Garment’s Inventory Management System

OVERVIEW: What We’ll Build

Modules in Our System:

1. **Authentication (Login + Roles)**
2. **Product + Variants**
3. **Raw Materials**
4. **Suppliers + Purchases**
5. **Production Orders**
6. **Stock Management**
7. **Sales Orders**
8. **Reports**

1,Authentication (Login + Roles)

Purpose:

Control who can access what. This protects your data and ensures the right people can do the right things.

Features:

- **User Registration & Login**
- **Password encryption (e.g., using BCrypt)**
- **JWT-based authentication (for stateless security)**

- **Role-based access:**
  - `ADMIN` – **Full access**
  - `STORE_MANAGER` – **Access to stock & materials**
  - `PRODUCTION_OFFICER` – **Access to production orders**
  - `SALES_OFFICER` – **Access to customer orders**

2.Product + Variants

Purpose:

Manage the finished garments your factory produces.

Features:

- **Add/edit/delete products: e.g., "Shirt", "Pant", "Saree"**
- **Define variants for each product (size, color, fabric)**
- **Generate or assign SKUs (unique identifiers)**
- **Track quantity per variant**
- **Attach product images (optional)**

Connected To:

- **Production Orders (produces variants)**
- **Sales Orders (sells variants)**
- **Stock Management (manages quantity)**

Raw Materials

Purpose:

Track the ingredients used in production, like fabric, threads, buttons, etc.

Features:

- **Add new raw materials**
- **Update stock (e.g., when purchases arrive or items are used)**

- **Define units (e.g., meters, kg, pieces)**
- **Set minimum stock level and trigger alerts**
- **Categorize materials (e.g., fabric, accessories, chemicals)**

  Connected To:
- **Suppliers + Purchases (adds stock)**
- **Production Orders (reduces stock)**

. Suppliers + Purchases

Purpose:

Manage the materials you buy from external suppliers.

Features:

- **Add/manage supplier profiles**
- **Create and track Purchase Orders (POs)**
- **Record material receipts (Goods Received Notes)**
- **Automatically update raw material stock when received**
- **View purchase history by supplier or material**

Connected To:

- **Raw Materials (stock increases after receiving)**
- **Reports (track spending)**

Production Orders

Purpose:

Track the garments being made from raw materials.

Features:

- **Create production jobs (e.g., "Make 100 Large Red Cotton Shirts")**
- **Assign raw materials to each job**
- **Update job status: `CREATED`, `IN_PROGRESS`, `COMPLETED`**

- **When job is completed, update finished goods stock**
- **Track how much material was used per job**

Connected To:

- **Raw Materials (consumes stock)**
- **Product Variants (produces stock)**

6. Stock Management

Purpose:

Real-time tracking of how much inventory you have — both raw materials and finished products.

Features:

- **Monitor current stock levels**
- **Transfer stock between warehouses (if multi-location)**
- **Handle stock adjustments (damaged, returned, stolen)**
- **Show alerts for low stock**

Connected To:

- **Raw Materials + Product Variants**
- **Reports (inventory valuation, fast/slow moving items)**

7.Sales Orders

Purpose:

Record and manage orders from customers (wholesale or retail).

Features:

- **Add customer orders (with items, quantities, price)**
- **Automatically deduct stock from product variants**

- **Generate invoice**
- **Track order status: `PENDING`, `DELIVERED`, `CANCELLED`**
- **Associate order with customer**

Connected To:

- **Product Variants (reduces stock)**
- **Reports (sales summary, revenue, etc.)**

8. Reports

Purpose:

Give you insights about your garments business — like a factory manager's dashboard.

Features:

Stock Summary (raw materials + finished goods)

- **Purchase Reports (by supplier, time period)**
- **Production Reports (pending jobs, completed)**
- **Sales Reports (top-selling products, customers)**
- **Alerts (low stock, overdue POs or jobs)**

Connected To:

All modules — pulls data from across the system.

How They Work Together (Mini Flow Example):

1. **You buy raw materials → Materials come in → Raw material stock increases**
2. **You create a production order → Materials used → Finished goods produced → Raw stock decreases, finished stock increases**
3. **You get a customer order → System checks stock → Deducts quantity → Generates invoice**
4. **Reports show your sales, inventory levels, and material usage.**

Design

com.garments.inventory
├── domain → Business entities (no Spring)
├── application → Services + Interfaces + DTOs
├── infrastructure → Controllers, JPA Repos, Entity classes
├── config → BeanConfig, SecurityConfig, etc.
└── GarmentsInventoryApp → Main Spring Boot class

Domain

1.Product.java

```
public class Product {
    private UUID id;
    private String name;
    private String category;
    private List<Variant> variants;
}
```

