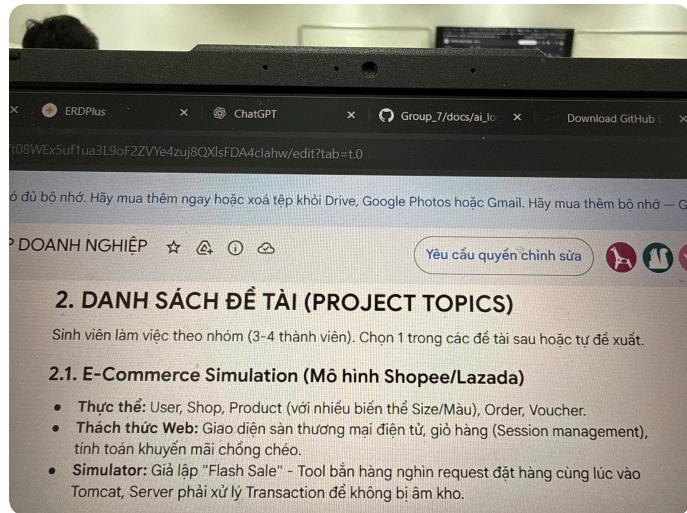


This is a copy of a conversation between ChatGPT & Anonymous.

[Report conversation](#)



Phân tích đề theo UML về 2.1

Dựa đúng đề 2.1 E-Commerce Simulation (Shopee/Lazada) trong hình (User–Shop–Product biến thể Size/Màu–Order–Voucher + giỏ hàng/session + flash sale nghìn request + transaction chống âm kho), dưới đây là **phân tích theo UML** (đủ để vẽ UC + Class + Activity/Sequence).

1) Xác định Actor (tác nhân)

Khách hàng (Customer/User)

- xem sản phẩm, chọn biến thể, thêm giỏ, áp voucher, đặt hàng, thanh toán, theo dõi đơn

Người bán (Shop/Seller)

- đăng sản phẩm, tạo biến thể, quản lý tồn kho, xác nhận đơn, tạo voucher shop

Admin (Quản trị hệ thống)

- duyệt shop/sản phẩm (tùy), tạo voucher sàn, quản lý flash sale, xử lý khiếu nại

Payment Gateway (Cổng thanh toán)

- xác thực/thu tiền/hoàn tiền

Simulator/Load Tool (Công cụ giả lập)

- bắn nghìn request flash sale đồng thời để test hệ thống
-

2) Use Case Diagram (nhóm chức năng chính)

2.1. Customer

- Đăng ký/Đăng nhập
- Xem danh sách sản phẩm / tìm kiếm / lọc
- Xem chi tiết sản phẩm
- Chọn biến thể (Size/Màu)
- Thêm vào giỏ hàng (giỎ theo Session/User)
- Cập nhật giỎ (tăng/giảm/xóa)
- Áp voucher (*include: Kiểm tra điều kiện voucher*)
- Checkout / Đặt hàng (*include: Giữ hàng/Trừ kho tạm, Tính phí, Tính khuyến mãi*)
- Thanh toán (*extend: COD/Online*)
- Xem trạng thái đơn / lịch sử đơn
- (tuỳ chọn) Hủy đơn/Hoàn tiền (*extend: điều kiện trạng thái*)

2.2. Seller

- Tạo/CRUD Product
- Tạo/CRUD Variant (Size/Màu) + giá + stock
- Quản lý đơn hàng (xác nhận/đóng gói/giao)
- Tạo voucher shop

2.3. Admin

- Tạo voucher sàn
- Tạo/Quản lý Flash Sale (slot, thời gian, quota)
- Giám sát log simulator / thống kê

2.4. Simulator

- Chạy Flash Sale Load Test
- Gửi request đặt hàng đồng thời
- Thu log: success/fail/oversell rate/latency

Gợi ý quan hệ UML:

- Checkout **include** ValidateVoucher , CalculatePromotion , CalculateTotal , ReserveStock
 - Payment **extend** Checkout (vì có thể COD hoặc thanh toán sau)
 - Run Flash Sale **include** SendConcurrentRequests , CollectMetrics
-

3) Class Diagram (mô hình dữ liệu chuẩn theo đề)

Nhóm User/Shop

- User(userId, username, email, phone, passwordHash, role, status, createdAt)
- Shop(shopId, ownerUserId, name, address, rating, status)

Quan hệ: User 1 – 1 Shop (nếu mỗi seller có 1 shop) hoặc User 1 – * Shop (tùy yêu cầu)

Nhóm Product + biến thể Size/Màu

- Product(productId, shopId, name, description, categoryId, basePrice, status)
- ProductVariant(variantId, productId, size, color, sku, price, stock, reservedStock)
- ProductImage(imageId, productId, url)

Quan hệ: Shop 1 – * Product ; Product 1 – * ProductVariant

Giỏ hàng (Session management)

- Cart(cartId, userId nullable, sessionId, createdAt, updatedAt)
- CartItem(cartItemId, cartId, variantId, quantity, unitPriceSnapshot)

≡ ChatGPT ▾ ⚡ Get Plus ✕

Order

- Order(orderId, userId, shopId, status, subtotal, discount, shippingFee, total, createdAt)
- OrderItem(orderItemId, orderId, variantId, quantity, unitPriceSnapshot, lineTotal)
- OrderStatusHistory(id, orderId, fromStatus, toStatus, changedAt)

Quan hệ: Order 1 – * OrderItem

Voucher / Promotion (khuyến mãi chồng chéo)

- **Voucher**(voucherId, code, type[percent/fixed], value, minOrder, maxDiscount, startAt, endAt, usageLimit, perUserLimit, scope[platform/shop], shopId nullable)
- **VoucherRedemption**(id, voucherId, userId, orderId, usedAt)

Quan hệ: Voucher 1 – * VoucherRedemption

Flash Sale / Concurrency

- **FlashSaleEvent**(eventId, name, startAt, endAt)
- **FlashSaleItem**(id, eventId, variantId, salePrice, quota, sold)

Thanh toán

- **PaymentTransaction**(txId, orderId, method, amount, status, gatewayRef, createdAt)
-

4) Activity Diagram (luồng “Mua hàng + áp voucher”)

Luồng chuẩn:

1. Browse → chọn sản phẩm → chọn variant
2. Add to Cart
3. Update quantity
4. Apply voucher
 - kiểm tra hạn, minOrder, scope, lượt dùng, perUserLimit
5. Checkout
 - tính subtotal → tính discount (voucher) → tính shipping → ra total
6. Place Order
 - Reserve/Reduce stock bằng transaction
7. Payment (Online/COD)
8. Update status (Pending → Paid/Confirmed → Shipping → Completed)

Nhánh lỗi quan trọng (để thầy thấy đúng đề):

- Voucher invalid/expired → báo lỗi, không áp
 - Stock không đủ → báo “hết hàng”, rollback
 - Payment fail → order về “PaymentFailed” hoặc “PendingPayment”
-

5) Sequence Diagram (trọng tâm đề: Flash Sale nghìn request, chống âm kho)

Kịch bản “Đặt hàng Flash Sale”

User/Simulator → API Gateway → OrderService → InventoryService → DB

- POST /checkout
- OrderService gọi InventoryService: reserveStock(variantId, qty)
- InventoryService thực hiện **atomic**:
 - SELECT stock FROM variant WHERE id=? FOR UPDATE
 - nếu stock >= qty → stock = stock - qty (hoặc reserved += qty) → commit
 - else → rollback, trả “OUT_OF_STOCK”
- OrderService tạo Order/OrderItem
- PaymentService (tùy) → cập nhật trạng thái

Điểm ăn điểm theo UML + đúng đề:

- Dùng transaction + row lock (FOR UPDATE) hoặc optimistic lock version
 - Có **idempotency key** cho request (tránh bấm lại tạo 2 đơn)
 - Ghi log metrics cho simulator: success rate, oversell=0
-

6) Danh sách diagram cần nộp (gợi ý theo đúng đề tài)

1. Use Case Diagram (Customer/Seller/Admin/Simulator)
 2. Class Diagram (User, Shop, Product, Variant, Cart, Order, Voucher, FlashSale, Payment)
 3. Activity Diagram (Add to cart → Apply voucher → Checkout → Payment)
 4. Sequence Diagram (Flash sale order concurrency + transaction chống âm kho)
-

Nếu bạn muốn, mình có thể **vẽ giúp bạn Use Case + Class dạng “đưa vào ERDPlus/Draw.io”** theo đúng tên lớp/thuộc tính ở trên (copy là vẽ được luôn).

