Backend Implementation Guide - Online Grocery Web App

Project: Multi-Warehouse E-Commerce Distribution System
Tech Stack: Next.js, Express.js, Prisma ORM, PostgreSQL
Team: Fadil (Feature 1), Aksa (Feature 2), Rega (Feature 3)

---

1. System Overview

1.1 Architecture Concept

Sistem ini adalah **warehouse-based distribution system** seperti iBox, Zalora, atau Uniqlo, dengan karakteristik:

- **Single company** dengan multiple warehouses di berbagai kota
- **Same products** dijual dari semua warehouse
- **Location-based fulfillment**: Order dipenuhi dari warehouse terdekat customer
- **Product variants**: Produk dengan variasi (warna, ukuran, memory)
- **Multi-tier discount**: Product discount, cart discount, shipping discount, dan voucher global

1.2 Key Entities

```
Users (Customer & Admins)
   ↓
Products → ProductVariants
   ↓
Warehouses → Inventory (stock per variant per warehouse)
   ↓
Cart → Order → Payment
   ↓
Discounts & Vouchers
```

1.3 User Roles

- **USER**: Customer yang bisa belanja
- **STORE_ADMIN**: Mengelola warehouse yang di-assign, lihat laporan stock warehouse tersebut
- **SUPER_ADMIN**: Full access semua warehouse, kelola admin, kelola produk

---

2. Feature Ownership & Integration Points

Feature 1: User & Authentication (Fadil)

Scope:

- User registration, login, verification (email/social)
- Address management (dengan coordinates)
- Shipping method management
- Admin assignment ke warehouse

Models Owned:

- `User`
- `VerificationToken`
- `Address`
- `ShippingMethod`
- `UserWarehouse`

⚠️ **Integration Points:**

- **Dengan Feature 2 (Aksa):**
  - User.role untuk authorization (Super Admin vs Store Admin)
  - UserWarehouse untuk assign admin ke warehouse
  - User sebagai creator di StockJournal
- **Dengan Feature 3 (Rega):**
  - User untuk Cart & Order ownership
  - Address untuk shipping calculation
  - ShippingMethod untuk order fulfillment

Feature 2: Product & Inventory (Aksa)

Scope:

- Product & variant management (CRUD)
- Category management
- Warehouse management
- Inventory tracking per variant per warehouse

- Stock journal (audit trail semua perubahan stock)
- Discount management (4 tipe)
- Voucher global management
- Sales & stock reports

Models Owned:

- `Category`
- `Product`
- `ProductImage`
- `ProductVariant`
- `Warehouse`
- `Inventory`
- `StockJournal`
- `Discount`
- `ProductDiscount`
- `Voucher`
- `UserVoucher`

⚠️ **Integration Points:**

- **Dengan Feature 1 (Fadil):**
  - UserWarehouse untuk cek authorization admin
  - User untuk StockJournal.createdBy
- **Dengan Feature 3 (Rega):**
  - ProductVariant untuk Cart & Order
  - Inventory untuk stock availability check
  - Discount & Voucher untuk checkout calculation
  - StockJournal untuk order fulfillment (stock OUT)

Feature 3: Cart, Order, Payment (Rega)

Scope:

- Shopping cart management
- Checkout process (find nearest warehouse, calculate shipping)
- Discount & voucher application
- Order creation & management
- Payment processing (manual transfer & gateway)
- Order status workflow
- Stock reservation & release

Models Owned:

- `Cart`
- `CartItem`
- `Order`
- `OrderItem`
- `Payment`
- `PaymentProof`
- `OrderHistory`
- `DiscountUsage`
- `VoucherUsage`

⚠️ **Integration Points:**

- **Dengan Feature 1 (Fadil):**
  - User untuk cart/order ownership
  - Address untuk warehouse distance calculation
  - ShippingMethod untuk shipping fee
- **Dengan Feature 2 (Aksa):**
  - ProductVariant untuk cart/order items
  - Inventory untuk stock check & reservation
  - Warehouse untuk order fulfillment
  - Discount & Voucher untuk price calculation
  - StockJournal (trigger creation saat order fulfilled)

---

3. Core Entity Workflows

3.1 Product & Variant Management

Create Product with Variants

Who: Super Admin only (Feature 2 - Aksa)

Flow:

1. Super Admin input product master data (name, description, category)
2. Input multiple variants dengan attributes berbeda
    - SKU (unique)
    - Name (e.g., "iPhone 17 Pro 256GB Black")
    - Color, size, weight
    - Price per variant
3. Upload multiple product images
4. System auto-create inventory records (quantity=0) untuk semua warehouse
5. Product & variants ready to sell

Example:

```
Product: "iPhone 17 Pro"
├── Variant 1: SKU "IPH17-PRO-256-BLK", Color "Black", Size "256GB", Price 15,000,000
├── Variant 2: SKU "IPH17-PRO-256-WHT", Color "White", Size "256GB", Price 15,000,000
└── Variant 3: SKU "IPH17-PRO-512-BLK", Color "Black", Size "512GB", Price 18,000,000

Auto-create Inventory:
- Variant 1 → Warehouse Jakarta: qty=0
- Variant 1 → Warehouse Bandung: qty=0
- Variant 1 → Warehouse Surabaya: qty=0
(dan seterusnya untuk semua variant + warehouse combinations)
```

Authorization:

- Super Admin: Full CRUD
- Store Admin: Read-only
- User: View active products only

3.2 Inventory Management

Stock Update (CRITICAL - ATOMIC TRANSACTION)

Who: Store Admin (warehouse assigned) atau Super Admin (Feature 2 - Aksa)

Flow - Stock IN (Pembelian/Restock):

1. Admin pilih warehouse (auto-selected jika Store Admin)
2. Pilih product variant yang mau di-restock
3. Input quantity IN dan reference number (e.g., "PO-2025-001")
4. **ATOMIC TRANSACTION:**

```
a. Get current inventory.quantity
b. Create StockJournal:
  - type: IN
  - quantity: +100
  - stockBefore: 50
  - stockAfter: 150
  - referenceNo: "PO-2025-001"
  - reason: "Purchase Order"
  - createdBy: adminUserId
c. Update Inventory:
  - quantity: 50 + 100 = 150
```

5. If transaction fails, rollback semua (journal tidak dibuat, stock tidak berubah)

Flow - Stock OUT (Manual Adjustment/Damaged):

- Same flow, tapi type: OUT dan quantity negative
- Validasi: stockAfter tidak boleh < 0

⚠️ CRITICAL RULE:

- **TIDAK BOLEH** update Inventory.quantity langsung
- **WAJIB** lewat StockJournal transaction
- **ATOMIC:** Create journal + update inventory dalam 1 transaction

Authorization:

```
Super Admin: Bisa update stock di SEMUA warehouse
Store Admin: Hanya warehouse yang di-assign (cek UserWarehouse)
User: Tidak ada akses
```

3.3 Discount System

4 Tipe Discount

Who: Super Admin only (Feature 2 - Aksa)

1. PRODUCT Discount

- Diskon untuk product variant tertentu
- Example: "Diskon iPhone 15%" apply ke variant tertentu
- Relasi: Discount → ProductDiscount → ProductVariant

2. CART Discount (Min Purchase)

- Diskon jika total belanja >= minPurchase
- Example: "Belanja 1jt diskon 100rb"
- Bisa set maxDiscount untuk limit

3. SHIPPING Discount

- Diskon ongkos kirim
- Example: "Gratis ongkir" = 100% discount

4. BUY_ONE_GET_ONE (BOGO)

- Beli X gratis Y
- Example: "Beli 2 Gratis 1"
- Relasi ke ProductVariant tertentu

Business Rules:

- Semua discount punya startDate & endDate
- isActive flag untuk enable/disable
- Bisa overlap (multiple discounts aktif bersamaan)

3.4 Voucher System

Voucher Global

Who: Super Admin (Feature 2 - Aksa)

Concept:

- Voucher dibuat oleh admin sebagai **kode promo global**
- User **claim** voucher → masuk ke UserVoucher
- Setiap user bisa pakai voucher sesuai maxUsagePerUser
- Total usage dibatasi maxTotalUsage

Flow - Create Voucher:

1. Admin create voucher dengan code unique (e.g., "NEWUSER50")
2. Set type (PRODUCT/CART/SHIPPING)
3. Set discount value & conditions
4. Set usage limits (per user & total)

Flow - User Claim Voucher (Feature 3 - Rega):

1. User input voucher code
2. System validate:
    - Voucher exists & active
    - User belum claim (atau belum exceed maxUsagePerUser)
    - Total usage < maxTotalUsage
3. Create UserVoucher record
4. User bisa pakai di checkout

