**US-1: Create Product API**

**As a** product manager  
 **I want to** create new products in the system  
 **So that** I can add new items to our inventory

**Acceptance Criteria:**

* API should accept product details (name, SKU, price, stock quantity, category, description)
* System should validate all required fields
* System should check for duplicate SKUs
* System should assign creation timestamp and user
* Should return the created product with generated ID
* Should create an audit log entry

**Technical Criteria:**

* Endpoint: POST /api/v1/products
* Request validation for all fields
* Response Code: 201 Created
* Security: Requires ADMIN role
* Rate Limit: 100 requests per minute

**US-2: Retrieve Products API**

**As a** system user  
 **I want to** view all products with pagination  
 **So that** I can browse through the product catalog efficiently

**Acceptance Criteria:**

* API should return paginated list of products
* Should support sorting by different fields
* Should support filtering by category
* Should include total count of products
* Should allow customizing page size

**Technical Criteria:**

* Endpoint: GET /api/v1/products
* Query Parameters: page, size, sortBy, category
* Response Code: 200 OK
* No authentication required
* Rate Limit: 1000 requests per minute

**US-3: Update Product API**

**As a** product manager  
 **I want to** update existing product details  
 **So that** I can maintain accurate product information

**Acceptance Criteria:**

* API should allow updating any product field
* System should validate all fields
* Should maintain update history
* Should prevent concurrent updates
* Should verify product exists

**Technical Criteria:**

* Endpoint: PUT /api/v1/products/{id}
* Optimistic locking for concurrency
* Response Code: 200 OK
* Security: Requires ADMIN role
* Rate Limit: 100 requests per minute

**US-4: Delete Product API**

**As a** product manager  
 **I want to** remove products from the system  
 **So that** I can maintain a clean product catalog

**Acceptance Criteria:**

* API should soft delete the product
* Should maintain deletion history
* Should verify product exists
* Should prevent deletion of products with active orders
* Should create audit log entry

**Technical Criteria:**

* Endpoint: DELETE /api/v1/products/{id}
* Response Code: 204 No Content
* Security: Requires ADMIN role
* Rate Limit: 50 requests per minute

**US-5: Product Search API**

**As a** system user  
 **I want to** search products by various criteria  
 **So that** I can find specific products quickly

**Acceptance Criteria:**

* API should support search by name, SKU, category
* Should support partial matches
* Should support price range filtering
* Should return paginated results
* Should support sorting of results

**Technical Criteria:**

* Endpoint: GET /api/v1/products/search
* Query Parameters: q, category, minPrice, maxPrice, page, size
* Response Code: 200 OK
* No authentication required
* Rate Limit: 500 requests per minute

# Product Management System Case Study

## Business Requirements

A retail company needs a Product Management System with the following capabilities:

* Manage product inventory (CRUD operations)
* Track product categories
* Handle product pricing and stock updates
* Maintain audit logs for all operations
* Support bulk operations
* Implement role-based access control

## Technical Requirements

### Architecture Overview

* Spring Boot 3.x
* Java 17
* MySQL Database
* Spring Data JPA
* Spring Security
* Spring Validation
* Lombok
* Swagger/OpenAPI for documentation

### Database Design

#### Tables

1. products

CREATE TABLE products (

    id BIGINT PRIMARY KEY AUTO\_INCREMENT,

    name VARCHAR(100) NOT NULL,

    description TEXT,

    sku VARCHAR(50) UNIQUE NOT NULL,

    price DECIMAL(10,2) NOT NULL,

    stock\_quantity INT NOT NULL,

    category\_id BIGINT,

    created\_at TIMESTAMP,

    updated\_at TIMESTAMP,

    created\_by VARCHAR(50),

    updated\_by VARCHAR(50)

);

Categories  
  
CREATE TABLE categories (

    id BIGINT PRIMARY KEY AUTO\_INCREMENT,

    name VARCHAR(50) NOT NULL UNIQUE,

    description TEXT

);

Audit\_logs  
  
CREATE TABLE audit\_logs (

    id BIGINT PRIMARY KEY AUTO\_INCREMENT,

    entity\_type VARCHAR(50),

    entity\_id BIGINT,

    action VARCHAR(20),

    changes JSON,

    performed\_by VARCHAR(50),

    performed\_at TIMESTAMP

);

**API Documentation**

### Endpoints

1. Create Product

POST /api/v1/products

Authorization: Required (ADMIN role)

Request Body: ProductDTO

Response: ProductResponse

1. Get All Products

GET /api/v1/products?page=0&size=10&sortBy=id

Authorization: Not required

Response: Page<ProductResponse>

1. Get Product by ID

GET /api/v1/products/{id}

Authorization: Not required

Response: ProductResponse

1. Update Product

PUT /api/v1/products/{id}

Authorization: Required (ADMIN role)

Request Body: ProductDTO

Response: ProductResponse

1. Delete Product

DELETE /api/v1/products/{id}

Authorization: Required (ADMIN role)

Response: 204 No Content

The case study follows Spring Boot best practices and includes:

* JPA entities with proper relationships
* DTO pattern for request/response handling
* Global exception handling
* Audit logging
* Pagination and sorting
* OpenAPI documentation [Optional]
* Comprehensive validation [Optional]

**Implementation Steps**

1. Project Setup (30 minutes)

* Create Spring Boot project
* Configure dependencies
* Set up database configuration

1. Entity and Repository Layer (30 minutes)

* Implement entities with validations
* Create repository interfaces
* Set up audit logging

1. Service Layer (45 minutes)

* Implement business logic
* Add validation handling
* Integrate audit logging

1. Controller Layer (30 minutes)

* Create REST endpoints
* Add security configurations [Optional]
* Implement exception handling

1. ~~Testing (45 minutes)~~

* ~~Write unit tests~~
* ~~Create integration tests~~
* ~~Perform manual testing~~

**Monitoring and Logging  [30 th Jan Task]**

1. Actuator Endpoints

* Health checks
* Metrics
* Application info

1. Logging

* Request/Response logging
* Error logging
* Audit logging

# Product Management API User Stories

## Epic: Product Management System

As a retail business owner, I want a robust product management system so that I can efficiently manage my product inventory and sales operations.

