

0.1 Thiết kế kiến trúc

0.1.1 Lựa chọn kiến trúc phần mềm

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

a, Mô hình kiến trúc tổng thể

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

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

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

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

Bao bọc bên ngoài lớp miền là Lớp Ứng dụng (Application Layer). Lớp này đóng vai trò điều phối các ca sử dụng (Use Cases) của hệ thống. Nhiệm vụ chính của nó là nhận yêu cầu từ các lớp bên ngoài, điều phối các đối tượng trong lớp miền thực hiện các tác vụ nghiệp vụ và trả về kết quả. Lớp ứng dụng định nghĩa các giao

diện (Interfaces) cho các dịch vụ hạ tầng nhưng không trực tiếp thực thi chúng.

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

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

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

c, Thiết kế chiến lược (Strategic Design)

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

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

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

Mẫu CQRS giúp hệ thống tách biệt rõ ràng luồng Đọc (Query) và Ghi (Command) dữ liệu. Phần Ghi sử dụng Entity Framework Core kết hợp với Domain

Model để đảm bảo tính toàn vẹn dữ liệu khi thực hiện các thay đổi nghiệp vụ. Trong khi đó, Phần Đọc sử dụng Dapper với SQL để truy vấn dữ liệu trực tiếp và trả về các DTO phẳng, tối ưu hóa tốc độ phản hồi cho người dùng.

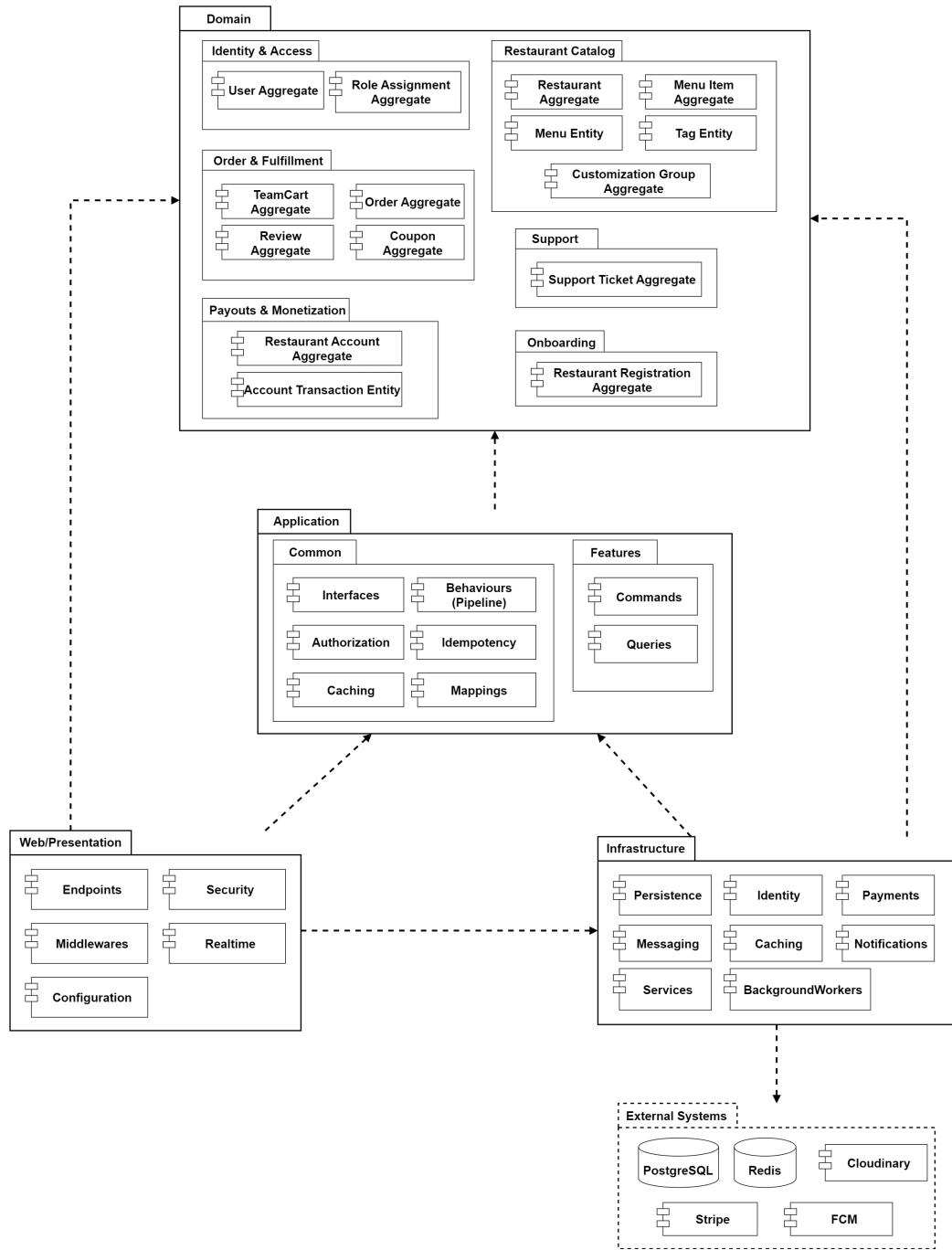
Bên cạnh đó, mẫu Domain Events kết hợp với Outbox Pattern được sử dụng để giải quyết bài toán về tính nhất quán cuối cùng (Eventual Consistency). Khi một nghiệp vụ quan trọng hoàn tất, hệ thống sinh ra một sự kiện miền và lưu vào bảng Outbox trong cùng một giao dịch cơ sở dữ liệu. Một tiến trình nền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

e, Kết luận

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

0.1.2 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.



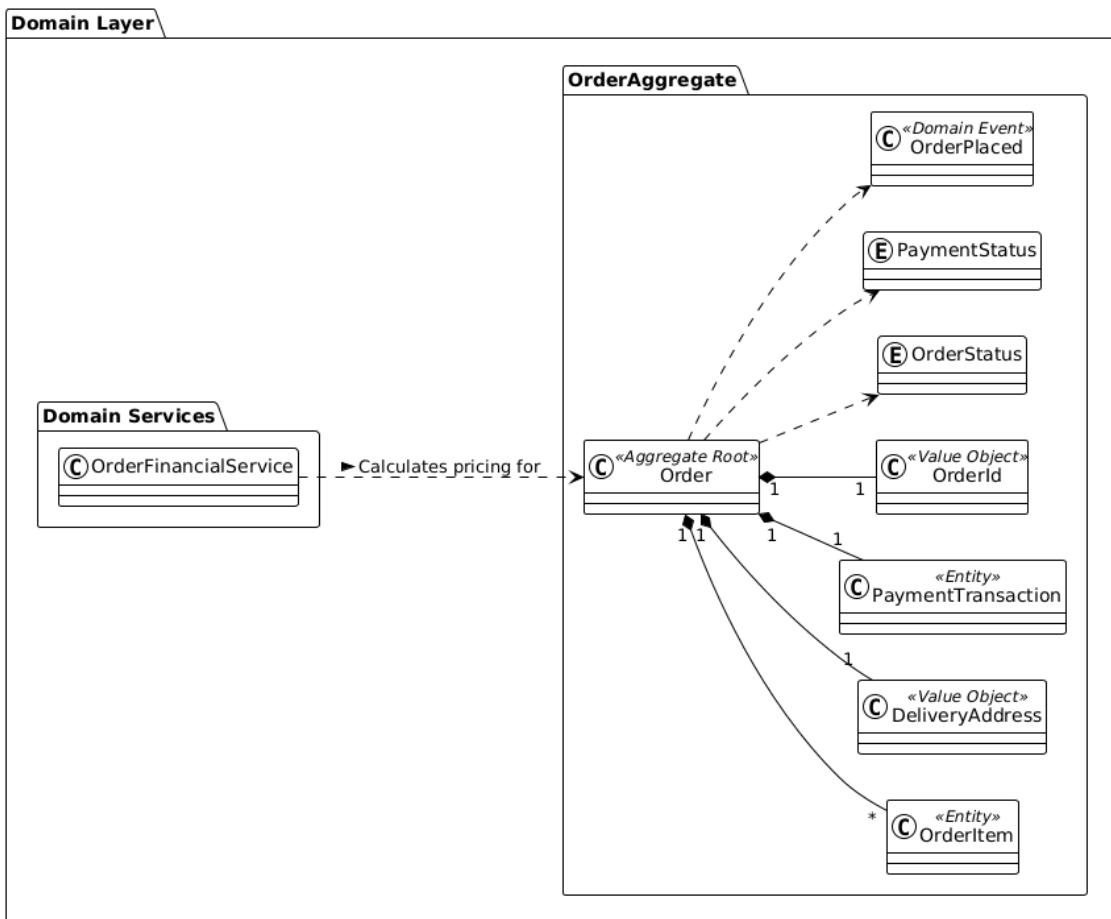
Hình 0.1: Biểu đồ phụ thuộc gói tổng quan YummyZoom (Backend)

0.1.3 Thiết kế chi tiết gói

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

a, Thiết kế lớp miền (Domain Layer)

Biểu đồ tại Hình 0.2 minh họa các thành phần cốt lõi trong lớp miền phục vụ cho nghiệp vụ khởi tạo đơn hàng. Tại đây, lớp Order đóng vai trò là gốc tập hợp (Aggregate Root), trung tâm quản lý mọi thay đổi và đảm bảo tính nhất quán của dữ liệu.

