Product Management API Task

A REST API service for managing products with role-based access control, comprehensive validation, and statistical reporting.

Tech Stack Options

Category	Options	Status
Runtime	Node.js	Required
API Style	REST API	Required
Language	JavaScript, TypeScript	Optional
Framework	Express.js, Fastify, NestJS	Optional
Database	MongoDB, PostgreSQL, MySQL, SQLite, In-Memory	Optional
ORM/ODM	Mongoose, Sequelize, Prisma, TypeORM, None	Optional
Validation	Joi, Zod, Ajv, express-validator, class-validator	Optional
Caching	node-cache, Redis, None	Optional
API Documentation	Swagger/OpenAPI, Postman Collection, HTTP files	Optional
Testing	Jest, Mocha, Supertest, None	Optional

Project Goals

Build a REST API service to manage products and provide aggregated statistics with the following requirements:

- ✅ **Role-Based Access Control** (mocked via headers)
- ✅ **Robust Validation** (prevent invalid data from breaking the server)
- ✅ **Comprehensive Error Handling** (proper HTTP status codes)
- ✅ **Unified Response Format** (consistent success/error responses)
- ✅ **Pagination** (mandatory for listing endpoints)
- ✅ **Console Logging** (log successful operations)
- ❌ **No AI Usage** (must be implemented manually)

Data Model

Product Schema

Field	Type	Required	Default	Constraints
id	Auto-generated	Yes	Auto-generated	Primary key (Integer/UUID/ObjectId)
sku	String	Yes	-	**UNIQUE**, alphanumeric, max 50 chars
name	String	Yes	-	Min 3 chars, max 200 chars
description	String	No	null	Max 1000 chars
category	String	Yes	-	Min 2 chars, max 100 chars
type	Enum	Yes	"public"	Values: "public" or "private"
price	Number (Decimal)	Yes	-	Must be > 0, max 2 decimal places
discountPrice	Number (Decimal)	No	null	If provided: must be ≥ 0 AND < price
quantity	Integer	Yes	-	Must be ≥ 0
createdAt	Timestamp	Yes	Auto-generated	ISO 8601 format
updatedAt	Timestamp	Yes	Auto-updated	ISO 8601 format

Validation Rules Summary

- **sku**: Required, unique, alphanumeric (A-Z, a-z, 0-9, hyphens, underscores), 3-50 chars
- **name**: Required, 3-200 chars, trimmed
- **description**: Optional, max 1000 chars
- **category**: Required, 2-100 chars, trimmed
- **type**: Required, must be exactly "public" or "private" (case-sensitive)
- **price**: Required, positive number > 0, max 2 decimal places
- **discountPrice**: Optional, if provided: must be ≥ 0 and strictly less than price
- **quantity**: Required, non-negative integer ≥ 0

API Routes

1. Create Product

```
POST /api/products
Headers:
  X-User-Role: admin
  Content-Type: application/json
Body:
{
  "sku": "LAPTOP-001",
  "name": "Gaming Laptop",
  "description": "High-performance gaming laptop",
  "category": "Electronics",
  "type": "public",
  "price": 1299.99,
  "discountPrice": 1099.99,
  "quantity": 50
}

Response: 201 Created
{
  "success": true,
  "message": "Product created successfully",
  "data": {
    "id": "123",
    "sku": "LAPTOP-001",
    "name": "Gaming Laptop",
    "description": "High-performance gaming laptop",
    "category": "Electronics",
    "type": "public",
    "price": 1299.99,
    "discountPrice": 1099.99,
    "quantity": 50,
    "createdAt": "2025-01-15T10:30:00Z",
    "updatedAt": "2025-01-15T10:30:00Z"
  }
}
```

Access Control:

- ✅ Admin only
- ❌ User: 403 Forbidden
- ❌ No role: 401 Unauthorized

---

2. Get All Products (with Pagination)

```
GET /api/products?
page=1&limit=10&category=Electronics&type=public&search=laptop&sort=price&
order=asc&minPrice=100&maxPrice=2000
Headers:
  X-User-Role: user|admin
```

```
Response: 200 OK
{
  "success": true,
  "message": "Products retrieved successfully",
  "data": [
    {
      "id": "123",
      "sku": "LAPTOP-001",
      "name": "Gaming Laptop",
      "category": "Electronics",
      "type": "public",
      "price": 1299.99,
      "discountPrice": 1099.99,
      "quantity": 50,
      "createdAt": "2025-01-15T10:30:00Z",
      "updatedAt": "2025-01-15T10:30:00Z"
    }
  ],
  "pagination": {
    "currentPage": 1,
    "totalPages": 5,
    "totalItems": 48,
    "itemsPerPage": 10,
    "hasNextPage": true,
    "hasPreviousPage": false
  }
}
```

