INNOVIOR

Project 4: E-commerce + Integrated Stock Management - Detailed Task Brief

1. Overview

Build a public e-commerce storefront tightly integrated with internal stock management. Provide catalog browsing, cart and checkout, orders, payments, and shipments while maintaining accurate inventory with reservations, concurrency protections, and anti-oversell guarantees. Deployed for internal operations and public customers; admin and fulfillment screens are internal-only (behind SSO/VPN).

2. Objectives

- Offer a modern product catalog with variants, categories, search, and filters.
- Enable cart and checkout with secure payments (Stripe test mode or Cash on Delivery).
- Maintain accurate inventory using reservation (qty_reserved) and deduction (qty_on_hand) flows.
- Provide order management, shipments with tracking, and email notifications.
- Prevent overselling under concurrent checkouts; implement idempotent payment and order logic.
- Provide reports (sales, top products, inventory status) and CSV exports.

3. Scope

In Scope:

- Customer accounts (register/login) and admin roles (Admin, Inventory Manager, Fulfillment, Support).
- Catalog: products, variants (SKU), categories, images, pricing, tax rates.
- Cart and checkout: address capture, shipping method, tax calculation, discount codes.
- Payments: Stripe (test mode) and/or Cash on Delivery; Stripe webhook handling.
- Inventory integration: reservation on add-to-cart/checkout; deduction on payment success; release on cancel/expiry.
- Orders and shipments: statuses, tracking, emails.
- Reports: sales summary, inventory status; CSV export.

Out of Scope (for v1):

- Marketplace (multi-vendor) features.
- Advanced promotions (buy X get Y, bundles) beyond simple coupons.

- Multi-currency, multi-language, advanced tax automation (basic fixed rates only).
- Returns/RMA workflows beyond basic order cancellation/refund.
- 3rd-party shipping carrier rate shopping and label purchase (use manual rates).

4. Stakeholders and Roles

- Admin: system settings, users/roles, taxes, shipping methods, promotions.
- Inventory Manager: products, variants, pricing, inventory adjustments.
- Fulfillment/Shipping: pick/pack/ship, create shipments, update tracking.
- Support: view orders, process cancellations/refunds, assist customers.
- Customer: browse catalog, place orders, view order history.

5. Assumptions & Constraints

- Tech: PHP 8.2+, Laravel 11, MySQL 8, Redis (queues/cache), Mail (SMTP).
- Frontend: Blade or Inertia/Vue for storefront and admin panel.
- Auth: Laravel Breeze (customers) and RBAC via spatie/laravel-permission (admin).
- Payments: Stripe in test mode; do not store raw card data (use tokens only).
- Concurrency: use DB row locks for inventory updates; idempotency keys for payments/webhooks.
- Timeframe: 4 weeks for MVP by one intern; weekly reviews and demos.

6. Architecture Overview

Presentation: Public Storefront (Blade/Inertia) and Internal Admin (Blade/Livewire).
Application: Laravel Controllers, Form Requests, Policies; Services (InventoryService, ReservationService, CheckoutService, PaymentService, ShippingService).
Data: MySQL with migrations, seeders, factories.
Infra: Redis for queues/cache; Mail for notifications; storage for media.
Scheduler: release expired reservations; nightly reports; email jobs.
CI: GitHub Actions for lint (Pint), static analysis (PHPStan), and tests (Pest/PHPUnit).

7. Modules and Functional Requirements

7.1 Catalog

Description: Product catalog with variants, categories, images, and pricing.

User Stories:

- As a customer, I can browse products by category and search by keyword.
- As a customer, I can view product details, select a variant (size/color), and see stock status.
- As an admin, I can create/edit/deactivate products, variants, prices, and images.

Acceptance Criteria:

- Variant SKU must be unique; inactive products/variants hidden from storefront.
- List pages support pagination, filters (category, price range), and sort (price, latest).
- Prices shown inclusive/exclusive of tax based on a simple config.

7.2 Inventory & Reservation

Description: Track inventory quantities and manage reservations to prevent oversell.

User Stories:

- As a customer, when I add to cart, the system soft-reserves stock (qty_reserved) for a time window.
- As the system, on payment success, I deduct reserved stock from qty_on_hand and reduce qty_reserved accordingly.
- As the system, on reservation expiry/cancel, I release qty_reserved back to available.