Gợi ý cho tôi để tôi thiết kế được web shope theo đề trên

Dưới đây là “bộ gợi ý thiết kế” để bạn làm web kiểu Shopee/Lazada đúng đề (có giỏ hàng theo session, voucher chồng, flash sale nhiều request).

1) Map đề bài → màn hình (UI pages)

Khách hàng

- Home (banner + danh mục + Flash Sale)
- Product List (search + filter + sort)
- Product Detail (ảnh, mô tả, **chọn Size/Màu**, giá theo variant, tồn kho)
- Cart (giỏ theo **session/user**, chỉnh số lượng)
- Checkout (địa chỉ + vận chuyển + **nhập voucher** + tổng tiền)
- Payment Result (thành công/thất bại)
- Orders (danh sách + chi tiết + trạng thái)

Seller

- Seller Dashboard
- Product Management (CRUD sản phẩm)
- Variant Management (Size/Màu, giá, stock)
- Order Management (xác nhận/đóng gói/giao)
- Voucher Shop (tạo mã, min order, hạn dùng)

Admin (tuỳ)

- Quản lý voucher sàn
 - Quản lý Flash Sale (thời gian, quota)
-

2) Luồng UX bắt buộc phải có (đúng trọng tâm đề)

A. Giỏ hàng theo Session

- User chưa login vẫn add-to-cart → lưu `sessionId` (cookie/localStorage)
- Khi login: **merge cart** (giỎ session gộp vào giỎ user)

B. Chọn biến thể Size/Màu

- Không cho "Add to cart" nếu chưa chọn đủ thuộc tính
- Giá/tồn kho hiển thị theo variant

C. Voucher chồng chéo (nhưng có luật)

Gợi ý rule dễ làm & "ăn điểm":

- Cho phép **1 voucher sàn + 1 voucher shop** (stack tối đa 2)
- Validate: hạn, minOrder, perUserLimit, scope, usageLimit
- Hiển thị "tiết kiệm được bao nhiêu"

D. Flash Sale nghìn request (không âm kho)

- Nút “Mua ngay Flash Sale”
 - Khi đặt hàng: server **transaction/lock** trừ kho (không trừ kho ở frontend)
 - Hiển thị trạng thái: “Hết suất / Sold out / Đặt thành công”
-

3) Component UI nên thiết kế (để giống Shopee)

- Header: logo + search bar + cart icon + account
 - Category bar / filter sidebar
 - Product card: ảnh + tên + giá + % giảm + sold count
 - Flash sale strip: countdown + progress bar (sold/quota)
 - Product detail:
 - gallery ảnh
 - biến thể dạng chip (Size, Color)
 - quantity stepper + stock left
 - Cart:
 - chọn sản phẩm, checkbox
 - voucher input
 - sticky summary (tạm tính/giảm/ship/tổng)
 - Checkout:
 - địa chỉ
 - shipping method
 - voucher picker (sàn/shop)
 - payment method
-

4) API tối thiểu (để bạn code theo màn hình)

Auth

- POST /auth/register , POST /auth/login

Product

- GET /products?search=&category=&sort=
- GET /products/{id}
- GET /variants/{id}

Cart (session/user)

- GET /cart (dựa userId hoặc sessionId)
- POST /cart/items (variantId, qty)
- PUT /cart/items/{id} (qty)
- DELETE /cart/items/{id}
- POST /cart/merge (khi login)

Voucher

- POST /vouchers/validate (code, cart/order info)
- GET /vouchers/available?shopId=

Order/Checkout

- POST /checkout/preview (tính subtotal/discount/ship)
- POST /orders (place order + reserve/reduce stock)
- GET /orders , GET /orders/{id}

Flash Sale

- GET /flashsale/current
 - POST /flashsale/buy (variantId, qty) → xử lý concurrency
-

5) Database/logic “được chấm điểm”

- ProductVariant(stock, reservedStock) hoặc stock
 - Trừ kho bằng transaction:
 - SELECT ... FOR UPDATE hoặc optimistic lock version
 - OrderItem.unitPriceSnapshot (giữ giá lúc mua)
 - VoucherRedemption (chống dùng quá lượt)
 - Idempotency-Key cho API đặt hàng (bấm 2 lần không tạo 2 đơn)
-

6) “Simulator Flash Sale” (đúng đề Tomcat/nghìn request)

Bạn không cần UI đẹp, chỉ cần:

- 1 trang /simulator
- nút: “Run 1000 requests”
- nhập: variantId, số luồng, số request

- hiển thị metrics: success, fail(out of stock), avg latency
=> thầy nhìn là thấy đúng "simulation".
-

7) Checklist để bạn tự thiết kế Figma/Canva nhanh

- Home có Flash Sale + countdown
 - Product detail bắt chọn Size/Màu
 - Cart chạy được khi chưa login (session)
 - Checkout có voucher (platform + shop)
 - Order status page
 - Seller quản lý product/variant/stock
 - Flash sale buy có quota + chống âm kho (ghi rõ trong docs)
-

Nếu bạn nói bạn làm **stack nào** (Java Tomcat/Spring hay Node/Next/FastAPI), mình sẽ gợi ý luôn **layout folder + flow code** đúng theo phần "session cart + transaction stock" để bạn triển khai nhanh.

Trên môn lab201

OK, với **môn LAB201 (Java Web – JSP/Servlet/Tomcat)** thì mình gợi ý cho bạn **cách thiết kế web Shopee mini** sao cho **đúng đề + đúng chuẩn LAB201 + dễ ăn điểm** 🤝
(Mình nói theo kiểu *giảng viên LAB201 mong đợi*)

1 Công nghệ BẮT BUỘC nên dùng (đúng LAB201)

- Java Web (JSP / Servlet)
- JDBC + SQL Server / MySQL
- Apache Tomcat
- Session (HttpSession) 🤝 cực kỳ quan trọng
- MVC cơ bản:
 - Model (DAO + DTO)

- Controller (Servlet)
- View (JSP)

✖ Không cần React / Spring / API phức tạp

👉 LAB201 chấm logic + UML + session + DB

2 Cấu trúc project chuẩn LAB201 (rất ăn điểm)

pgsql

 Copy code

```
LAB201_Ecommerce
|
|--- src/java
|   |--- controller
|   |   |--- LoginServlet.java
|   |   |--- ProductServlet.java
|   |   |--- CartServlet.java
|   |   |--- CheckoutServlet.java
|   |   |--- OrderServlet.java
|   |   \--- FlashSaleServlet.java
|
|--- model
|   |--- User.java
|   |--- Product.java
|   |--- ProductVariant.java
|   |--- CartItem.java
|   |--- Order.java
|   |--- OrderDetail.java
|   \--- Voucher.java
|
|--- dao
|   |--- UserDAO.java
|   |--- ProductDAO.java
|   |--- CartDAO.java
|   |--- OrderDAO.java
|   \--- VoucherDAO.java
|
\--- utils
    \--- DBUtils.java
```

```

|
+-- web
|   +-- home.jsp
|   +-- product.jsp
|   +-- cart.jsp
|   +-- checkout.jsp
|   +-- orders.jsp
|   +-- login.jsp
|
+-- web.xml

