) [11].

Vai trò của SignalR trong hệ thống:

- **Đồng bộ TeamCart tức thời:** Khi một thành viên trong nhóm thực hiện hành động (ví dụ: Thêm món "Trà sữa" vào giỏ), SignalR Server sẽ ngay lập tức đẩy (push) thông tin cập nhật xuống tất cả các thiết bị của thành viên khác trong cùng nhóm. Quá trình này diễn ra gần như tức thời, tạo cảm giác mọi người đang cùng thao tác trên một màn hình chung.
- **Thông báo đơn hàng mới:** Giúp ứng dụng Web của nhà hàng nhận thông báo đơn hàng mới (Real-time Notification) mà không cần tải lại trang (F5), giúp quy trình tiếp nhận đơn hàng diễn ra nhanh chóng.

Lý do lựa chọn SignalR so với các giải pháp khác:

- **So với Polling (Hỏi định kỳ):** Kỹ thuật Polling yêu cầu Client liên tục gửi request lên Server (ví dụ mỗi 5 giây) để kiểm tra dữ liệu mới. Điều này gây lãng phí băng thông mạng và tài nguyên máy chủ khủng khiếp khi số lượng người dùng tăng cao. SignalR sử dụng kết nối bền vững (Persistent Connection), chỉ gửi dữ liệu khi thực sự có thay đổi.
- **So với Firebase Realtime Database:** Mặc dù Firebase rất mạnh, việc phụ thuộc hoàn toàn vào dịch vụ bên thứ ba (BaaS) có thể dẫn đến chi phí vận

hành tăng vọt khi quy mô dữ liệu lớn. SignalR được tích hợp sẵn miễn phí trong ASP.NET Core, cho phép nhóm tự chủ hoàn toàn về hạ tầng và dữ liệu.

- **Cơ chế tự động điều phối (Auto-negotiation):** SignalR thông minh tự động lựa chọn phương thức kết nối tốt nhất mà Client và Server hỗ trợ (ưu tiên WebSockets, nếu không hỗ trợ sẽ tự chuyển sang Server-Sent Events hoặc Long Polling), đảm bảo ứng dụng hoạt động ổn định trên mọi môi trường mạng.

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

Bên cạnh kết nối thời gian thực bằng SignalR, hệ thống tích hợp dịch vụ **Firebase Cloud Messaging (FCM)** [12] để giải quyết bài toán gửi thông báo đến người dùng ngay cả khi ứng dụng đang chạy nền (Background) hoặc đã tắt hoàn toàn.

Vai trò của FCM trong hệ thống:

- **Cập nhật trạng thái đơn hàng:** Khách hàng sẽ nhận được thông báo ngay lập tức khi đơn hàng chuyển đổi trạng thái (ví dụ: Nhà hàng đã xác nhận, Tài xế đang giao). Việc này giúp người dùng an tâm mà không cần mở ứng dụng liên tục để kiểm tra.
- **Tương tác TeamCart:** Gửi lời mời tham gia nhóm hoặc thông báo khi "Chủ phòng" chốt đơn đến các thành viên khác.

Lý do kết hợp FCM cùng SignalR: Mặc dù SignalR rất mạnh trong việc đồng bộ dữ liệu trực tuyến (Online), nhưng kết nối WebSockets sẽ bị ngắt khi người dùng tắt màn hình điện thoại hoặc chuyển sang ứng dụng khác để tiết kiệm pin. FCM là giải pháp bổ trợ hoàn hảo nhờ sử dụng cơ chế của hệ điều hành (APNs trên iOS và GCM trên Android) để đánh thức thiết bị và hiển thị thông báo, đảm bảo thông tin quan trọng luôn đến được tay người dùng.

## 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, đội 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, nhóm phát triển quyết định sử dụng mô hình Monolithic Modular.

Mô hình này giúp cân bằng giữa tốc độ phát triển và khả năng mở rộng:

- **Phát triển nhanh và nhất quán:** Toàn bộ mã nguồn nằm trong một Solution thống nhất, giúp việc gỡ lỗi (debug), kiểm thử và tái cấu trúc (refactor) trở nên dễ dàng hơn.
- **Ranh giới nghiệp vụ rõ ràng (Modular Boundaries):** Các module được tổ chức độc lập về mặt logic dựa trên các 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.
- **Tập trung vào nghiệp vụ cốt lõi:** 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 ánh xạ từ lý thuyết vào thực tế như sau:

#### 1. Tầng Miền (Domain Layer) - *Project: YummyZoom.Domain*

Đây là lõi trung tâm của hệ thống, nơi chứa các quy tắc nghiệp vụ bất biến của doanh nghiệp. Tầng này hoàn toàn không phụ thuộc vào bất kỳ thư viện bên ngoài hay công nghệ cơ sở dữ liệu nào.

- **Thành phần:** Entities (Thực thể), Value Objects (Đối tượng giá trị), Aggregates (Hợp nhất), Domain Events.
- **Ví dụ thực tế:** Aggregate `TeamCart` đóng gói logic nghiệp vụ của giỏ hàng nhóm, đảm bảo các quy tắc như: một giỏ hàng chỉ có một chủ phòng (Host), và chỉ Host mới có quyền chốt đơn hoặc khóa giỏ hàng.

## 2. Tầng Ứng dụng (Application Layer) - *Project: YummyZoom.Application*

Đóng vai trò lớp vỏ bao quanh Domain, điều phối các yêu cầu từ người dùng và chuyển đổi chúng thành các thao tác nghiệp vụ.

- **Thành phần:** Use Cases (được triển khai dưới dạng Command/Query Handlers), Validators (Kiểm tra dữ liệu), Interfaces cho Infrastructure.
- **Ví dụ thực tế:** `AddItemToTeamCartCommand` là một Use Case nhận yêu cầu thêm món, kích hoạt phương thức `AddItem` trong Domain Entity, và sau đó lưu thay đổi thông qua `ITeamCartRepository`.

## 3. Tầng Hạ tầng (Infrastructure Layer) - *Project: YummyZoom.Infrastructure*

Nơi chứa các thực thi của các interfaces kỹ thuật được định nghĩa ở tầng Application. Đây là nơi các công nghệ cụ thể được "cắm" vào hệ thống.

