📦 Case Study: Inventory Management System (IMS)

1. Overview

The **Inventory Management System (IMS)** is designed to streamline **stock tracking, supplier management, purchase orders, and sales operations**.
It enables real-time stock monitoring, generates sales reports, and issues alerts for low stock levels.
The system is **role-based** (Admin, Store Manager, Sales Staff) and secured with **JWT Authentication**.

2. Scope

- **Admin**:
  - Manage products (add/update/delete)
  - Manage suppliers
  - Handle purchase orders
- **Store Manager**:
  - Track inventory levels
  - Approve/reject restocking requests
  - Receive goods against purchase orders
- **Sales Staff**:
  - Create invoices for sales
  - Update stock levels after sales
  - View past sales

3. Tech Stack

- **Backend**: Spring Boot, Spring Security + JWT, Hibernate/JPA, Swagger (API Docs)
- **Frontend**: React + Bootstrap, Axios (API integration)
- **Database**: PostgreSQL
- **Reports & Charts**: Recharts / Chart.js

- **API Docs**: Swagger UI

4. System Architecture

- **React Frontend** → Axios → **Spring Boot REST API** → PostgreSQL
- **JWT Security**: Role-based authentication
- **Swagger Docs**: Auto-generated for all endpoints
- **Reports**: Generated by backend & displayed in charts on frontend

5. Key Features with API Integration

🔷 Authentication & Security

- **Login API**: `POST /auth/login`
- **Register Staff**: `POST /auth/register`
- JWT used for all subsequent API calls.

🔷 Product Management (Admin)

- Add Product → `POST /api/products`
- Update Product → `PUT /api/products/{id}`
- Get All Products → `GET /api/products`
- Delete Product → `DELETE /api/products/{id}`

🔷 Supplier Management

- Add Supplier → `POST /api/suppliers`
- Get Suppliers → `GET /api/suppliers`
- Update Supplier → `PUT /api/suppliers/{id}`
- Delete Supplier → `DELETE /api/suppliers/{id}`

🔷 Purchase Order Management

- Create PO → `POST /api/purchase-orders`
- Approve PO (Manager) → `PUT /api/purchase-orders/{id}/approve`
- Receive Goods → `PUT /api/purchase-orders/{id}/receive`
- List POs → `GET /api/purchase-orders`

🔷 Inventory Tracking

- Check Stock Levels → `GET /api/inventory`
- Low Stock Alert → `GET /api/inventory/low-stock`
- Update Stock → `PUT /api/inventory/{productId}`

🔷 Sales Management

- Create Invoice → `POST /api/sales`
- Get Sales History → `GET /api/sales/history`
- Generate Report → `GET /api/reports/sales`

🔷 Reports

- Daily Sales Report → `GET /api/reports/sales/daily`
- Monthly Stock Trends → `GET /api/reports/stock/monthly`
- Most Sold Products → `GET /api/reports/top-products`

6. Swagger Integration

- Integrated using **springdoc-openapi-ui** dependency.
- Endpoint: `http://localhost:8080/swagger-ui.html`
- Provides **API documentation** with request/response models.

7. Frontend (React + Bootstrap)

- **Login/Register Page**: JWT stored in `localStorage`

- **Dashboard**: Different UI for Admin, Manager, Sales Staff
- **Product Management Page**: CRUD with Bootstrap forms
- **Inventory Dashboard**: Real-time stock visualization (Recharts)
- **Sales Page**: Create invoices & track history

8. Sample Folder Structure

📂 Backend (Spring Boot + JWT + Swagger)

```
ims-backend/
 ├── controller       # REST Controllers
 ├── service          # Business logic
 ├── repository       # JPA Repositories
 ├── model            # Entities (Product, Supplier,
PurchaseOrder, Sale, User)
 ├── security         # JWT configs & filters
 ├── exception        # Custom exceptions
 └── resources
     └── application.yml
```

📂 Frontend (React + Bootstrap)

```
ims-frontend/
 ├── src/
 │    ├── components/
 │    │    ├── Login.js
 │    │    ├── Dashboard.js
 │    │    ├── ProductList.js
 │    │    ├── Inventory.js
 │    │    ├── Sales.js
 │    │    └── Reports.js
 │    ├── services/ (Axios API calls)
 │    └── App.js
```

9. Security (JWT Implementation)

- **Login** generates JWT → stored in browser `localStorage`.
- **Axios Interceptors** → attach JWT in `Authorization: Bearer <token>` headers.

- **Role-based access**:
  - `/api/admin/**` → Admin only
  - `/api/manager/**` → Store Manager only
  - `/api/sales/**` → Sales Staff only

10. Benefits

✅ Real-time inventory visibility
✅ Secure role-based access with JWT
✅ Auto-documented APIs with Swagger
✅ Easy-to-use UI with React + Bootstrap
✅ Sales & stock trend reports with charts
✅ Prevents over/under-stocking with alerts