Acceptance Criteria:

- Inventory invariant: qty_on_hand >= 0 and qty_reserved >= 0 at all times.
- Concurrent carts cannot oversell; row-level locking and atomic updates are enforced.
- Reservations expire automatically after N minutes (configurable) via scheduler job.
- Admin UI to adjust inventory with reason and audit log.

7.3 Cart

Description: Manage cart items and compute totals.

User Stories:

- As a customer, I can add, update, and remove items in my cart.
- As a customer, I can apply a coupon code and see the updated totals.
- As a customer, I can see shipping estimate and taxes before placing the order.

Acceptance Criteria:

- Cart totals include subtotal, discounts, tax, and shipping.
- Quantity changes validate against available stock (on-hand minus existing reservations).
- Coupons validate eligibility (date, min subtotal, usage limit).

7.4 Checkout & Payments

Description: Secure checkout with address capture, shipping method selection, and payments.

User Stories:

- As a customer, I can enter billing/shipping addresses and select a shipping method.
- As a customer, I can pay using Stripe (test mode) or choose Cash on Delivery.
- As the system, I finalize the order and send confirmation emails after payment success.

Acceptance Criteria:

- Stripe integration uses Payment Intents; webhook verifies final status (SUCCEEDED/FAILED).
- Idempotency: repeated webhook/events do not duplicate charges or orders.
- On payment success: move reservation to deduction; on failure/cancel: release reservations.
- Checkout validation rechecks stock before confirming order.

7.5 Orders & Fulfillment

Description: Manage order lifecycle and shipments.

User Stories:

- As an admin, I can view all orders, filter by status/date/customer.
- As fulfillment, I can create a shipment with a carrier and tracking number.
- As support, I can cancel orders not yet shipped and issue refunds (Stripe).

Acceptance Criteria:

- Order statuses: PENDING -> PAID -> FULFILLED -> (optional) RETURNED/CANCELLED.
- Shipment statuses: READY -> SHIPPED -> DELIVERED -> RETURNED.
- Email notifications: order confirmation, shipment tracking.

7.6 Promotions & Coupons

Description: Support simple coupon codes for discounts.

Acceptance Criteria:

- Coupon types: PERCENT or AMOUNT; start/end dates; min subtotal; max redemptions; per-customer limit.
- Applied discounts stored on order; coupon usage counted atomically.

7.7 Customer Accounts

Description: Customer profiles, addresses, and order history.

Acceptance Criteria:

- Customers can update profile and manage multiple addresses.
- Order history page lists past orders with status and invoice details.

7.8 Admin & Reports

Description: Internal dashboards and standard reports.

Reports:

- Sales Summary by day/week/month.
- Top Products by revenue/quantity.
- Inventory Status (on hand, reserved, low stock).

INNOVIOR

8. Data Model (Entities)

Table	Key Fields
products	id, name, description, category_id, active, timestamps
variants	id, product_id, sku (unique), option_values (json), price, compare_at_price (nullable), active, timestamps
categories	id, name, parent_id (nullable), slug, timestamps
images	id, product_id, path, alt_text, sort_order
inventory	id, variant_id, qty_on_hand, qty_reserved, reorder_point, timestamps
customers	id, name, email (unique), phone, password_hash, timestamps
addresses	id, customer_id, type (BILLING|SHIPPING), name, line1, line2, city, region, postal_code, country, phone, timestamps
carts	id, customer_id, status (ACTIVE|ORDERED|ABANDONED), updated_at
cart_items	id, cart_id, variant_id, qty, unit_price, timestamps
orders	id, customer_id, order_number (unique), status (PENDING|PAID|FULFILLED|CANCELLED|REFUNDED), subtotal, discount, tax, shipping, total, currency, billing_address_id, shipping_address_id, timestamps
order_items	id, order_id, variant_id, qty, unit_price, total
payments	id, order_id, provider (STRIPE|COD), status (INIT|SUCCEEDED|FAILED|REFUNDED), transaction_id (nullable), amount, currency, timestamps
shipments	id, order_id, carrier, tracking_no, status (READY|SHIPPED|DELIVERED|RETURNED), shipped_at (nullable), delivered_at (nullable), timestamps
coupons	id, code (unique), type (PERCENT|AMOUNT), value, min_subtotal, starts_at, ends_at, max_redemptions, per_customer_limit, active, timestamps
order_coupons	id, order_id, coupon_id, discount_amount, timestamps
tax_rates	id, country, region, rate, active

shipping_methods	id, code, name, base_rate, per_item_rate, active
users	id, name, email, password, ... (admin users)

