**BÁO CÁO ĐỒ ÁN CUỐI KỲ – MINI E‑COMMERCE (Strategy & Decorator)**

Sinh viên: Nguyễn Hoàng Kiệt – MSSV 2212398

Ngày nộp: 26/10/2025

**CHƯƠNG I. TỔNG QUAN**

**1.1. Giới thiệu & mục tiêu**

Dự án “Mini E‑Commerce” xây dựng một hệ thống thương mại điện tử thu gọn phục vụ các tác vụ: quản lý sản phẩm, giỏ hàng, thanh toán giả lập, lựa chọn phương thức vận chuyển và bổ sung dịch vụ vận chuyển. Báo cáo tập trung vào áp dụng hai mẫu thiết kế hướng đối tượng: Strategy cho tính phí vận chuyển linh hoạt và Decorator cho việc kết hợp các dịch vụ bổ sung.

Mục tiêu: (i) minh chứng khả năng thay đổi thuật toán tính phí mà không đổi nơi gọi (Strategy) [1], (ii) mở rộng tính năng vận chuyển theo kiểu xếp chồng không xâm lấn mã cũ (Decorator) [1], (iii) đảm bảo nguyên lý SOLID để mã nguồn dễ bảo trì [2].

**1.2. Kiến trúc hệ thống**

Hệ thống áp dụng kiến trúc phân tầng hướng domain: API (presentation), Business (services), Core (domain + patterns), Infrastructure (data). Các lớp Controller phơi bày RESTful API [3]; Service triển khai nghiệp vụ; Core định nghĩa Entity/Interface/Strategy/Decorator; Infrastructure triển khai lưu trữ SQLite [4]. Giao tiếp giữa các tầng sử dụng DTO nhằm tách biệt biểu diễn và domain.

**1.3. Chức năng chính**

Người dùng: (a) đăng ký/đăng nhập (JWT) [5][7]; (b) duyệt danh mục và chi tiết sản phẩm; (c) quản lý giỏ; (d) chọn phương thức vận chuyển và nhận ước tính phí từ Strategy; (e) thực hiện checkout và lưu đơn. Quản trị viên: (a) CRUD sản phẩm; (b) cấu hình tham số các chiến lược vận chuyển; (c) xem báo cáo gọn.

**1.4. Công nghệ sử dụng**

• Nền tảng: .NET (Web API), C#, SQLite (file‑based RDBMS) [4], Entity Framework Core (ORM) [9].

• Bảo mật: JWT (RFC 7519) [5], BCrypt cho băm mật khẩu [6].

• Tài liệu hóa: Swagger/OpenAPI 3.x [8].

• Kiểm thử: xUnit/NUnit (tùy dự án), Postman collection cho thử nghiệm API.

**CHƯƠNG II. CƠ SỞ LÝ THUYẾT**

**2.1. Nguyên lý SOLID**

SOLID là tập 5 nguyên tắc thiết kế hướng đối tượng giúp tăng tính mô‑đun và khả năng thay đổi: (S) Single Responsibility, (O) Open/Closed, (L) Liskov Substitution, (I) Interface Segregation, (D) Dependency Inversion [2].

Áp dụng vào dự án: ShippingStrategy tuân thủ LSP (mỗi chiến lược thay thế lẫn nhau); Service phụ thuộc abstraction IShippingStrategy (DIP); Decorator đóng gói một trách nhiệm (SRP) và mở rộng mà không sửa lớp cũ (OCP).

**2.2. Các mẫu thiết kế sử dụng**

**2.2.1. Decorator Pattern**

Decorator cho phép bổ sung hành vi động bằng cách “bọc” đối tượng gốc bằng các lớp trang trí tuân thủ cùng interface; có thể xếp chồng nhiều lớp để tạo hiệu ứng tổng hợp [1]. Trong dự án, các dịch vụ bổ sung như bảo hiểm, gói quà, yêu cầu chữ ký… được triển khai như các Decorator bọc quanh IShippingStrategy, tạo nên phí cộng dồn minh bạch.

**2.2.2. Strategy Pattern**

Strategy định nghĩa họ thuật toán có thể hoán đổi tại runtime; client chỉ biết interface chung, không phụ thuộc hiện thực cụ thể [1]. Dự án dùng Strategy để tính phí vận chuyển theo các biến số: khối lượng, cự ly, vùng, khung giờ cao điểm, cutoff trong ngày, v.v.