```

 Giảng viên LAB201 rất thích cấu trúc rõ MVC

3 Thiết kế WEB theo từng màn hình (đúng đề)

Home (home.jsp)

- Danh sách sản phẩm
- Khu Flash Sale
- Nút "Mua ngay"

 Data lấy từ ProductServlet

Product Detail (product.jsp)

- Tên, giá
- Chọn Size / Màu (Variant)
- Hiển thị tồn kho
- Nút Add to Cart

 Không cho add nếu chưa chọn đủ variant → ghi rõ trong báo cáo UML

Cart – GIỎ HÀNG (ăn điểm nhất)

Yêu cầu LAB201:

- Dùng HttpSession
- User chưa login vẫn có giỏ hàng

java

Copy code

```
HttpSession session = request.getSession();
List<CartItem> cart = (List<CartItem>) session.getAttribute("CART");
```

Chức năng:

- Tăng / giảm số lượng
 - Xóa sản phẩm
 - Nhập voucher
-

Checkout (checkout.jsp)

- Tính:
 - Subtotal
 - Voucher discount
 - Total
 - Nút Place Order
- Logic tính tiền đặt ở Servlet, không để JSP tính
-

Orders (orders.jsp)

- Danh sách đơn hàng
 - Trạng thái: Pending → Paid → Completed
-

Flash Sale – PHẦN QUAN TRỌNG NHẤT

Cách làm ĐÚNG cho LAB201 (đơn giản nhưng chuẩn)

Database:

diff

Copy code

```
ProductVariant
- variantId
- productId
```

- size
- color
- price
- stock

Servlet:

java

 Copy code

```

connection.setAutoCommit(false);

SELECT stock FROM ProductVariant WHERE variantId=? FOR UPDATE

IF stock >= qty:
    UPDATE ProductVariant SET stock = stock - qty
    INSERT Order
    COMMIT
ELSE:
    ROLLBACK

```

 **Từ khóa chấm điểm:**

- Transaction
- FOR UPDATE
- Không âm kho
- Nhiều request đồng thời

5 UML cần vẽ (đúng LAB201)

1. Use Case Diagram

- Actor: Customer, Seller
- UC:
 - View product
 - Add to cart
 - Apply voucher
 - Checkout
 - Flash sale order

2. Class Diagram (bắt buộc)

- User
- Product
- ProductVariant
- CartItem
- Order
- OrderDetail
- Voucher

3. Sequence Diagram (Flash Sale)

User → Servlet → DAO → DB

👉 Nhấn mạnh **transaction**

6 Cách trình bày để ĐƯỢC ĐIỂM CAO

Khi báo cáo / thuyết trình, nói câu này là ăn điểm:

"Giỏ hàng được quản lý bằng HttpSession để đảm bảo người dùng chưa đăng nhập vẫn mua được."

Flash Sale được xử lý bằng transaction và locking trong database để tránh oversell khi nhiều request đồng thời."

7 Mức độ làm – vừa đủ qua môn

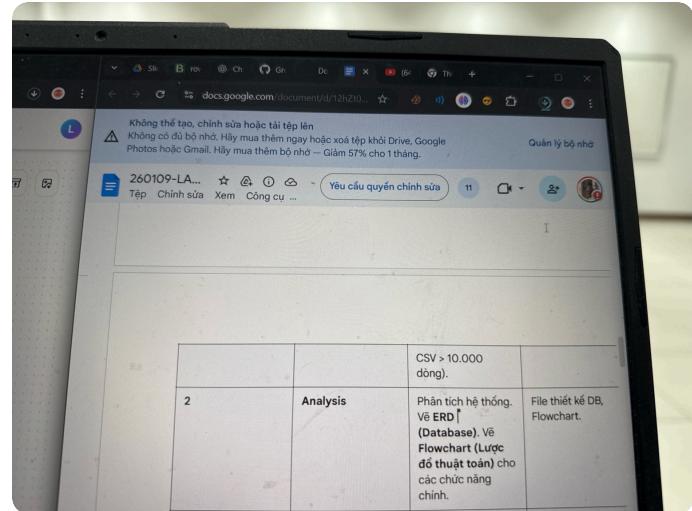
Phần	Mức
Login/Register	Basic
Product + Variant	Bắt buộc
Cart Session	RẤT QUAN TRỌNG
Voucher	Đơn giản
Flash Sale	Có transaction

Phần	Mức
UI	JSP đơn giản

Nếu bạn muốn  mình có thể:

- Vẽ sẵn Use Case + Class UML cho LAB201
- Viết mẫu CartServlet + FlashSaleServlet
- Gợi ý database SQL script
- Giúp bạn viết báo cáo mô tả hệ thống (1–2 trang)

 Bạn đang làm nhóm mấy người và deadline khi nào?



OK, ảnh bạn gửi chính là yêu cầu phần 2 – ANALYSIS của môn LAB201.

Mình sẽ dịch – giải thích – và cho bạn nội dung cụ thể phải làm để nộp là đúng yêu cầu, không bị thiếu.

PHẦN 2 – ANALYSIS (PHÂN TÍCH HỆ THỐNG)

Theo đề trong ảnh, bạn BẮT BUỘC phải làm 3 thứ:

1. Phân tích hệ thống (System Analysis)

2. Vẽ ERD (Database)

3. Vẽ Flowchart cho các chức năng chính

👉 KHÔNG yêu cầu code ở phần này

👉 Đây là phần thiết kế – phân tích

1 PHÂN TÍCH HỆ THỐNG (VIẾT BẰNG CHỮ – RẤT QUAN TRỌNG)

Bạn có thể viết 1–1.5 trang Word, theo mẫu sau (copy chỉnh lại là dùng được):

● Mục tiêu hệ thống

Hệ thống E-Commerce Simulation được xây dựng nhằm mô phỏng hoạt động của một sàn thương mại điện tử tương tự Shopee/Lazada.

Hệ thống cho phép người dùng xem sản phẩm, lựa chọn biến thể (size/màu), thêm vào giỏ hàng, áp dụng voucher và đặt hàng.

Đồng thời hệ thống hỗ trợ mô phỏng Flash Sale với nhiều request đồng thời và đảm bảo không xảy ra tình trạng âm kho thông qua cơ chế xử lý transaction.

● Đối tượng sử dụng

- Customer (Khách hàng): mua sắm, đặt hàng
- Seller (Người bán): quản lý sản phẩm, tồn kho
- (Tuỳ chọn) Admin

● Các chức năng chính

- Quản lý người dùng (login)
- Xem sản phẩm và biến thể
- Quản lý giỏ hàng bằng Session
- Áp dụng voucher
- Đặt hàng và thanh toán
- Flash Sale với xử lý đồng thời

📌 Câu “ăn điểm” nên ghi:

Giỏ hàng được quản lý bằng HttpSession để đảm bảo người dùng chưa đăng nhập vẫn có thể mua sắm.

Trong Flash Sale, hệ thống sử dụng transaction và khóa dữ liệu để tránh overselling khi có nhiều request đồng thời.