2,Variant.java

```
public class Variant {

    private UUID id;

    private UUID productId;

    private String size;

    private String color;

    private String fabric;

    private int quantity;

}
```

3.RawMaterial.java

```
public class RawMaterial {
    private UUID id;
```

```
    private String name;
    private String unit; // e.g. meters, kg
    private int currentStock;
    private int reorderLevel;
}
```

4.ProductionOrder.java

```
public class ProductionOrder {
    private UUID id;
    private UUID productVariantId;
    private int quantity;
    private String status; // CREATED, IN_PROGRESS, COMPLETED
}
```

6.SalesOrder.java

```
public class SalesOrder {
    private UUID id;
    private String customerName;
    private List<SalesItem> items;
    private double totalAmount;
    private String status; // CONFIRMED, DELIVERED
}
```

Application

- `ProductService.java`
- `RawMaterialService.java`
- `ProductionService.java`
- DTOs: `ProductDTO`, `VariantDTO`, `SalesOrderDTO`

Infrastructure

Controllers

- `ProductController.java`
- `SalesController.java`

JPA Entity Classes

- `ProductJpaEntity`, `VariantJpaEntity`

- `SalesOrderJpaEntity`, etc.

Repositories

- Spring `JpaRepository` interfaces (e.g. `ProductJpaRepository`)
- Adapters that implement the domain's repository interfaces

CONFIGURATION LAYER

BeanConfig.java

DATABASE (PostgreSQL) TABLES

Table	Description
`products`	List of garments
`variants`	Size/color/fabric variations
`raw_materials`	Fabric, buttons, threads
`suppliers`	Raw material suppliers
`purchases`	Purchase orders
purchase_items	
`production_orders`	Tracks ongoing jobs
Sales_items	
`sales_orders`	Final product customer orders

..

`1.products`

Stores high-level info about a garment type (e.g., T-Shirt)

Field	Type	Description
id	UUID (PK)	Unique product ID
name	TEXT	Product name
category	TEXT	Men/Women/Kids/Unisex
description	TEXT	Optional longer description

Field	Type	Description
created_at	TIMESTAMP	Date of creation

2. `variants`

Variants of a product — by size, color, fabric, etc.

Field	Type	Description
id	UUID (PK)	Unique variant ID
product_id	UUID (FK)	References `products(id)`
size	TEXT	Size (S, M, L, XL, etc.)
color	TEXT	Color name (e.g., Red)
fabric	TEXT	Fabric type (e.g., Cotton)
quantity	INTEGER	Current stock level
sku	TEXT	Stock Keeping Unit (e.g., "TS-RED-M")
created_at	TIMESTAMP	Date of creation

3.`raw_materials`

Inventory of fabric, thread, buttons, zippers, etc.

Field	Type	Description
id	UUID (PK)	Unique ID
name	TEXT	Material name
unit	TEXT	e.g., meter, kg, piece
current_stock	INTEGER	How much is currently in stock
reorder_level	INTEGER	Alert threshold
category	TEXT	e.g., fabric, accessory
created_at	TIMESTAMP	Date of entry

4. `suppliers`

Stores information about raw material suppliers.

Field	Type	Description
id	UUID (PK)	Unique supplier ID

Field	Type	Description
name	TEXT	Supplier name
email	TEXT	Email address
phone	TEXT	Contact number
address	TEXT	Location/address
created_at	TIMESTAMP	Registration date

5. `purchases`

Purchase orders made to suppliers.

Field	Type	Description
id	UUID (PK)	Purchase order ID
supplier_id	UUID (FK)	References `suppliers(id)`
order_date	DATE	When the order was placed
status	TEXT	CREATED, RECEIVED, CANCELLED
total_amount	DECIMAL	Sum of all material costs
created_at	TIMESTAMP	Record creation date

6. `purchase_items`

Items inside each purchase order (many-to-one with `purchases`)

Field	Type	Description
id	UUID (PK)	Unique line item ID
purchase_id	UUID (FK)	References `purchases(id)`
material_id	UUID (FK)	References `raw_materials(id)`
quantity	INTEGER	Ordered quantity
unit_price	DECIMAL	Price per unit

7. `production_orders`

Tracks manufacturing jobs and material usage.

Field	Type	Description
id	UUID (PK)	Production job ID
variant_id	UUID (FK)	Which product variant is being produced
quantity	INTEGER	How many pieces to make
status	TEXT	CREATED, IN_PROGRESS, COMPLETED
start_date	DATE	When job started
end_date	DATE	Completion date

8. `sales_orders`

Stores customer orders for finished products.