Query Parameters:

Parameter	Type	Default	Description
page	Integer	1	Page number (min: 1)
limit	Integer	10	Items per page (min: 1, max: 100)
category	String	-	Filter by exact category name
type	String	-	Filter by type ("public" or "private")
search	String	-	Search in name and description (case-insensitive)
sort	String	-	Sort field (name, price, quantity, createdAt)
order	String	asc	Sort order ("asc" or "desc")
minPrice	Number	-	Minimum price filter
maxPrice	Number	-	Maximum price filter

Access Control:

- ✅ Admin: Sees ALL products

- ✅ User: Sees ONLY **public** products (automatic filter applied)
- ❌ No role: 401 Unauthorized

---

3. Get Single Product

```
GET /api/products/:id
Headers:
  X-User-Role: user|admin

Response: 200 OK
{
  "success": true,
  "message": "Product retrieved successfully",
  "data": {
    "id": "123",
    "sku": "LAPTOP-001",
    "name": "Gaming Laptop",
    "description": "High-performance gaming laptop",
    "category": "Electronics",
    "type": "public",
    "price": 1299.99,
    "discountPrice": 1099.99,
    "quantity": 50,
    "createdAt": "2025-01-15T10:30:00Z",
    "updatedAt": "2025-01-15T10:30:00Z"
  }
}
```

Access Control:

- ✅ Admin: Can view ANY product
- ✅ User: Can view ONLY **public** products (404 for private products)
- ❌ No role: 401 Unauthorized

---

4. Update Product

```
PUT /api/products/:id
Headers:
  X-User-Role: admin
  Content-Type: application/json
Body:
{
  "name": "Updated Gaming Laptop",
  "price": 1399.99,
  "quantity": 45
}

Response: 200 OK
{
  "success": true,
  "message": "Product updated successfully",
  "data": {
    "id": "123",
    "sku": "LAPTOP-001",
    "name": "Updated Gaming Laptop",
    "description": "High-performance gaming laptop",
    "category": "Electronics",
    "type": "public",
    "price": 1399.99,
    "discountPrice": 1099.99,
    "quantity": 45,
    "createdAt": "2025-01-15T10:30:00Z",
    "updatedAt": "2025-01-15T11:45:00Z"
  }
}
```

Notes:

- SKU cannot be updated (immutable after creation)
- Partial updates allowed (only send fields to update)
- All validation rules apply

Access Control:

- ✅ Admin only
- ❌ User: 403 Forbidden
- ❌ No role: 401 Unauthorized

---

5. Delete Product

```
DELETE /api/products/:id
Headers:
  X-User-Role: admin

Response: 200 OK
{
  "success": true,
  "message": "Product deleted successfully",
  "data": {
    "id": "123",
    "sku": "LAPTOP-001"
  }
}
```

Access Control:

- ✅ Admin only
- ❌ User: 403 Forbidden
- ❌ No role: 401 Unauthorized

---

6. Get Product Statistics

```
GET /api/products/stats
Headers:
  X-User-Role: admin

Response: 200 OK
{
  "success": true,
  "message": "Statistics retrieved successfully",
  "data": {
    "totalProducts": 150,
    "totalInventoryValue": 125000.50,
    "totalDiscountedValue": 98000.00,
    "averagePrice": 833.34,
    "outOfStockCount": 12,
    "productsByCategory": [
      {
        "category": "Electronics",
        "count": 85,
        "totalValue": 95000.00
      },
      {
        "category": "Clothing",
        "count": 65,
        "totalValue": 30000.50
      }
    ],
    "productsByType": [
      {
        "type": "public",
        "count": 120,
        "totalValue": 100000.00
      },
      {
        "type": "private",
        "count": 30,
        "totalValue": 25000.50
      }
    ]
  }
}
```

Statistics Calculations:

- **totalProducts**: Total count of all products
- **totalInventoryValue**: Sum of (price × quantity) for all products
- **totalDiscountedValue**: Sum of (discountPrice × quantity) where discountPrice exists
- **averagePrice**: Average product price (totalInventoryValue / totalProducts)
- **outOfStockCount**: Count of products where quantity = 0
- **productsByCategory**: Breakdown by category with count and total value
- **productsByType**: Breakdown by type (public/private) with count and total value