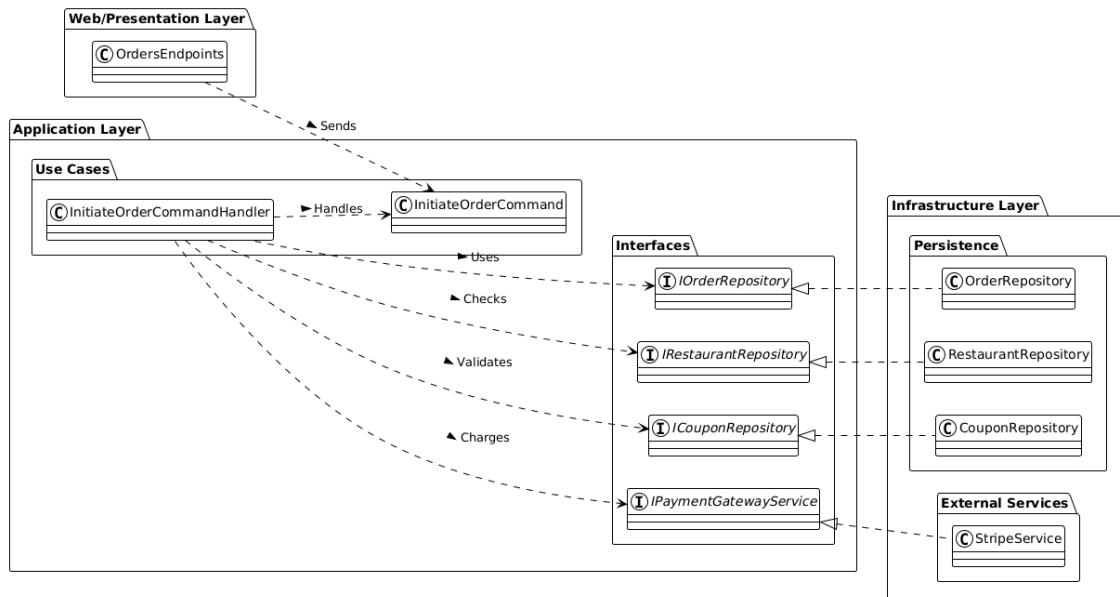


Hình 0.2: Thiết kế gói lớp miền cho luồng Khởi tạo Đơn hàng

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

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

Sự tương tác giữa các tầng còn lại để hiện thực hóa luồng nghiệp vụ được trình bày chi tiết tại Hình 0.3. Biểu đồ này thể hiện rõ vai trò điều phối của lớp ứng dụng và sự tách biệt về công nghệ của lớp hạ tầng.



Hình 0.3: Thiết kế gói lớp ứng dụng và hạ tầng cho luồng Khởi tạo Đơn hàng

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

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

c, Luồng thực thi tổng thể

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

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

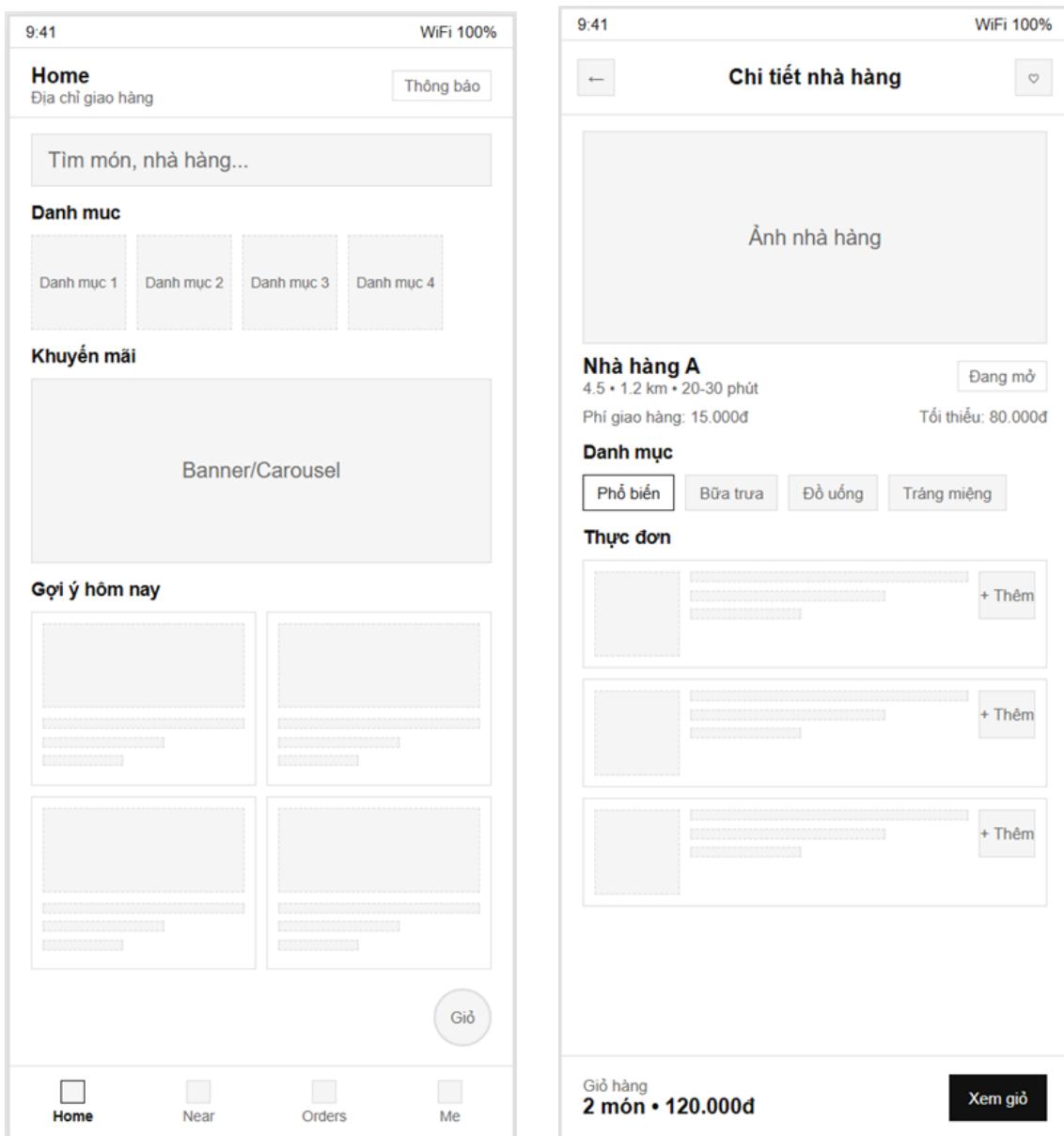
0.2 Thiết kế chi tiết

0.2.1 Thiết kế giao diện

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

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

Các màn hình tiêu biểu được lựa chọn nhằm phản ánh đầy đủ hành trình người dùng từ khám phá đến thanh toán, bao gồm trang Home/Menu và Restaurant Detail (Hình 0.4).

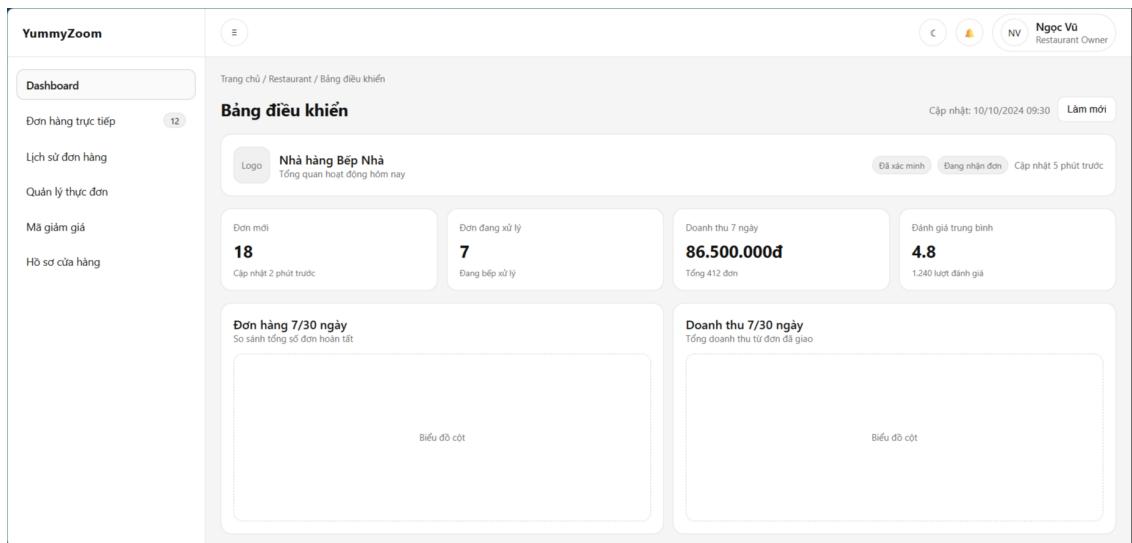


Hình 0.4: Minh họa màn hình Home/Menu (trái) và Restaurant Detail (phải) trên ứng dụng di động