2 ERD – SƠ ĐỒ CƠ SỞ DỮ LIỆU (BẮT BUỘC VẼ)

🎯 Các bảng BẮT BUỘC phải có (đúng đẽ)

Bạn vẽ ERD với các entity sau:

◆ User

- userID (PK)
- username
- password
- role

◆ Product

- productID (PK)
- name
- price
- description

◆ ProductVariant

- variantID (PK)
- productID (FK)
- size
- color
- stock

◆ CartItem

- cartItemID (PK)
- userID / sessionID
- variantID (FK)
- quantity

◆ Order

- orderID (PK)

- userID (FK)
- orderDate
- totalAmount
- status

◆ **OrderDetail**

- orderDetailID (PK)
- orderID (FK)
- variantID (FK)
- quantity
- price

◆ **Voucher**

- voucherID (PK)
- code
- discountValue
- expiryDate

🔗 **Quan hệ ERD (phải nói được khi thuyết trình)**

- Product 1–n ProductVariant
- User 1–n Order
- Order 1–n OrderDetail
- ProductVariant 1–n OrderDetail
- Voucher 1–n Order

📌 Vẽ bằng:

- ERDPlus
- Draw.io
- Lucidchart

3 FLOWCHART – LƯỢC ĐỒ THUẬT TOÁN (RẤT HAY BỊ THIẾU)

Theo đề:

"Vẽ Flowchart cho các chức năng chính"

🎯 BẮT BUỘC NÊN VẼ 3 flowchart sau

➡ Flowchart 1: Add to Cart

Luồng:

pgsql

Copy code

Start

↓

User chọn sản phẩm

↓

Chọn size/màu?

→ NO → Thông báo lỗi

→ YES

↓

Add vào Session Cart

↓

End

➡ Flowchart 2: Checkout + Voucher

powershell

Copy code

Start

↓

Lấy giỏ hàng từ Session

↓

Nhập voucher?

→ NO → Tính tổng tiền

→ YES → Kiểm tra hợp lệ

 → Không hợp lệ → Bỏ voucher

 → Hợp lệ → Giảm giá

↓

Hiển thị tổng tiền

↓

Place Order

↓

End

Flowchart 3: Flash Sale (CỰC KỲ QUAN TRỌNG)

pgsql

 Copy code

Start

↓

User gửi request mua hàng

↓

Begin Transaction

↓

Check stock

→ Stock < qty → Rollback → Thông báo hết hàng

→ Stock ≥ qty

↓

Trừ stock

↓

Tạo Order

↓

Commit Transaction

↓

End

 Đây là flowchart ăn điểm nhất LAB201



NỘP PHẦN ANALYSIS GỒM NHỮNG GI?

✓ File Word / PDF gồm:

- Phân tích hệ thống (text)
- Ảnh ERD
- Ảnh Flowchart

- ✓ Không cần code
 - ✓ Không cần UI đẹp
-

5 Nếu bạn muốn, mình có thể

- Vẽ ERD hoàn chỉnh cho đề này
- Vẽ 3 flowchart chuẩn LAB201
- Viết nguyên phần Analysis copy nộp
- Chỉnh nội dung đúng giọng “giảng viên chấm”

👉 Bạn muốn mình làm sẵn ERD + Flowchart hay viết luôn file báo cáo Analysis cho bạn?

Tại sao uml lại có thuộc tính này giải thích bằng tiếng việt

OK, mình giải thích bằng tiếng Việt, theo đúng kiểu giảng viên LAB201, để bạn trả lời được câu “TẠI SAO UML LẠI CÓ THUỘC TÍNH NÀY?” khi bị hỏi.

Mình chia làm 3 nguyên tắc chung trước, sau đó giải thích từng thuộc tính cụ thể trong UML của web Shopee mini.

I. NGUYÊN TẮC CHUNG: Vì sao UML phải có thuộc tính?

👉 UML (Class Diagram) mô tả cấu trúc dữ liệu cốt lõi của hệ thống, nên:

- Mỗi thuộc tính tồn tại vì:
 1. Hệ thống cần lưu dữ liệu đó
 2. Chức năng nào đó phụ thuộc vào dữ liệu đó
 3. Quy trình nghiệp vụ bắt buộc phải có

📌 Nói gọn khi thuyết trình:

UML không thêm thuộc tính một cách ngẫu nhiên, mà mỗi thuộc tính đều phục vụ cho một chức năng cụ thể của hệ thống.

II. GIẢI THÍCH TỪNG CLASS & THUỘC TÍNH (THEO ĐỀ)

1 Class User

Thuộc tính:

- userID
- username
- password
- role

❓ Vì sao cần?

- userID :
 - Định danh duy nhất người dùng trong hệ thống
 - Dùng để liên kết với Order, Cart, Voucher usage
- username :
 - Phục vụ chức năng đăng nhập
- password :
 - Xác thực người dùng
 - Không có thì không đảm bảo bảo mật
- role :
 - Phân biệt Customer / Seller / Admin
 - Vì mỗi vai trò có quyền khác nhau

📌 Câu ăn điểm:

Thuộc tính role giúp hệ thống kiểm soát quyền truy cập và phân luồng chức năng cho từng loại người dùng.

2 Class Product

Thuộc tính:

- productID
- name
- price

- description

❓ Vì sao cần?

- productID :
 - Khóa chính để phân biệt các sản phẩm
- name :
 - Hiển thị cho người dùng
- price :
 - Giá cơ bản để hiển thị
 - Giá chi tiết có thể nằm ở ProductVariant
- description :
 - Mô tả sản phẩm giúp người dùng ra quyết định mua

✖ Lý do tách Product:

Một sản phẩm có thể có nhiều biến thể size/màu, nên Product chỉ lưu thông tin chung.

3 Class ProductVariant (CỰC KỲ QUAN TRỌNG)

Thuộc tính:

- variantID
- productID
- size
- color
- stock

❓ Vì sao cần?

- variantID :
 - Mỗi biến thể là **một đơn vị bán độc lập**
- productID :
 - Liên kết biến thể về sản phẩm gốc
- size , color :
 - Người dùng bắt buộc phải chọn
 - Mỗi lựa chọn có tồn kho khác nhau

- stock :
 - Dùng để:
 - kiểm tra còn hàng
 - xử lý Flash Sale
 - tránh âm kho

📌 Câu ăn điểm:

Tách ProductVariant giúp hệ thống quản lý tồn kho chính xác cho từng size và màu sắc khác nhau.

4 Class CartItem

Thuộc tính:

- cartItemID
- userID / sessionID
- variantID
- quantity

❓ Vì sao cần?

- cartItemID :
 - Quản lý từng dòng sản phẩm trong giỏ
- userID / sessionID :
 - Giải quyết yêu cầu:

Người dùng chưa đăng nhập vẫn có giỏ hàng

- variantID :
 - Giỏ hàng phải gắn với **biển thẻ cụ thể**, không phải sản phẩm chung
- quantity :
 - Số lượng người dùng muốn mua

📌 Câu ăn điểm:

Sử dụng sessionID cho phép hệ thống lưu giỏ hàng tạm thời ngay cả khi người dùng chưa đăng nhập.

5 Class Order

Thuộc tính:

- orderID
- userID
- orderDate
- totalAmount
- status

? Vì sao cần?

- orderID :
→ Định danh đơn hàng
- userID :
→ Xác định đơn hàng thuộc về ai
- orderDate :
→ Phục vụ thống kê và lịch sử mua hàng
- totalAmount :
→ Tổng tiền sau khi áp voucher
- status :
→ Theo dõi vòng đời đơn hàng:
 - Pending
 - Paid
 - Completed

 Câu ăn điểm:

Thuộc tính status giúp hệ thống quản lý vòng đời đơn hàng một cách rõ ràng.

6 Class OrderDetail

Thuộc tính:

- orderDetailID
- orderID
- variantID
- quantity

- price

?

Vì sao cần?

