**Smart Label System - Backend Documentation**

**1. Project Overview**

This backend powers the Smart Label System, managing:

• User accounts and authentication

• Product and category management

• Price update tracking and history

• Regulatory alert generation

• Sales recording

• Report generation and export

**Backend Stack**

• **Runtime**: Node.js

• **Framework**: Express.js

• **ORM**: Sequelize

• **Database**: MySQL

• **Auth**: JWT-based authentication

⸻

**2. API Modules Overview**

**2.1 Authentication Module**

**Path**: /api/auth

**Endpoints**:

• POST /register: Register a new user (Status: ✅ Implemented)

• POST /login: Authenticate user and return JWT token (Status: ✅ Implemented)

**Roles**:

• Admin

• Staff

• Regulatory

**Token Usage**:

• **Header**: Authorization: Bearer <token>

**Middleware**:

• **authenticate**: Validates JWT token (Status: ✅ Implemented)

• **authorize([roles])**: Restricts access to specific user roles (Status: ✅ Implemented)

⸻

**2.2 Product Management Module**

**Path**: /api/products

**Endpoints**:

• POST /: Add a new product (Admin, Staff) (Status: ✅ Implemented)

• GET /: Retrieve all products (Authenticated) (Status: ✅ Implemented)

• PUT /:id/price: Update a product’s price and log the change (Admin, Staff) (Status: ✅ Implemented)

**Product Fields**:

• Name

• Description

• NutritionalFacts

• CategoryID

• Price

• ExpiryDate

• StockQuantity

**Price Update Logging**:

• Triggers a new record in PriceChanges

• Tracks OldPrice, NewPrice, ChangedByUserID, ChangeDateTime (Status: ✅ Implemented)

⸻

**2.3 Category Management Module**

**Path**: /api/categories

**Endpoints**:

• POST /: Create a new category (Admin only) (Status: ✅ Implemented)

• GET /: List all categories (Authenticated users) (Status: ✅ Implemented)

**Controller**: category.controller.js

• **createCategory**: Create a new category (Status: ✅ Implemented)

• **getAllCategories**: Retrieve all categories (Status: ✅ Implemented)

**Routes**: category.routes.js

• Uses authenticate and authorize(['Admin']) middleware (Status: ✅ Implemented)

**Linked Model**:

• ProductCategory: Used in product creation and regulatory limits (Status: ✅ Implemented)

⸻

**2.4 Price Change Tracking Module**

**Status**: ✅ Implemented

**Purpose**:

• Logs every product price update

• Data stored in PriceChanges table

• Enables rollback, audits, and historical pricing reports

**Data Fields**:

• ProductID

• OldPrice

• NewPrice

• ChangeDateTime

• ChangedByUserID

**Related Endpoints**:

• PUT /api/products/:id/price: Update product price (Logs price change) (Status: ✅ Implemented)

⸻

**2.5 Regulatory Alert System**

**Status**: ✅ Implemented

**Purpose**:

• Automatically creates alerts when product price exceeds category’s max allowed price

• Used to enforce pricing compliance in real-time

**Logic**:

• Checks RegulatoryLimits for the product category

• If Product.Price > MaxPrice → creates Alert of type PriceExceeded

• Alert is saved with Status = Pending

**Related Tables**:

• RegulatoryLimits

• Alerts

**Endpoint**:

• GET /api/alerts: List all alerts (Admin, Regulatory) (Status: ✅ Implemented)

• Optional query: ?status=Pending (Status: ✅ Implemented)

⸻

**2.6 Sales Logging Module**

**Status**: ✅ Implemented

**Purpose**:

• Record each sale of a product

• Helps in stock management and sales reports

**Planned Endpoints**:

• POST /api/sales: Log a sale (Admin, Staff) (Status: ✅ Implemented)

• GET /api/sales: List sales history (Authenticated) (Status: ✅ Implemented)

**Related Table**:

• Sales: Stores sale details (Status: ✅ Implemented)

⸻

**2.7 Report Generation Module**

**Status**: ✅ Implemented

**Purpose**:

• Export reports for:

• Pricing history

• Compliance checks

• Sales trends

**Fields**:

• ReportType

• GeneratedByUserID

• GeneratedDateTime

• FilePath

**Endpoint**:

• POST /api/reports: Generate a new report (Admin only) (Status: ✅ Implemented)

• GET /api/reports: Retrieve all reports (Authenticated) (Status: ✅ Implemented)

⸻

**3. Sequelize Models Summary**

**Users**

• UserID

• UserName

• Password

• Email

• Role

**ProductCategories**

• CategoryID

• CategoryName

**Products**

• ProductID

• Name

• Description

• NutritionalFacts

• CategoryID

• Price

• ExpiryDate

• StockQuantity

**PriceChanges**

• ChangeID

• ProductID

• OldPrice

• NewPrice

• ChangeDateTime

• ChangedByUserID

**RegulatoryLimits**

• LimitID

• ProductCategoryID

• MaxPrice

• RegulationDate

**Alerts**

• AlertID

• ProductID

• AlertType

• AlertDateTime

• Status

**Sales**

• SaleID

• ProductID

• Quantity

• SaleDateTime

**Reports**

• ReportID

• GeneratedByUserID

• ReportType

• GeneratedDateTime

• FilePath

⸻

**4. Environment Configuration (.env)**

PORT=5050

DB\_HOST=localhost

DB\_USER=smartuser

DB\_PASSWORD=smartpassword

DB\_NAME=smart\_label\_db

JWT\_SECRET=smartlabelsecret

⸻

**5. Project Commands**

• **Start dev server**:

npm run dev

• **Start production server**:

npm run start

• **Access MySQL**:

mysql -u root -p

• **Drop table (if needed)**:

DROP TABLE table\_name;

⸻

**6. Next Development Goals**

• Implement more granular role-based access control for users.

• Enhance report generation capabilities with additional formats (e.g., CSV, Excel).

• Expand sales tracking with customer information.

• Optimize database queries for large datasets.

• Integrate with external systems (e.g., inventory management or POS systems).

⸻

**Maintained by: Backend Developer - Smart Label System**

**Repository: (to be connected)**

**Documentation updated: April 8, 2025**

⸻