Access Control:

- ✅ Admin only
- ❌ User: 403 Forbidden
- ❌ No role: 401 Unauthorized

Optional Caching:

- Cache for 5 minutes to improve performance
- Invalidate cache on any CREATE, UPDATE, or DELETE operation

---

Unified Response Format

Success Response Structure

```json
{
  "success": true,
  "message": "Descriptive success message",
  "data": {},
  "pagination": {}
}
```

Fields:

- `success`: Always `true` for successful responses
- `message`: Human-readable success message

- `data` : Response payload (object, array, or null)
- `pagination` : Only included for paginated endpoints (GET /api/products)

Error Response Structure

```
{
  "success": false,
  "message": "User-friendly error message",
  "error": {
    "code": "ERROR_CODE",
    "details": {}
  }
}
```

Fields:

- `success` : Always `false` for error responses
- `message` : Human-readable error message for end users
- `error.code` : Machine-readable error code (e.g., "VALIDATION_ERROR", "NOT_FOUND")
- `error.details` : Additional error information (validation errors, field names, etc.)

HTTP Status Codes

Status Code	Usage	Example
200 OK	Successful GET, PUT, DELETE requests	Product retrieved/updated/deleted
201 Created	Successful POST (resource created)	Product created successfully
204 No Content	Successful DELETE (optional alternative)	Product deleted (no response body)
400 Bad Request	Validation errors, invalid input	Invalid price, negative quantity
401 Unauthorized	Missing or invalid X-User-Role header	No role provided, invalid role value
403 Forbidden	User lacks permission for the action	User trying to create/update/delete
404 Not Found	Resource doesn't exist	Product ID not found
409 Conflict	Duplicate resource (unique constraint)	SKU already exists

Status Code	Usage	Example
500 Internal Server Error	Unexpected server errors	Database connection failure

---

Error Handling Examples

1. Validation Error: Negative Quantity

```
POST /api/products
Body: { "sku": "TEST-001", "name": "Test", "category": "Test", "price":
100, "quantity": -5 }

Response: 400 Bad Request
{
  "success": false,
  "message": "Validation failed",
  "error": {
    "code": "VALIDATION_ERROR",
    "details": [
      {
        "field": "quantity",
        "message": "Quantity must be greater than or equal to 0"
      }
    ]
  }
}
```

2. Validation Error: Discount Price Greater Than Price

```
POST /api/products
Body: { "sku": "TEST-001", "name": "Test", "category": "Test", "price":
100, "discountPrice": 150, "quantity": 10 }

Response: 400 Bad Request
{
  "success": false,
  "message": "Validation failed",
  "error": {
    "code": "VALIDATION_ERROR",
    "details": [
      {
        "field": "discountPrice",
        "message": "Discount price must be less than the original price"
      }
    ]
  }
}
```

3. Validation Error: Multiple Fields

```
POST /api/products
Body: { "sku": "", "name": "AB", "price": -10, "quantity": 5 }

Response: 400 Bad Request
{
  "success": false,
  "message": "Validation failed",
  "error": {
    "code": "VALIDATION_ERROR",
    "details": [
      {
        "field": "sku",
        "message": "SKU is required and must be 3-50 characters"
      },
      {
        "field": "name",
        "message": "Name must be at least 3 characters long"
      },
      {
        "field": "category",
        "message": "Category is required"
      },
      {
        "field": "price",
        "message": "Price must be greater than 0"
      }
    ]
  }
}
```

4. Duplicate SKU Error

```
POST /api/products
Body: { "sku": "EXISTING-SKU", ... }

Response: 409 Conflict
{
  "success": false,
  "message": "Product with this SKU already exists",
  "error": {
    "code": "DUPLICATE_SKU",
    "details": {
      "field": "sku",
      "value": "EXISTING-SKU"
    }
  }
}
```

5. Unauthorized Access (No Role Header)

```
GET /api/products

Response: 401 Unauthorized
{
  "success": false,
  "message": "Authentication required",
  "error": {
    "code": "UNAUTHORIZED",
    "details": "X-User-Role header is missing or invalid"
  }
}
```

6. Forbidden Access (Insufficient Permissions)

```
DELETE /api/products/123
Headers: X-User-Role: user

Response: 403 Forbidden
{
  "success": false,
  "message": "You do not have permission to perform this action",
  "error": {
    "code": "FORBIDDEN",
    "details": "Admin role required for this operation"
  }
}
```