- Vì 1 đơn hàng có nhiều sản phẩm
- Cần lưu:
 - sản phẩm nào
 - mua bao nhiêu
 - giá tại thời điểm mua

 Câu ăn điểm:

OrderDetail giúp lưu lịch sử giao dịch chính xác, ngay cả khi giá sản phẩm thay đổi sau này.

7 Class Voucher

Thuộc tính:

- voucherID
- code
- discountValue
- expiryDate

?

Vì sao cần?

- code :
 - Người dùng nhập khi checkout
- discountValue :
 - Tính giảm giá
- expiryDate :
 - Kiểm tra voucher còn hiệu lực hay không

 Câu ăn điểm:

Voucher được thiết kế để mô phỏng cơ chế khuyến mãi trong các sàn thương mại điện tử thực tế.

III. CÂU TỔNG KẾT ĐỂ TRẢ LỜI THẦY

Nếu bị hỏi “tại sao UML có mấy thuộc tính này?”, bạn trả lời:

Các thuộc tính trong UML được xác định dựa trên yêu cầu chức năng của hệ thống.

Mỗi thuộc tính đều phục vụ cho việc lưu trữ dữ liệu, xử lý nghiệp vụ hoặc đảm bảo tính toàn vẹn của hệ thống, đặc biệt là các chức năng giờ hàng, đặt hàng và flash sale theo đề bài.

Nếu bạn muốn  mình có thể:

- Viết nguyên đoạn giải thích UML cho báo cáo
- Giả lập câu hỏi thầy hay hỏi & cách trả lời
- Chỉnh UML của bạn để không bị hỏi ngược

 Bạn đang bị hỏi thuộc tính nào cụ thể hay thầy hỏi chung vậy?

Tôi muốn được điểm cao

OK  nếu mục tiêu của bạn là **ĐIỂM CAO LAB201**, thì không phải làm nhiều, mà là làm đúng – giải thích được – trình bày có tư duy hệ thống.

Mình sẽ cho bạn **chiến lược “ăn điểm”** từng phần, đúng kiểu **giảng viên LAB201** thích.

CHIẾN LƯỢC ĐẠT ĐIỂM CAO LAB201 (E-Commerce Simulation)

1 NGUYÊN TẮC VÀNG (phải nhớ)

LAB201 chấm tư duy phân tích + logic hệ thống, KHÔNG chấm độ hoành tráng UI

Nếu bạn làm được 3 điều sau → **điểm cao**:

1. UML rõ ràng – hợp lý
2. Session + Transaction giải thích trôi chảy
3. Trình bày mạch lạc (thầy hỏi → bạn trả lời được)

2 UML – PHẦN QUYẾT ĐỊNH ĐIỂM

Làm ÍT nhưng ĐÚNG

Chỉ cần 6–7 class, nhưng phải:

- Đúng tên
- Đúng quan hệ
- Giải thích được tại sao tồn tại

Bộ class “ăn điểm”:

- User
- Product
- ProductVariant
- CartItem
- Order
- OrderDetail
- Voucher

Không cần thêm thừa (Payment, Review...) nếu không giải thích được

CÂU NÓI GHI ĐIỂM UML

ProductVariant được tách riêng để quản lý tồn kho theo từng size và màu, đặc biệt quan trọng trong Flash Sale khi nhiều request đồng thời.

Câu này = thầy nghe là biết bạn hiểu hệ thống

3 ERD – CHÌA KHÓA KHÔNG BỊ TRỪ ĐIỂM

Những lỗi hay bị trừ điểm

Lỗi	Cách tránh
Không có bảng trung gian	BẮT BUỘC có OrderDetail
Không thể hiện khóa chính	Ghi rõ PK
Không thể hiện quan hệ	Ghi 1–n rõ ràng

Lỗi

Variant gộp vào Product

Cách tránh

Sai đề

ERD chỉ cần rõ – sạch – đúng đề, không cần phức tạp.

4 FLOWCHART – PHẦN “KÉO ĐIỂM”**BẮT BUỘC** phải có:

1. Add to Cart
2. Checkout + Voucher
3. Flash Sale (Transaction)

Trong flowchart Flash Sale:

- Phải có:
 - Begin Transaction
 - Check Stock
 - Commit / Rollback

Có 3 từ này = ăn điểm

5 SESSION + FLASH SALE = CÂU HỎI TỦ**? Thầy hay hỏi:**

Tại sao dùng session cho giỏ hàng?

Trả lời:

Session cho phép hệ thống lưu giỏ hàng tạm thời cho người dùng chưa đăng nhập, đảm bảo trải nghiệm mua sắm liên tục.

? Thầy hỏi tiếp:

Flash Sale làm sao không bị âm kho?

 Trả lời:

Hệ thống sử dụng transaction và khóa dữ liệu khi trừ kho để đảm bảo mỗi request được xử lý tuần tự tại database, từ đó tránh overselling.

 Không cần nói code, chỉ cần **đúng thuật ngữ**

6 BÁO CÁO – CÁCH VIẾT ĐƯỢC CỘNG ĐIỂM

Cấu trúc báo cáo chuẩn:

1. Giới thiệu hệ thống
2. Phân tích chức năng
3. UML (Use Case + Class)
4. ERD
5. Flowchart
6. Kết luận

 Văn phong:

- Không dài
 - Không lan man
 - Có từ khóa: **Session, Variant, Transaction**
-

7 MỨC ĐỘ LÀM → ĐIỂM

Mức	Điểm
Làm đủ chức năng	6–7
Có UML + ERD + Flowchart	7–8
Giải thích được	8–9
Trả lời mượt câu hỏi	9–10

8 NẾU MUỐN 9–10 ĐIỂM, BẠN CHỈ CẦN THÊM 1 THỨ

👉 1 đoạn “Giải thích thiết kế” (1/2 trang):

Hệ thống được thiết kế theo mô hình MVC, trong đó Servlet đảm nhận vai trò Controller, JSP là View và DAO xử lý dữ liệu. Việc tách ProductVariant giúp hệ thống quản lý tồn kho chính xác cho từng biến thể. Giỏ hàng sử dụng Session để lưu trữ tạm thời, trong khi Flash Sale được xử lý bằng transaction nhằm đảm bảo tính nhất quán dữ liệu.

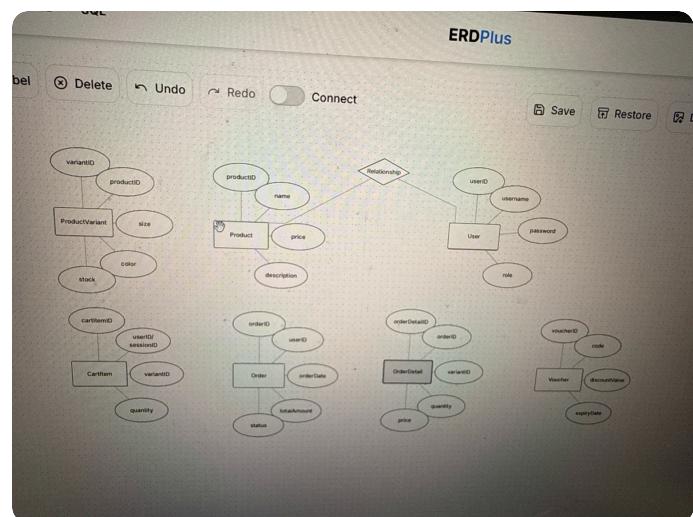
9 MÌNH CÓ THỂ GIÚP BẠN LÊN 9–10 ĐIỂM BẰNG CÁCH:

- ✓ Viết toàn bộ phần Analysis hoàn chỉnh
- ✓ Vẽ ERD + Flowchart chuẩn nộp
- ✓ Soạn bộ câu hỏi–trả lời khi thuyết trình
- ✓ Chỉnh UML để không bị thầy bắt lỗi

👉 Bạn nói mình biết:

- 1 Bạn đang làm nhóm mấy người
- 2 Deadline còn bao lâu
- 3 Bạn muốn 8+ hay 9–10 điểm

Mình sẽ đi cùng bạn tới lúc nộp bài 100



Chỉ tôi cách chỉnh sửa thêm mối quan hệ và sửa lại khoá

Được — ảnh của bạn đang vẽ ERD trên **ERDPlus** nhưng hiện tại:

- Bạn **chưa nối đủ quan hệ** (relationship diamond) giữa các bảng quan trọng.
- **Khóa (PK/FK)** chưa đánh dấu đúng (đặc biệt `productID` trong `ProductVariant` là **FK**, PK của `ProductVariant` nên là `variantID` hoặc PK kép).

