# **Low-Level Design (LLD) Document - Product Management**

## **1. Introduction**

This document provides a detailed low-level design for the **Product Management** module of the e-commerce application. It outlines the internal structure, data flow, API contracts, and component interactions for managing products within the system.

## **2. Module Overview**

The Product Management module handles the core operations for product creation, updating, deletion, and retrieval. It also includes support for categories, pricing, and inventory management.

## **3. Key Components**

### **3.1 Frontend Components (Angular)**

* **ProductCardComponent:** Displays product information.
* **ProductListComponent:** Lists all available products.
* **ProductDetailComponent:** Shows detailed information about a specific product.
* **ProductFormComponent:** Used for adding and editing products.

### **3.2 State Management (NgRx)**

* **Actions:** LoadProducts, AddProduct, UpdateProduct, DeleteProduct
* **Reducers:** Handles state updates based on actions
* **Effects:** Manages side effects like API calls
* **Selectors:** Extracts specific slices of the state

## **4. Database Schema (MongoDB)**

### **4.1 Product Collection**

{

"\_id": "ObjectId",

"name": "String",

"description": "String",

"category": "String",

"price": "Number",

"images": ["String"],

"stock": "Number",

"rating": "Number",

"createdAt": "Date",

"updatedAt": "Date"

}

## **5. API Contracts (Node.js + Express)**

### **5.1 Product APIs**

* **GET /api/products** - Fetch all products
* **GET /api/products/:id** - Fetch a single product by ID
* **POST /api/products** - Create a new product
* **PUT /api/products/:id** - Update a product
* **DELETE /api/products/:id** - Delete a product

### **5.2 Sample Payloads**

#### **Request (Create Product)**

{

"name": "Laptop",

"description": "Powerful gaming laptop",

"category": "Electronics",

"price": 1200.00,

"images": ["image1.jpg", "image2.jpg"],

"stock": 50

}

#### **Response (Product List)**

[

{

"\_id": "60f71792b65f1a001c5a7b8f",

"name": "Laptop",

"category": "Electronics",

"price": 1200.00,

"stock": 50

},

{

"\_id": "60f71792b65f1a001c5a7b90",

"name": "Headphones",

"category": "Electronics",

"price": 150.00,

"stock": 200

}

]

## **6. Data Flow**

1. User requests product data -> ProductService fetches data from API -> NgRx Store is updated -> ProductListComponent displays the data.
2. Add Product -> ProductFormComponent dispatches action -> NgRx Effect sends API request -> Store is updated.
3. Update Product -> Same as Add Product but with PUT request.
4. Delete Product -> Dispatch action -> Effect sends DELETE request -> Store is updated.

## **7. Error Handling**

* Validate request payloads before database operations
* Return meaningful error messages
* Use try-catch blocks in service methods

## **8. Security Considerations**

* JWT authentication for all API routes
* Role-based access control (Admin only for product creation, update, and deletion)

## **9. Testing Strategy**

* Unit testing for components
* Integration testing for API endpoints
* Mock data for frontend components

## **10. Future Enhancements**

* Add product reviews and ratings
* Real-time stock updates
* Image optimization and caching