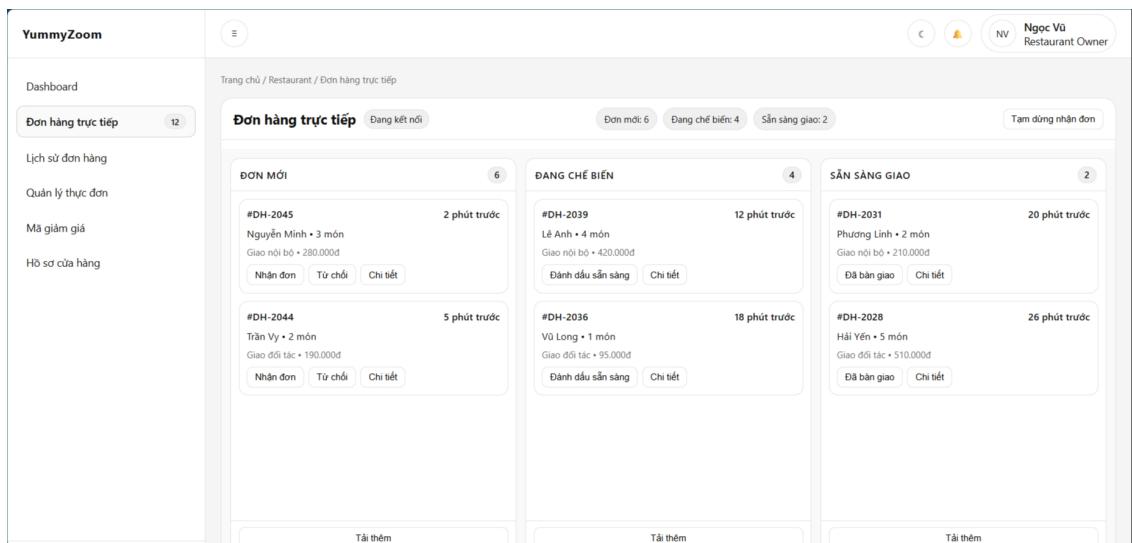
b, Thiết kế giao diện hệ thống quản trị trên web

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

Các màn hình minh họa tập trung vào những chức năng cốt lõi của quản trị, gồm trang Restaurant Dashboard (Hình 0.5) và trang Live Orders (Hình 0.6).



Hình 0.5: Minh họa màn hình Restaurant Dashboard trên hệ thống quản trị



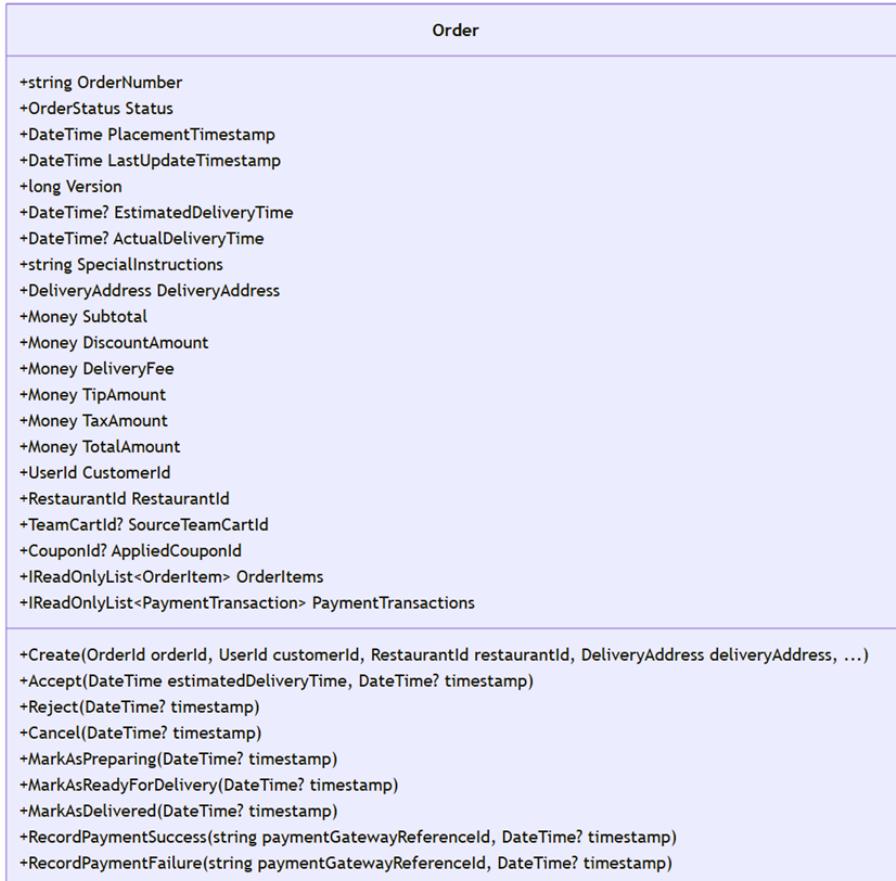
Hình 0.6: Minh họa màn hình Live Orders trên hệ thống quản trị

0.2.2 Thiết kế lớp

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

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

TaxAmount, TotalAmount) cùng các phương thức chuyển trạng thái như Accept, Reject, MarkAsPreparing, MarkAsReadyForDelivery và MarkAsDelivered, đồng thời phát sinh các sự kiện miền tương ứng phục vụ đồng bộ trạng thái theo thời gian thực. Hình 0.7 mô tả cấu trúc lớp Order và các mối quan hệ chính.



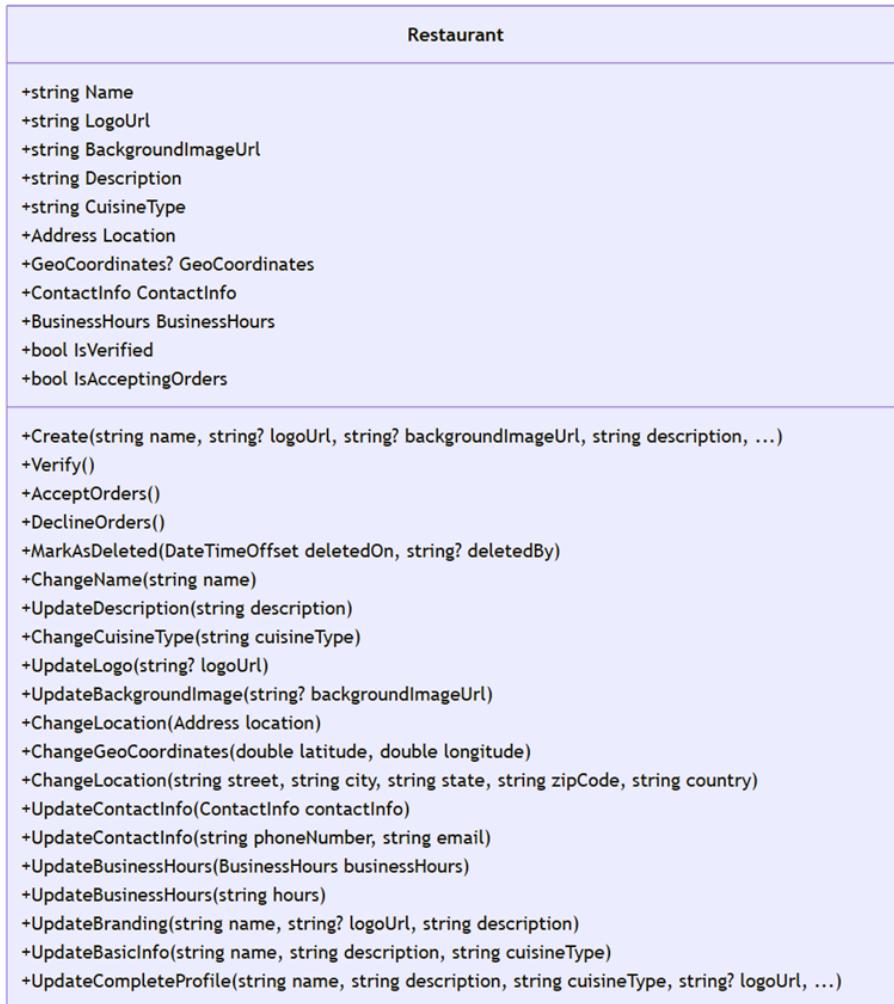
Hình 0.7: Biểu đồ lớp của Order (Aggregate root)

Lớp TeamCart mô hình hóa giỏ hàng nhóm, cho phép nhiều người cùng đặt món và phối hợp thanh toán. Các thuộc tính cốt lõi gồm thông tin thành viên, danh sách món, trạng thái giỏ và các cấu phần tài chính như TipAmount và QuoteVersion. Các phương thức như AddItem, LockForPayment, FinalizePricing và RecordSuccessfulOnlinePayment giúp đảm bảo tính nhất quán của luồng thanh toán nhóm. Hình 0.8 trình bày thiết kế lớp TeamCart.