- **Thành phần:** `DbContext` (Entity Framework Core), Repositories Implementation, Services giao tiếp bên ngoài (`EmailService`, `CloudinaryService`, `StripeService`, `FCMService`).
- **Chức năng:** Truy xuất dữ liệu từ SQL Server/PostgreSQL, tương tác với hệ thống file, gửi email xác nhận, xử lý thanh toán qua Stripe, gửi thông báo đẩy với FCM.

## 4. Tầng Giao diện/Trình bày (Web/Presentation Layer) - *Project: YummyZoom.Web*

Lớp vỏ ngoài cùng, chịu trách nhiệm giao tiếp với Client (Mobile App, Web Admin).

- **Thành phần:** Minimal APIs, SignalR Hubs, Middlewares.
- **Nhiệm vụ:** Nhận HTTP Request, xác thực token, gọi xuống Application Layer để xử lý và trả về HTTP Response chuẩn (JSON).

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

Áp dụng chiến lược của DDD, hệ thống được chia nhỏ thành các **Bounded Contexts** (Ngữ cảnh giới hạn), mỗi context giải quyết một vấn đề nghiệp vụ cụ thể và có Ubiquitous Language (Ngôn ngữ chung) riêng:

- **Identity Context:** Quản lý người dùng, phân quyền và xác thực (Authentic-

tion/Authorization).

- **Catalog Context:** Quản lý thực đơn, danh mục, thông tin nhà hàng, món ăn và các tùy chọn (Toppings).
- **TeamCart Context:** Ngữ cảnh quan trọng và phức tạp nhất, xử lý logic chia sẻ giỏ hàng, đồng bộ trạng thái thời gian thực và phân chia hóa đơn.
- **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:

**CQRS (Command Query Responsibility Segregation):** Hệ thống tách biệt rõ ràng luồng Đọc (Query) và Ghi (Command) dữ liệu:

- **Phần Ghi (Write Side):** Sử dụng **Entity Framework Core** kết hợp với Domain Model để đảm bảo tính toàn vẹn và nhất quán dữ liệu (Data Consistency) khi thực hiện các thay đổi nghiệp vụ.
- **Phần Đọc (Read Side):** Sử dụng **Dapper** với SQL để truy vấn dữ liệu trực tiếp và trả về các DTO phẳng. Đồng thời, hệ thống kết hợp sử dụng các **Read Models** được cập nhật song song với các bảng ghi, giúp tránh việc thực hiện các phép JOIN bảng quá mức. Cách tiếp cận này đảm bảo khả năng phục vụ hiệu quả các yêu cầu dữ liệu phức tạp từ ứng dụng người dùng với tốc độ phản hồi tối ưu.

**Domain Events & Outbox Pattern:** Giải quyết bài toán về tính nhất quán cuối cùng (Eventual Consistency) và xử lý bất đồng bộ:

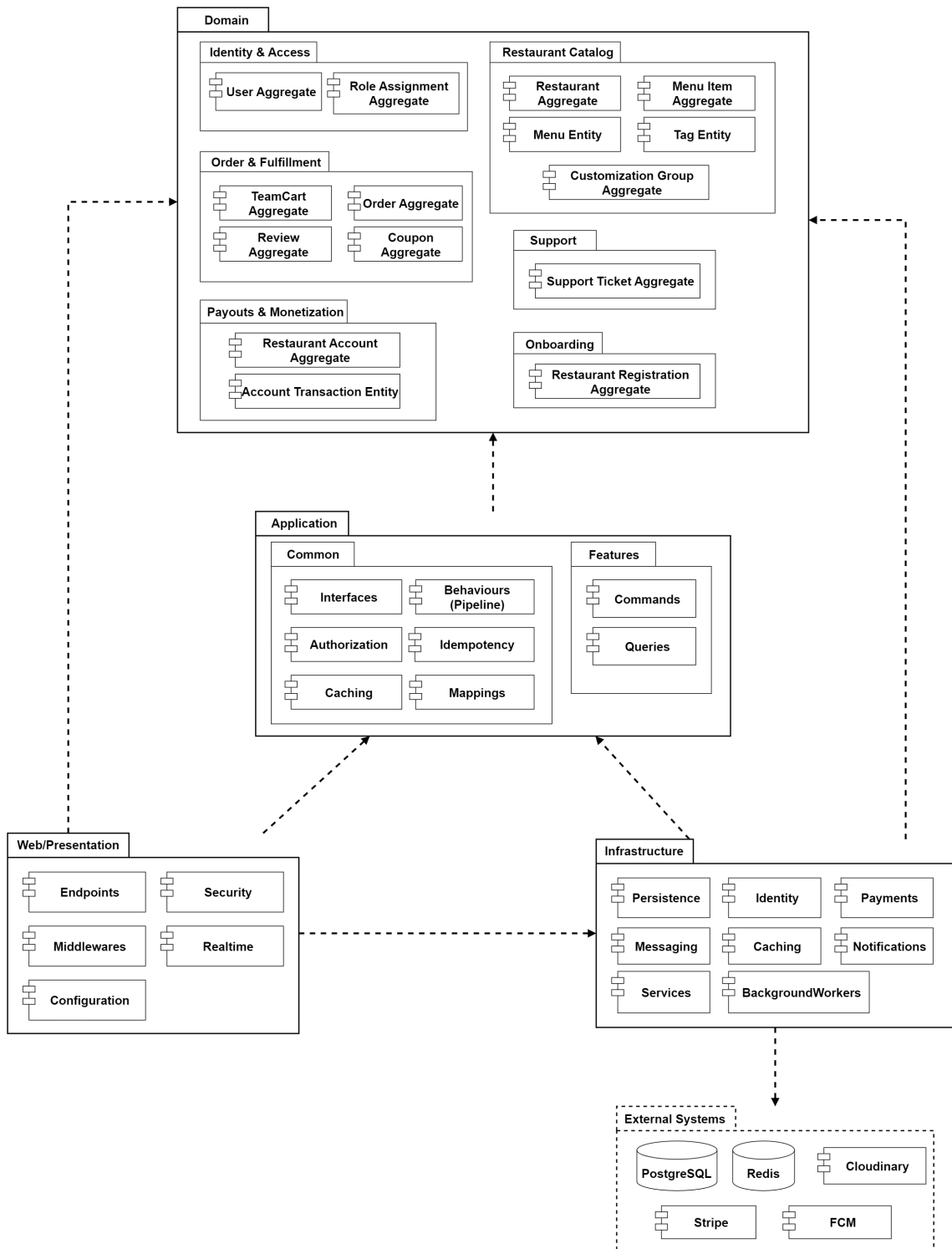
- **Cơ chế:** Khi một nghiệp vụ quan trọng hoàn tất (ví dụ: OrderCreated), hệ thống không gọi trực tiếp các dịch vụ bên thứ ba (như gửi email) mà sinh ra một Domain Event. Event này được lưu vào bảng OutboxMessages trong cùng một transaction database với dữ liệu chính.
- **Tác dụng:** Một tiến trình nền (Background Worker) sẽ đọc bảng Outbox và thực hiện các tác vụ phụ sau đó. Điều này đảm bảo rằng giao dịch chính luôn nhanh và an toàn; nếu việc gửi email thất bại, nó sẽ được thử lại (Retry) mà không làm mất đơn hàng.