Example:

```
Voucher: "WELCOME10"
- Code: WELCOME10
- Type: CART
- Discount: 10% (max 50rb)
- Min Purchase: 100rb
- Max usage per user: 1
- Max total usage: 1000

User A claim → UserVoucher created
User A checkout pakai WELCOME10 → VoucherUsage created
User A coba pakai lagi → Error: "Voucher sudah digunakan"
```

3.5 Cart Management

Add to Cart

Who: Verified User only (Feature 3 - Rega)

Flow:

1. User browse products → see variants
2. User pilih variant (color, size)
3. User set quantity
4. **Check stock availability:**

```
Total Available = SUM(Inventory.quantity - Inventory.reserved)
WHERE productVariantId = selected
AND warehouseId IN (active warehouses)
```

5. If available >= quantity → Create/Update CartItem
6. If not enough → Error: "Stock tidak cukup"

⚠️ **Integration Point:**

- **Dengan Feature 2:** Query Inventory untuk check availability
- **Authorization:** User must be authenticated & verified

---

4. Critical Flow: Complete Checkout Process

4.1 Checkout Initiation

Who: Verified User (Feature 3 - Rega)

Prerequisites:

1. User has items in cart
2. User has at least 1 address with coordinates
3. User selected shipping method

4.2 Warehouse Selection

Logic:

Step 1: Get customer address coordinates

```
Customer Address:
- Latitude: -6.200000
- Longitude: 106.816666
```

Step 2: Find warehouses dengan stock cukup

```
For each cart item:
  Get ProductVariant
  Query: SELECT * FROM Inventory
         WHERE productVariantId = item.variantId
         AND (quantity - reserved) >= item.quantity
         AND warehouseId IN (active warehouses)
```

Step 3: Calculate distance ke customer

```
For each warehouse yang punya stock cukup:
  Calculate distance(customerLat, customerLng, warehouseLat, warehouseLng)
  Filter: distance <= warehouse.maxServiceRadius (default 10 KM)
```

Step 4: Select nearest warehouse

```
nearest = warehouse dengan distance terkecil
```

Validation:

- Jika tidak ada warehouse dalam radius → Error: "Lokasi di luar jangkauan layanan"
- Jika tidak ada warehouse dengan stock cukup → Error: "Stock tidak tersedia"

4.3 Discount & Voucher Calculation

Step 1: Calculate Subtotal

```
For each cart item:
  itemPrice = productVariant.price
  itemSubtotal = itemPrice × quantity

Total Subtotal = SUM(all itemSubtotal)
```

Step 2: Apply BOGO Discount

```
Get active BOGO discounts untuk product variants di cart
For each BOGO:
  If cart has buyQuantity of product:
    Add getQuantity item gratis (atau discount equivalent)
```

Step 3: Apply Product Discount

```
Get active PRODUCT discounts
For each cart item:
  If item.productVariant has active discount:
    Calculate discount amount
    itemDiscount = calculate based on discountType (% or nominal)
    itemTotal = itemSubtotal - itemDiscount
```

Step 4: Apply Cart Discount (Min Purchase)

```
Get active CART discounts
For each cart discount:
  If subtotal >= discount.minPurchase:
    Calculate discount amount
    Apply maxDiscount limit if set

Select highest discount
Subtotal = Subtotal - cartDiscount
```

Step 5: Apply User Voucher (if provided)

```
If user input voucher code:
  Get UserVoucher (unused, not expired)
  Validate voucher conditions
  Calculate voucher discount
  Subtotal = Subtotal - voucherDiscount
  Mark UserVoucher as used
```

Step 6: Calculate Shipping Fee

```
Call RajaOngkir API:
- Origin: warehouse.city
- Destination: customer.city
- Weight: sum of all product weights
- Courier: selected shipping method

ShippingFee = API response
```

Step 7: Apply Shipping Discount

```
Get active SHIPPING discount
If exists:
  ShippingFee = ShippingFee - calculate discount
```

Step 8: Calculate Final Total

```
Total = Subtotal + ShippingFee + Tax - totalDiscount
```

Priority Urutan Discount:

```
1. BOGO (dapat item gratis)
2. Product Discount (per item)
3. Cart Discount (min purchase)
4. User Voucher
5. Shipping Discount
```