TeamCart
<pre>+TeamCartId Id +RestaurantId RestaurantId +UserId HostUserId +TeamCartStatus Status +ShareableLinkToken ShareToken +DateTime? Deadline +DateTime CreatedAt +DateTime ExpiresAt + IReadOnlyList<TeamCartMember> Members + IReadOnlyList<TeamCartItem> Items + IReadOnlyList<MemberPayment> MemberPayments +long QuoteVersion +Money GrandTotal + IReadOnlyDictionary<UserId, Money> MemberTotals +Money TipAmount +CouponId? AppliedCouponId</pre>
<pre>+Create(UserId hostUserId, RestaurantId restaurantId, string hostName, DateTime? deadline) +AddMember(UserId userId, string name, MemberRole role) +SetDeadline(UserId requestingUserId, DateTime deadline) +IsExpired() +AddItem(UserId userId, MenuItemId menuItemId, MenuCategoryId menuCategoryId, ...) +UpdateItemQuantity(UserId requestingUserId, TeamCartItem item, int newQuantity) +RemoveItem(UserId requestingUserId, TeamCartItem item) +LockForPayment(UserId requestingUserId) +FinalizePricing(UserId requestingUserId) +MarkAsExpired() +ValidateJoinToken(string token) +CommitToCashOnDelivery(UserId userId, Money amount) +RecordSuccessfulOnlinePayment(UserId userId, Money amount, string transactionId) +RecordFailedOnlinePayment(UserId userId, Money amount) +ApplyTip(UserId requestingUserId, Money tipAmount) +ApplyCoupon(UserId requestingUserId, CouponId couponId) +RemoveCoupon(UserId requestingUserId) +ComputeQuoteLite(IReadOnlyDictionary<UserId, Money> memberItemSubtotals, Money feesTotal, ...) +GetMemberQuote(UserId userId) +MarkAsConverted()</pre>

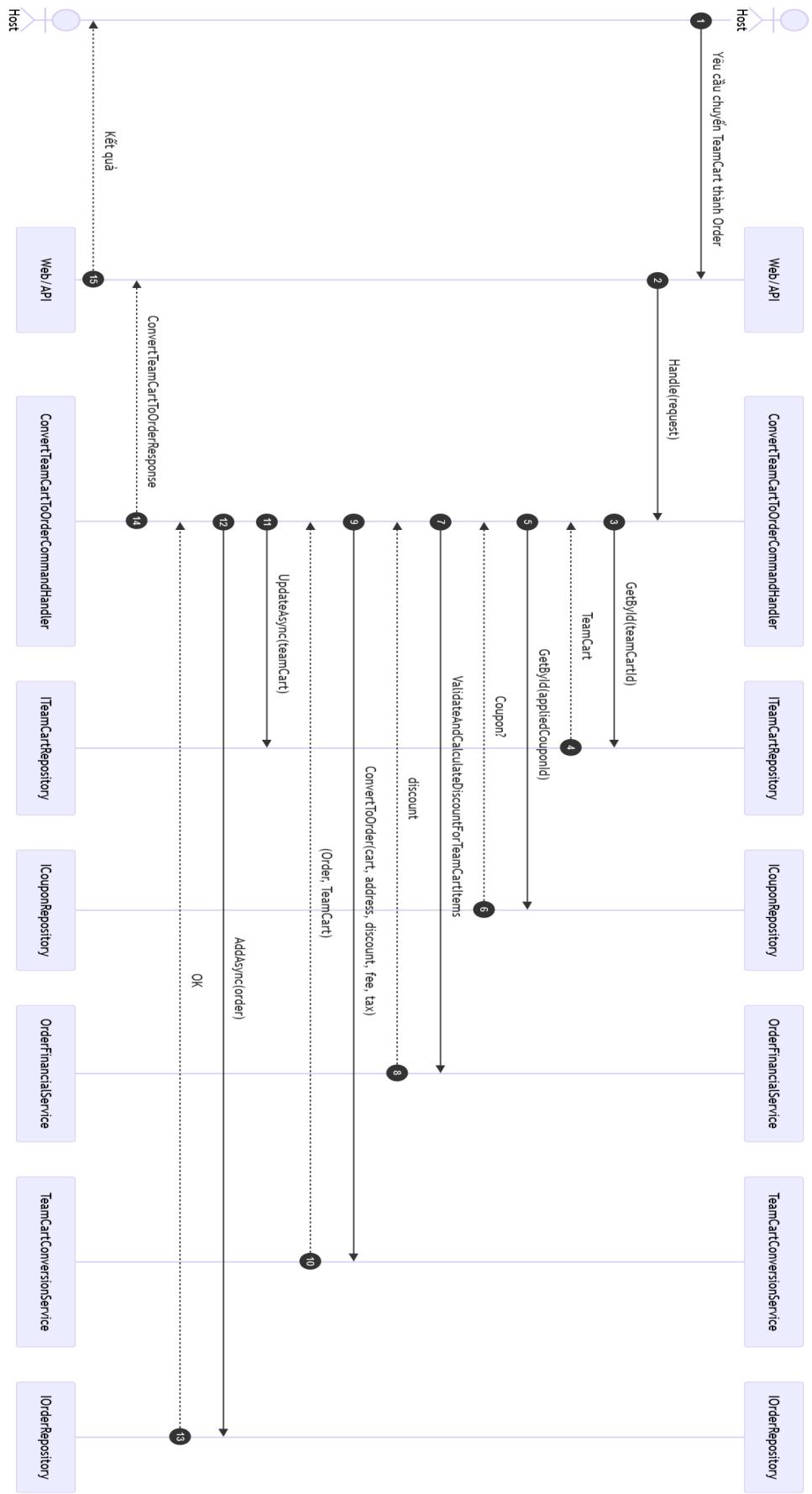
Hình 0.8: Biểu đồ lớp của TeamCart (Aggregate root)

Lớp Restaurant quản lý hồ sơ nhà hàng, các thông tin định danh và trạng thái hoạt động như đã xác thực hay đang nhận đơn. Thiết kế này đảm bảo các ràng buộc nghiệp vụ khi thay đổi thông tin thương hiệu, địa điểm, khung giờ hoạt động và trạng thái xác thực. Lớp này lần lượt được minh họa tại Hình 0.9.

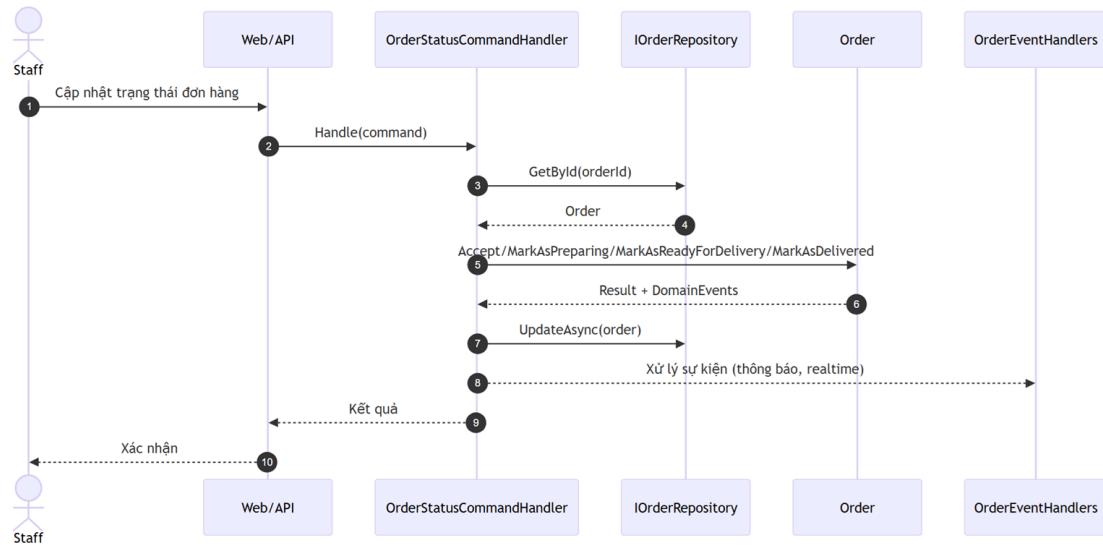


Hình 0.9: Biểu đồ lớp của Restaurant (Aggregate root)

Để minh họa cách các lớp phối hợp trong các ca sử dụng quan trọng, hai luồng chính được lựa chọn gồm chuyển TeamCart thành Order và cập nhật trạng thái đơn hàng. Luồng chuyển TeamCart thành Order nhấn mạnh vai trò của Domain Service trong việc ánh xạ dữ liệu và tính toán tài chính trước khi tạo Order và cập nhật TeamCart. Luồng cập nhật trạng thái đơn hàng thể hiện các chuyển trạng thái hợp lệ trong Order và phát sinh sự kiện miền để phục vụ cập nhật thời gian thực. Các luồng này được trình bày lần lượt tại Hình 0.10 và Hình 0.11.



