POS System - Flow & Connection Summary

System Architecture Overview

```
SUPER_ADMIN → Company → Stores → Products → Variants → Inventory → Orders
                ↓         ↓         ↓          ↓          ↓          ↓
              Users → Cashiers → Shifts → Orders → Items → Payments
```

1. AUTHENTICATION FLOW

1. User Login (POST /api/auth/login)
   ↓
2. Get JWT Token (contains: userId, role, permissions, companyId, storeId)
   ↓
3. Use Token in Headers: Authorization: Bearer <token>
   ↓
4. Access APIs based on Role & Permissions

Key Points:

- Token contains all user context (role, company, store, permissions)
- Every protected API checks token validity
- Different roles see different data (filtered by company/store)

2. COMPANY & STORE HIERARCHY

```
SUPER_ADMIN creates Company
   ↓
ADMIN creates Stores (under that Company)
   ↓
Users are assigned to Company + Store
   ↓
All data (products, inventory, orders) belongs to a Store
```

Key Points:

- **Company**: Top-level organization (e.g., "ABC Retail Chain")
- **Store**: Physical location (e.g., "ABC Store - Downtown")
- Each user works within ONE company and ONE store
- Data isolation: Users only see data from their company/store

3. PRODUCT MANAGEMENT FLOW

```
1. Create Product (base product info)
   ↓
2. Create Product Variants (sizes, colors, etc.)
   ↓
3. Create Inventory for each Variant
   ↓
4. Variants are ready for sale
```

Example:

```
Product: "T-Shirt"
  ↓
Variants:
  - Small (Red) → SKU: TS-SM-R, Barcode: 123456
  - Small (Blue) → SKU: TS-SM-B, Barcode: 123457
  - Large (Red) → SKU: TS-LG-R, Barcode: 123458
  ↓
Inventory:
  - Small (Red): 50 pcs in Store A
  - Small (Blue): 30 pcs in Store A
  - Large (Red): 20 pcs in Store A
```

Key Points:

- **Product**: Template (name, base prices, tax rate)
- **Variant**: Actual sellable item (SKU, barcode, specific price)
- **Inventory**: Stock quantity per variant per store
- Orders use **Variants**, not Products

4. PRICING LOGIC

```
Product has:
  - baseRetailPrice: 100
  - baseWholesalePrice: 80
  - taxPercent: 15

Variant can override:
  - retailPrice: 110 (or use baseRetailPrice)
  - wholesalePrice: 90 (or use baseWholesalePrice)

Order determines price by:
  - customerType: RETAIL → use retailPrice
  - customerType: WHOLESALE → use wholesalePrice

Final calculation:
  unitPrice × quantity = subtotal
  subtotal × (taxPercent/100) = taxAmount
  subtotal − discountAmount = afterDiscount
  afterDiscount + taxAmount = totalAmount
```

5. CASH SHIFT FLOW (Important!)

```
1. Cashier Opens Shift
   POST /api/shifts/open
   { "openingCash": 1000 }
   ↓
2. Shift Status: OPEN (shift records shiftId)
   ↓
3. Cashier Creates Orders during shift
   (All orders linked to this shiftId)
   ↓
4. Cashier Closes Shift
   POST /api/shifts/close
   { "closingCash": 5500 }
   ↓
5. System Calculates:
   - Expected Cash = openingCash + (cash payments during shift)
   - Difference = closingCash - expectedCash
   ↓
6. Shift Status: CLOSED
```

Key Points:

- Cashiers MUST open a shift before creating orders
- One cashier can have only ONE active shift at a time
- All orders created during shift are linked to that shift
- Closing shift shows cash discrepancies (over/short)

