Backend System – Project Documentation

1. Tech Stack

A modern, production-ready backend stack:

Backend Framework

- **Java 17**
- **Spring Boot 4**
- **Spring Web**
- **Spring Data JPA (Hibernate 7)**
- **Jakarta Validation**

Database

- **MySQL 8.0+**

Tools

- Postman
- GitHub
- Spring Tool Suite
- Maven (Dependency Management)

2.System Overview

This project is a **Product Management Backend System** providing **RESTful APIs** for managing product catalog data.

It supports:

- Create product
- Update product
- Delete product (soft delete)
- Fetch product by ID
- Search + Pagination
- Validation
- Global error handling
- Consistent API response structure

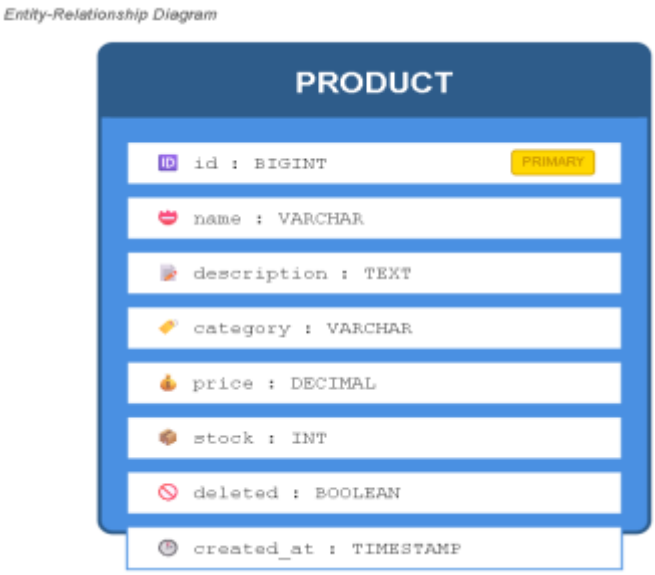
Designed following **clean architecture**, **layered architecture**, and **industry-grade practices**.

3. Database Schema

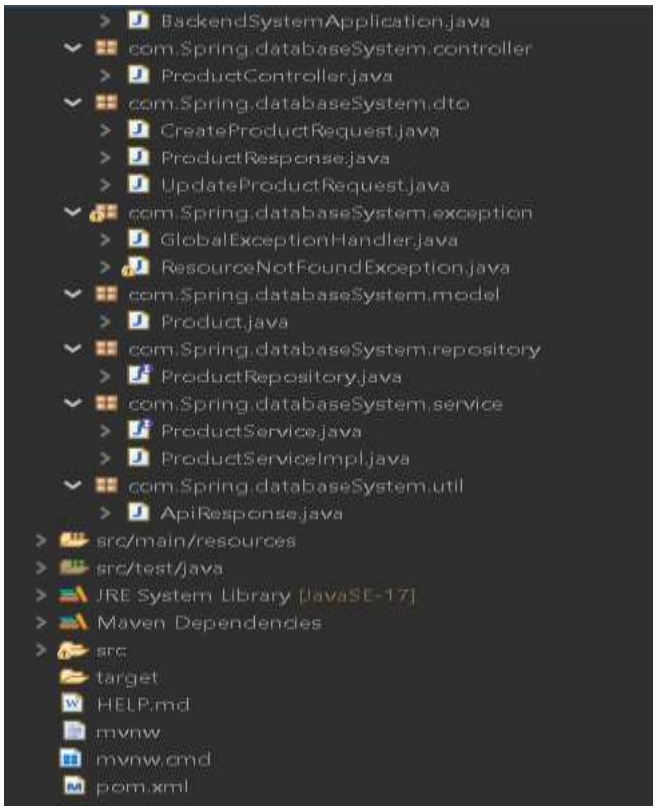
Table: products

Column Name	Type	Description
Id	BIGINT (PK, AUTO_INCREMENT)	Unique product ID
name	VARCHAR(255)	Product name
description	TEXT	Product details
category	VARCHAR(255)	Product category
price	DECIMAL(10,2)	Price
stock	INT	Stock count
deleted	BOOLEAN	Soft delete flag
created_at	TIMESTAMP	Auto-generated timestamp

4. ER Diagram



5. Folder Structure



6. API Documentation

http://localhost:8080/items

1.create Product

POST /items

```json
{
  "name": "Laptop",
  "description": "High-end performance laptop",
  "category": "Electronics",
  "price": 75000,
  "stock": 25
}
```

Response:

```json
{
  "status": "success",
  "message": "Created",
  "data": {
    "id": 1,
    "name": "Laptop",
    "category": "Electronics"
  }
}
```