Hình 0.10: Biểu đồ trình tự chuyển TeamCart thành Order

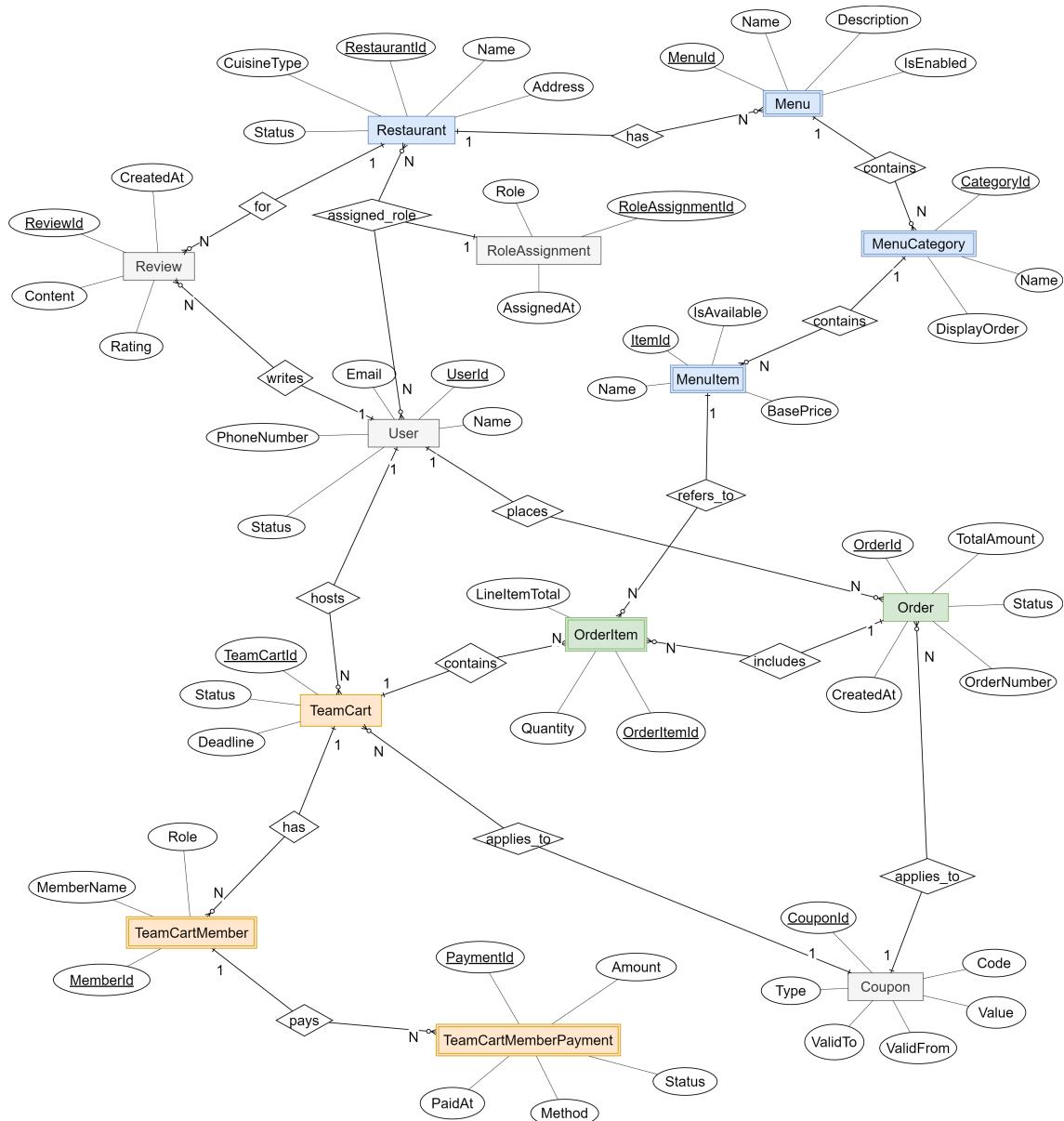


Hình 0.11: Biểu đồ trình tự cập nhật trạng thái đơn hàng

0.2.3 Thiết kế cơ sở dữ liệu

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

Biểu đồ ERD khái niệm theo Chen tập trung vào các thực thể cốt lõi như Người dùng, Nhà hàng, Thực đơn, Món ăn, Đơn hàng và Giỏ nhóm, cùng các quan hệ nghiệp vụ giữa chúng. Mỗi thực thể chỉ giữ lại một số thuộc tính chính như định danh, trạng thái, thông tin mô tả và giá trị tiền tệ, giúp làm rõ bức tranh nghiệp vụ mà chưa bị ràng buộc bởi chi tiết triển khai. Hình 0.12 mô tả toàn bộ ERD khái niệm, trong đó các quan hệ chính như Nhà hàng–Thực đơn, Người dùng–Đơn hàng, Đơn hàng–Mục đơn hàng và Giỏ nhóm–Thành viên được biểu diễn rõ ràng theo bội số.