6. ORDER CREATION FLOW (POS Billing)

```
Step 1: Create Order (Cart)
POST /api/orders
{ "customerType": "RETAIL" }
→ Returns: orderId, status: "DRAFT"

Step 2: Add Items (Scan/Search Products)
POST /api/orders/:orderId/items
{ "productVariantId": "uuid", "quantity": 2 }
→ System checks inventory
→ Calculates prices (retail/wholesale)
→ Applies tax
→ Updates order totals

Step 3: Update Items (if needed)
PATCH /api/orders/:orderId/items/:itemId
{ "quantity": 3, "discountAmount": 10 }

Step 4: Confirm Order
POST /api/orders/:orderId/confirm
→ Status changes: DRAFT → PENDING
→ Order locked for payment

Step 5: Process Payment
POST /api/payments
{ "orderId": "uuid", "method": "CASH", "amount": 450 }
→ Inventory deducted automatically
→ Order status: PENDING → PAID

Step 6: Print Receipt
(Frontend handles printing)
```

Key Points:

- **DRAFT**: Order is a cart (can add/remove items)
- **PENDING**: Order confirmed (ready for payment)
- **PAID**: Payment received (inventory deducted)
- **CANCELLED**: Order cancelled (inventory restored if was paid)

7. INVENTORY MANAGEMENT

```
Manual Adjustment:
PATCH /api/inventory/:id/adjust
{ "adjustment": -5, "reason": "Damaged" }
→ Quantity decreased by 5

Automatic Adjustment (on order payment):
Order Payment → System automatically:
  - Deducts inventory for each item
  - Updates inventory.quantity

Low Stock Alert:
GET /api/inventory/store/:storeId/low-stock
→ Returns variants where quantity < reorderLevel

Out of Stock:
GET /api/inventory/store/:storeId/out-of-stock
→ Returns variants where quantity = 0
```

8. ROLE-BASED ACCESS FLOW

SUPER_ADMIN:

```
Can manage everything:
- Create/manage companies
- View all companies, stores, users
- Full system access
```

ADMIN (Company Level):

```
Can manage their company:
- Create/manage stores
- Create/manage users
- Manage products & inventory
- View all reports for their company
```

MANAGER (Store Level):

Can manage their store:
- Manage products & inventory
- Open/close shifts
- Create orders & process payments
- View reports for their store

CASHIER (Store Level):

Can operate POS:
- Open/close their own shifts
- Create orders
- Process payments
- Print receipts
- View only their own orders/shifts

9. TYPICAL USER FLOWS

Flow A: SUPER_ADMIN Setup

1. Login as SUPER_ADMIN
2. Create Company: "ABC Retail"
3. Create Stores: "Store A", "Store B"
4. Create Users:
   - Admin for "ABC Retail"
   - Manager for "Store A"
   - Cashier for "Store A"

Flow B: ADMIN Product Setup

1. Login as ADMIN
2. Create Product: "T-Shirt"
3. Create Variants: Small, Medium, Large
4. Set Inventory: 100, 150, 80
5. Products ready for sale

Flow C: CASHIER Daily Operation

1. Login as CASHIER
2. Open Shift: openingCash = 1000
3. Customer arrives:
   - Create Order (DRAFT)
   - Scan barcode → Add items
   - Confirm Order (PENDING)
   - Receive payment → Process Payment (PAID)
   - Print receipt
4. Repeat for all customers
5. End of day: Close Shift
   - Count cash: closingCash = 5500
   - System shows expected vs actual
   - Shift closed

Flow D: MANAGER End-of-Day Reports

1. Login as MANAGER
2. View shift summary:
   GET /api/shifts/:id/summary
   - Total sales
   - Payment breakdown (cash, card, etc.)
   - Cash discrepancies
3. View orders:
   GET /api/orders?shiftId=xxx
   - All orders for that shift
4. Check inventory:
   GET /api/inventory/store/:storeId/low-stock
   - Reorder products if needed