2.Product List

GET /items?page=0&limit=10&search=&category=

```json
{
  "status": "success",
  "message": "OK",
  "data": {
    "content": [ ... ],
    "pageNumber": 0,
    "totalPages": 1,
    "totalElements": 10
  }
}
```

3. Get Product by ID

GET /items/{id}

4. Update Product

Request

```json
{
  "name": "Updated Laptop"
}
```

Response:

```json
{
  "status": "success",
  "message": "Updated",
  "data": { ... }
}
```

Delete Product

DELETE /items/{id}

✓ Marks the record as `deleted = true`
✓ Record stays in database for audit
✓ API does not show deleted items in results

7. Major Code Snippets

Product Entity

```java
@Entity
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String description;
    private String category;

    private BigDecimal price;
    private Integer stock;

    private boolean deleted = false;

    @CreationTimestamp
    private Timestamp createdAt;
}
```

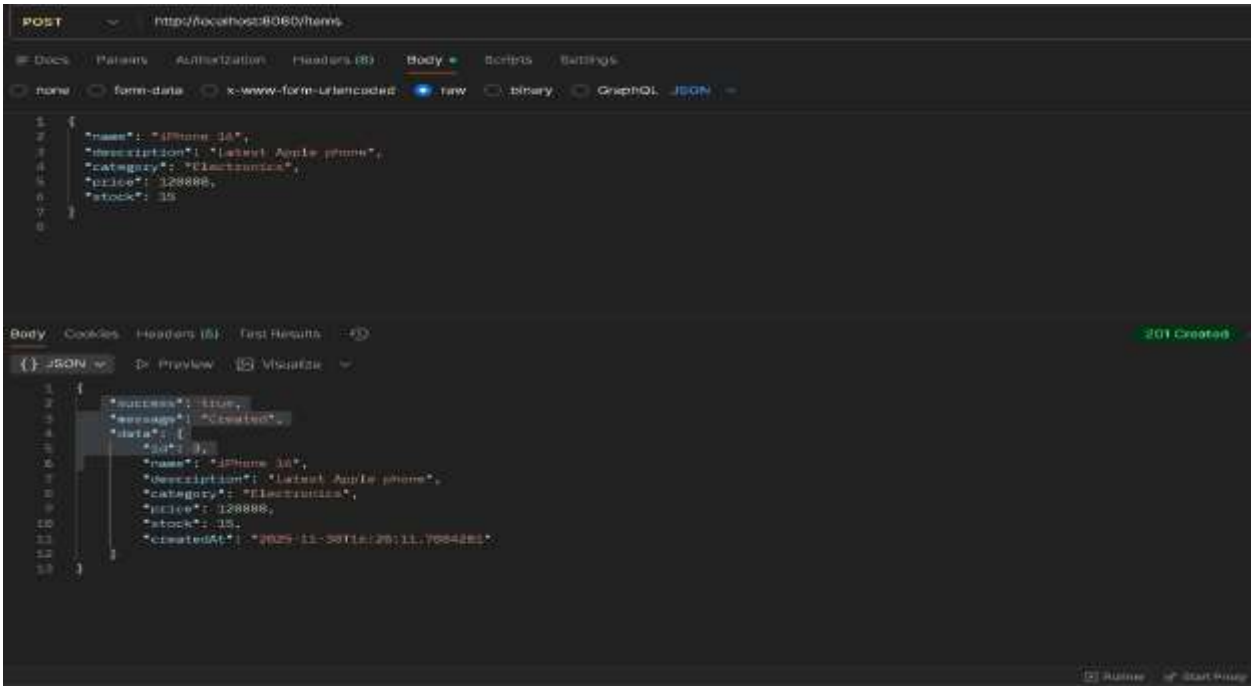
Create DTO

```java
@Data
public class CreateProductRequest {

    @NotBlank(message = "Name is required")
    private String name;

    private String description;
    private String category;

    @NotNull(message = "Price is required")
    private BigDecimal price;

    @NotNull(message = "Stock is required")
    private Integer stock;
}
```