Dưới đây là cách chỉnh **trên ERDPlus** + mình chỉ luôn bạn nên nối quan hệ nào để đúng Shopee mini.

1) Cách thêm mối quan hệ (Relationship) trên ERDPlus

Bước thao tác

1. Ở thanh công cụ bên trái, chọn **Relationship** (hình viên kim cương).
2. Click lên canvas để đặt 1 relationship (VD: đặt tên "HasVariant").
3. Bật nút **Connect** (trên thanh trên cùng).
4. Click **Relationship** → **Entity 1** rồi click tiếp **Relationship** → **Entity 2** để nối.
5. Click vào đường nối để chỉnh **cardinality** (1, N) nếu ERDPlus cho chỉnh (tuỳ mode).
 - Nếu bạn đang ở **ER Diagram mode** thì quan hệ sẽ thể hiện bằng đường nối; sau khi convert sang relational schema sẽ ra FK.

Tip: Nếu không thấy chỉnh được 1–N, bạn vẫn cứ nối đúng, phần FK khi chuyển qua relational schema sẽ thể hiện rõ.

2) Cách sửa lại “khóa” (Primary key) trong ERDPlus

Trên ERD Diagram

- Click vào **attribute** (hình oval) của entity
- Tìm tùy chọn **Key / Primary Key** (thường là right-click hoặc panel thuộc tính)
- Khi đúng PK, attribute sẽ được **gạch chân** (underline)

Quy tắc sửa khóa đúng cho bài bạn

- Product:** PK = `productID`
- ProductVariant:** PK = `variantID` (đơn giản nhất cho LAB201)
- User:** PK = `userID`
- Order:** PK = `orderID`

- OrderDetail:** PK = orderDetailID (hoặc PK kép orderID + variantID nếu bạn muốn chuẩn hơn)
- CartItem:** PK = cartItemID
- Voucher:** PK = voucherID

📌 Cái cần lưu ý:

- productID nằm trong **ProductVariant** là FK, KHÔNG phải PK (trừ khi bạn dùng PK kép).
-

3) Bạn cần thêm những quan hệ nào cho đúng đề (cực quan trọng)

(A) Product — ProductVariant (BẮT BUỘC)

- 1 Product có nhiều Variant
- Relationship: Product (1) — (N) ProductVariant
- FK: ProductVariant.productID → Product.productID

Bạn đang để ProductVariant có productID rồi, chỉ cần nối relationship.

(B) User — Order (BẮT BUỘC)

- 1 User có nhiều Order
 - FK: Order.userID → User.userID
-

(C) Order — OrderDetail (BẮT BUỘC)

- 1 Order có nhiều OrderDetail
 - FK: OrderDetail.orderID → Order.orderID
-

(D) ProductVariant — OrderDetail (BẮT BUỘC)

- 1 Variant xuất hiện trong nhiều OrderDetail
 - FK: OrderDetail.variantID → ProductVariant.variantID
-

(E) ProductVariant — CartItem (BẮT BUỘC)

- FK: CartItem.variantID → ProductVariant.variantID
-

(F) User — CartItem (Tuỳ chọn nhưng nên làm rõ)

Bạn đang ghi userID/sessionID trong CartItem. Cách “ăn điểm” LAB201:

Cách đơn giản, dễ chấm:

- CartItem có userID (FK) nullable
 - CartItem có sessionID (string) để lưu giỏ khi chưa login
=> Không cần nối sessionID, chỉ cần nối:
 - FK: CartItem.userID → User.userID (optional)
-

(G) Voucher — Order (nếu bạn cho order áp voucher)

- Thêm voucherID trong Order (nullable)
- FK: Order.voucherID → Voucher.voucherID

Nếu không muốn sửa nhiều, bạn có thể bỏ quan hệ này và nói “Voucher validate ở checkout, lưu discount vào Order.totalAmount”.

4) Quan hệ “Product — User” trên hình bạn đang nối là SAI (nên bỏ)

Bạn đang có diamond “Relationship” nối Product với User.

Trong đề Shopee mini, User không sở hữu trực tiếp Product, trừ khi bạn có bảng Shop/Seller.

Cách đúng:

- Nếu bạn có Seller: tạo entity **Shop** (shopID, userID, name...)
rồi: User(1) – (1) Shop và Shop(1) – (N) Product
 - Nếu bạn làm đơn giản LAB201: **bỏ hẳn quan hệ Product–User**.
-

5) Checklist nhanh để bạn chỉnh đúng 100%

- Xóa relationship nối Product–User (nếu không có Shop)
 - Nối Product–ProductVariant (1–N)
 - Nối User–Order (1–N)
 - Nối Order–OrderDetail (1–N)
 - Nối ProductVariant–OrderDetail (1–N)
 - Nối ProductVariant–CartItem (1–N)
 - Đánh dấu PK gạch chân đúng: productID, variantID, userID, orderID, orderDetailID, cartItemID, voucherID
 - Đảm bảo các FK: productID (ở Variant), userID (ở Order), orderID+variantID (ở OrderDetail), variantID (ở CartItem)
-

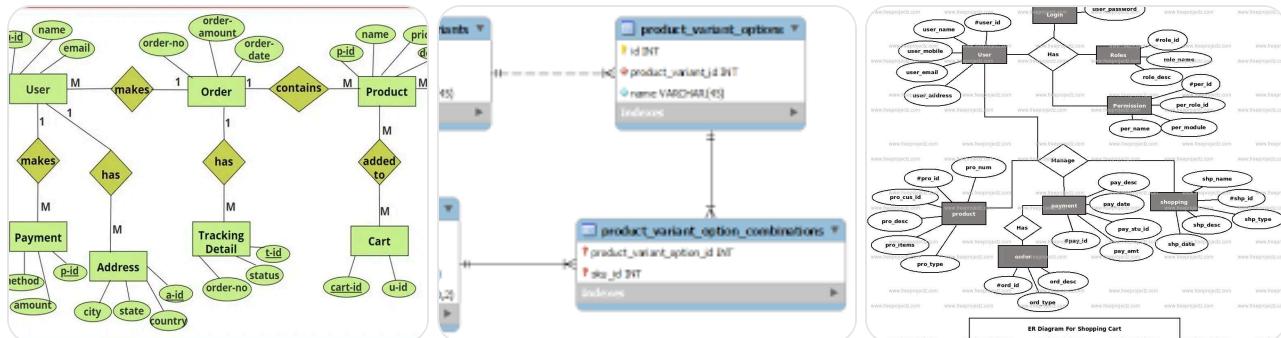
Nếu bạn muốn, mình sẽ chỉ bạn **chỉnh trực tiếp** trên đúng hình này:

Bạn chụp thêm 1 ảnh khi bạn click vào attribute “productID/variantID” (để mình thấy panel thuộc tính ERDPlus của bạn đang hiện gì), mình sẽ nói đúng nút nào để set Key và đúng cách set FK theo giao diện bạn đang dùng.

