

EXP-4.3:

AIM:

Create a MongoDB collection to represent an e-commerce catalog. Each product document should include fields such as name, price, category, and an array of nested variants. Each variant should include details like color, size, and stock. Insert a few sample product documents demonstrating different variants. Then, implement queries to retrieve all products, filter products by category, and project specific variant details. Use MongoDB shell queries or Mongoose methods to show how nested documents can be accessed and manipulated effectively.

CODE-

1. Sample Product Document Structure:

Each product has fields for name, price, category, and an array of nested variant objects with color, size, and stock quantity:

```
[
  {
    "name": "T-Shirt",
    "price": 19.99,
    "category": "Apparel",
    "variants": [
      {"color": "Red", "size": "M", "stock": 10},
      {"color": "Blue", "size": "L", "stock": 5}
    ]
  },
  {
    "name": "Sneakers",
    "price": 59.99,
    "category": "Footwear",
    "variants": [
      {"color": "White", "size": 9, "stock": 20},
```

```

    {"color": "Black", "size": 10, "stock": 15}
  ]
},
{
  "name": "Backpack",
  "price": 39.99,
  "category": "Accessories",
  "variants": [
    {"color": "Green", "size": "Standard", "stock": 7}
  ]
}
]

```

2. Mongoose Schema Design:

```

const mongoose = require('mongoose');
const variantSchema = new mongoose.Schema({
  color: String,
  size: mongoose.Schema.Types.Mixed, // can be number or string
  stock: Number
});
const productSchema = new mongoose.Schema({
  name: { type: String, required: true },
  price: { type: Number, required: true },
  category: { type: String, required: true },
  variants: [variantSchema]
});
const Product = mongoose.model('Product', productSchema);
module.exports = Product;

```

3. Sample Queries Using Mongoose:

a) Insert Sample Products:

```
await Product.insertMany([
```

```
{
  name: "T-Shirt",
  price: 19.99,
  category: "Apparel",
  variants: [
    { color: "Red", size: "M", stock: 10 },
    { color: "Blue", size: "L", stock: 5 }
  ]
},
{
  name: "Sneakers",
  price: 59.99,
  category: "Footwear",
  variants: [
    { color: "White", size: 9, stock: 20 },
    { color: "Black", size: 10, stock: 15 }
  ]
},
{
  name: "Backpack",
  price: 39.99,
  category: "Accessories",
  variants: [
    { color: "Green", size: "Standard", stock: 7 }
  ]
}
```

```
]);
```

