**8. Kiến trúc phần mềm theo mô hình 3-tier**

**Lớp Presentation (UI/Web)**

* Web App:
  + Trang Danh mục: lọc/sắp xếp theo tên, giá; xem chi tiết nhanh (quick view).
  + Trang Chi tiết: tên, mô tả, tồn kho, thông tin kho (địa chỉ), “sản phẩm nổi bật”, đánh giá, nút “Mua hàng”.
  + Trang Giỏ hàng: danh sách dòng hàng + bảng tổng hợp (tổng chi phí, khuyến mãi, vận chuyển, tiết kiệm, tổng đơn).
  + Trang Thanh toán: xác thực thông tin + nút “Xác nhận thanh toán”.
  + Trang Quản trị: quản lý sản phẩm/kho, gán sản phẩm vào kho, xem/khóa giỏ hàng.
  + Trang Đăng nhập/đăng xuất: điều hướng Buyer → Catalog, SysAdmin → Admin.
* UI: chỉ xử lý hiển thị/validate cơ bản (required, định dạng), gọi API lớp Application.

**Lớp Application (Service/Business)**

* API Gateway (tuỳ chọn): gom các API, rate limit, auth.
* Các dịch vụ ứng dụng (có thể triển khai trong 1 service theo module hoặc tách nhỏ dần):
  1. CatalogService
     + Trả về thông tin hiển thị + tồn kho tổng hợp (lấy từ InventoryService).
  2. CartService
     + Thêm, cập nhật số lượng, xoá ...
     + Tự tính bảng tổng hợp: tổng trước KM, khuyến mãi tiết kiệm, phí ship, tổng đơn.
     + Gọi Pricing/PromotionService để tính khuyến mãi, InventoryService để reserve tạm khi thêm vào giỏ (tuỳ chính sách) hoặc chỉ kiểm tra khả dụng.
  3. CheckoutService
     + Xác thực tồn kho và giá, gọi PaymentService (mock/gateway), sau đó phát sự kiện “OrderPlaced”.
     + Gửi email qua NotificationService.
  4. InventoryService
     + CRUD kho; quản trị gán sản phẩm vào kho.
     + API kiểm tra/giảm tồn khi đặt hàng, hoàn trả khi huỷ.
  5. RatingService
     + Trung bình đánh giá sản phẩm.
  6. AuthService / AccessControl
     + Đăng nhập/đăng xuất; cấp JWT; phân quyền Buyer/SysAdmin (RBAC).
  7. AdminService
     + CRUD sản phẩm, gán sản phẩm vào kho hiện có.
     + Xem tất cả giỏ hàng, vô hiệu hoá giỏ không hợp lệ.
  8. NotificationService
     + Gửi email (SMTP/Email API) khi checkout.
  9. Pricing/PromotionService (nếu đã có logic KM/ship tách biệt)
     + Tính toán khuyến mãi, phí vận chuyển.
* Luồng chính:
  1. UI → CartService: thêm sản phẩm → CartService kiểm tra hợp lệ + (tuỳ chọn) reserve tồn qua InventoryService → tính tổng hợp qua Pricing/PromotionService → trả giỏ + bảng tổng hợp.
  2. Checkout: UI → CheckoutService → xác thực giá/tồn → PaymentService → nếu OK: Inventory trừ tồn, tạo Order, Notification gửi email.
* Tích hợp & giao tiếp:
  1. Đồng bộ: REST/GraphQL giữa các service.
  2. Bất đồng bộ: message queue (để gửi email/hậu xử lý sau thanh toán).

**Lớp Data (Persistence/Infra)**

* CSDL quan hệ (ví dụ PostgreSQL/MySQL):
  + Bảng: Products, Warehouses, ProductWarehouses (map sản phẩm-kho + tồn), Carts, CartItems, Orders, OrderItems, Users, Roles, Ratings, Promotions (tuỳ chọn).
* ORM/Repository: Repository + Unit of Work; transaction khi checkout.
* Indexing: chỉ mục theo name, price, productId, warehouseId, cartId.
* Seed Data lúc khởi động: tạo SysAdmin, 2 Buyer, dữ liệu mẫu sản phẩm/kho/đánh giá.
* Cache: Redis cho catalog và session giỏ khách (guest cart).
* Search mở rộng (tuỳ chọn): Elasticsearch/PG trigram cho tìm kiếm.