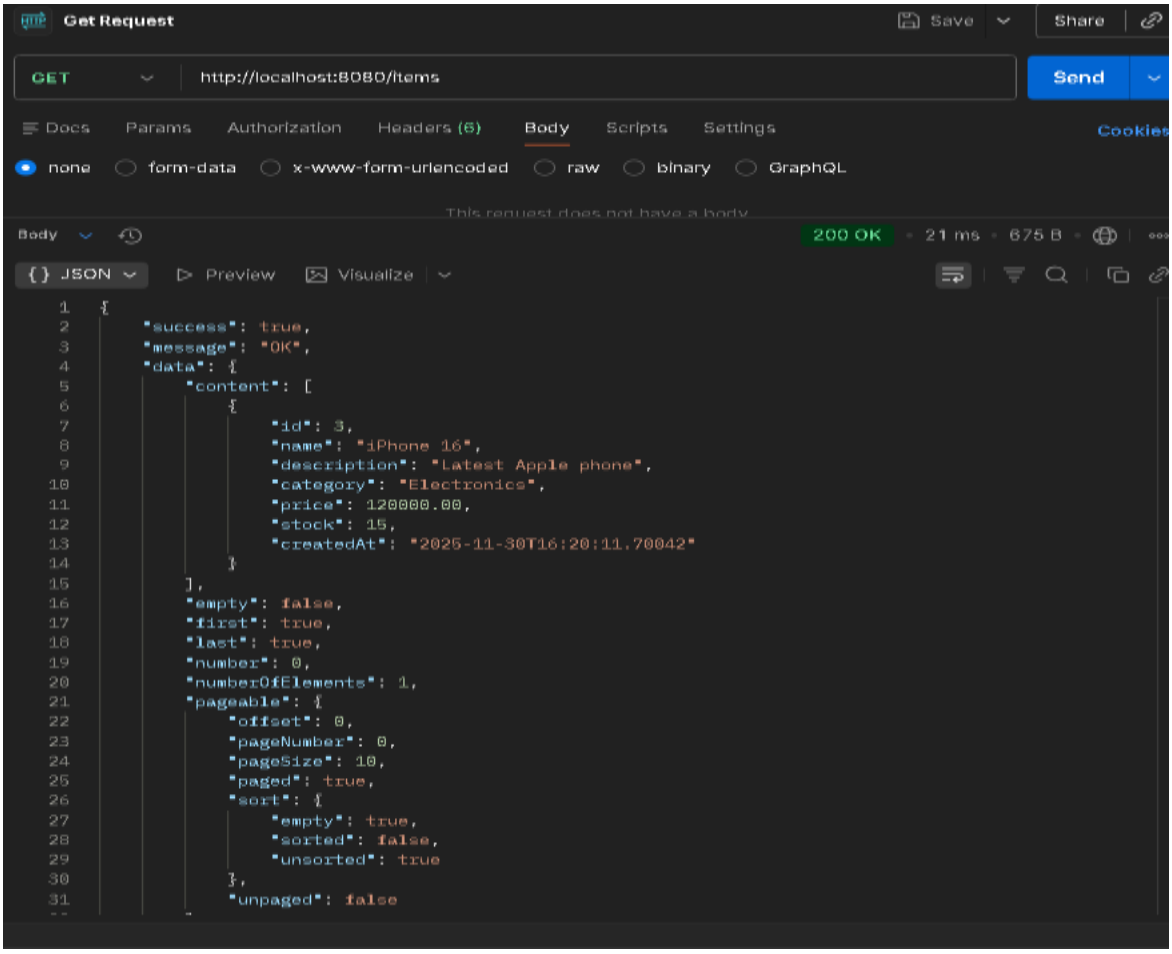
```java
@Override
public void delete(Long id) {
    Product p = repo.findByIdAndDeletedFalse(id)
            .orElseThrow(() -> new ResourceNotFoundException("Product not found"));

    p.setDeleted(true);   // <-- NOT removing from DB
    repo.save(p);         // <-- Updating the record
}
```

8. Postman Examples

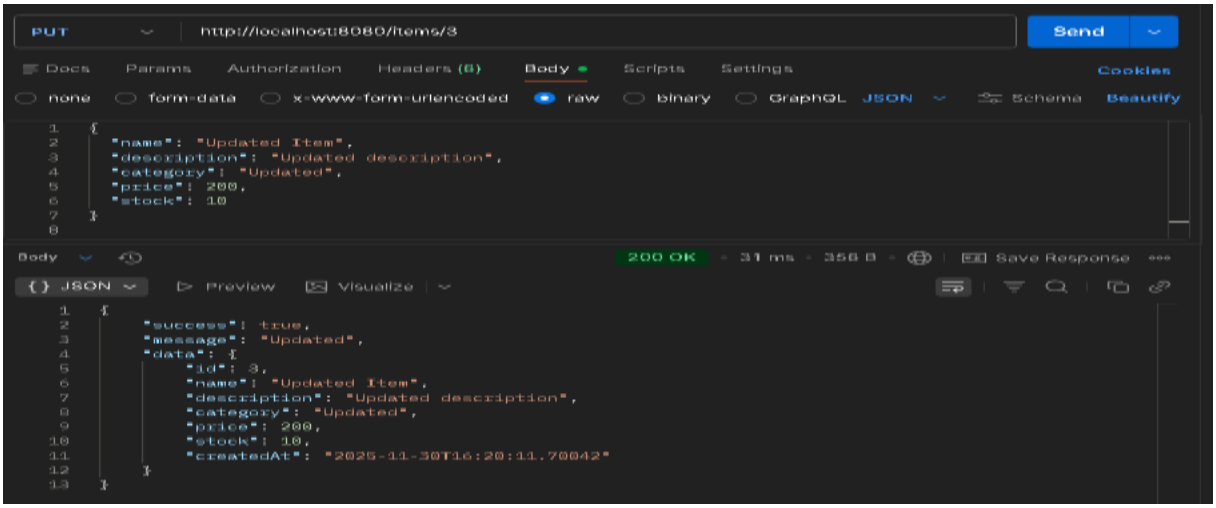
POST:



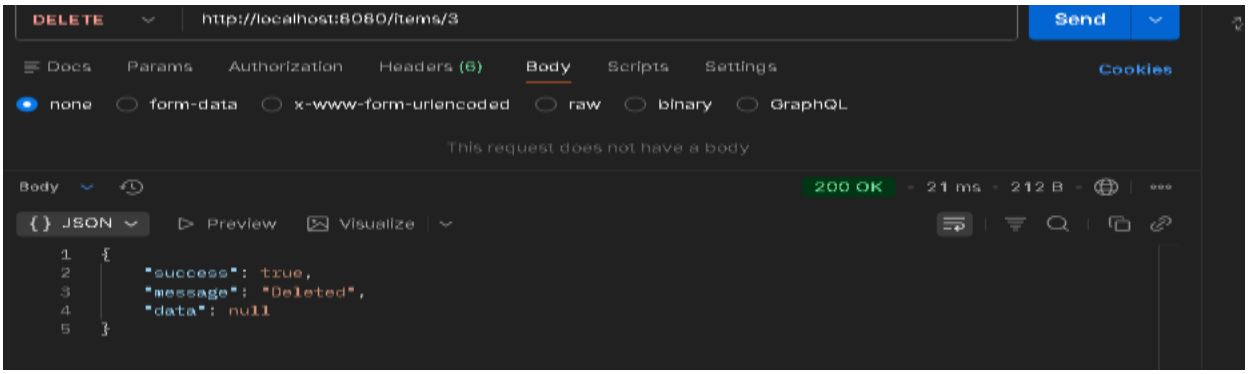
GET:


Get Request
Save
Share
GET
http://localhost:8080/items
Send
Docs
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body
Body
200 OK
21 ms
675 B
JSON
Preview
Visualize
{
"success": true,
"message": "OK",
"data": {
"content": [
{
"id": 3,
"name": "iPhone 16",
"description": "Latest Apple phone",
"category": "Electronics",
"price": 120000.00,
"stock": 15,
"createdAt": "2025-11-30T16:20:11.70042"
}
],
"empty": false,
"first": true,
"last": true,
"number": 0,
"numberOfElements": 1,
"pageable": {
"offset": 0,
"pageNumber": 0,
"pageSize": 10,
"paged": true,
"sort": {
"empty": true,
"sorted": false,
"unsorted": true
},
"unpaged": false

PUT:


PUT
http://localhost:8080/items/3
Send
Docs
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"name": "Updated Item",
"description": "Updated description",
"category": "Updated",
"price": 200,
"stock": 10
}
Body
200 OK
31 ms
358 B
Save Response
JSON
Preview
Visualize
{
"success": true,
"message": "Updated",
"data": {
"id": 3,
"name": "Updated Item",
"description": "Updated description",
"category": "Updated",
"price": 200,
"stock": 10,
"createdAt": "2025-11-30T16:20:11.70042"
}
}

DELETE:



9. GitHub Repository

https://github.com/Umamahesh-1726/Database-System

10. Challenges Faced & Solutions

1. MySQL Access Denied

- Incorrect password caused connection failure
- Solved by updating credentials in `application.properties`

2. Port Already Running

- Tomcat collided with another app
- Changed port → `server.port=1726`

3. Constructor Injection Error

- Field was not initialized
- Fixed using **Constructor Injection**

4. Soft Delete Not Reflecting

- Repository didn't filter deleted = false
- Added correct query method in repository

5. Git Errors (refspec main missing)

- No commit existed

11. Final Summary

This backend system is a fully functional Spring Boot application designed to manage product data with clean API design, DTO separation, soft delete functionality, proper validations, and database integration using MySQL. The project follows scalable architecture, provides a structured REST API, includes defensive error handling, and delivers production-ready backend functionality suitable for enterprise-level systems.