| **Module** | **Endpoint** | **Method** | **Access** |
| --- | --- | --- | --- |
| Auth | /api/auth/login | POST | Public |
| Products | /api/products | GET/POST/PUT/DELETE | Admin |
| Invoices | /api/invoices | POST/GET | Admin |
| Customers | /api/customers | GET/POST | Admin |
| Payments | /api/payments | POST | Admin |
| Reports | /api/reports/sales | GET | Admin |
| Barcode Lookup | /api/products/barcode/{code} | GET | Admin |

Example API Endpoints Summary

🧱 **Technology Stack Summary**

| **Layer** | **Technology Used** |
| --- | --- |
| **Backend** | Spring Boot, Spring Data JPA |
| **Database** | MySQL or PostgreSQL |
| **Security** | Spring Security + JWT |
| **Frontend** | React or Angular |
| **Templating (if server-side UI)** | Thymeleaf |
| **PDF/Barcode** | iText, ZXing |
| **Payment Gateway** | Razorpay / Stripe |

**Complete Backend Flow Diagram (Textual Overview)**

[Frontend (React/Angular)]

|

(API Calls via JSON)

|

v

[Spring Boot Controllers]

|

v

[Service Layer - Business Logic]

|

v

[Repositories (Spring Data JPA)]

|

v

[Database (MySQL/PostgreSQL)]

## 1. ****Security (Login/Logout & Role Management)****

### 🔁 **Authentication Flow:**

1. Admin logs in

POST /api/auth/login

|

2. JWT token issued (Spring Security)

|

3. Token attached to headers for all future requests

|

4. Spring Security filter validates token on every API hit

|

5. Role-based access (ADMIN, CASHIER)

✔ Uses: Spring Security + JWT + BCryptPasswordEncoder

## 🧾 2. ****Invoice Generation (With GST)****

### 🧮 **Flow:**

1. Admin selects products, quantities, and customer

2. POST /api/invoices

3. Backend calculates:

- Total

- GST (e.g., 3% for gold)

- Net amount

4. Saves invoice and invoice items

5. Reduces product stock

6. Returns InvoiceResponse

✅ Entities: Invoice, InvoiceItem, Customer, Product  
✅ GST Logic in InvoiceService  
✅ PDF invoice (optional) using libraries like iText

## 📦 3. ****Stock Management****

### 🔄 **Operations:**

- Add new items: POST /api/products

- Update quantity/price: PUT /api/products/{id}

- Track inventory: GET /api/products

- Automatically reduce stock on invoice generation

✅ Product Entity: name, type, metalType, weight, price, quantityInStock

## 👤 4. ****Customer Management****

### 💼 **Features:**

- Add new customers: POST /api/customers

- View customer list: GET /api/customers

- Fetch customer by phone/email: GET /api/customers/search

✅ Useful for repeat customers and loyalty tracking

## 📊 5. ****Reporting****

### 🧾 **Reports:**

- Daily/Monthly Sales: GET /api/reports/sales

- Low Stock Alert: GET /api/reports/stock

- Customer-wise Sales: GET /api/reports/customers

✅ Implement using custom JPQL queries or DTOs  
✅ Optionally export as Excel/PDF

## 🧾 6. ****API for Barcode Integration****

### 🔍 **Barcode Workflow:**

- Each product has a unique barcode

- Barcode stored in Product Entity

- API: GET /api/products/barcode/{code}

- Barcode scanner sends scanned code to fetch product

✅ Can integrate libraries like ZXing (Java) or use 3rd-party scanner hardware  
✅ Useful for fast billing at the counter

## 💳 7. ****Payment Integration****

### 🪙 **Payment Options:**

- Cash

- UPI (Google Pay, PhonePe)

- Card

- EMI (Optional)

API Flow:

POST /api/payments

- Invoice ID

- Payment mode

- Amount paid

✅ Entity: Payment linked to Invoice  
✅ Option to integrate Razorpay/Stripe SDK for online payment

## 🖥️ 8. ****User Interface (Frontend-Backed Integration)****

### UI Modules:

* **Login Page** (with token handling)
* **Dashboard** (sales summary, inventory alerts)
* **Invoice Form** (product selection, customer dropdown)
* **Stock Management** (add/update/view products)
* **Customer CRUD**
* **Report Viewer**

✅ Tech Stack: Angular or React (with Axios or Fetch)  
✅ RESTful API calls to Spring Boot backend