10. DATA RELATIONSHIPS

```
Company (1) ──┬──→ (Many) Stores
              └──→ (Many) Users
              └──→ (Many) Products

Store (1) ────┬──→ (Many) Users
              └──→ (Many) Inventory
              └──→ (Many) Orders
              └──→ (Many) Shifts

Product (1) ──→ (Many) Variants

Variant (1) ──┬──→ (Many) Inventory (one per store)
              └──→ (Many) OrderItems

User (1) ─────→ (Many) Shifts (cashier)
              └──→ (Many) Orders (cashier)

Shift (1) ────→ (Many) Orders

Order (1) ────┬──→ (Many) OrderItems
              └──→ (Many) Payments
```

11. IMPORTANT BUSINESS RULES

1. **One Active Shift Rule**:
    - Cashier can have only ONE open shift at a time
    - Must close current shift before opening new one
2. **Inventory Deduction**:
    - Inventory NOT deducted when order is created
    - Inventory deducted ONLY when payment is completed
    - If order cancelled after payment → inventory restored
3. **Price Selection**:
    - RETAIL customer → uses retailPrice
    - WHOLESALE customer → uses wholesalePrice
    - Prices can be overridden at variant level

4. **Order Status Progression**:
    - DRAFT → PENDING → PAID (normal flow)
    - Can cancel from DRAFT or PENDING (free)
    - PAID can be REFUNDED (inventory restored)
5. **Role Data Isolation**:
    - CASHIER: See only their own shifts & orders
    - MANAGER: See all data for their store
    - ADMIN: See all data for their company
    - SUPER_ADMIN: See everything
6. **Tax Calculation**:
    - Tax applied AFTER discount
    - Formula: (subtotal - discount) × (taxRate/100)
7. **Barcode Scanning**:
    - Each variant has unique barcode
    - Scan barcode → GET /api/product-variants/barcode/:barcode
    - Returns variant with current inventory
    - Add to order if stock available

12. FRONTEND INTEGRATION TIPS

Login Page:

```
POST /api/auth/login
→ Save token to localStorage/cookies
→ Save user data (role, companyId, storeId)
→ Redirect based on role:
   - SUPER_ADMIN → Company Management
   - ADMIN → Store Management
   - MANAGER → Product/Reports Dashboard
   - CASHIER → POS Screen
```

POS Screen (Cashier):

1. Check active shift:
   `GET /api/shifts/current`
   → If null: Show "Open Shift" button
   → If shift: Load shift details

2. Search products (by name/SKU):
   `GET /api/products/store/:storeId/search?q=shirt`

3. Scan barcode:
   `GET /api/product-variants/barcode/:barcode`
   → Add to cart

4. Cart management:
   - Add item: `POST /api/orders/:orderId/items`
   - Update qty: `PATCH /api/orders/:orderId/items/:itemId`
   - Remove: `DELETE /api/orders/:orderId/items/:itemId`

5. Checkout:
   - Confirm: `POST /api/orders/:orderId/confirm`
   - Payment: `POST /api/payments`
   - Print receipt

Inventory Dashboard:

1. Show all inventory:
   `GET /api/inventory/store/:storeId`

2. Low stock alerts:
   `GET /api/inventory/store/:storeId/low-stock`

3. Adjust stock:
   `PATCH /api/inventory/:id/adjust`
   `{ adjustment: -5, reason: "Damaged" }`

Reports Dashboard:

1. Daily sales:
   `GET /api/orders?from=2025-01-27&to=2025-01-27&status=PAID`

2. Shift summary:
   `GET /api/shifts/:id/summary`

3. Product performance:
   - Get all orders
   - Group by product
   - Calculate totals

QUICK REFERENCE

Authentication: JWT token in header

Base URL: `/api`

Date Format: ISO 8601 (e.g., `2025-01-27T10:00:00Z`)

Currency: Decimal (e.g., `100.50`)

Pagination: `?page=1&limit=20`

Filtering: `?status=PAID&from=2025-01-01&to=2025-01-31`

This summary shows how all parts connect. Share this with your frontend team to understand the complete system flow!