Product CatLog REST API

Vedanshi Sahu

Github link- https://github.com/11roxanne/LeadToRev_Assignment

A RESTful API for managing a structured product catalog with complex data structures and MongoDB integration.

Tech Stack

- Java 17- was already present in the system
- Spring Boot 3.x- downloaded the extension
- MongoDB- was already present in the system
- Maven/Gradle (for dependency management)
- Spring Data MongoDB- downloaded the extension
- Spring Web- downloaded the extension
- Postman

Data Structure used in Database

Objects (documents in MongoDB):

- The main Product document itself
- availability (a single object with inStock and quantity)
- Each individual object inside the attributes array
- Each individual object inside the ratings array

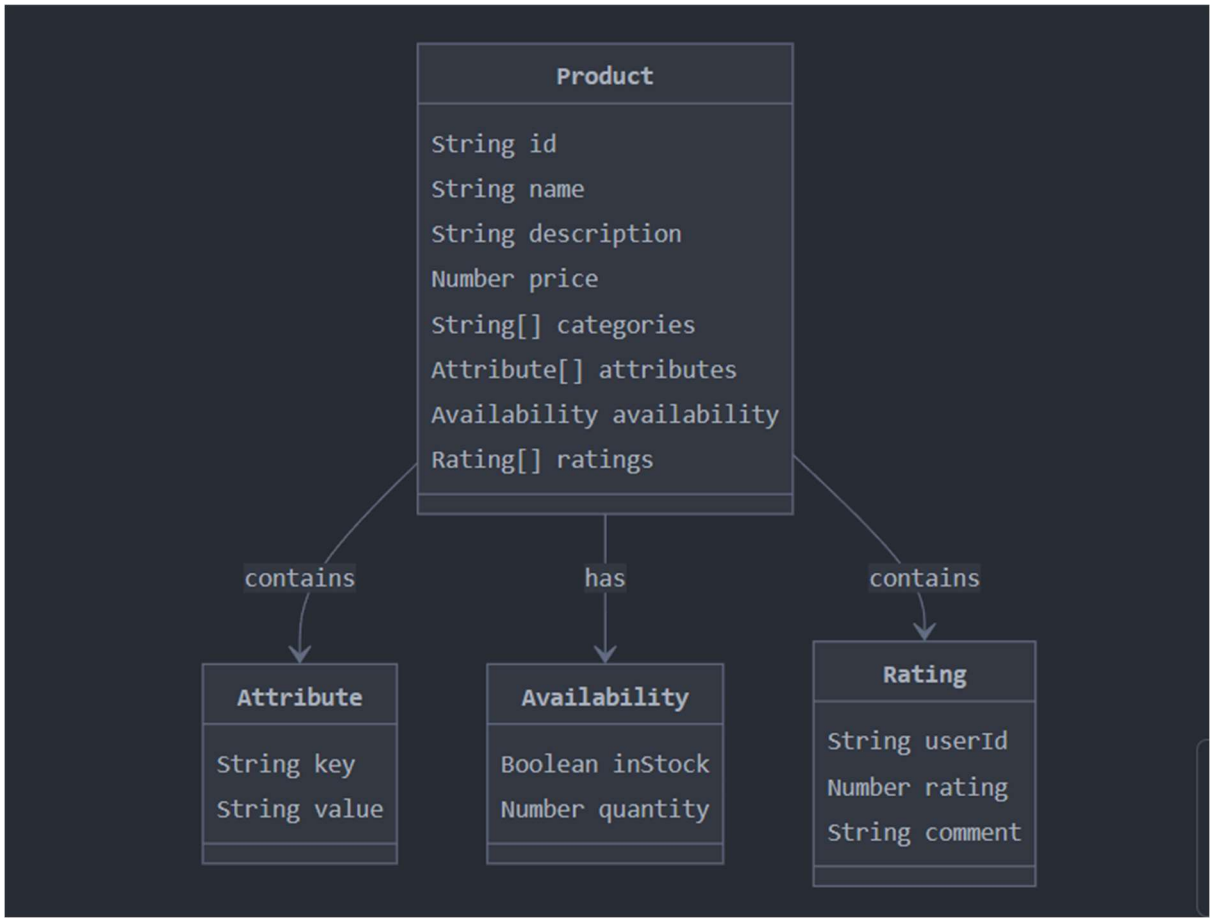
Arrays:

- categories (array of strings)
- attributes (array of objects)
- ratings (array of objects)

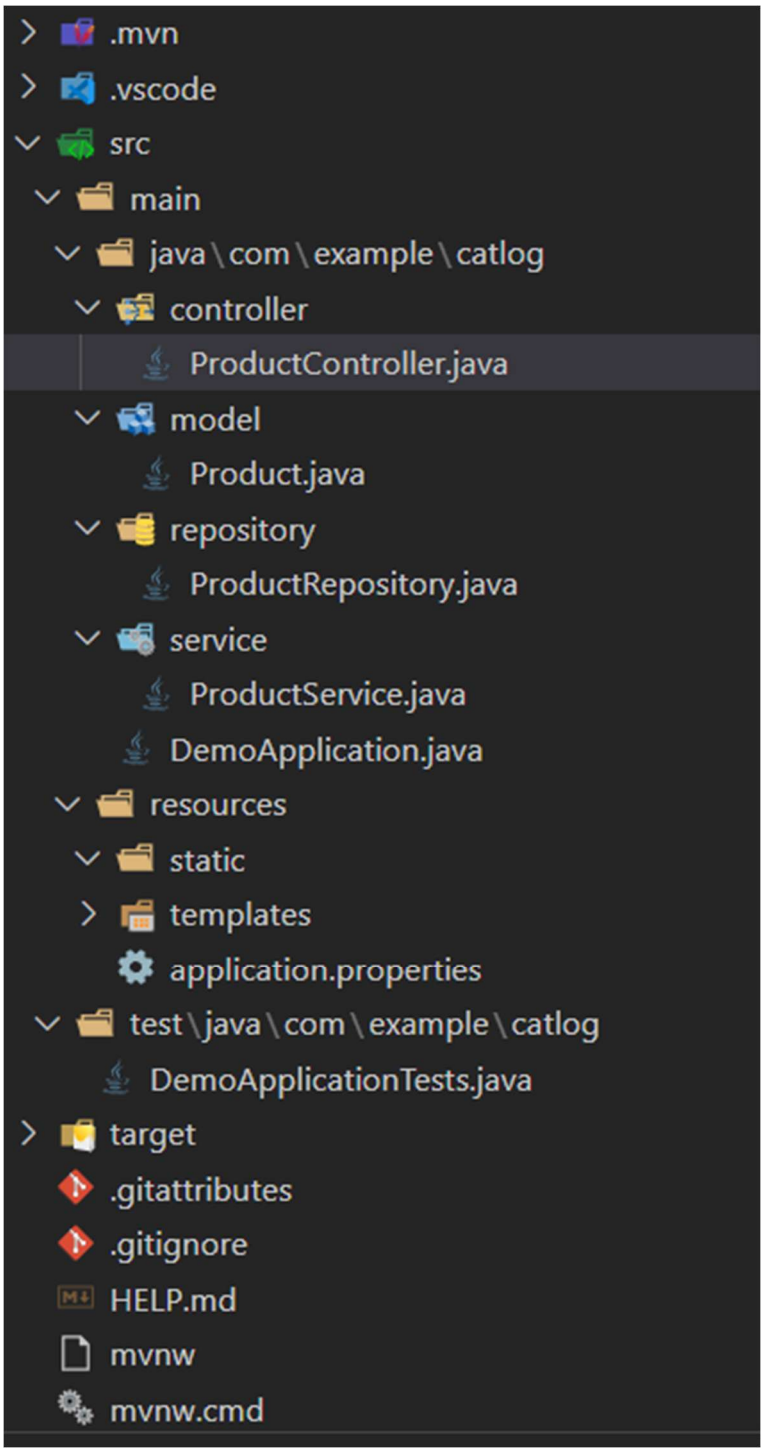
Simple fields (not arrays or objects):

- id (string)
- name (string)
- description (string)
- price (number)

Data Model diagram of Database



File Structure of the project



1. **<u>src folder - Main source code directory</u>**

    - Controller file - Handles HTTP requests, defines API endpoints for products

- Model file - The data model/entity class that defines the structure of a product
- Repository file - manages database operations with MongoDB
- Services File - Contains business logic and connects controller with repository
- DemoApplication.java: Main Spring Boot application class with @SpringBootApplication

2. **resources folder**
   - static- For static files (CSS, JavaScript, images)
   - application.properties- Configuration file for Spring Boot (database connection, server settings, etc.)