### **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 chi tiết cách tổ chức mã nguồn và hiện thực hóa kiến trúc Clean Architecture thông qua các biểu đồ gói (Package Diagrams). Tôi đã lựa chọn 2 luồng nghiệp vụ tiêu biểu đại diện cho hai phân hệ quan trọng nhất: Restaurant (Nhà hàng) và Order (Đơn hàng) để phân tích.

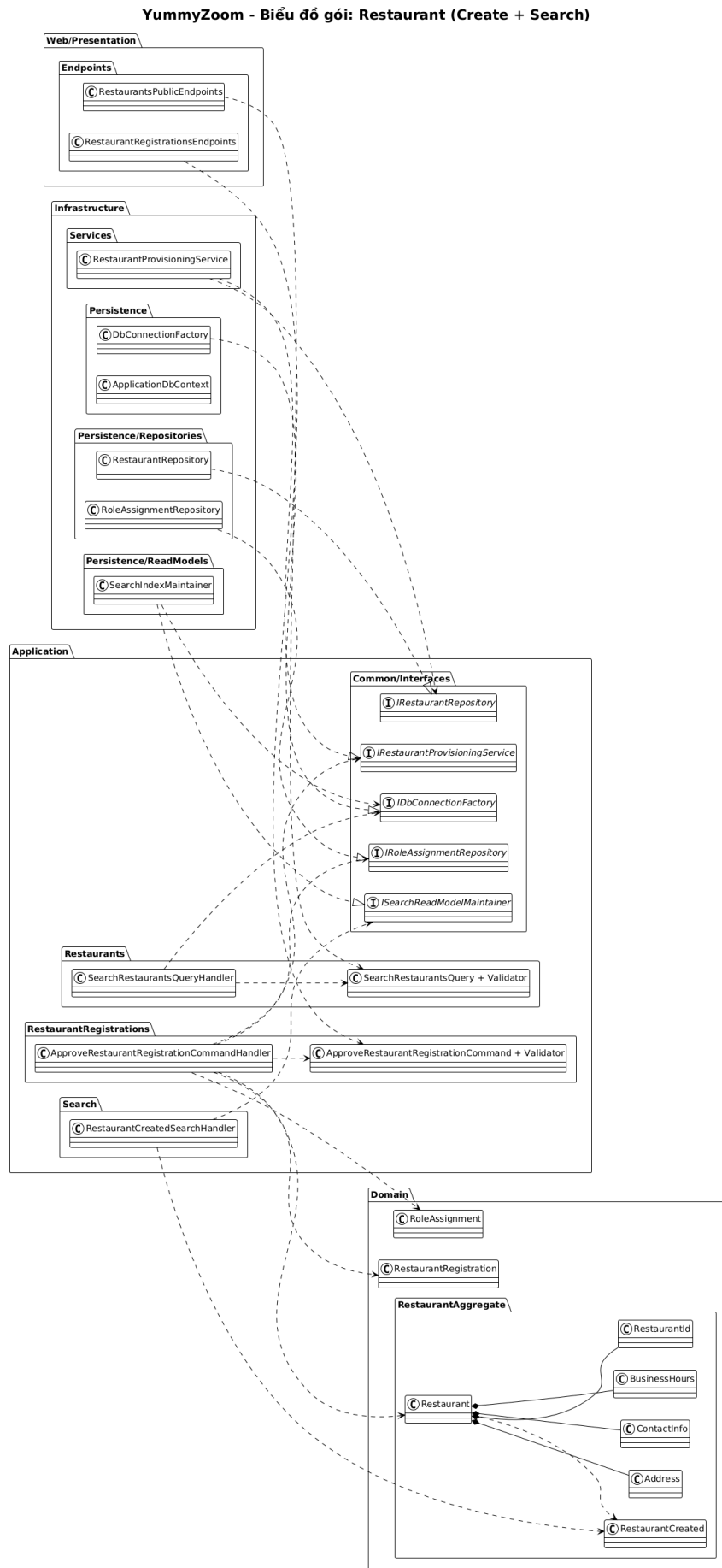
Các biểu đồ được xây dựng dựa trên các quy tắc sau nhằm đảm bảo tính trực

quan và tránh sự quá tải thông tin do số lượng lớp lớn của hệ thống:

- **Phạm vi:** Mỗi biểu đồ tập trung vào một luồng nghiệp vụ cụ thể (bao gồm 1 Command và 1 Query), không bao gồm toàn bộ các lớp trong mã nguồn.
- **Cấu trúc:** Các thành phần được vẽ và sắp xếp dọc theo 4 tầng của Clean Architecture (Web, Application, Domain, Infrastructure).
- **Chi tiết:** Chỉ thể hiện tên lớp và các mối quan hệ chính, lược bỏ các phương thức và thuộc tính chi tiết.

### **a, Biểu đồ 1: Restaurant (Create Restaurant + Search Restaurants)**

Biểu đồ này minh họa luồng xử lý cho hai chức năng: Phê duyệt đăng ký nhà hàng (Create) và Tìm kiếm nhà hàng (Search).