4.4 Stock Reservation

Step 1: Reserve stock (ATOMIC TRANSACTION)

```
BEGIN TRANSACTION

For each order item:
  UPDATE Inventory
  SET reserved = reserved + item.quantity
  WHERE productVariantId = item.variantId
  AND warehouseId = selectedWarehouse.id

  Validate: (quantity - reserved) >= 0

COMMIT
```

Step 2: Set auto-cancel timer

```
If paymentMethod = MANUAL_TRANSFER:
  autoCancelAt = now + 1 hour
```

⚠️ Integration Point:

- **Dengan Feature 2:** Update Inventory.reserved (Aksa provide endpoint)

4.5 Order Creation

Step 1: Create Order record

```
Order:
- orderNumber: generate unique (e.g., "ORD-20250108-0001")
- userId: current user
- shippingAddressId: selected address
- shippingWarehouseId: nearest warehouse
- shippingMethodId: selected method
- subtotal, shippingFee, tax, totalDiscount, total
- paymentMethod: MANUAL_TRANSFER / PAYMENT_GATEWAY
- orderStatus: PENDING_PAYMENT
- autoCancelAt: now + 1h (jika manual transfer)
```

Step 2: Create OrderItems

```
For each cart item:
  OrderItem:
  - productVariantId
  - sku (snapshot)
  - productName (snapshot)
  - variantName (snapshot)
  - price (snapshot)
  - quantity
  - discount
  - total
```

Step 3: Record Discount & Voucher Usage

```
DiscountUsage: (for each discount applied)
- orderId
- discountId
- snapshot discount details
- discountAmount (actual amount)

VoucherUsage: (if voucher used)
- orderId
- voucherId
- snapshot voucher details
- discountAmount
```

Step 4: Clear Cart

```
Delete all CartItems for this user
```

4.6 Payment Processing

Manual Transfer Flow

Step 1: User upload bukti transfer (Feature 3 - Rega)

```
1. User upload image (max 1MB, jpg/jpeg/png)
2. Upload to Cloudinary
3. Create PaymentProof record:
   - paymentId
   - uploadedById: userId
   - fileName, mimeType, sizeBytes
   - url (Cloudinary)
   - verified: false
4. Update Order.orderStatus = AWAITING_PAYMENT_PROOF
```

Step 2: Admin verify payment (Feature 2 or 3)

```
1. Admin review bukti transfer
2. If valid:
   - Update PaymentProof.verified = true
   - Update Payment.status = PAID
   - Update Order.paymentStatus = PAID
   - Update Order.orderStatus = PROCESSING
   - Cancel autoCancelAt timer
3. If invalid:
   - Reject with reason
   - Order.orderStatus back to PENDING_PAYMENT
```

Payment Gateway Flow (Midtrans)

```
1. System create Midtrans payment link
2. User redirected to Midtrans
3. User pay
4. Midtrans callback to webhook
5. System auto-update:
   - Payment.status = PAID
   - Order.paymentStatus = PAID
   - Order.orderStatus = PROCESSING
```

Auto-Cancel (Background Job)

```
Cron job check every 5 minutes:

For each Order WHERE:
  - orderStatus = PENDING_PAYMENT
  - autoCancelAt <= now

Action:
  - Update orderStatus = CANCELLED
  - Unreserve stock:
    UPDATE Inventory
    SET reserved = reserved - orderItem.quantity
```

⚠️ Integration Point:

- **Feature 3 (Rega):** Payment processing
- **Feature 2 (Aksa):** Admin verification (optional, bisa juga Rega)

4.7 Order Fulfillment

Admin Process Order

Who: Store Admin (warehouse assigned) or Super Admin (Feature 2 or 3)

Step 1: Prepare Items (orderStatus: PROCESSING)

```
1. Admin lihat order yang PROCESSING
2. Admin prepare/pack items
3. Admin ubah status = SHIPPED
4. Input tracking number (optional)
```

Step 2: Create Stock Journal OUT (ATOMIC)

```
BEGIN TRANSACTION

For each OrderItem:

  1. Create StockJournal:
     - type: OUT
     - quantity: orderItem.quantity
     - stockBefore: inventory.quantity
     - stockAfter: inventory.quantity - quantity
     - referenceNo: order.orderNumber
     - reason: "Sales Order"
     - relatedOrderId: order.id
     - createdBy: adminUserId

  2. Update Inventory:
     - quantity = quantity - orderItem.quantity
     - reserved = reserved - orderItem.quantity

  Validate: quantity >= 0 after update

COMMIT
```

Step 3: Set auto-confirm timer

```
Order.autoConfirmAt = now + 48 hours
```

⚠️ Integration Point:

- **Feature 2 (Aksa):** StockJournal creation (critical!)