9. API Endpoints (Examples)

- Catalog: GET /api/products, GET /api/products/{id}, GET /api/categories
- Cart: GET /api/cart, POST /api/cart/items, PUT /api/cart/items/{id}, DELETE /api/cart/items/{id}
- Coupons: POST /api/cart/apply-coupon, DELETE /api/cart/remove-coupon
- Checkout: POST /api/checkout (calculates totals, creates PaymentIntent or COD)
- Orders: GET /api/orders/{id}, POST /api/orders/{id}/cancel
- Payments: POST /webhooks/stripe
- Inventory (admin): GET /api/inventory?variant_id=, PUT /api/inventory/{variant_id}/adjust
- Shipments (admin): POST /api/orders/{id}/ship, GET /api/shipments/{id}
- Reports (admin): GET /api/reports/sales?from=&to=, GET /api/reports/inventory-status

10. Non-Functional Requirements

- Security: Auth for customers and RBAC for admin; never store PAN/card data (Stripe only).
- Auditability: Log key actions (inventory adjustments, refunds, shipment updates).
- Performance: Product list pages < 1.5s; checkout < 2s under normal load.
- Reliability: Inventory updates in DB transactions with row locks; retries with backoff for webhooks.
- Observability: Structured logs; correlation IDs on checkout and payment flows.
- SEO (basic): Clean URLs and meta tags for product/category pages.

11. Security & Compliance

- Protect secrets via .env; never commit credentials; rotate Stripe keys as needed.
- CSRF protection for web forms; rate limit write endpoints.
- Validate/sanitize inputs with Form Requests; server-side enforce prices/discounts.
- PII protection: limit who can access customer data; data minimization in logs.
- PCI note: use Stripe-hosted elements; do not store card numbers or CVV.

12. Testing Plan

Unit Tests:

- InventoryService invariants (reserve/deduct/release cannot go negative).
- PriceCalculator (tax, shipping, coupon discount) correctness.
- Coupon eligibility (date ranges, min subtotal, usage limits).
- Payment idempotency key handling.

Feature/Integration Tests:

- Add-to-cart reserves stock; reservation expires releases stock.
- Concurrent checkout attempts do not oversell (simulate parallel).
- Stripe webhook processed exactly once; order reflects final payment state.
- Order -> Shipment updates send emails and status transitions are valid.

Test Data & Factories: categories, products (with variants), customers, carts, coupons, and initial inventory.

13. Milestones & Timeline (4 Weeks)

Week	Focus	Key Deliverables
Week 1	Catalog & Accounts	Migrations/seeders; products/variants/categories CRUD; customer auth; basic storefront UI
Week 2	Cart & Checkout	Cart endpoints/UI; reservation logic; totals calc; Stripe/COD integration (test mode)
Week 3	Orders & Fulfillment	Order management; Stripe webhook; shipments; emails; coupons; admin dashboards
Week 4	Reports & Quality	Reports; CSV export; feature tests; README; demo data; deployment script; final demo

14. Deliverables & Definition of Done

- Git repository with README (setup, run, test, seed) and .env.example.
- Diagrams (/diagrams): ERD and architecture.
- Automated tests with coverage report (Pest/PHPUnit).
- Activity/audit logs for critical admin/customer actions.
- Demo video or screenshots covering browse -> checkout -> ship.

15. Development Standards

- PSR-12 code style; Laravel Pint on CI; PHPStan level 6+.
- Use config() not env() in code; secrets via .env; example env provided.
- Form Requests for validation; Policies for authorization; Route model binding.

- Use database transactions and row-level locking for inventory updates.
- Implement domain events (OrderPaid, ReservationExpired) and queue jobs where applicable.

16. Seed Data (Minimum)

- 3 Categories; 10 Products with 2 variants each; images.
- Initial inventory for each variant (e.g., qty_on_hand 20, qty_reserved 0).
- 2 Customers with addresses; 1 admin user.
- 1 Active coupon (e.g., 10% off) with limits.
- Sample orders and a shipment for demo.

17. Glossary

- On Hand: Physical quantity available for sale (after deductions).
- Reserved: Quantity held for carts/checkout pending payment; auto-expires.
- Oversell: Selling more than available stock; must be prevented via locks.
- Payment Intent: Stripe object managing the payment lifecycle.
- COD: Cash on Delivery payment method.