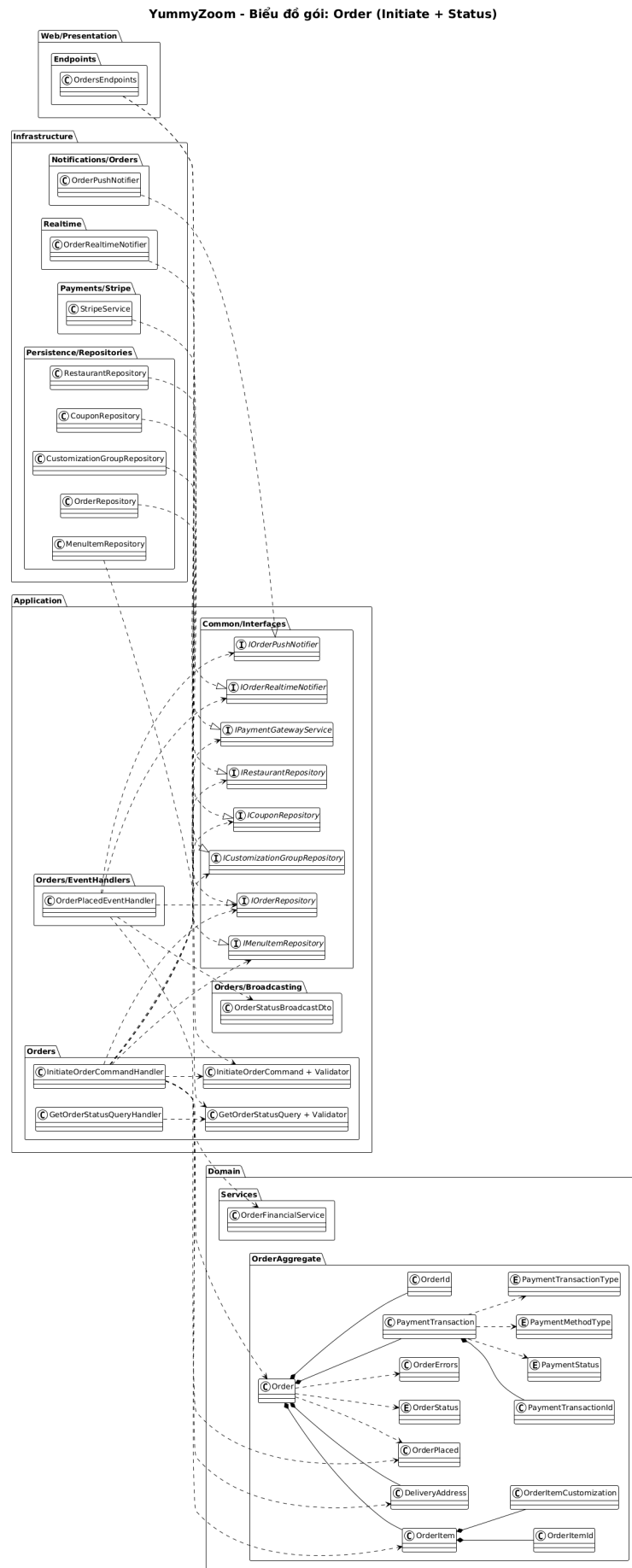
Hình 4.2: Biểu đồ gói chi tiết phân hệ Restaurant (Create & Search)

**Mô tả luồng logic:** Biểu đồ thể hiện cách các tầng phối hợp để xử lý yêu cầu:

- **Luồng Ghi (Command):** Bắt đầu từ yêu cầu POST đến Endpoints/RestaurantRegistrations để phê duyệt đăng ký nhà hàng, yêu cầu phê duyệt được chuyển đến ApproveRestaurantRegistrationCommandHandler (Application). Handler này tương tác với Domain (RestaurantAggregate, RestaurantRegistration) để đảm bảo quy tắc nghiệp vụ, rồi một đối tượng Restaurant được tạo và lưu trữ qua RestaurantRepository (Infrastructure) xuống database. Khi thành công, sự kiện RestaurantCreated được phát ra và được xử lý bởi RestaurantCreatedSearchHandler (Application) để cập nhật bảng SearchIndex trong database để phục vụ cho thao tác tìm kiếm sau này.
- **Luồng Đọc (Query):** Khi yêu cầu GET đến Restaurants để tìm kiếm nhà hàng theo các tiêu chí, SearchRestaurantsQueryHandler (Application) thực hiện truy vấn tối ưu bằng SQL vào bảng SearchIndex là một Read Model tối ưu hóa để phục vụ tìm kiếm. Dữ liệu phục vụ tìm kiếm được cập nhật bất đồng bộ bởi SearchIndexMaintainer (Infrastructure) ngay khi nhận được sự kiện RestaurantCreated, minh họa cho mô hình CQRS tách biệt Ghi và Đọc.

#### **b, Biểu đồ 2: Order (Initiate Order + Get Order Status)**

Biểu đồ này mô tả quy trình khởi tạo đơn hàng mới và truy vấn trạng thái đơn hàng, đóng vai trò cốt lõi trong giao dịch thương mại của hệ thống.



**Hình 4.3:** Biểu đồ gói chi tiết phân hệ Order (Initiate & Status)

**Mô tả luồng logic:** Biểu đồ minh họa sự phối hợp chặt chẽ giữa các thành phần để xử lý đơn hàng:

- **Luồng Ghi (Command):** Yêu cầu đặt hàng từ người dùng (Web Endpoints) được đóng gói thành `InitiateOrderCommand`. `InitiateOrderCommandHandler` đóng vai trò điều phối chính: nó sử dụng `OrderFinancialService` (một Domain Service) để tính toán chi phí, xác thực qua `IPaymentGatewayService`, và tương tác với các repositories (`IOrderRepository`, `ICouponRepository`) để lưu trữ Aggregate Order. Thành công kích hoạt sự kiện `OrderPlaced`, dẫn đến việc `OrderPlacedEventHandler` gọi các dịch vụ thông báo (`IOrderRealtimeNotifier`, `IOrderPushNotifier`).
- **Luồng Đọc (Query):** `GetOrderStatusQueryHandler` tiếp nhận yêu cầu truy vấn và trả về thông tin trạng thái đơn hàng, đảm bảo tính cập nhật cho phía người dùng.

## 4.2 Thiết kế chi tiết

### 4.2.1 Thiết kế giao diện

Phần này có độ dài từ hai đến ba trang. Sinh viên đặc tả thông tin về màn hình mà ứng dụng của mình hướng tới, bao gồm độ phân giải màn hình, kích thước màn hình, số lượng màu sắc hỗ trợ, v.v. Tiếp đến, sinh viên đưa ra các thống nhất/chuẩn hóa của mình khi thiết kế giao diện như thiết kế nút, điều khiển, vị trí hiển thị thông điệp phản hồi, phối màu, v.v. Sau cùng sinh viên đưa ra một số hình ảnh minh họa thiết kế giao diện cho các chức năng quan trọng nhất. Lưu ý, sinh viên không nhầm lẫn giao diện thiết kế với giao diện của sản phẩm sau cùng.