- Stock harus OUT saat order SHIPPED, bukan saat order created

---

4.8 Order Completion

User Confirm Delivery

Who: User (Feature 3 - Rega)

Flow:

```
1. User click "Terima Pesanan"
2. Update Order:
   - orderStatus = COMPLETED
   - completedAt = now
3. Cancel autoConfirmAt timer
```

Auto-Confirm (Background Job)

```
Cron job check every 6 hours:

For each Order WHERE:
  - orderStatus = SHIPPED
  - autoConfirmAt <= now

Action:
  - Update orderStatus = COMPLETED
  - Update completedAt = now
```

4.9 Order Cancellation

Cancel Before Payment

Who: User (Feature 3 - Rega)

Flow:

```
1. User click "Batalkan Pesanan"
2. Validate: orderStatus = PENDING_PAYMENT
3. BEGIN TRANSACTION:

   a. Unreserve stock:
      For each OrderItem:
        UPDATE Inventory
        SET reserved = reserved - item.quantity

   b. Update Order:
      - orderStatus = CANCELLED
      - cancelledAt = now

   COMMIT
```

Cancel After Payment (by Admin)

Who: Admin (Feature 2 or 3)

Flow:

```
1. Admin click "Batalkan Pesanan"
2. Validate: orderStatus != SHIPPED and != COMPLETED
3. BEGIN TRANSACTION:

   a. If orderStatus = PROCESSING:
      Unreserve stock (same as above)

   b. Update Order:
      - orderStatus = CANCELLED
      - cancelledAt = now

   c. Update Payment:
      - status = REFUNDED

   COMMIT

4. Admin manually refund payment outside system
```

Business Rules:

- User hanya bisa cancel SEBELUM upload bukti bayar
- Admin bisa cancel sampai status PROCESSING
- Tidak bisa cancel jika status SHIPPED atau COMPLETED

5. Reports & Analytics

5.1 Sales Report

Who: Super Admin (all warehouses) or Store Admin (assigned warehouse only) - Feature 2 (Aksa)

Report Types:

1. Monthly Sales Summary

```
Query:
- Period: Month/Year
- Filter: warehouseId (optional)
- Status: COMPLETED only

Metrics:
- Total Orders
- Total Revenue
- Total Items Sold
- Average Order Value
- Total Discount Given
- Total Shipping Fee
```

2. Sales by Category

```
Group by: Product.categoryId
Show:
- Category Name
- Total Orders
- Total Revenue
- Total Items
- % of Total Sales
```

3. Sales by Product

```
Group by: ProductVariant
Show:
- Product Name + Variant
- Quantity Sold
- Total Revenue
- Average Price
- % of Total Sales
- Most sold variant
```

Export: PDF & Excel

5.2 Stock Report

Who: Super Admin or Store Admin - Feature 2 (Aksa)

Report Types:

1. Stock Summary

```
Query:
- Period: Month
- Filter: warehouseId, productVariantId

Show per product variant:
- Opening Stock (start of month)
- Total IN (from StockJournal type=IN)
- Total OUT (from StockJournal type=OUT)
- Closing Stock (end of month)
- Stock Value (closing × price)
```

2. Stock Movement Detail

```
Query: All StockJournal entries
Show:
- Date
- Product Variant
- Type (IN/OUT)
- Quantity
- Stock Before/After
- Reference Number
- Reason
- Created By
```

3. Low Stock Alert

```
Query: Inventory WHERE quantity < threshold
Show:
- Product Variant
- Warehouse
- Current Stock
- Reserved Stock
- Available Stock
```

Export: PDF & Excel

⚠️ Laporan Stock Calculation:

```
Opening Stock = Last StockJournal.stockAfter before period start
Total IN = SUM(quantity WHERE type=IN AND date IN period)
Total OUT = SUM(quantity WHERE type=OUT AND date IN period)
Closing Stock = Opening + Total IN - Total OUT

Must match with: Current Inventory.quantity
```