7. Resource Not Found

```
GET /api/products/999

Response: 404 Not Found
{
  "success": false,
  "message": "Product not found",
  "error": {
    "code": "NOT_FOUND",
    "details": {
      "resource": "Product",
      "id": "999"
    }
  }
}
```

8. User Accessing Private Product

```
GET /api/products/123
Headers: X-User-Role: user
(Product 123 has type: "private")

Response: 404 Not Found
{
  "success": false,
  "message": "Product not found",
  "error": {
    "code": "NOT_FOUND",
    "details": {
      "resource": "Product",
      "id": "123"
    }
  }
}
```

Note: Return 404 instead of 403 to avoid leaking information about private product existence.

Role-Based Access Control (RBAC)

Role Header Format

```
X-User-Role: admin
X-User-Role: user
```

Valid Values: `admin`, `user` (case-insensitive)

Access Control Matrix

Endpoint	Admin	User	No Role
POST /api/products	✅	❌	❌
GET /api/products	✅	✅	❌
GET /api/products/:id	✅	✅	❌

Endpoint	Admin	User	No Role
PUT /api/products/:id	✅	❌	❌
DELETE /api/products/:id	✅	❌	❌
GET /api/products/stats	✅	❌	❌

Admin Role (`X-User-Role: admin`)

- ✅ Full CRUD access to all products
- ✅ View all products (public + private)
- ✅ Access statistics endpoint
- ✅ No automatic filtering applied

User Role (`X-User-Role: user`)

- ✅ Read-only access to **public** products
- ❌ Cannot create, update, or delete products
- ❌ Cannot access statistics
- ⚠️ Automatic filter applied: `type = "public"`

No Role / Invalid Role

- ❌ All endpoints return `401 Unauthorized`
- Error message: "Authentication required"

Additional Requirements

1. Input Validation

- ✅ Validate ALL user inputs before processing
- ✅ Return detailed validation errors with field names
- ✅ Prevent SQL injection / NoSQL injection
- ✅ Sanitize string inputs (trim whitespace)
- ✅ Validate data types (number, string, enum)
- ✅ Validate ranges (min/max values)

2. Error Handling

- ✅ Never expose internal errors to users
- ✅ Use try-catch blocks for all database operations
- ✅ Handle edge cases (empty database, invalid IDs)
- ✅ Return appropriate HTTP status codes

- ✅ Never let the server crash from user input

3. Data Integrity

- ✅ Enforce unique SKU constraint
- ✅ Prevent negative prices/quantities
- ✅ Validate price > discountPrice relationship
- ✅ Ensure type is either "public" or "private"
- ✅ Handle concurrent updates gracefully

4. Performance Considerations

- ✅ Implement pagination for listing endpoints
- ✅ Add indexes on frequently queried fields (sku, category, type)
- ⭐ Optional: Cache statistics endpoint (5 minutes)
- ⭐ Optional: Implement rate limiting

5. Code Quality

- ✅ Use consistent naming conventions
- ✅ Separate concerns (routes, controllers, services, models)
- ✅ Write clean, readable code
- ✅ Add comments for complex logic
- ✅ Handle edge cases explicitly

Project Deliverables

1. **Source Code**
   - Clean, well-organized code structure
   - Proper separation of concerns
   - Comments where necessary
2. **API Documentation**
   - ⭐ Optional: Swagger/OpenAPI specification
   - ⭐ Optional: Postman collection
   - ⭐ Optional: HTTP/REST files
3. **README.md**
   - Setup instructions
   - Environment variables
   - How to run the project
   - API endpoint examples
4. **Environment Configuration**

    - `.env.example` file with required variables
    - Database connection details
    - Port configuration

Bonus Features (Optional)

- ⭐ TypeScript implementation
- ⭐ Input sanitization middleware
- ⭐ Request rate limiting
- ⭐ Unit and integration tests
- ⭐ Docker containerization

Important Reminders

❌ **DO NOT use AI tools** (ChatGPT, Copilot, etc.) to generate code
✅ **DO implement** all validation and error handling manually
✅ **DO test** all edge cases thoroughly
✅ **DO write** clean, readable code
✅ **DO handle** errors gracefully without crashing the server
✅ **DO log** successful operations to console
✅ **DO use** consistent response format throughout

Good luck! Focus on writing clean, working code with proper validation and error handling.