### 4.2.2 Thiết kế lớp

Phần này có độ dài từ ba đến bốn trang. Sinh viên trình bày thiết kế chi tiết các thuộc tính và phương thức cho một số lớp chủ đạo/quan trọng nhất của ứng dụng (từ 2-4 lớp). Thiết kế chi tiết cho các lớp khác, nếu muốn trình bày, sinh viên đưa vào phần phụ lục.

Để minh họa thiết kế lớp, sinh viên thiết kế luồng truyền thông điệp giữa các đối tượng tham gia cho 2 đến 3 use case quan trọng nào đó bằng biểu đồ trình tự (hoặc biểu đồ giao tiếp).

### 4.2.3 Thiết kế cơ sở dữ liệu

Phần này có độ dài từ hai đến bốn trang. Sinh viên thiết kế, vẽ và giải thích biểu đồ thực thể liên kết (E-R diagram). Từ đó, sinh viên thiết kế cơ sở dữ liệu tùy theo hệ quản trị cơ sở dữ liệu mà mình sử dụng (SQL, NoSQL, Firebase, v.v.)

### 4.3 Xây dựng ứng dụng

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

Sinh viên liệt kê các công cụ, ngôn ngữ lập trình, API, thư viện, IDE, công cụ kiểm thử, v.v. mà mình sử dụng để phát triển ứng dụng. Mỗi công cụ phải được chỉ rõ phiên bản sử dụng. SV nên kẻ bảng mô tả tương tự như Bảng 4.1. Nếu có nhiều nội dung trình bày, sinh viên cần xoay ngang bảng.

Mục đích	Công cụ	Địa chỉ URL
IDE lập trình	Eclipse Oxygen a64 bit	<a href="http://www.eclipse.org/">http://www.eclipse.org/</a>
v.v.	v.v.	v.v.

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

#### 4.3.2 Kết quả đạt được

Sinh viên trước tiên mô tả kết quả đạt được của mình là gì, ví dụ như các sản phẩm được đóng gói là gì, bao gồm những thành phần nào, ý nghĩa, vai trò?

Sinh viên cần thống kê các thông tin về ứng dụng của mình như: số dòng code, số lớp, số gói, dung lượng toàn bộ mã nguồn, dung lượng của từng sản phẩm đóng gói, v.v. Tương tự như phần liệt kê về công cụ sử dụng, sinh viên cũng nên dùng bảng để mô tả phần thông tin thống kê này.

#### 4.3.3 Minh họa các chức năng chính

Sinh viên lựa chọn và đưa ra màn hình cho các chức năng chính, quan trọng, và thú vị nhất. Mỗi giao diện cần phải có lời giải thích ngắn gọn. Khi giải thích, sinh viên có thể kết hợp với các chú thích ở trong hình ảnh giao diện.

### 4.4 Kiểm thử

Phần này có độ dài từ hai đến ba trang. Sinh viên thiết kế các trường hợp kiểm thử cho hai đến ba chức năng quan trọng nhất. Sinh viên cần chỉ rõ các kỹ thuật kiểm thử đã sử dụng. Chi tiết các trường hợp kiểm thử khác, nếu muốn trình bày, sinh viên đưa vào phần phụ lục. Sinh viên sau cùng tổng kết về số lượng các trường hợp kiểm thử và kết quả kiểm thử. Sinh viên cần phân tích lý do nếu kết quả kiểm thử không đạt.

### 4.5 Triển khai

Sinh viên trình bày mô hình và/hoặc cách thức triển khai thử nghiệm/thực tế. Ứng dụng của sinh viên được triển khai trên server/thiết bị gì, cấu hình như thế nào. Kết quả triển khai thử nghiệm nếu có (số lượng người dùng, số lượng truy cập, thời gian phản hồi, phản hồi người dùng, khả năng chịu tải, các thống kê, v.v.)

## CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

Chương này có độ dài tối thiểu 5 trang, tối đa không giới hạn.<sup>1</sup> Sinh viên cần trình bày tất cả những nội dung đóng góp mà mình thấy tâm đắc nhất trong suốt quá trình làm ĐATN. Đó có thể là một loạt các vấn đề khó khăn mà sinh viên đã từng bước giải quyết được, là giải thuật cho một bài toán cụ thể, là giải pháp tổng quát cho một lớp bài toán, hoặc là mô hình/kiến trúc hữu hiệu nào đó được sinh viên thiết kế.

Chương này **là cơ sở quan trọng** để các thầy cô đánh giá sinh viên. Vì vậy, sinh viên cần phát huy tính sáng tạo, khả năng phân tích, phản biện, lập luận, tổng quát hóa vấn đề và tập trung viết cho thật tốt. Mỗi giải pháp hoặc đóng góp của sinh viên cần được trình bày trong một mục độc lập bao gồm ba mục con: (i) dẫn dắt/giới thiệu về bài toán/vấn đề, (ii) giải pháp, và (iii) kết quả đạt được (nếu có).

Sinh viên lưu ý **không trình bày lặp lại nội dung**. Những nội dung đã trình bày chi tiết trong các chương trước không được trình bày lại trong chương này. Vì vậy, với nội dung hay, mang tính đóng góp/giải pháp, sinh viên chỉ nên tóm lược/mô tả sơ bộ trong các chương trước, đồng thời tạo tham chiếu chéo tới đề mục tương ứng trong Chương 5 này. Chi tiết thông tin về đóng góp/giải pháp được trình bày trong mục đó.

Ví dụ, trong Chương 4, sinh viên có thiết kế được kiến trúc đáng lưu ý gì đó, là sự kết hợp của các kiến trúc MVC, MVP, SOA, v.v. Khi đó, sinh viên sẽ chỉ mô tả ngắn gọn kiến trúc đó ở Chương 4, rồi thêm các câu có dạng: “Chi tiết về kiến trúc này sẽ được trình bày trong phần 5.1”.

---

<sup>1</sup>Trong trường hợp phần này dưới 5 trang thì sinh viên nên gộp vào phần kết luận, không tách ra một chương riêng rẽ nữa.

## **CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

### **6.1 Kết luận**

Sinh viên so sánh kết quả nghiên cứu hoặc sản phẩm của mình với các nghiên cứu hoặc sản phẩm tương tự.