3. **test folder**
   - Contains test cases for your application
   - DemoApplicationTests.java: Basic test class for your application

API Endpoints

1. Product Management

Add a Product - POST

Get a Product by ID - GET /products/{id}

Update a Product - PUT /products/{id}

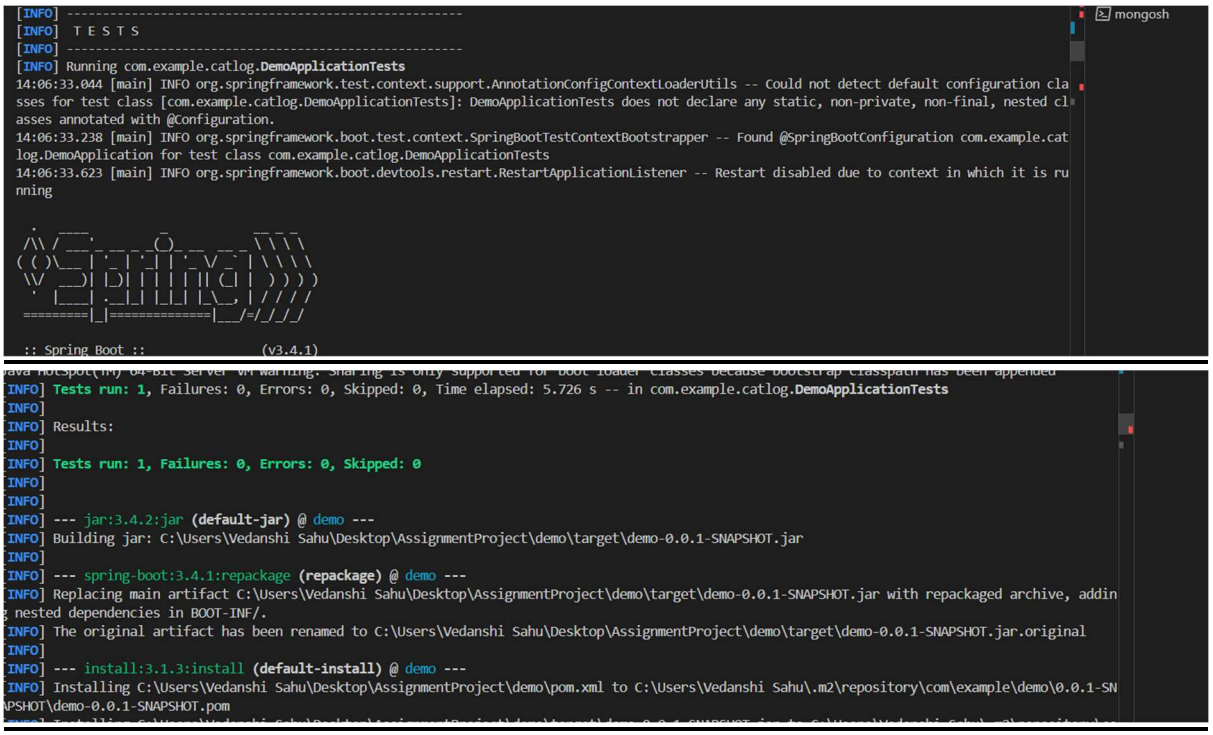
DELETE /products/{id}

2. Search and Filters

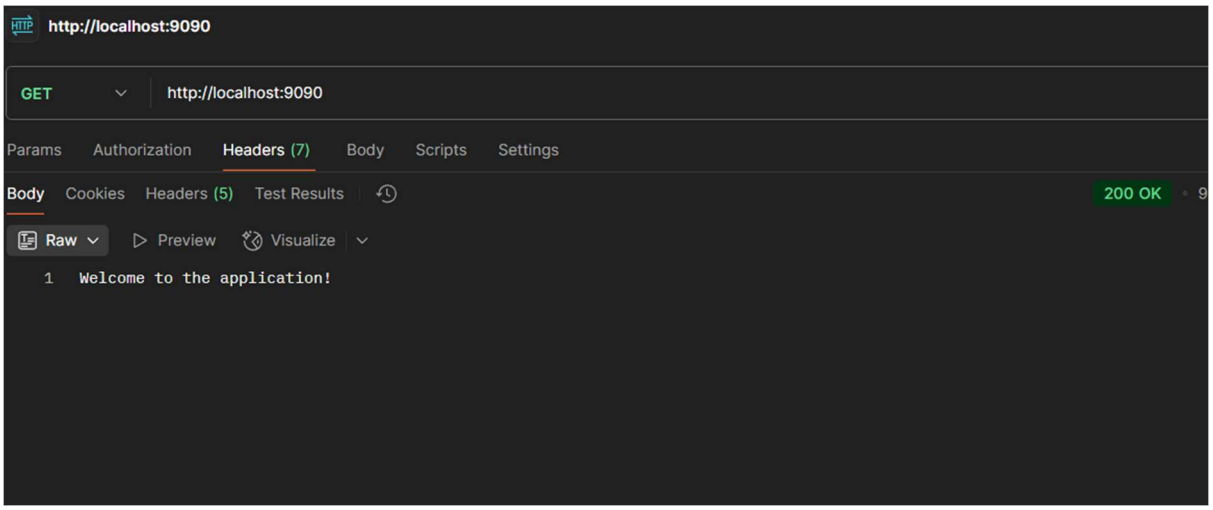
Search Products - GET /products/search

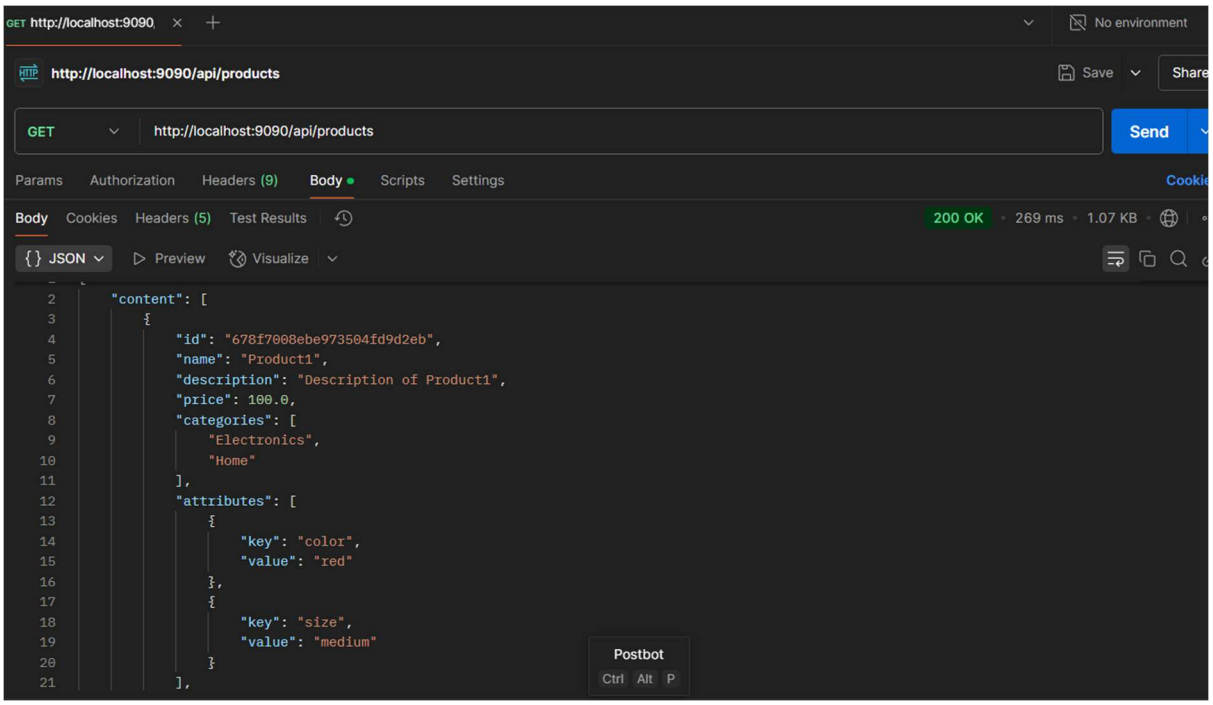
Project Outputs

1. Web Output

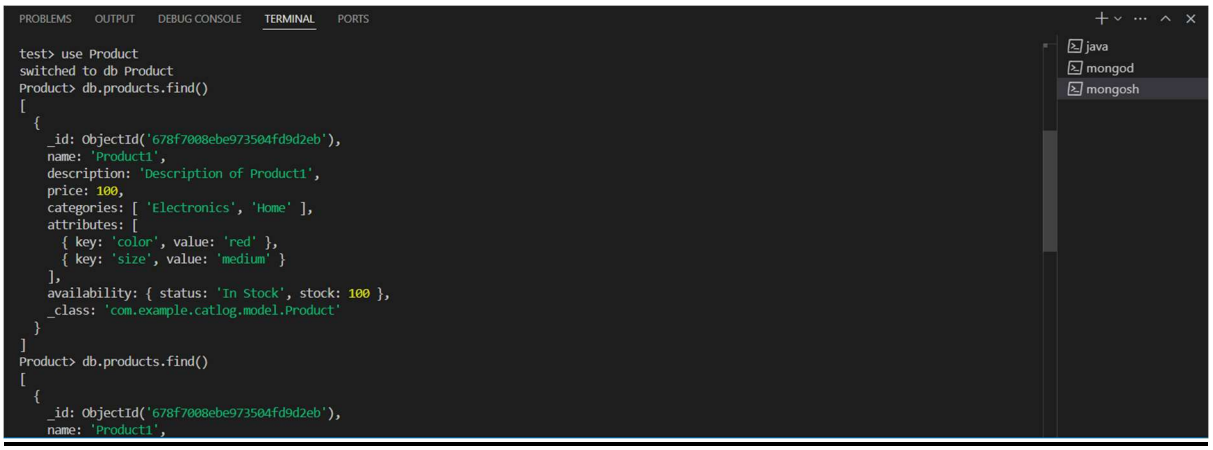


2. Checking API endpoints through Postman


http://localhost:9090
GET
http://localhost:9090
Params
Authorization
Headers (7)
Body
Scripts
Settings
Body
Cookies
Headers (5)
Test Results
200 OK
Raw
Preview
Visualize
1 Welcome to the application!


GET http://localhost:9090
No environment
http://localhost:9090/api/products
Save
