EXPERIMENT 4.3

AIM: **E-commerce Catalog with Nested Document Structure in MongoDB**

Full-stack MERN example implementing a nested MongoDB document structure (Catalog → Categories → Subcategories → Products). Includes backend (Express + Mongoose) and frontend (React) code and run instructions.

Overview

This project demonstrates how to store and manipulate a nested catalog structure in MongoDB using Mongoose embedded documents. Important features:

- Catalog document contains categories
- Categories contain subcategories
- Subcategories contain product documents
- Complete CRUD APIs for each level (catalog, category, subcategory, product)
- Simple React frontend to view and modify the catalog

Quick Setup

1. Copy the `backend/` and `frontend/` directories into your workspace.
2. In `backend/.env` set `MONGO_URI` and optionally `PORT`.
3. `cd backend && npm install && npm run dev`
4. `cd frontend && npm install && npm start`

Backend

`backend/package.json`

```
{
  "name": "catalog-backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js",
    "seed": "node seed.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.0.0",
    "express": "^4.18.2",
    "mongoose": "^7.0.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.0"
  }
}
```

backend/.env.example

```
MONGO_URI=mongodb://localhost:27017/catalog_db
PORT=5000
```

backend/server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
require('dotenv').config();

const catalogRoutes = require('./routes/catalogRoutes');

const app = express();
app.use(cors());
app.use(express.json());

app.use('/api/catalogs', catalogRoutes);

const PORT = process.env.PORT || 5000;

mongoose.connect(process.env.MONGO_URI, { useNewUrlParser: true,
useUnifiedTopology: true })
  .then(() => {
    console.log('MongoDB connected ✅');
    app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
  })
  .catch(err => console.error(err));
```

backend/models/Catalog.js

```
const mongoose = require('mongoose');
const { Schema, Types } = mongoose;

const ProductSchema = new Schema({
  _id: { type: Schema.Types.ObjectId, default: () => new Types.ObjectId()
},
  name: { type: String, required: true },
  price: { type: Number, required: true },
  description: { type: String },
  images: [String],
  specs: [
    {
      key: String,
      value: String
    }
  ],
  stock: { type: Number, default: 0 }
}, { timestamps: true });

const SubcategorySchema = new Schema({
  _id: { type: Schema.Types.ObjectId, default: () => new Types.ObjectId()
},
  name: { type: String, required: true },
  products: [ProductSchema]
}, { timestamps: true });

const CategorySchema = new Schema({
  _id: { type: Schema.Types.ObjectId, default: () => new Types.ObjectId()
},
  name: { type: String, required: true },
  subcategories: [SubcategorySchema]
}, { timestamps: true });

const CatalogSchema = new Schema({
  name: { type: String, required: true },
  description: String,
  categories: [CategorySchema]
}, { timestamps: true });

module.exports = mongoose.model('Catalog', CatalogSchema);
```

backend/routes/catalogRoutes.js

```
const express = require('express');
const router = express.Router();
const controller = require('../controllers/catalogController');

// Catalog-level
router.post('/', controller.createCatalog);
router.get('/', controller.listCatalogs);
router.get('/:catalogId', controller.getCatalog);
router.put('/:catalogId', controller.updateCatalog);
router.delete('/:catalogId', controller.deleteCatalog);

// Categories
router.post('/:catalogId/categories', controller.addCategory);
router.put('/:catalogId/categories/:categoryId',
controller.updateCategory);
router.delete('/:catalogId/categories/:categoryId',
controller.deleteCategory);

// Subcategories
router.post('/:catalogId/categories/:categoryId/subcategories',
controller.addSubcategory);
router.put('/:catalogId/categories/:categoryId/subcategories/:subcatId',
controller.updateSubcategory);
router.delete('/:catalogId/categories/:categoryId/subcategories/:subcatId',
controller.deleteSubcategory);

// Products (nested under subcategory)
router.post('/:catalogId/categories/:categoryId/subcategories/:subcatId/pro
ducts', controller.addProduct);
router.put('/:catalogId/categories/:categoryId/subcategories/:subcatId/prod
ucts/:productId', controller.updateProduct);
router.delete('/:catalogId/categories/:categoryId/subcategories/:subcatId/p
roducts/:productId', controller.deleteProduct);

// Helper: fetch product by id (scans nested arrays)
router.get('/:catalogId/products/:productId', controller.getProductById);

module.exports = router;
```

backend/controllers/catalogController.js

```
const Catalog = require('../models/Catalog');
const mongoose = require('mongoose');