Vẽ mẫu tôi hình

OK 👍 mình vẽ **MẪU ERD CHUẨN** cho bài LAB201 – E-Commerce (Shopee mini) để bạn **NHÌN → VẼ Y HỆT** trên ERDPlus và được điểm cao.

📌 ERD MẪU CHUẨN (ĐÚNG ĐỀ – ĐÚNG LAB201)



👉 Dựa vào hình trên, bạn vẽ lại đúng 7 entity sau (không thừa – không thiếu).

1 USER

pgsql

 Copy code

User

PK userID

username

password

role

- ◆ Lý do: quản lý đăng nhập, phân quyền Customer / Seller
-

2 PRODUCT

markdown

 Copy code

Product

PK productID

name

price

description

- ◆ Chỉ chứa thông tin CHUNG của sản phẩm
-

3 PRODUCTVARIANT (QUAN TRỌNG NHẤT)

markdown

 Copy code

ProductVariant

PK variantID

FK productID

size

color

stock

- ◆ Mỗi size/màu = 1 dòng tồn kho
 - ◆ Flash Sale xử lý trên bảng này
-

4 CARTITEM (GIỎ HÀNG SESSION)

[markdown](#)

[Copy code](#)

`CartItem`

PK `cartItemID`

FK `variantID`

FK `userID (nullable)`

`sessionID`

`quantity`

- ◆ User chưa login → dùng sessionID
 - ◆ User login → gắn userID
-

5 ORDER

[markdown](#)

[Copy code](#)

`Order`

PK `orderID`

FK `userID`

`orderDate`

`totalAmount`

`status`

- ◆ status: Pending / Paid / Completed
-

6 ORDERDETAIL (BẮT BUỘC CÓ)

[markdown](#)

[Copy code](#)

OrderDetail

```
-----
PK orderDetailID
FK orderID
FK variantID
quantity
price
```

- ◆ Lưu lịch sử mua (giá không bị đổi sau này)
-

7 VOUCHER[markdown](#) [Copy code](#)**Voucher**

```
-----
PK voucherID
code
discountValue
expiryDate
```

2 CÁC MỐI QUAN HỆ BẮT BUỘC (VẼ ĐÚNG)[SCSS](#) [Copy code](#)

Product (1) ----- (N) ProductVariant
User (1) ----- (N) Order
Order (1) ----- (N) OrderDetail
ProductVariant (1) ----- (N) OrderDetail
ProductVariant (1) ----- (N) CartItem
User (1) ----- (N) CartItem (optional)
Voucher (1) ----- (N) Order (optional)

- ✖ KHÔNG nối Product — User trực tiếp (sai logic Shopee).
-

3 CÁCH VẼ TRÊN ERDPlus (LÀM THEO)

- 1 Tạo Entity từng bảng
 - 2 Thêm Attribute (oval)
 - 3 Gạch chân PK (Right click → Key / Primary Key)
 - 4 Tạo Relationship (hình kim cương)
 - 5 Bật Connect → nối entity
 - 6 Đặt tên relationship (Has, Contains, Includes...)
-

4 CÂU NÓI “ĂN ĐIỂM” KHI CHỈ VÀO HÌNH

Bạn chỉ vào ProductVariant và nói:

Em tách ProductVariant riêng để quản lý tồn kho theo từng size và màu, điều này giúp hệ thống xử lý Flash Sale chính xác và tránh tình trạng âm kho khi có nhiều request đồng thời.

👉 Câu này = **điểm cao**

5 CHECKLIST TRƯỚC KHI NỘP

- Đủ 7 entity
 - PK gạch chân rõ
 - Có OrderDetail
 - Có ProductVariant
 - Không nối Product–User
 - Quan hệ 1–N hợp lý
-

Nếu bạn muốn 🔥

👉 mình vẽ lại đúng layout giống ERDPlus (đặt vị trí từng bảng y như nên vẽ để đẹp & dễ chấm)

👉 hoặc mình soi hình ERD của bạn và sửa tới khi đạt 9–10 điểm

Chỉ cần nói: “Sửa ERD cho tôi để ăn điểm cao” **100**

Hình của uml mẫu

Ok — mình gửi bạn UML mẫu (đúng LAB201 + đúng đề Shopee mini) dưới dạng PlantUML để bạn copy vào trang PlantUML/StarUML/Draw.io là ra “**hình UML**” luôn.

1) UML Use Case mẫu (Shopee mini)

plantuml

Copy code

```
@startuml
left to right direction
actor "Customer" as C
actor "Seller" as S
actor "Admin" as A

rectangle "E-Commerce Simulation" {
    (Register/Login) as UC1
    (View Products) as UC2
    (View Product Detail) as UC3
    (Select Variant Size/Color) as UC4
    (Add to Cart) as UC5
    (Update Cart) as UC6
    (Apply Voucher) as UC7
    (Checkout / Place Order) as UC8
    (View Orders) as UC9

    (Manage Products) as UC10
    (Manage Variants & Stock) as UC11
    (Manage Orders - Seller) as UC12
    (Create Voucher) as UC13

    (Run Flash Sale Simulation) as UC14
    (Flash Sale Buy) as UC15
}

C --> UC1
```

```

C --> UC2
C --> UC3
C --> UC5
C --> UC6
C --> UC7
C --> UC8
C --> UC9
UC3 --> UC4 : <<include>>

```

```

S --> UC10
S --> UC11
S --> UC12
S --> UC13

```

```

A --> UC14
C --> UC15
UC15 --> UC8 : <<include>>
UC7 --> UC8 : <<extend>>
@enduml

```

2) UML Class Diagram mẫu (đúng với ERD bạn đang vẽ)

plantuml

 Copy code

```

@startuml
class User {
    +int userID <<PK>>
    +String username
    +String password
    +String role
}

class Product {
    +int productID <<PK>>
    +String name
    +double price
    +String description
}

```

```
class ProductVariant {  
    +int variantID <<PK>>  
    +int productID <<FK>>  
    +String size  
    +String color  
    +int stock  
}  
  
class CartItem {  
    +int cartItemID <<PK>>  
    +Integer userID <<FK, nullable>>  
    +String sessionID  
    +int variantID <<FK>>  
    +int quantity  
}  
  
class Order {  
    +int orderID <<PK>>  
    +int userID <<FK>>  
    +Date orderDate  
    +double totalAmount  
    +String status  
    +Integer voucherID <<FK, nullable>>  
}  
  
class OrderDetail {  
    +int orderDetailID <<PK>>  
    +int orderID <<FK>>  
    +int variantID <<FK>>  
    +int quantity  
    +double price  
}  
  
class Voucher {  
    +int voucherID <<PK>>  
    +String code  
    +double discountValue  
    +Date expiryDate  
}
```

```
' العلاقات (multiplicity)
Product "1" -- "0..*" ProductVariant : has
ProductVariant "1" -- "0..*" CartItem : in cart
User "1" -- "0..*" Order : places
Order "1" -- "1..*" OrderDetail : contains
ProductVariant "1" -- "0..*" OrderDetail : purchased as
User "0..1" -- "0..*" CartItem : owns (optional)
Voucher "0..1" -- "0..*" Order : applied to
@enduml
```

3) UML Sequence Diagram mẫu (Flash Sale – “ăn điểm” vì có Transaction)

plantuml

 Copy code

```
@startuml
actor Customer
participant "FlashSaleServlet" as FS
participant "OrderDAO" as ODAO
participant "VariantDAO" as VDAO
database DB

Customer -> FS : POST /flashsale/buy(variantId, qty)
FS -> DB : BEGIN TRANSACTION
FS -> VDAO : checkStockForUpdate(variantId)
VDAO -> DB : SELECT stock FROM ProductVariant\nWHERE variantID=? FOR UPDATE
DB --> VDAO : stock

alt stock >= qty
    FS -> VDAO : decreaseStock(variantId, qty)
    VDAO -> DB : UPDATE ProductVariant SET stock = stock - qty
    FS -> ODAO : createOrder(userId, total)
    ODAO -> DB : INSERT Order, OrderDetail
    FS -> DB : COMMIT
    FS --> Customer : Success (Order created)
else stock < qty
    FS -> DB : ROLLBACK
    FS --> Customer : Fail (Out of stock)
```

```
end
@enduml
```

Cách bạn “lấy hình” nhanh

- Vào bất kỳ PlantUML online renderer (hoặc dùng plugin VSCode PlantUML)
 - Paste code → Export PNG → nộp vào report
-

Nếu bạn muốn mình vẽ đúng layout giống ERDPlus của bạn (đặt vị trí từng box để nhìn đẹp, thầy nhìn là ưng), bạn nói bạn cần **Use Case** hay **Class** để mình chỉnh “cho ra hình chuẩn nộp”.