Luồng điển hình:

Thêm sản phẩm vào giỏ  
UI → POST /cart/items (productId, qty) → CartService:

* Validate input & quyền.
* Gọi InventoryService kiểm tra tồn (theo kho mặc định hoặc best-match).
* Gộp dòng nếu đã tồn tại; cập nhật Cart + tính lại Summary (gọi Pricing/Promotion).
* Lưu DB; trả về giỏ + summary mới.

Thanh toán  
UI → POST /checkout → CheckoutService:

* Xác thực giỏ & tồn kho lần cuối (lock/transaction).
* Trừ kho, tạo Order, gọi cổng thanh toán.
* Phát sự kiện; Worker gửi email.

**9. Test-case cho chức năng “Thêm sản phẩm vào giỏ hàng”**

Giả định: API POST /carts/{userId}/items với payload { productId, quantity }. Hệ thống kiểm tra tồn kho khả dụng, cập nhật giỏ, và trả về giỏ + bảng tổng hợp.

**Test case 1 — Thêm sản phẩm hợp lệ**

* Dữ liệu đầu vào:
  + userId = U1 (Buyer hợp lệ, giỏ đang “active”)
  + productId = P1001 tồn kho khả dụng ≥ 5, quantity = 2
* Bước thực hiện:

Gọi POST /carts/U1/items với { productId: P1001, quantity: 2 }.

* Kết quả mong đợi:
  + HTTP 200.
  + cart.items có mục cho P1001 với qty = 2.
  + Bảng tổng hợp được cập nhật chính xác:
    - totalBeforeDiscount = price(P1001) \* 2
    - discount, shippingFee tính theo rule hiện hành
    - grandTotal = totalBeforeDiscount - discount + shippingFee
  + (Tuỳ chính sách) tồn kho không bị trừ ngay, hoặc reserve tạm được ghi nhận.
  + Log/audit tạo record “ADD\_ITEM”.

**Test case 2 — Vượt quá tồn kho khả dụng**

* Dữ liệu đầu vào:
  + userId = U1
  + productId = P1002 chỉ còn inventory = 3, yêu cầu quantity = 5
* Bước thực hiện:

POST /carts/U1/items với { productId: P1002, quantity: 5 }.

* Kết quả mong đợi:
  + HTTP 409 (Conflict) hoặc 400 kèm thông báo “Số lượng vượt quá tồn kho”.
  + Giỏ hàng không thay đổi.
  + Bảng tổng hợp giữ nguyên.
  + Không tạo reserve/không trừ tồn.

**Test case 3 — Sản phẩm không tồn tại/đã bị vô hiệu hoá**

* Dữ liệu đầu vào:
  + userId = U1
  + productId = P9999 (không tồn tại hoặc is\_active = false)
  + quantity = 1
* Bước thực hiện:

POST /carts/U1/items với { productId: P9999, quantity: 1 }.

* Kết quả mong đợi:
  + HTTP 404 (Not Found) hoặc 400 với thông báo “Sản phẩm không hợp lệ”.
  + Không có thay đổi nào trong giỏ.

**Test case 4 — Cộng dồn khi thêm trùng sản phẩm**

* Dữ liệu đầu vào:
  + Trong giỏ đã có P1001 với qty = 1.
  + Thêm { productId: P1001, quantity: 2 }.
* Bước thực hiện:

POST /carts/U1/items với { P1001, 2 }.

* Kết quả mong đợi:
  + Mục P1001 được cộng dồn thành qty = 3 nếu tổng ≤ tồn kho.
  + Bảng tổng hợp cập nhật chính xác, khuyến mãi (nếu bậc thang số lượng) được áp lại.

**Test case 5 — Kiểm thử quyền truy cập giỏ**

* Dữ liệu đầu vào:
  + userId = U2 cố gắng thêm vào giỏ của U1.
* Bước thực hiện:

POST /carts/U1/items với token của U2.

* Kết quả mong đợi:
  + HTTP 403 (Forbidden). Không thay đổi dữ liệu.