Sinh viên phân tích trong suốt quá trình thực hiện ĐATN, mình đã làm được gì, chưa làm được gì, các đóng góp nổi bật là gì, và tổng hợp những bài học kinh nghiệm rút ra nếu có.

### **6.2 Hướng phát triển**

Trong phần này, sinh viên trình bày định hướng công việc trong tương lai để hoàn thiện sản phẩm hoặc nghiên cứu của mình.

Trước tiên, sinh viên trình bày các công việc cần thiết để hoàn thiện các chức năng/nhiệm vụ đã làm. Sau đó sinh viên phân tích các hướng đi mới cho phép cải thiện và nâng cấp các chức năng/nhiệm vụ đã làm.

## MỘT SỐ LƯU Ý VỀ TÀI LIỆU THAM KHẢO

Lưu ý: Sinh viên không được đưa bài giảng/slide, các trang Wikipedia, hoặc các trang web thông thường làm tài liệu tham khảo.

Một trang web được phép dùng làm tài liệu tham khảo **chỉ khi** nó là công bố chính thống của cá nhân hoặc tổ chức nào đó. Ví dụ, trang web đặc tả ngôn ngữ XML của tổ chức W3C <https://www.w3.org/TR/2008/REC-xml-20081126/> là TLTK hợp lệ.

Có năm loại tài liệu tham khảo mà sinh viên phải tuân thủ đúng quy định về cách thức liệt kê thông tin như sau. Lưu ý: các phần văn bản trong cặp dấu < > dưới đây chỉ là hướng dẫn khai báo cho từng loại tài liệu tham khảo; sinh viên cần xóa các phần văn bản này trong DATN của mình.

<**Bài báo đăng trên tạp chí khoa học:** Tên tác giả, tên bài báo, tên tạp chí, volume, từ trang đến trang (nếu có), nhà xuất bản, năm xuất bản >

[13] E. H. Hovy, "Automated discourse generation using discourse structure relations," *Artificial intelligence*, vol. 63, no. 1-2, pp. 341–385, 1993

<**Sách:** Tên tác giả, tên sách, volume (nếu có), lần tái bản (nếu có), nhà xuất bản, năm xuất bản>

[14] L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.

[15] N. T. Hải, *Mạng máy tính và các hệ thống mở*. Nhà xuất bản giáo dục, 1999.

<**Tập san Báo cáo Hội nghị Khoa học:** Tên tác giả, tên báo cáo, tên hội nghị, ngày (nếu có), địa điểm hội nghị, năm xuất bản>

[16] M. Poesio and B. Di Eugenio, "Discourse structure and anaphoric accessibility," in *ESSLLI workshop on information structure, discourse structure and discourse semantics*, Copenhagen, Denmark, 2001, pp. 129–143.

<**Đồ án tốt nghiệp, Luận văn Thạc sĩ, Tiến sĩ:** Tên tác giả, tên đồ án/luận văn, loại đồ án/luận văn, tên trường, địa điểm, năm xuất bản>

[17] A. Knott, "A data-driven methodology for motivating a set of coherence relations," Ph.D. dissertation, The University of Edinburgh, UK, 1996.

<**Tài liệu tham khảo từ Internet:** Tên tác giả (nếu có), tựa đề, cơ quan (nếu có), địa chỉ trang web, thời gian lần cuối truy cập trang web>

[18] T. Berners-Lee, *Hypertext transfer protocol (HTTP)*. [Online]. Available:

`ftp://info.cern.ch/pub/www/doc/http-spec.txt.Z` (visited on 09/30/2010).

[19] Princeton University, *Wordnet*. [Online]. Available: `http://www.cogsci.princeton.edu/~wn/index.shtml` (visited on 09/30/2010).

## TÀI LIỆU THAM KHẢO

- [1] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Upper Saddle River, NJ: Prentice Hall, 2017.
- [2] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston, MA: Addison-Wesley, 2003.
- [3] 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>
- [4] Microsoft, *.net aspire documentation*, Microsoft Learn, May 2024. [Online]. Available: <https://learn.microsoft.com/en-us/dotnet/aspire/>
- [5] PostgreSQL Global Development Group, *Postgresql 16.0 documentation*, 2023. [Online]. Available: <https://www.postgresql.org/docs/16/index.html>
- [6] Cloudinary, *Cloudinary documentation*, Cloudinary.com, 2024. [Online]. Available: <https://cloudinary.com/documentation>
- [7] 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>
- [8] Google, *Flutter architectural overview*, Flutter.dev. [Online]. Available: <https://docs.flutter.dev/resources/architectural-overview>
- [9] Google, *Angular documentation: Architecture overview*, Angular.io. [Online]. Available: <https://angular.io/guide/architecture>
- [10] PrimeTek, *Primeng documentation*, PrimeFaces.org. [Online]. Available: <https://primeng.org/installation>
- [11] Microsoft, *Introduction to asp.net core signalr*, Microsoft Learn, 2023. [Online]. Available: <https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction>
- [12] Google, *Firebase cloud messaging*, Firebase.google.com, 2024. [Online]. Available: <https://firebase.google.com/docs/cloud-messaging>
- [13] E. H. Hovy, "Automated discourse generation using discourse structure relations," *Artificial intelligence*, vol. 63, no. 1-2, pp. 341–385, 1993.
- [14] L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.
- [15] N. T. Hải, *Mạng máy tính và các hệ thống mở*. Nhà xuất bản giáo dục, 1999.

- [16] M. Poesio and B. Di Eugenio, “Discourse structure and anaphoric accessibility,” in *ESSLLI workshop on information structure, discourse structure and discourse semantics, Copenhagen, Denmark*, 2001, pp. 129–143.
- [17] A. Knott, “A data-driven methodology for motivating a set of coherence relations,” Ph.D. dissertation, The University of Edinburgh, UK, 1996.
- [18] T. Berners-Lee, *Hypertext transfer protocol (HTTP)*. Accessed: Sep. 30, 2010. [Online]. Available: <ftp://info.cern.ch/pub/www/doc/http-spec.txt.Z>
- [19] Princeton University, *Wordnet*. Accessed: Sep. 30, 2010. [Online]. Available: <http://www.cogsci.princeton.edu/~wn/index.shtml>

## **PHỤ LỤC**

## A. HƯỚNG DẪN VIẾT ĐỒ ÁN TỐT NGHIỆP

### Quy định chung

Dưới đây là một số quy định và hướng dẫn viết đồ án tốt nghiệp mà bắt buộc sinh viên phải đọc kỹ và tuân thủ nghiêm ngặt.