Share
GET
http://localhost:9090/api/products
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Body
Cookies
Headers (5)
Test Results
200 OK
269 ms
1.07 KB
{} JSON
Preview
Visualize
"content": [
{
"id": "678f7008ebe973504fd9d2eb",
"name": "Product1",
"description": "Description of Product1",
"price": 100.0,
"categories": [
"Electronics",
"Home"
],
"attributes": [
{
"key": "color",
"value": "red"
},
{
"key": "size",
"value": "medium"
}
],
Postbot
Ctrl Alt P

3. Database


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
java
mongod
mongosh
test> use Product
switched to db Product
Product> db.products.find()
[
{
_id: ObjectId('678f7008ebe973504fd9d2eb'),
name: 'Product1',
description: 'Description of Product1',
price: 100,
categories: ['Electronics', 'Home'],
attributes: [
{ key: 'color', value: 'red' },
{ key: 'size', value: 'medium' }
],
availability: { status: 'In Stock', stock: 100 },
_class: 'com.example.catlog.model.Product'
}
]
Product> db.products.find()
[
{
_id: ObjectId('678f7008ebe973504fd9d2eb'),
name: 'Product1',

Product.products

2 DOCUMENTS 1 INDEXES

Documents | Aggregations | Schema | Indexes | Validation

Filter | Type a query: { field: 'value' } or Generate query | Explain | Reset | Find | </> | Options

ADD DATA | EXPORT DATA | 1 - 2 of 2


```
_id: ObjectId('678f7008ebe973504fd9d2eb')
name: "Product1"
description: "Description of Product1"
price: 100
▸ categories: Array (2)
▸ attributes: Array (2)
▸ availability: Object
_class: "com.example.catlog.model.Product"
```

```
_id: ObjectId('678f7081ebe973504fd9d2ec')
name: "Product2"
description: "A sleek and modern laptop with high performance."
price: 1200
▸ categories: Array (2)
▸ attributes: Array (3)
▸ availability: Object
_class: "com.example.catlog.model.Product"
```