### US-1: Create Product API

**As a** product manager  
 **I want to** create new products in the system  
 **So that** I can add new items to our inventory

**Acceptance Criteria:**

* API should accept product details (name, SKU, price, stock quantity, category, description)
* System should validate all required fields
* System should check for duplicate SKUs
* System should assign creation timestamp and user
* Should return the created product with generated ID
* Should create an audit log entry

**Technical Criteria:**

* Endpoint: POST /api/v1/products
* Request validation for all fields
* Response Code: 201 Created
* Security: Requires ADMIN role
* Rate Limit: 100 requests per minute

### US-2: Retrieve Products API

**As a** system user  
 **I want to** view all products with pagination  
 **So that** I can browse through the product catalog efficiently

**Acceptance Criteria:**

* API should return paginated list of products
* Should support sorting by different fields
* Should support filtering by category
* Should include total count of products
* Should allow customizing page size

**Technical Criteria:**

* Endpoint: GET /api/v1/products
* Query Parameters: page, size, sortBy, category
* Response Code: 200 OK
* No authentication required
* Rate Limit: 1000 requests per minute

### US-3: Update Product API

**As a** product manager  
 **I want to** update existing product details  
 **So that** I can maintain accurate product information

**Acceptance Criteria:**

* API should allow updating any product field
* System should validate all fields
* Should maintain update history
* Should prevent concurrent updates
* Should verify product exists

**Technical Criteria:**

* Endpoint: PUT /api/v1/products/{id}
* Optimistic locking for concurrency
* Response Code: 200 OK
* Security: Requires ADMIN role
* Rate Limit: 100 requests per minute

### US-4: Delete Product API

**As a** product manager  
 **I want to** remove products from the system  
 **So that** I can maintain a clean product catalog

**Acceptance Criteria:**

* API should soft delete the product
* Should maintain deletion history
* Should verify product exists
* Should prevent deletion of products with active orders
* Should create audit log entry

**Technical Criteria:**

* Endpoint: DELETE /api/v1/products/{id}
* Response Code: 204 No Content
* Security: Requires ADMIN role
* Rate Limit: 50 requests per minute

### ~~US-5: Bulk Product Upload API~~

**~~As a~~** ~~product manager~~ **~~I want to~~** ~~upload multiple products at once~~ **~~So that~~** ~~I can efficiently add new product lines~~

**~~Acceptance Criteria:~~**

* ~~API should accept CSV/Excel file~~
* ~~Should validate all records before processing~~
* ~~Should provide detailed error report~~
* ~~Should be transactional (all or nothing)~~
* ~~Should handle duplicate SKUs~~

**~~Technical Criteria:~~**

* ~~Endpoint: POST /api/v1/products/bulk~~
* ~~Multipart file upload~~
* ~~Response Code: 202 Accepted~~
* ~~Security: Requires ADMIN role~~
* ~~Rate Limit: 10 requests per minute~~

### US-6: Product Search API

**As a** system user  
 **I want to** search products by various criteria  
 **So that** I can find specific products quickly

**Acceptance Criteria:**

* API should support search by name, SKU, category
* Should support partial matches
* Should support price range filtering
* Should return paginated results
* Should support sorting of results

**Technical Criteria:**

* Endpoint: GET /api/v1/products/search
* Query Parameters: q, category, minPrice, maxPrice, page, size
* Response Code: 200 OK
* No authentication required
* Rate Limit: 500 requests per minute

## Error Scenarios

### Create/Update Product

* Duplicate SKU
* Invalid category
* Invalid price format
* Missing required fields
* Concurrent update conflicts

### Retrieve Products

* Invalid pagination parameters
* Invalid sort field
* Non-existent category filter

### Delete Product

* Product not found
* Product has dependencies
* Insufficient permissions

### Bulk Upload

* Invalid file format
* Data validation errors
* File size too large
* Duplicate records
* Wrapper classes
* JPA Working
* Dependency injection – why constructor?
* @Service

CREATE TABLE products (

id BIGINT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

description TEXT,

sku VARCHAR(50) UNIQUE NOT NULL,

price DECIMAL(10,2) NOT NULL,

stock\_quantity INT NOT NULL,

category\_id BIGINT,

created\_at TIMESTAMP,

updated\_at TIMESTAMP,

created\_by VARCHAR(50),

updated\_by VARCHAR(50),

is\_Deleted boolean DEFAULT FALSE,

CONSTRAINT fk\_products\_category FOREIGN KEY (category\_id) REFERENCES categories(id) ON DELETE SET NULL

);

insert into products(id,name,description ,sku,price,stock\_quantity,

category\_id,created\_at,updated\_at,created\_by,updated\_by)

values (1,'ball','football','199','500','50','1','2000-01-01','2001-01-01','john','Doe');

select \* from products;

CREATE TABLE categories (

id BIGINT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL UNIQUE,

description TEXT

);

CREATE TABLE audit\_logs (

id BIGINT PRIMARY KEY AUTO\_INCREMENT,

entity\_type VARCHAR(50),

entity\_id BIGINT,

action VARCHAR(20),

changes JSON,

performed\_by VARCHAR(50),

performed\_at TIMESTAMP

);