Sinh viên cần đảm bảo tính thống nhất toàn báo cáo (font chữ, căn dòng hai bên, hình ảnh, bảng, margin trang, đánh số trang, v.v.). Để làm được như vậy, sinh viên chỉ cần sử dụng các định dạng theo đúng template DATN này. Khi paste nội dung văn bản từ tài liệu khác của mình, sinh viên cần chọn kiểu Copy là “Text Only” để định dạng văn bản của template không bị phá vỡ/vi phạm.

Tuyệt đối cấm sinh viên đạo văn. Sinh viên cần ghi rõ nguồn cho tất cả những gì không tự mình viết/vẽ lên, bao gồm các câu trích dẫn, các hình ảnh, bảng biểu, v.v. Khi bị phát hiện, sinh viên sẽ không được phép bảo vệ DATN.

Tất cả các hình vẽ, bảng biểu, công thức, và tài liệu tham khảo trong DATN nhất thiết phải được SV giải thích và tham chiếu tới ít nhất một lần. Không chấp nhận các trường hợp sinh viên đưa ra hình ảnh, bảng biểu tùy hứng và không có lời mô tả/giải thích nào.

Sinh viên tuyệt đối không trình bày DATN theo kiểu viết ý hoặc gạch đầu dòng. DATN không phải là một slide thuyết trình; khi người đọc không hiểu sẽ không có ai giải thích hộ. Sinh viên cần viết thành các đoạn văn và phân tích, diễn giải đầy đủ, rõ ràng. Câu văn cần đúng ngữ pháp, đầy đủ chủ ngữ, vị ngữ và các thành phần câu. Khi thực sự cần liệt kê, sinh viên nên liệt kê theo phong cách khoa học với các ký tự La Mã. Ví dụ, nhiều sinh viên luôn cảm thấy hối hận vì (i) chưa cố gắng hết mình, (ii) chưa sắp xếp thời gian học/chơi một cách hợp lý, (iii) chưa tìm được người yêu để chia sẻ quãng đời sinh viên vất vả, và (iv) viết DATN một cách cẩu thả.

Trong một số trường hợp nhất thiết phải dùng các bullet để liệt kê, sinh viên cần thống nhất Style cho toàn bộ các bullet các cấp mà mình sử dụng đến trong báo cáo. Nếu dùng bullet cấp 1 là hình tròn đen, toàn bộ báo cáo cần thống nhất cách dùng như vậy; ví dụ như sau:

- Đây là mục 1 – Thực sự không còn cách nào khác tôi mới dùng đến việc bullet trong báo cáo.
- Đây là mục 2 – Nghĩ lại thì tôi có thể không cần dùng bullet cũng được. Nên tôi sẽ xóa bullet và tổ chức lại hai mục này trong báo cáo của mình cho khoa học hơn. Tôi muốn thầy cô và người đọc cảm nhận được tâm huyết của tôi

trong từng trang báo cáo ĐATN.

### A.1 Ngành học

Sinh viên lưu ý viết đúng ngành/chuyên ngành trên bìa và trên gáy theo đúng quy định của Trường. Ngành học hay chuyên ngành học phụ thuộc vào ngành học mà sinh viên đăng ký. Sinh viên có thể đăng nhập trên trang quản lý học tập của mình để xem lại chính xác ngành học của mình.

Một số ví dụ sinh viên có thể tham khảo dưới đây, trong trường hợp có chuyên ngành thì sinh viên không cần ghi chuyên ngành:

- Đối với kỹ sư chính quy:
  - Từ K61 trở về trước: Ngành Kỹ thuật phần mềm
  - Từ K62 trở về sau: Ngành Khoa học máy tính
- Đối với cử nhân:
  - Ngành Công nghệ thông tin
- Đối với chương trình EliteTech:
  - Chương trình Việt Nhật/KSTN: Ngành Công nghệ thông tin
  - Chương trình ICT Global: Ngành Information Technology
  - Chương trình DS&AI: Ngành Khoa học dữ liệu

### A.2 Đánh dấu (bullet) và đánh số (numering)

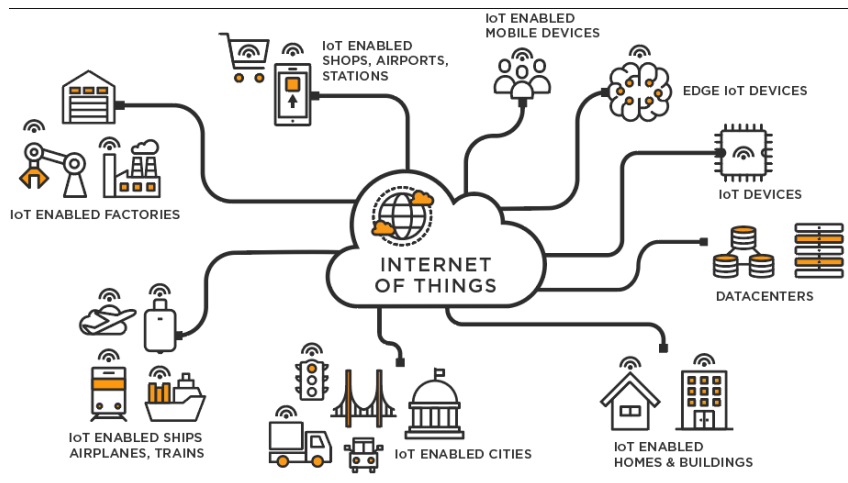
Việc sử dụng danh sách trong LaTeX khá đơn giản và không yêu cầu sinh viên phải thêm bất kỳ gói bổ sung nào. LaTeX cung cấp hai môi trường liệt kê đó là:

- Đánh dấu (bullet) là kiểu liệt kê không có thứ tự. Để sử dụng kiểu liệt kê đánh dấu, chúng ta khai báo như sau

```
\begin{itemize}
\item Nội dung thứ nhất được viết ở đây.
\item Nội dung thứ hai được viết ở đây.
\item ...
\end{itemize}
```

- Đánh số (numering) là kiểu liệt kê có thứ tự. Để sử dụng kiểu liệt kê đánh số, chúng ta khai báo như sau

```
\begin{enumerate}
\item Nội dung thứ nhất được viết ở đây.
\item Nội dung thứ hai được viết ở đây.
\item ...
```



Hình A.1: Internet vạn vật

\end{enumerate}

Chú ý các nội dung trình bày trong cả hai môi trường liệt kê theo sau lệnh `\item`. Ngoài ra LaTeX còn cung cấp một số kiểu liệt kê khác, sinh viên có thể tham khảo tại <https://www.overleaf.com/learn/latex/Lists>

### A.3 Cách thêm bảng

Col1	Col2	Col2	Col3
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

Bảng A.1: Table to test captions and labels.