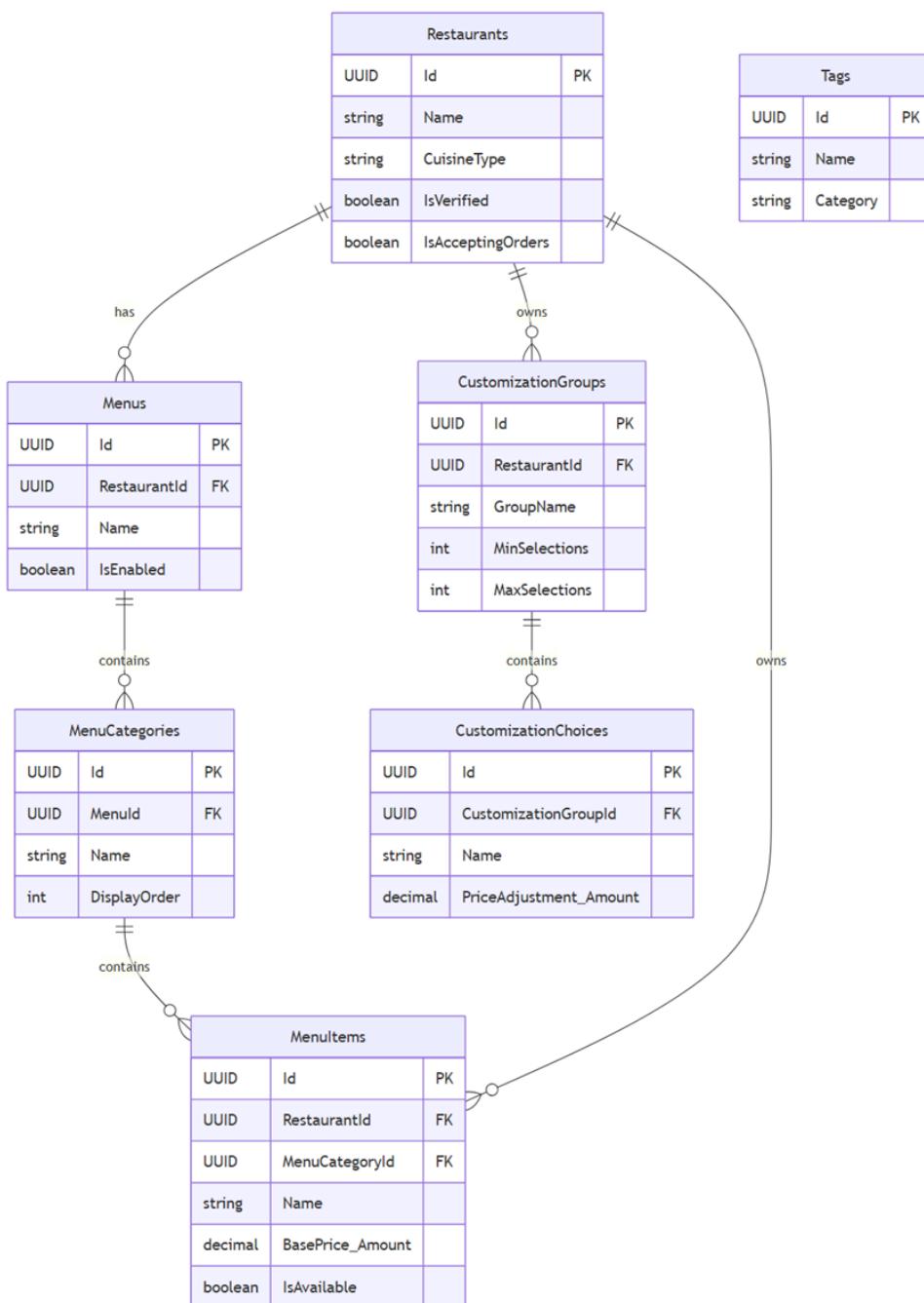


Hình 0.12: ERD khái niệm theo mô hình Chen cho YummyZoom

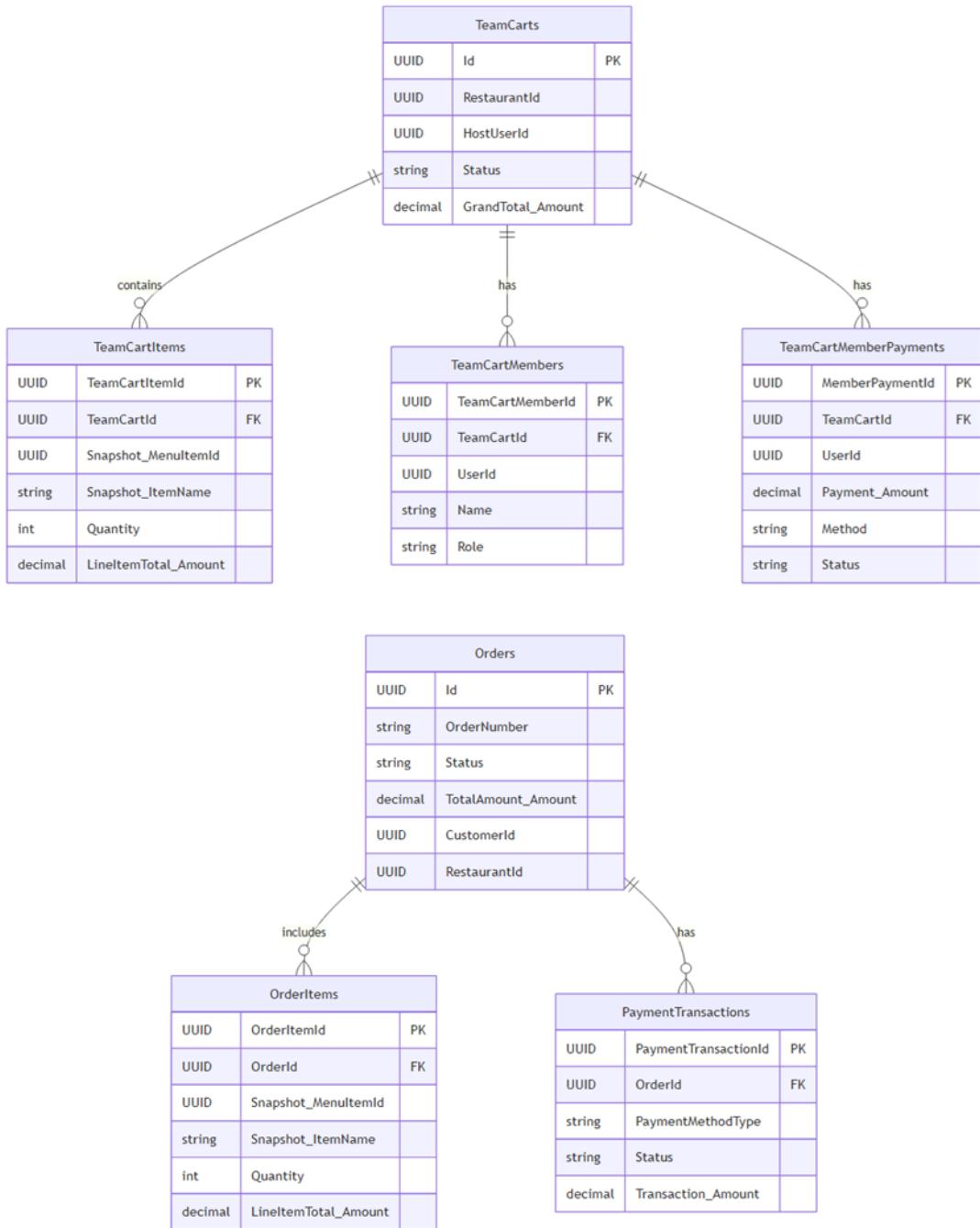
Từ mô hình khái niệm, hệ thống áp dụng quy tắc ánh xạ (mapping) từ mô hình miền (domain model) sang cơ sở dữ liệu quan hệ. Mỗi aggregate root được ánh xạ thành một bảng chính, ví dụ Order tương ứng bảng Orders, Restaurant tương ứng bảng Restaurants, TeamCart tương ứng bảng TeamCarts. Các thực thể có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

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

Ở mức triển khai vật lý, số lượng bảng lớn được chia nhỏ theo từng nhóm chức năng để trình bày rõ ràng. Hình 0.13 mô tả nhóm Restaurant Catalog, gồm Restaurants, Menus, MenuCategories, MenuItemItems và các bảng tùy chọn món; nhóm này phản ánh cấu trúc dữ liệu phục vụ duyệt thực đơn và quản trị nội dung. Hình 0.14 trình bày nhóm Order & TeamCart, làm rõ cấu trúc lưu trữ đơn hàng, mục đơn hàng, thanh toán, giỏ nhóm và thành viên.



Hình 0.13: ERD hiện đại cho nhóm Restaurant Catalog



Hình 0.14: ERD hiện đại cho nhóm Order và TeamCart

0.3 Xây dựng ứng dụng

0.3.1 Thư viện và công cụ sử dụng

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

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

(Tiếp tục trang sau)

(Tiếp theo từ trang trước)

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

Bảng 0.1: Danh sách thư viện và công cụ sử dụng

0.3.2 Kết quả đạt được

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

Sản phẩm backend được đóng gói dưới dạng dịch vụ ASP.NET Core Web API. Các số liệu thống kê chính được tổng hợp trong Bảng 0.2.

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

Bảng 0.2: Thống kê kết quả backend

b, Mobile (Flutter/Dart)

Ứng dụng di động dành cho khách hàng được phát triển bằng Flutter/Dart. Các số liệu thống kê chính được tổng hợp trong Bảng 0.3.

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

Bảng 0.3: Thông kê kết quả mobile

c, Web Admin (Angular/TypeScript)

Ứng dụng web quản trị được phát triển bằng Angular/TypeScript. Các số liệu thống kê chính được tổng hợp trong Bảng 0.4.