↪ (E) ProductVariant — CartItem (BẮT BUỘC)

Mỗi quan hệ gì

Mình giải thích đúng trọng tâm câu bạn hỏi nhé:

(E) ProductVariant — CartItem (BẮT BUỘC) là mối quan hệ gì?

Trả lời ngắn gọn (để nói với giảng viên)

👉 Đây là mối quan hệ 1 – N (One-to-Many)

ProductVariant (1) —— (N) CartItem

1 Giải thích bằng tiếng Việt – theo nghiệp vụ Shopee

- **ProductVariant:**
Là một biến thể cụ thể của sản phẩm (ví dụ: Áo thun – Size M – Màu đen)
- **CartItem:**
Là một dòng trong giỏ hàng, ghi nhận:

- biến thể nào
- số lượng bao nhiêu
- thuộc giỏ của user/session nào

Vì sao là 1 – N?

- 1 ProductVariant có thể:
 - được **nhiều người khác nhau** thêm vào giỏ
 - xuất hiện **nhiều lần** trong nhiều giỏ hàng khác nhau

→ Do đó:

Một ProductVariant có thể liên kết với **nhiều CartItem**,
nhưng **mỗi CartItem chỉ gắn với đúng 1 ProductVariant**.

2 Biểu diễn trong ERD / UML như thế nào?

Trong ERD

- FK nằm ở CartItem

text

 Copy code

CartItem.variantID → ProductVariant.variantID

Quan hệ:

SCSS

 Copy code

ProductVariant (1) ----- (N) CartItem

3 Vì sao mối quan hệ này BẮT BUỘC? (câu ăn điểm)

Bạn nói câu này khi thầy hỏi 

Vì giỏ hàng phải lưu theo **biến thể cụ thể** (size, màu), không thể lưu theo Product chung.

Do đó CartItem bắt buộc phải liên kết với ProductVariant để đảm bảo tồn kho và giá được xử lý chính xác.

4 Nếu KHÔNG có quan hệ này → sai chỗ nào?

✗ Nếu CartItem chỉ liên kết với Product:

- Không biết user chọn size nào, màu nào
- Không kiểm soát được tồn kho
- Flash Sale không thể trừ kho chính xác

→ Sai logic Shopee → dễ bị trừ điếm

5 Cách vẽ đúng trên ERDPlus

1. Chọn Relationship (hình kim cương)

2. Nối:

- ProductVariant → Relationship
- Relationship → CartItem

3. Đặt tên: Contains / AddedTo

4. Khi convert sang relational schema → CartItem có FK variantID

6 Tóm lại (1 câu duy nhất để nhớ)

ProductVariant – CartItem là quan hệ 1–N, vì một biến thể có thể xuất hiện trong nhiều giỏ hàng khác nhau.

Nếu bạn muốn, mình có thể:

- Kiểm tra từng mối quan hệ trong ERD của bạn (đúng/sai)
- Soạn bảng “Entity – Relationship – Lý do” để bạn học thuộc đi bảo vệ

👉 Bạn muốn mình kiểm tiếp mối quan hệ nào nữa không?

Voucher có mối quan hệ ko

Có — Voucher CÓ mối quan hệ, nhưng tùy cách bạn thiết kế. Dưới đây là cách trả lời chuẩn để được điểm cao LAB201, kèm lý do rõ ràng.

Trả lời ngắn gọn (để nói với giảng viên)

Voucher có mối quan hệ với Order theo quan hệ 1 – N (One-to-Many), vì một voucher có thể được dùng cho nhiều đơn hàng, nhưng mỗi đơn hàng chỉ áp dụng tối đa một voucher.

1 Quan hệ đúng nhất cho LAB201

Voucher — Order

SCSS

 Copy code

Voucher (1) ----- (N) Order

Khóa ngoại:

CSS

 Copy code

Order.voucherID → Voucher.voucherID

Giải thích nghiệp vụ:

- Một mã voucher (VD: SALE10)
 - có thể được nhiều người dùng cho nhiều đơn khác nhau
- Nhưng:
 - mỗi Order chỉ dùng 1 voucher (đơn giản, dễ chấm, đúng LAB201)

Đây là cách nên làm để:

- UML rõ
 - ERD gọn
 - Dễ giải thích
 - Không bị hỏi xoáy
-

2 Khi nào Voucher CÓ THỂ KHÔNG cần vẽ quan hệ?

Trường hợp KHÔNG vẽ (vẫn chấp nhận được nếu giải thích tốt):

- Voucher chỉ dùng để:

- validate khi checkout
- tính discount
- không lưu lịch sử voucher đã dùng

➡ Khi đó:

- Voucher **không** liên kết DB
- Discount được ghi thẳng vào:

CSS

 Copy code
`Order.totalAmount`

📌 Nếu chọn cách này, bạn phải nói rõ:

Voucher chỉ được sử dụng để tính toán trong quá trình checkout, hệ thống không lưu thông tin voucher đã áp dụng cho đơn hàng.

⚠️ Cách này **đơn giản nhưng điểm không cao bằng** cách có quan hệ.

3 Cách “ăn điểm” hơn (nâng cao – nói miệng, không cần làm)

Nếu thầy hỏi sâu, bạn có thể nói thêm:

Trong hệ thống thực tế, voucher có thể cần bảng trung gian `VoucherUsage` để giới hạn số lần sử dụng. Tuy nhiên trong phạm vi LAB201, em thiết kế voucher theo quan hệ 1–N với Order để đơn giản hóa hệ thống.

👉 Câu này = **tư duy tốt, không bị làm quá**

4 Tóm tắt nhanh (để nhớ)

Câu hỏi	Trả lời
Voucher có mối quan hệ không?	Có
Quan hệ với bảng nào?	Order
Kiểu quan hệ	1 – N
FK nằm ở đâu?	<code>Order.voucherID</code>

Câu hỏi**Trả lời****Bắt buộc không?****Không bắt buộc, nhưng NÊN có để ăn điểm****5 Câu kết “chuẩn bài”**

Voucher được thiết kế có mối quan hệ với Order nhằm ghi nhận việc áp dụng khuyến mãi cho từng đơn hàng và đảm bảo tính rõ ràng trong xử lý nghiệp vụ.

Nếu bạn muốn, mình sẽ:

- Chỉ bạn vẽ quan hệ Voucher–Order trên ERDPlus đúng nút
- Soạn 1 đoạn giải thích Voucher để bỏ vào báo cáo
- Giả lập câu hỏi giảng viên hay hỏi về Voucher

👉 Bạn muốn làm mức an toàn (7–8) hay mức ăn điểm (9–10)?