**2.3. So sánh & lý do chọn mẫu**

Strategy giải quyết yêu cầu “thay thuật toán không động chạm nơi gọi”, trong khi Decorator giải quyết yêu cầu “kết hợp linh hoạt các dịch vụ bổ sung”. Kết hợp hai mẫu giúp hệ thống vừa linh hoạt trong lõi tính phí (Strategy), vừa giàu tùy biến (Decorator), đồng thời vẫn đảm bảo tuân thủ SOLID [1][2].

**CHƯƠNG III. ÁNH XẠ MẪU THIẾT KẾ VÀO TÍNH NĂNG DỰ ÁN**

**3.1. Chức năng chính của hệ thống**

• Quản lý sản phẩm: CRUD, tồn kho, trọng lượng, danh mục.

• Giỏ hàng: thêm/xóa/cập nhật số lượng; tính subtotal.

• Vận chuyển: hiển thị các phương thức khả dụng và phí ước tính.

• Checkout: tạo đơn hàng, lưu breakdown (subtotal, giảm, thuế, phí ship, tổng).

**3.1.1. Luồng hoạt động của các lớp (rút gọn)**

Controller → Service → (IShippingStrategy | Decorators) → Repository/DbContext (SQLite) → trả về DTO.

Service chọn chiến lược theo mã phương thức; nếu có dịch vụ bổ sung, Service bọc chiến lược bằng chuỗi Decorator theo thứ tự cấu hình.

**3.1.2. Luồng hoạt động chức năng (ví dụ Checkout)**

(1) Client gửi giỏ + methodCode + danh sách add‑ons → (2) Service dựng OrderContext (weight, distance, region, timeslot) → (3) Resolve IShippingStrategy từ registry → (4) Áp chuỗi Decorator theo yêu cầu → (5) Tính shippingFee và breakdown chi tiết → (6) Tính grandTotal → (7) Lưu Order (SQLite) → (8) Trả DTO kết quả.

**3.2. Ràng buộc và quy tắc nghiệp vụ**

• Cắt giờ same‑day (ví dụ 15:00); sau cutoff thì từ chối hoặc báo phí rất cao.

• Hạn mức cộng dồn add‑ons: quy định thứ tự áp dụng, chống xung đột.

• Ràng buộc địa lý: hệ số vùng (Urban/Suburb/Rural).

• Nhật ký tính phí: lưu lại tên chiến lược và các add‑ons đã áp.

**3.3. Ánh xạ mẫu thiết kế vào code**

**3.3.1. Strategy Pattern – Tính phí vận chuyển**

Interfaces/IShippingStrategy.cs định nghĩa Calculate(OrderContext) và Name/GetDetails. Implementations gồm Standard, Express (có surge giờ cao điểm), SameDay (có cutoff), Eco (giá thấp, thời gian lâu).

**3.3.2. Decorator Pattern – Dịch vụ bổ sung vận chuyển**

Decorators/ShippingDecorator.cs là lớp trừu tượng bọc IShippingStrategy. Các hiện thực: Insurance, GiftWrapping, SignatureRequired, PriorityHandling, WeekendDelivery… mỗi lớp bổ sung một phần phí, đồng thời gọi xuống đối tượng được bọc.

**CHƯƠNG IV. PHÂN TÍCH ỨNG DỤNG**

**4.1. Kiến trúc và cấu trúc mã nguồn**

**4.1.1. Kiến trúc phân tầng**

• API: Controllers phơi bày REST endpoints (OpenAPI/Swagger) [8].

• Business: Services triển khai use case; dùng DI để nhận IShippingStrategy [10].

• Core: Entities (User, Product, Order...), Interfaces, Strategies, Decorators, Models (OrderContext).

• Infrastructure: DbContext, Repository/Configurations cho EF Core + SQLite [9].

**4.1.2. Cấu trúc thư mục**

1\_Source/src/

MiniECommerce.Core/ (Entities, Interfaces, Models, Strategies, Decorators)

MiniECommerce.Infrastructure/ (Data: DbContext, DbInitializer)

MiniECommerce.API/ (Controllers, Services, DTOs)

3\_Database/ (ecommerce.db, schema.sql, seed\_data.sql)

**4.1.3. Dependency Injection và Service Registration**

Hệ thống đăng ký IShippingStrategy và các implementations trong container; Service nhận vào abstraction (DIP) [10]. Việc thêm chiến lược mới chỉ cần đăng ký thêm implementation mà không sửa client (OCP) [2].

**4.2. API Endpoints và chức năng (tiêu biểu)**

• Authentication: POST /api/auth/register, /api/auth/login → JWT [5][7].