b)Retrieve All Products:

```
const products = await Product.find({});
```

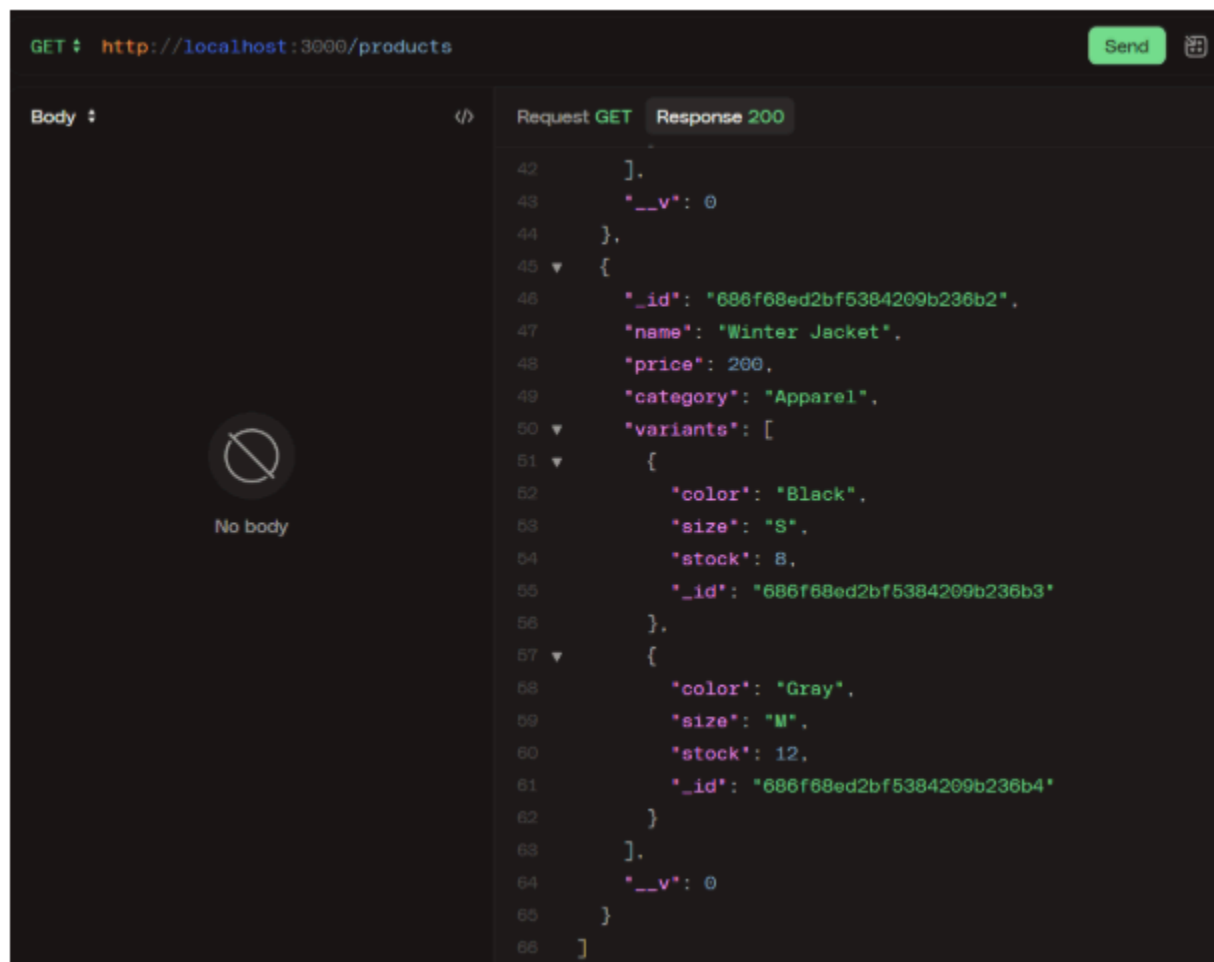
c)Filter Products by Category (e.g., "Apparel"):

```
const apparelProducts = await Product.find({ category: "Apparel" });
```

d)Project Specific Variant Details (e.g., only variants with color and stock):

```
const productsWithVariantDetails = await Product.find(  
  {},  
  { name: 1, 'variants.color': 1, 'variants.stock': 1 }  
);
```

OUTPUTS:



The screenshot shows a REST client interface with a GET request to `http://localhost:3000/products` and a 200 response. The response body is a JSON array containing two product objects. The first product is a 'Winter Jacket' with a price of 200, categorized as 'Apparel', and has two variants: one in Black (size S, stock 8) and one in Gray (size M, stock 12). The second product is not fully visible but appears to be another item.

```
42 ],  
43   "__v": 0  
44 },  
45 {  
46   "_id": "686f68ed2bf5384209b236b2",  
47   "name": "Winter Jacket",  
48   "price": 200,  
49   "category": "Apparel",  
50   "variants": [  
51     {  
52       "color": "Black",  
53       "size": "S",  
54       "stock": 8,  
55       "_id": "686f68ed2bf5384209b236b3"  
56     },  
57     {  
58       "color": "Gray",  
59       "size": "M",  
60       "stock": 12,  
61       "_id": "686f68ed2bf5384209b236b4"  
62     }  
63   ],  
64   "__v": 0  
65 }  
66 ]
```



GET  http://localhost:3000/products/category/Electronics Send 



Body   Request GET Response 200

▶ HTTP/1.1 200 OK (6 headers)

```
1  ▼ [
2  ▼  {
3      "_id": "686f63eb90ac2728b3f11082",
4      "name": "Smartphone",
5      "price": 699,
6      "category": "Electronics",
7      "__v": 0,
8      "variants": []
9  }
10 ]
```


 No body

GET  http://localhost:3000/products/by-color/Blue Send 

Body   Request GET Response 200

▶ HTTP/1.1 200 OK (6 headers)

```
1  ▼ [
2  ▼  {
3      "_id": "686f68ed2bf5384209b236af",
4      "name": "Running Shoes",
5      "price": 120,
6      "category": "Footwear",
7      "variants": [
8      ▼  {
9          "color": "Red",
10         "size": "M",
11         "stock": 10,
12         "_id": "686f68ed2bf5384209b236b0"
13     },
14     ▼  {
15         "color": "Blue",
16         "size": "L",
17         "stock": 5,
18         "_id": "686f68ed2bf5384209b236b1"
19     }
20 ],
21     "__v": 0
22 }
23 ]
```

 No body