Bảng A.1 là ví dụ về cách tạo bảng. Tất cả các bảng biểu phải được đề cập đến trong phần nội dung và phải được phân tích và bình luận. Chú ý: Tạo bảng trong LaTeX khá phức tạp và mất thời gian, vì vậy sinh viên có thể sử dụng các công cụ hỗ trợ tạo bảng (Ví dụ: <https://www.tablesgenerator.com/>). Sinh viên có thể tìm hiểu sâu hơn về cách chèn ảnh trong LaTeX tại link <https://www.overleaf.com/learn/latex/Tables>.

### A.4 Chèn hình ảnh

Hình A.1 là ví dụ về cách chèn ảnh. Lưu ý chú thích của hình vẽ được đặt ngay dưới hình vẽ. Sinh viên có thể tìm hiểu sâu hơn về cách chèn ảnh trong LaTeX tại [https://www.overleaf.com/learn/latex/Inserting\\_Images](https://www.overleaf.com/learn/latex/Inserting_Images).

Chú ý, tất cả các hình vẽ phải được đề cập đến trong phần nội dung và phải được phân tích và bình luận.

## A.5 Tài liệu tham khảo

### Cách liệt kê

Áp dụng cách liệt kê theo quy định của IEEE. Ví dụ của việc trích dẫn như sau **scott2013sdn**. Cụ thể, sinh viên sử dụng lệnh `\cite{}` như sau **ashton2009internet**. Chỉ những tài liệu được trích dẫn thì mới xuất hiện trong phần Tài liệu tham khảo. Tài liệu tham khảo cần có nguồn gốc rõ ràng và phải từ nguồn đáng tin cậy. Hạn chế trích dẫn tài liệu tham khảo từ các website, từ wikipedia.

### Các loại tài liệu tham khảo

Các nguồn tài liệu tham khảo chính là sách, bài báo trong các tạp chí, bài báo trong các hội nghị khoa học và các tài liệu tham khảo khác trên internet.

## A.6 Cách viết phương trình và công thức toán học

Các gói `amsmath`, `amssymb`, `amsfonts` hỗ trợ viết phương trình/công thức toán học đã được bổ sung sẵn ở phần đầu của file `main.tex`. Một ví dụ về tạo phương trình (A.1) như sau

$$F(x) = \int_b^a \frac{1}{3}x^3 \quad (\text{A.1})$$

Phương trình A.1 là ví dụ về phương trình tích phân. Một phương trình khác không được đánh số thứ tự (gán nhãn)

$$x[t_n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X[f_k] e^{j2\pi nk/N}$$

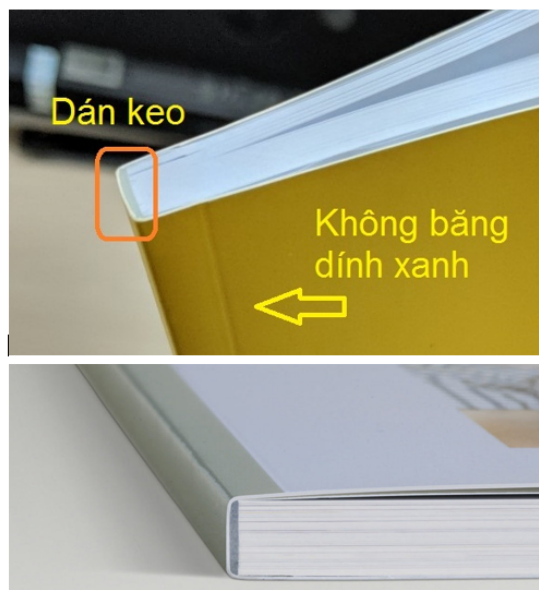
Phương trình này thể hiện phép biến đổi Fourier rời rạc ngược (IDFT).

## A.7 Qui cách đóng quyển

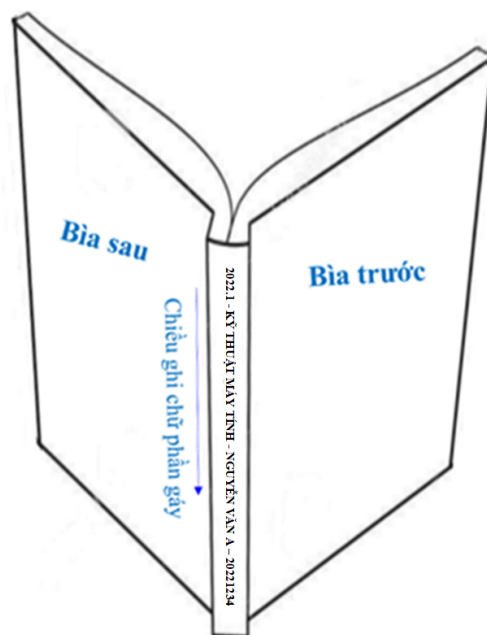
Phần bìa trước chế bản theo qui định; bìa trước và bìa sau là giấy liền khổ. Sử dụng keo nhiệt để dán gáy khi đóng quyển thay vì sử dụng băng dính và dập ghim như mô tả ở Hình A.2 Phần gáy ĐATN cần ghi các thông tin tóm tắt sau: Kỳ làm ĐATN - Ngành đào tạo - Họ và tên sinh viên - Mã số sinh viên. Ví dụ:

2022.1 - KỸ THUẬT MÁY TÍNH - NGUYỄN VĂN A - 20221234

Qui cách ghi chữ phần gáy như hình dưới đây:



**Hình A.2:** Quy cách đóng quyển đồ án



**Hình A.3:** Quy cách đóng quyển đồ án

## **B. ĐẶC TẢ USE CASE**

Nếu trong nội dung chính không đủ không gian cho các use case khác (ngoài các use case nghiệp vụ chính) thì đặc tả thêm cho các use case đó ở đây.

### **B.1 Đặc tả use case “Thống kê tình hình mượn sách”**

...

### **B.2 Đặc tả use case “Đăng ký làm thẻ mượn”**

...