• Products: GET/POST/PUT/DELETE /api/products → CRUD sản phẩm.

• Cart: POST/PATCH/DELETE /api/cart/items; GET /api/cart → subtotal.

• Shipping: GET /api/shipping/options → ước tính theo Strategy; POST /api/shipping/quote → breakdown.

• Orders: POST /api/orders/checkout → tạo đơn; GET /api/orders/{id} → chi tiết; GET /api/orders → lịch sử.

• Reports: GET /api/reports/daily → thống kê gọn.

**4.3. Triển khai chi tiết theo mẫu thiết kế**

**4.3.1. Strategy Pattern – Tính phí vận chuyển**

a) Mô tả vấn đề và giải pháp: yêu cầu thay đổi/so sánh thuật toán (standard/express/same‑day/eco) theo dữ liệu ngữ cảnh mà không đổi client; dùng strategy registry + DI để tách biệt.

b) Sơ đồ lớp (mô tả chữ): IShippingStrategy ⇢ {Standard, Express, SameDay, Eco}; OrderService phụ thuộc abstraction.

c) Implementations:

• Standard: fee = base + perKg\*weight, nhân hệ số vùng.

• Express: fee = base + perKg\*weight, nhân surge khi trong khung giờ cao điểm.

• SameDay: kiểm tra cutoff; fee = baseHigh + perKm\*distance.

• Eco: hệ số ưu đãi cho đơn nhẹ/dài ngày.

d) Sử dụng trong OrderService: resolve theo methodCode, gọi Calculate(ctx), nhận Details để ghi log breakdown.

e) Code minh hoạ và test: kiểm thử đơn vị biên (weight=0, distance=0, boundary time tại giờ cao điểm và cutoff).

**4.3.2. Decorator Pattern – Dịch vụ bổ sung vận chuyển**

a) Vấn đề/giải pháp: cần bổ sung các phí tùy chọn (bảo hiểm, gói quà, chữ ký…) có thể kết hợp linh hoạt; Decorator bọc IShippingStrategy để cộng dồn chi phí mà không sửa các lớp strategy.

b) Sơ đồ lớp (mô tả chữ): ShippingDecorator (abstract) wraps IShippingStrategy ⇢ {Insurance, GiftWrapping, SignatureRequired, PriorityHandling, WeekendDelivery, ...}.

c) Implementations: mỗi decorator tăng thêm một khoản phí theo điều kiện (tỷ lệ phần trăm/định mức).

d) Composition: xếp chồng theo thứ tự cấu hình để có kết quả quyết định (ví dụ tính bảo hiểm theo subtotal trước rồi mới cộng phụ phí cuối tuần).

e) Test: kiểm thử tổ hợp 2–3 decorator; so sánh breakdown mong đợi.

**4.4. Cơ sở dữ liệu**

**4.4.1. Công nghệ: SQLite là RDBMS nhúng, lưuFile, chuẩn ACID, không cần server [4]. EF Core ánh xạ đối tượng‑quan hệ, hỗ trợ migrations và LINQ [9].**

**4.4.2. ERD (mô tả chữ): Users(1)–(n)Orders; Orders(1)–(n)OrderItems; Products(1)–(n)OrderItems; ShippingMethods độc lập; CartItems liên kết User và Product.**

**4.4.3. Bảng chính: Users(Id, Email, PasswordHash, Role), Products(Id, Name, Price, Category, Stock, Weight), Orders(Id, UserId, Subtotal, ShippingFee, GrandTotal, Status), OrderItems(OrderId, ProductId, Qty, UnitPrice, LineTotal), ShippingMethods(Code, ParamsJSON).**

**4.4.4. Seed: schema.sql và seed\_data.sql cung cấp dữ liệu mẫu; có reset\_database.sql để làm sạch.**

**4.5. Bảo mật**

**4.5.1. JWT Authentication: token theo RFC 7519, gồm header‑payload‑signature, thường ký bằng HS256; lưu ngắn hạn trên client [5].**

**4.5.2. Role‑based Authorization: phân quyền User/Admin tại controller/action.**

**4.5.3. Password Hashing: BCrypt với salt và work factor, tăng sức kháng tấn công vét cạn [6]. Khuyến nghị tuân thủ OWASP JWT/Password guidelines [7].**