Field	Type	Description
id	UUID (PK)	Sales order ID
customer_name	TEXT	Customer's full name
status	TEXT	PENDING, DELIVERED, CANCELLED
order_date	DATE	Order date
total_amount	DECIMAL	Total order value

9. `sales_items`

Line items in each customer sales order.

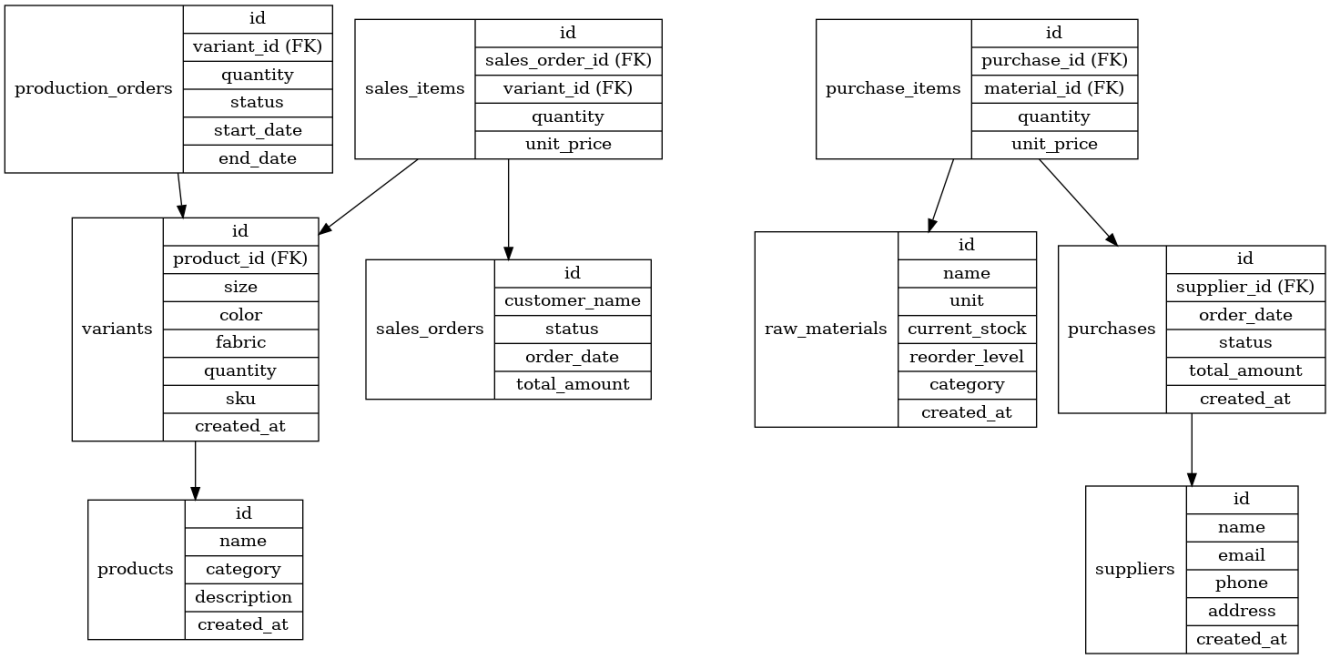
Field	Type	Description
id	UUID (PK)	Unique ID
sales_order_id	UUID (FK)	References `sales_orders(id)`
variant_id	UUID (FK)	References `variants(id)`
quantity	INTEGER	How many pieces sold
unit_price	DECIMAL	Price per piece

Constraints & Relationships Summary

- `variants.product_id → products.id`
- `purchase_items.purchase_id → purchases.id`
- `purchase_items.material_id → raw_materials.id`
- `purchases.supplier_id → suppliers.id`
- `production_orders.variant_id → variants.id`
- `sales_items.sales_order_id → sales_orders.id`
- `sales_items.variant_id → variants.id`

ER table…

Table	Included in ER Diagram?	Notes
`products`	✅ Yes	Core entity
`variants`	✅ Yes	Linked to `products` (One Product → Many Variants)
`raw_materials`	✅ Yes	Linked to `purchase_items`
`suppliers`	✅ Yes	Linked to `purchases`
`purchases`	✅ Yes	Linked to `suppliers`, referenced by `purchase_items`
`purchase_items`	✅ Yes	Junction table: `purchases` + `raw_materials`
`production_orders`	✅ Yes	Linked to `variants`
`sales_orders`	✅ Yes	Linked to `sales_items`
`sales_items`	✅ Yes	Junction table: `sales_orders` + `variants`



products ──< variants ──< production_orders

raw_materials ──< purchase_items >── purchases ──< suppliers

variants ──< sales_items >── sales_orders