---

6. Authorization Matrix

Action	User	Store Admin	Super Admin
User Management			
Register/Login	✅	✅	✅
Manage own profile	✅	✅	✅
Create Store Admin	❌	❌	✅
Assign Admin to Warehouse	❌	❌	✅
Product & Category			
View products	✅	✅	✅
Create/Edit/Delete Product	❌	❌	✅
Create/Edit/Delete Category	❌	❌	✅
Inventory & Stock			
View stock (all warehouses)	❌	❌	✅
View stock (assigned warehouse)	❌	✅	✅
Update stock (all warehouses)	❌	❌	✅
Update stock (assigned warehouse)	❌	✅	✅
View stock reports	❌	✅ (own)	✅ (all)
Discount & Voucher			

Action	User	Store Admin	Super Admin
View active discounts	✅	✅	✅
Create/Edit/Delete Discount	❌	❌	✅
Create/Edit/Delete Voucher	❌	❌	✅
Claim voucher	✅	❌	❌
Cart & Order			
Add to cart	✅	❌	❌
Checkout	✅	❌	❌
View own orders	✅	❌	❌
View all orders (assigned warehouse)	❌	✅	✅
Process order	❌	✅ (own)	✅ (all)
Cancel order (before payment)	✅	❌	❌
Cancel order (after payment)	❌	✅	✅
Payment			
Upload proof	✅	❌	❌
Verify payment	❌	✅ (own)	✅ (all)
Reports			
Sales report (all warehouses)	❌	❌	✅
Sales report (assigned warehouse)	❌	✅	✅
Stock report (all warehouses)	❌	❌	✅
Stock report (assigned warehouse)	❌	✅	✅

---

7. Critical Business Rules

7.1 Stock Management

1. **ATOMIC TRANSACTION:** StockJournal creation + Inventory update must be atomic
2. **NO DIRECT UPDATE:** Never update Inventory.quantity without StockJournal
3. **NEGATIVE CHECK:** Stock never boleh < 0
4. **RESERVED STOCK:** Reserved dikurangi saat order completed (SHIPPED), bukan saat created

7.2 Order Management

1. **WAREHOUSE SELECTION:** Always pilih warehouse terdekat dengan stock cukup
2. **DISTANCE VALIDATION:** Customer must dalam radius maxServiceRadius
3. **AUTO-CANCEL:** Pending payment auto-cancel after 1 hour (manual transfer)

4. **AUTO-CONFIRM:** Shipped order auto-confirm after 48 hours
5. **CANCEL RESTRICTION:** Cannot cancel after status SHIPPED

7.3 Discount & Voucher

1. **PRIORITY:** BOGO → Product → Cart → Voucher → Shipping
2. **SNAPSHOT:** Save discount details di DiscountUsage/VoucherUsage (untuk laporan)
3. **VOUCHER LIMIT:** Enforce maxUsagePerUser dan maxTotalUsage
4. **EXPIRY:** Check voucher expiresAt saat redemption

7.4 Payment

1. **MANUAL TRANSFER:** Require proof upload + admin verification
2. **GATEWAY:** Auto-update via webhook
3. **REFUND:** Manual process outside system (admin refund bank transfer)

7.5 Data Integrity

1. **SOFT DELETE:** Product use isDeleted flag, never hard delete
2. **SNAPSHOT:** OrderItem save snapshot (name, price) saat order created
3. **AUDIT TRAIL:** StockJournal record every stock change dengan stockBefore/After
4. **CASCADE:** Use onDelete: Cascade untuk relasi parent-child

---

8. API Structure Recommendations

8.1 Endpoint Naming Convention

```
GET    /api/products                  # List products
GET    /api/products/:id              # Get product detail + variants
POST   /api/products                  # Create product (Super Admin)
PUT    /api/products/:id              # Update product
DELETE /api/products/:id              # Soft delete product

GET    /api/products/:id/variants     # List variants
POST   /api/products/:id/variants     # Create variant
PUT    /api/variants/:id              # Update variant
DELETE /api/variants/:id              # Delete variant

GET    /api/warehouses/:id/inventory  # List inventory per warehouse
POST   /api/inventory/stock-in        # Stock IN transaction
POST   /api/inventory/stock-out       # Stock OUT transaction
```

```
GET    /api/inventory/stock-journal     # Stock journal history

GET    /api/cart                        # Get user cart
POST   /api/cart/items                  # Add to cart
PUT    /api/cart/items/:id              # Update cart item
DELETE /api/cart/items/:id              # Remove from cart

POST   /api/checkout                    # Checkout (complex process)
GET    /api/orders                      # List user orders
GET    /api/orders/:id                  # Order detail
POST   /api/orders/:id/upload-proof    # Upload payment proof
PUT    /api/orders/:id/verify-payment  # Admin verify payment
PUT    /api/orders/:id/ship             # Admin ship order
PUT    /api/orders/:id/confirm          # User confirm delivery
DELETE /api/orders/:id                  # Cancel order

GET    /api/reports/sales               # Sales report
GET    /api/reports/stock               # Stock report
```