**4.6. Giao diện và testing**

**4.6.1. Swagger UI: tự sinh tài liệu từ OpenAPI; hỗ trợ thử nhanh endpoints [8].**

**4.6.2. Postman Collection: nhóm request theo module, thêm ví dụ và biến môi trường.**

**4.6.3. Luồng test chính: (i) đăng ký, login; (ii) thêm sản phẩm, dựng giỏ; (iii) yêu cầu báo giá shipping theo 4 chiến lược; (iv) thêm add‑ons (decorator); (v) checkout và đối chiếu breakdown.**

**4.7. Đặc tả use case & kiểm thử**

**4.7.1. Use Case Diagram (mô tả): Tác nhân User và Admin tương tác các chức năng tương ứng; Shipping nằm trong flow Checkout.**

**4.7.2. Đặc tả UC chính: UC01(Đăng ký/Đăng nhập), UC02(Quản lý sản phẩm), UC03(Quản lý giỏ), UC04(Chọn phương thức vận chuyển), UC05(Thêm dịch vụ bổ sung), UC06(Đặt hàng). Mỗi UC mô tả tiền điều kiện, dòng sự kiện, hậu điều kiện, ngoại lệ.**

**4.7.3. Kết quả kiểm thử: log kết quả và screenshot Swagger/Postman minh hoạ các tổ hợp Strategy + Decorator hoạt động đúng.**

**CHƯƠNG V. TỔNG KẾT**

**5.1. Đánh giá ứng dụng**

Ưu điểm: mã tuân thủ SOLID, tách lớp và giao diện rõ ràng; dễ thay đổi thuật toán vận chuyển; bổ sung dịch vụ không xâm lấn mã cũ; dữ liệu SQLite đơn giản triển khai; API rõ ràng nhờ OpenAPI/Swagger.

Nhược điểm: chưa tích hợp cổng thanh toán thật; chưa tối ưu hoá hiệu năng khi số lượng decorator lớn; báo cáo giới hạn.

**5.2. Hướng phát triển**

• Bổ sung thêm chiến lược (international, cold‑chain), dynamic pricing theo mùa.

• Quy tắc xếp chồng decorator có DSL cấu hình.

• Tối ưu câu lệnh SQL và thêm chỉ mục; tách đọc/ghi nếu cần.

• Tích hợp UI web/mobile, thêm kiểm thử E2E.

**5.3. Kết luận**

Áp dụng đồng thời Strategy và Decorator vào bài toán vận chuyển thương mại điện tử cho thấy lợi ích thực tiễn của mẫu thiết kế: giảm liên kết chặt, tăng khả năng mở rộng và kiểm thử. Kiến trúc phân tầng, JWT, BCrypt và SQLite là lựa chọn phù hợp cho một dự án học thuật có thể mở rộng về sau [1][2][4][5][6][8][9][10].

**CHƯƠNG VI. TÀI LIỆU THAM KHẢO**

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, “Design Patterns: Elements of Reusable Object‑Oriented Software,” Addison‑Wesley, 1994. Available: https://www.informit.com/store/design-patterns-elements-of-reusable-object-oriented-9780201633610

[2] R. C. Martin, “Design Principles and Design Patterns,” Object Mentor, 2000. Available: https://web.archive.org/web/20190202123744/http://www.objectmentor.com/resources/articles/Principles\_and\_Patterns.pdf

[3] R. T. Fielding, “Architectural Styles and the Design of Network‑based Software Architectures,” Ph.D. dissertation, UC Irvine, 2000. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding\_dissertation.pdf

[4] SQLite Documentation, “About SQLite,” 2025. Available: https://www.sqlite.org/about.html

[5] M. Jones, J. Bradley, and N. Sakimura, “JSON Web Token (JWT),” RFC 7519, IETF, May 2015. Available: https://www.rfc-editor.org/rfc/rfc7519

[6] N. Provos and D. Mazieres, “A Future‑Adaptable Password Scheme,” USENIX 1999. Available: https://www.usenix.org/legacy/events/usenix99/provos/provos\_html/index.html

[7] OWASP, “JSON Web Token (JWT) Cheat Sheet for Java,” 2024. Available: https://cheatsheetseries.owasp.org/cheatsheets/JSON\_Web\_Token\_for\_Java\_Cheat\_Sheet.html

[8] OpenAPI Initiative, “OpenAPI Specification, v3.1.0,” 2021. Available: https://spec.openapis.org/oas/latest.html

[9] Microsoft Docs, “Entity Framework Core – Overview,” 2025. Available: https://learn.microsoft.com/ef/core/

[10] M. Fowler, “Inversion of Control Containers and the Dependency Injection pattern,” 2004. Available: https://martinfowler.com/articles/injection.html