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

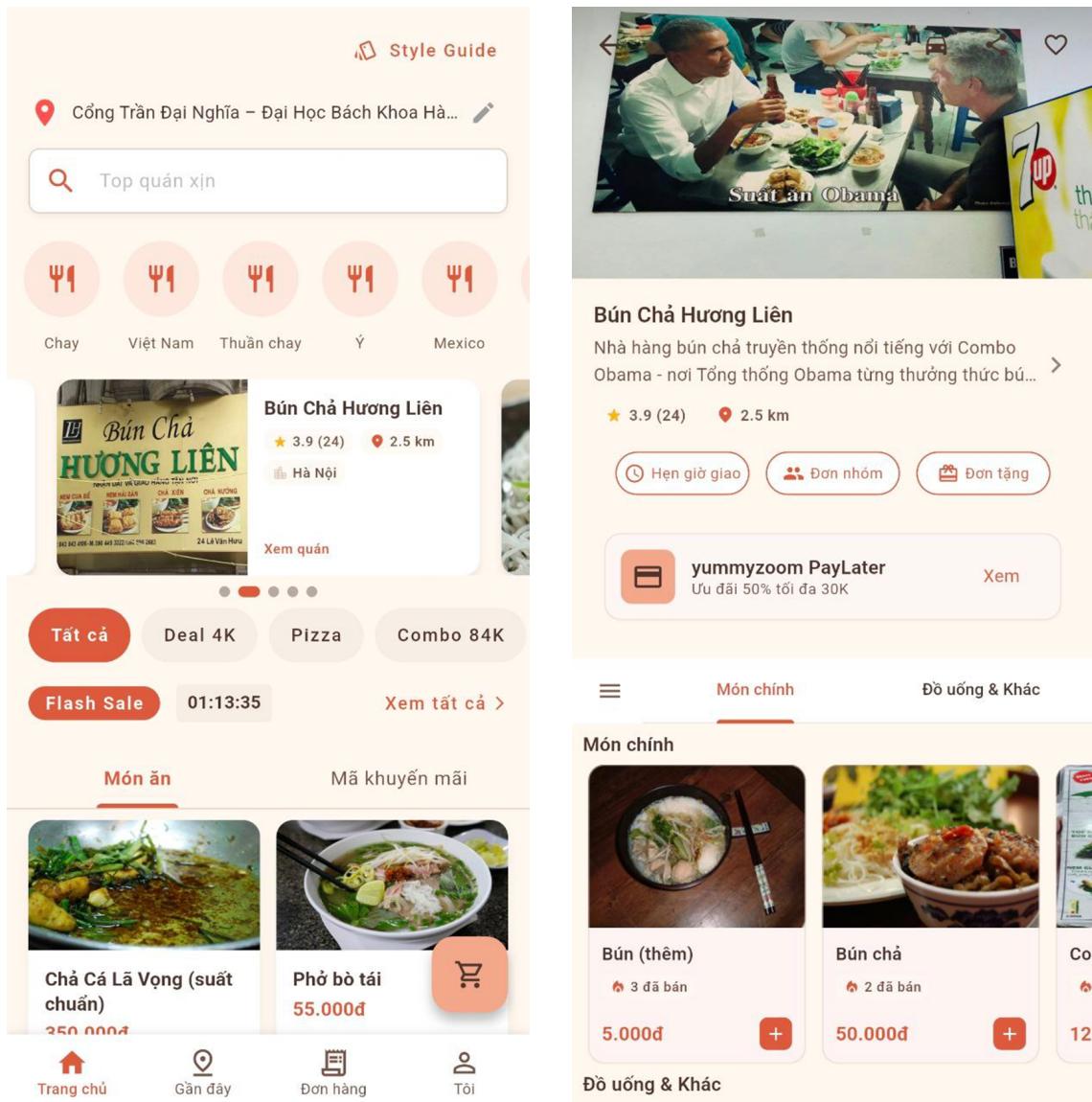
Bảng 0.4: Thông kê kết quả web admin

0.3.3 Minh họa các chức năng chính

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

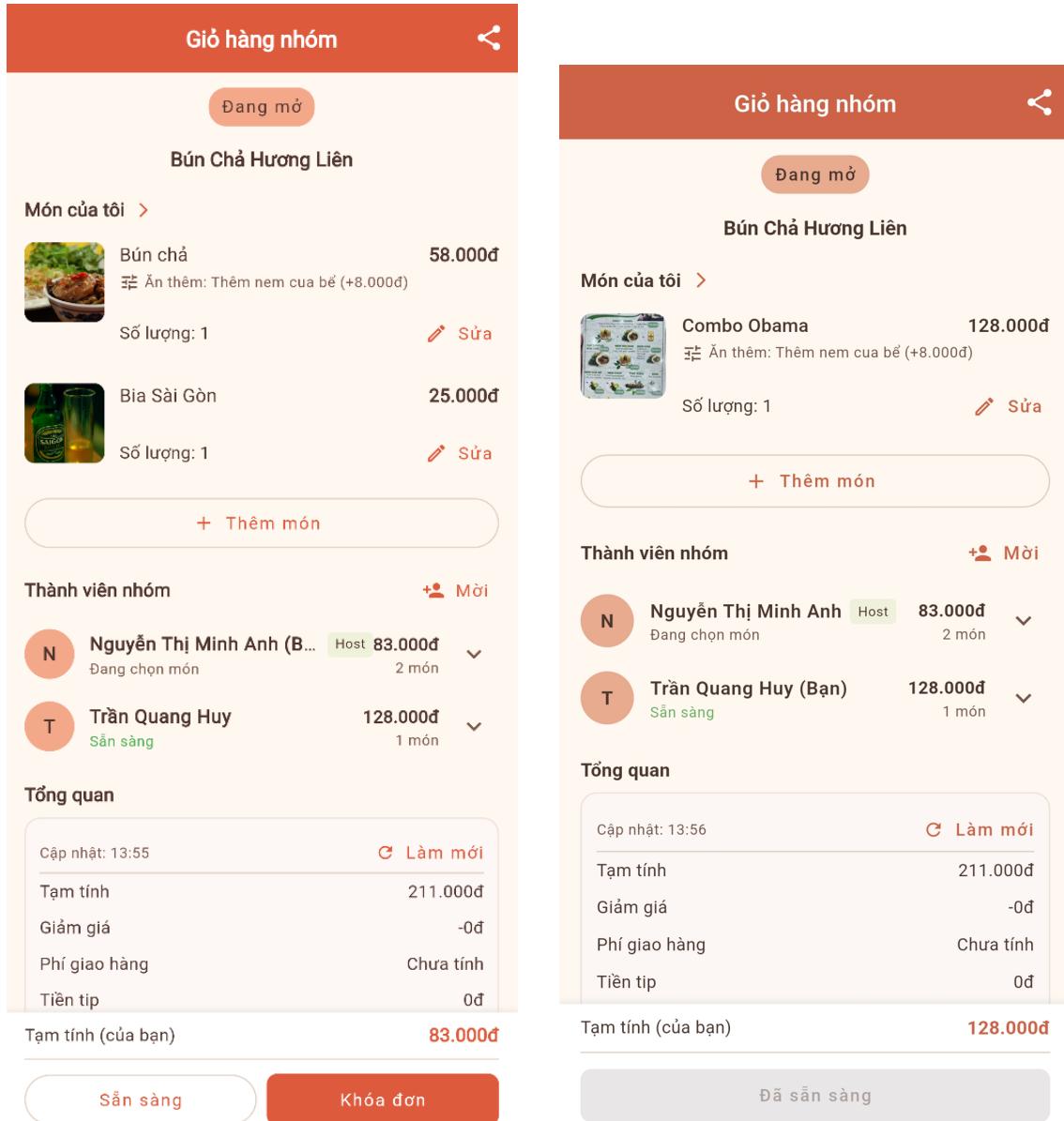
a, Ứng dụng di động

Giao diện trang chủ (Home) và chi tiết nhà hàng (Restaurant Detail) là điểm chạm đầu tiên của người dùng. Hình 0.15 minh họa cách bố trí thông tin trực quan, giúp người dùng nhanh chóng tiếp cận danh sách nhà hàng và khám phá thực đơn. Màn hình chi tiết nhà hàng hiển thị đầy đủ thông tin, đánh giá và danh sách món ăn được phân nhóm khoa học, hỗ trợ thao tác thêm món nhanh chóng.



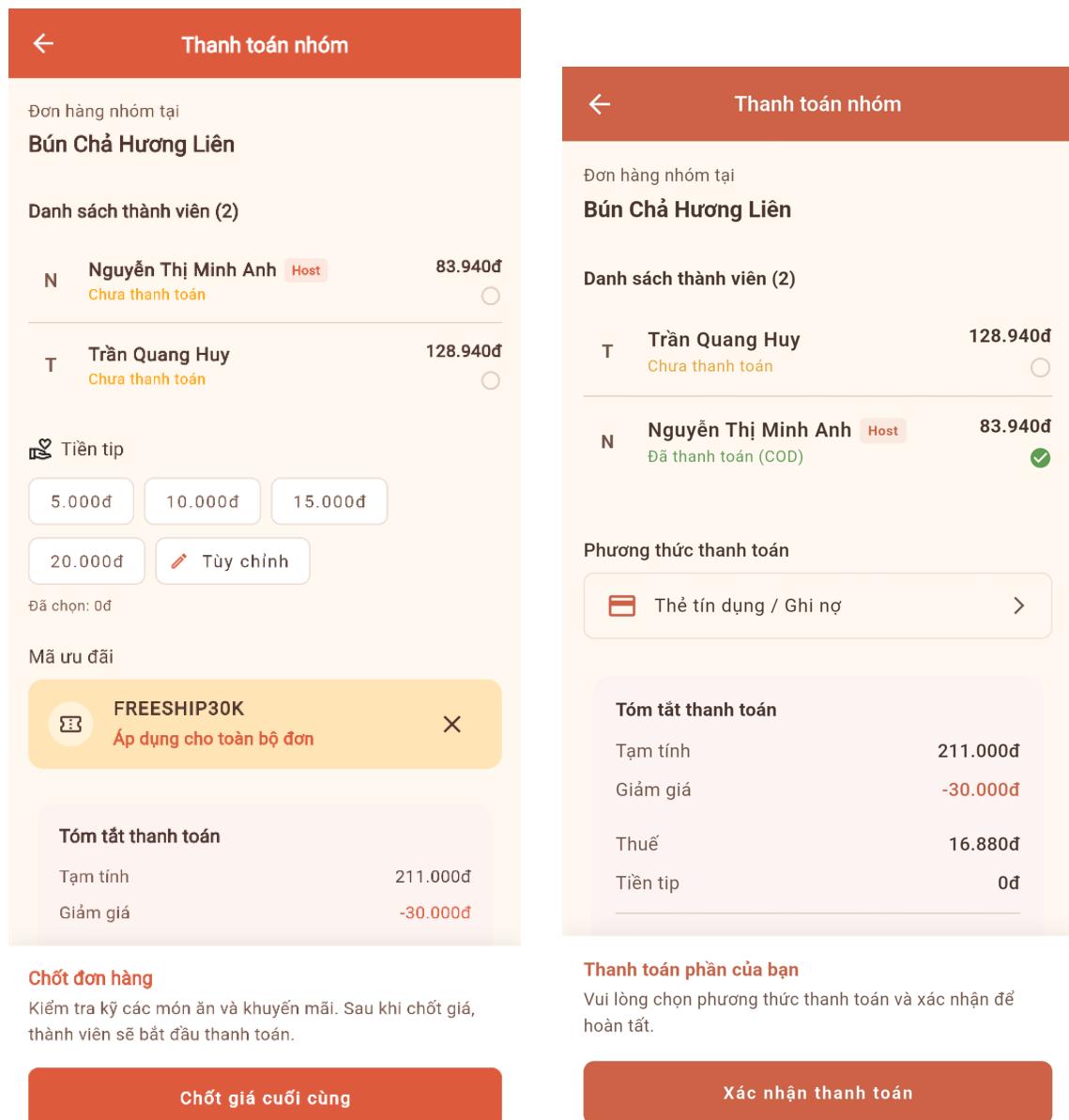
Hình 0.15: Màn hình Trang chủ (Trái) và Chi tiết nhà hàng (Phải)