8.2 Response Structure

```javascript
```

```
// Success
{
  success: true,
  data: { ... },
  message: "Success message"
}

// Error
{
  success: false,
  error: "Error message",
  code: "ERROR_CODE"
}

// List with pagination
{
  success: true,
  data: [ ... ],
  pagination: {
    page: 1,
    limit: 20,
    total: 100,
    totalPages: 5
  }
}
```

8.3 Validation Layers

1. **Client-side:** Form validation (required, format, length)
2. **Server-side:** express-validator untuk request validation
3. **Business Logic:** Check constraints (stock availability, authorization, etc)

---

9. Implementation Priorities

Sprint 1 (Week 1-2):

Fadil (F1):

- User registration, login, email verification
- Address CRUD
- Admin assignment ke warehouse

Aksa (F2):

- Category CRUD
- Product + ProductVariant CRUD (basic)
- Warehouse setup
- Inventory initialization

Rega (F3):

- Cart basic (add, update, remove)

Sprint 2 (Week 3-4):

Fadil (F1):

- Social login integration
- Shipping method setup

Aksa (F2):

- Stock IN/OUT transaction (with journal)
- Discount CRUD (4 tipe)
- Voucher CRUD

Rega (F3):

- Checkout flow (warehouse selection)
- Order creation
- Stock reservation

Sprint 3 (Week 4-5):

Fadil (F1):

- Profile management

Aksa (F2):

- Sales report
- Stock report
- Low stock alert

Rega (F3):

- Payment upload & verification
- Order status management
- Auto-cancel & auto-confirm (background jobs)

Sprint 4 (Week 5-6):

All Team:

- Integration testing
- Bug fixes
- Performance optimization
- Documentation finalization

---

10. Testing Checklist

Unit Testing

- [ ] Product & Variant CRUD
- [ ] Stock transaction (atomic)
- [ ] Discount calculation logic
- [ ] Voucher claim & redemption
- [ ] Warehouse distance calculation
- [ ] Order total calculation

Integration Testing

- [ ] Complete checkout flow
- [ ] Stock reservation & fulfillment
- [ ] Discount & voucher application
- [ ] Payment verification
- [ ] Order status workflow
- [ ] Auto-cancel & auto-confirm

Authorization Testing

- [ ] Super Admin vs Store Admin permissions
- [ ] Store Admin only access assigned warehouse
- [ ] User cannot access admin endpoints

Edge Cases

- [ ] Multiple users checkout same product simultaneously

- [ ] Order cancel after stock reserved
- [ ] Discount overlap handling
- [ ] Voucher usage limit exceeded
- [ ] Warehouse out of service radius
- [ ] Stock insufficient during checkout

---

11. Deployment Considerations

Environment Variables

```
DATABASE_URL=postgresql://...
JWT_SECRET=...
CLOUDINARY_URL=...
MIDTRANS_SERVER_KEY=...
MIDTRANS_CLIENT_KEY=...
RAJAONGKIR_API_KEY=...
MAIL_HOST=...
MAIL_PORT=...
MAIL_USER=...
MAIL_PASSWORD=...
```

Database Migration

```bash
npx prisma migrate deploy
npx prisma generate
```

Background Jobs Setup

- Auto-cancel orders (every 5 minutes)
- Auto-confirm orders (every 6 hours)
- Low stock alerts (daily)
- Report generation (scheduled)

Monitoring

- Stock anomaly detection (negative stock)
- Payment verification delays
- Order fulfillment time
- API response time

- Error rate per endpoint

Document Version: 1.0
Last Updated: 2025-01-08
Prepared by: Backend Team (Fadil, Aksa, Rega)