Tính năng Đặt hàng nhóm (TeamCart) là điểm nhấn công nghệ của dự án, cho phép nhiều người dùng cùng thao tác trên một giỏ hàng trong thời gian thực. Hình 0.16 thể hiện sự đồng bộ trạng thái giữa màn hình của chủ phòng (Host) và thành viên (Member). Khi bất kỳ ai thêm hoặc bớt món, danh sách món ăn và tổng tiền sẽ được cập nhật tức thì trên tất cả thiết bị nhờ công nghệ SignalR, tạo cảm giác cộng tác mượt mà như đang ngồi cùng nhau.



Hình 0.16: Giao diện TeamCart: Màn hình Chủ phòng (Trái) và Thành viên (Phải)

Quy trình thanh toán cho đơn hàng nhóm được chia thành hai giai đoạn rõ ràng: chốt giá và thanh toán. Hình 0.17 minh họa màn hình Chốt giá nơi chủ phòng xem lại tổng kết đơn hàng, bao gồm các khoản phí và tiền tip. Sau khi chủ phòng xác nhận, tất cả thành viên trong nhóm được chuyển sang màn hình thanh toán để thanh toán phần của mình an toàn bằng cách chọn thanh toán bằng tiền mặt hoặc thanh toán qua ví điện tử, tích hợp Stripe để xử lý giao dịch thẻ tín dụng, đảm bảo tính toàn vẹn của dữ liệu tài chính.



Hình 0.17: Quy trình Checkout: Giai đoạn Chốt giá (Trái) và Thanh toán (Phải)

b, Hệ thống quản trị

Đối với đối tác nhà hàng, hệ thống cung cấp các công cụ quản lý mạnh mẽ và chuyên sâu. Hình 0.18 minh họa giao diện Quản lý thực đơn với thiết kế phân tách giữa danh sách món ăn và thư viện tùy chọn (Modifiers). Người quản lý có thể dễ dàng cấu hình các nhóm tùy chọn (ví dụ: Mức đường, Mức đá, Topping) và gán

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

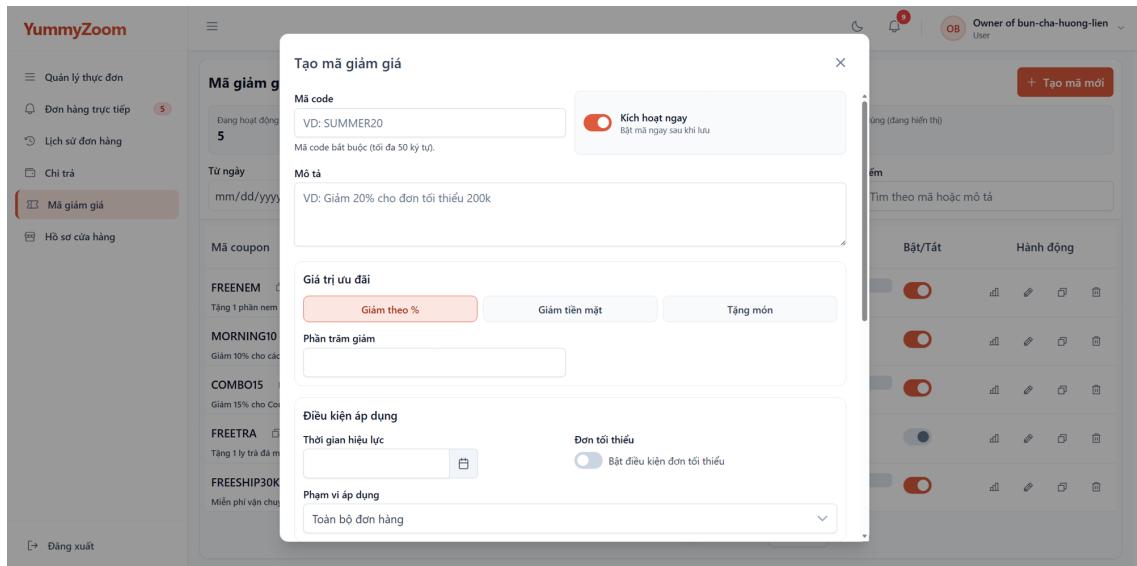
Hình 0.18: Giao diện Quản lý Menu và Thư viện tùy chọn

Trung tâm vận hành của nhà hàng là màn hình "Live Orders" (Đơn hàng trực tiếp), được thiết kế theo phong cách Kanban trực quan như Hình 0.19. Giao diện này chia làm ba làn: Đơn mới, Đang chế biến, và Sẵn sàng giao, giúp nhân viên bếp và thu ngân dễ dàng điều phối trạng thái đơn hàng chỉ bằng thao tác kéo thả hoặc nhấn nút. Thiết kế Responsive đảm bảo tính năng này hoạt động hiệu quả trên cả máy tính bảng và điện thoại di động của nhân viên đứng quầy.

Hình 0.19: Trung tâm xử lý đơn hàng trực tiếp (Live Orders Dashboard)

Cuối cùng, tính năng Quản lý khuyến mãi (Coupon Management) cho phép nhà hàng chủ động tạo các chiến dịch marketing. Hình 0.20 hiển thị danh sách các mã giảm giá cùng trạng thái hoạt động. Hệ thống hỗ trợ thiết lập chi tiết các loại ưu

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



Hình 0.20: Giao diện Quản lý Mã giảm giá và Chương trình khuyến mãi

0.4 Kiểm thử

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

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

0.4.1 Kiểm thử chức năng Đặt hàng nhóm (TeamCart)

- Mô tả chức năng:** Cho phép người dùng tạo một giỏ hàng chung, mời bạn bè tham gia qua mã QR hoặc liên kết chia sẻ. Các thành viên có thể cùng thêm/bớt món ăn và xem thay đổi của nhau trong thời gian thực.
- Kỹ thuật kiểm thử sử dụng:** Kiểm thử hộp đen, Kiểm thử tích hợp (xác minh tính đồng bộ thời gian thực qua SignalR).
- Các trường hợp kiểm thử:**

STT	Tình huống kiểm thử	Đầu vào	Kết quả mong đợi
1	Tạo mới TeamCart	Nhấn nút "Bắt đầu nhóm"	Giỏ hàng nhóm được tạo, mã QR và Link mời được sinh ra. Người tạo trở thành Chủ phòng (Host).
2	Tham gia nhóm bằng mã	Nhập mã 6 chữ số hợp lệ	Người dùng tham gia thành công vào nhóm, hiển thị danh sách thành viên hiện tại.
3	Đồng bộ thêm món (Real-time)	Thành viên A thêm 1 món	Màn hình của Chủ phòng và các thành viên khác tự động cập nhật món mới và tổng tiền ngay lập tức mà không cần tải lại.
4	Khóa giỏ hàng để chốt đơn	Chủ phòng nhấn "Chốt đơn"	Trạng thái giỏ hàng chuyển sang "Đang chốt". Các thành viên khác không thể thêm/bớt món, nhận thông báo giỏ hàng đã bị khóa.

Bảng 0.5: Kiểm thử chức năng Đặt hàng nhóm

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

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

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

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

Bảng 0.6: Kiểm thử chức năng Quy trình Đơn hàng

0.4.3 Kiểm thử chức năng Tìm kiếm và Lọc món ăn

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

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

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

Bảng 0.7: Kiểm thử chức năng Tìm kiếm và Lọc

0.4.4 Tổng kết kết quả kiểm thử

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

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

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

0.5 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.)