// ------------- Catalog ----------------
exports.createCatalog = async (req, res) => {
  try {
    const catalog = new Catalog(req.body);
    await catalog.save();
    res.status(201).json(catalog);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.listCatalogs = async (req, res) => {
  try {
    const catalogs = await Catalog.find();
    res.json(catalogs);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.getCatalog = async (req, res) => {
  try {
    const catalog = await Catalog.findById(req.params.catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    res.json(catalog);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.updateCatalog = async (req, res) => {
  try {
    const updated = await Catalog.findByIdAndUpdate(req.params.catalogId,
req.body, { new: true });
    res.json(updated);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.deleteCatalog = async (req, res) => {
  try {
    await Catalog.findByIdAndDelete(req.params.catalogId);
    res.json({ message: 'Catalog deleted' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// ------------- Categories ----------------
exports.addCategory = async (req, res) => {
  try {
    const { catalogId } = req.params;
    const catalog = await Catalog.findById(catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    catalog.categories.push({ name: req.body.name });
    await catalog.save();
    res.status(201).json(catalog);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.updateCategory = async (req, res) => {
  try {
    const { catalogId, categoryId } = req.params;
    const catalog = await Catalog.findById(catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    const cat = catalog.categories.id(categoryId);
    if (!cat) return res.status(404).json({ error: 'Category not found' });
    cat.name = req.body.name ?? cat.name;
    await catalog.save();
    res.json(cat);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.deleteCategory = async (req, res) => {
  try {
    const { catalogId, categoryId } = req.params;
    const catalog = await Catalog.findById(catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    catalog.categories.id(categoryId).remove();
    await catalog.save();
    res.json({ message: 'Category removed' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// ------------- Subcategories ----------------
exports.addSubcategory = async (req, res) => {
  try {
    const { catalogId, categoryId } = req.params;
    const catalog = await Catalog.findById(catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    const cat = catalog.categories.id(categoryId);
    if (!cat) return res.status(404).json({ error: 'Category not found' });
    cat.subcategories.push({ name: req.body.name });
    await catalog.save();
    res.status(201).json(cat);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.updateSubcategory = async (req, res) => {
  try {
    const { catalogId, categoryId, subcatId } = req.params;
    const catalog = await Catalog.findById(catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    const sub =
catalog.categories.id(categoryId).subcategories.id(subcatId);
    if (!sub) return res.status(404).json({ error: 'Subcategory not found'
});
    sub.name = req.body.name ?? sub.name;
    await catalog.save();
    res.json(sub);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.deleteSubcategory = async (req, res) => {
  try {
    const { catalogId, categoryId, subcatId } = req.params;
    const catalog = await Catalog.findById(catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    const cat = catalog.categories.id(categoryId);
    cat.subcategories.id(subcatId).remove();
    await catalog.save();
    res.json({ message: 'Subcategory removed' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// ------------- Products ----------------
exports.addProduct = async (req, res) => {
  try {
    const { catalogId, categoryId, subcatId } = req.params;
    const catalog = await Catalog.findById(catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    const sub =
catalog.categories.id(categoryId).subcategories.id(subcatId);
    if (!sub) return res.status(404).json({ error: 'Subcategory not found'
});
    sub.products.push(req.body);
    await catalog.save();
    res.status(201).json(sub);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.updateProduct = async (req, res) => {
  try {
    const { catalogId, categoryId, subcatId, productId } = req.params;
    const catalog = await Catalog.findById(catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    const prod =
catalog.categories.id(categoryId).subcategories.id(subcatId).products.id(pr
oductId);
    if (!prod) return res.status(404).json({ error: 'Product not found' });
    // copy allowed fields
    ['name','price','description','images','specs','stock'].forEach(k => {
      if (req.body[k] !== undefined) prod[k] = req.body[k];
    });
    await catalog.save();
    res.json(prod);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

exports.deleteProduct = async (req, res) => {
  try {
    const { catalogId, categoryId, subcatId, productId } = req.params;
    const catalog = await Catalog.findById(catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    const sub =
catalog.categories.id(categoryId).subcategories.id(subcatId);
    sub.products.id(productId).remove();
    await catalog.save();
    res.json({ message: 'Product removed' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// Helper: get product by scanning nested arrays
exports.getProductById = async (req, res) => {
  try {
    const { catalogId, productId } = req.params;
    const catalog = await Catalog.findById(catalogId);
    if (!catalog) return res.status(404).json({ error: 'Catalog not found'
});
    let found = null;
    for (const cat of catalog.categories) {
      for (const sub of cat.subcategories) {
        const p = sub.products.id(productId);
        if (p) {
          found = { category: cat, subcategory: sub, product: p };
          break;
        }
      }
      if (found) break;
    }
    if (!found) return res.status(404).json({ error: 'Product not found'
});
    res.json(found);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```

`backend/seed.js` (optional quick seed)

```
// Run with: npm run seed
const mongoose = require('mongoose');
require('dotenv').config();
const Catalog = require('./models/Catalog');

const seed = async () => {
  await mongoose.connect(process.env.MONGO_URI);
  await Catalog.deleteMany({});
  const catalog = new Catalog({
    name: 'Main Store',
    description: 'Seed catalog',
    categories: [
      {
        name: 'Electronics',
        subcategories: [
          {
            name: 'Phones',
            products: [
              { name: 'iPhone 14', price: 999, description: 'Latest Apple
phone', stock: 5 },
              { name: 'Pixel 7', price: 599, description: 'Google phone',
stock: 10 }
            ]
          },
          {
            name: 'Laptops',
            products: [
              { name: 'MacBook Pro', price: 1999, stock: 2 }
            ]
          }
        ]
      },
      {
        name: 'Fashion',
        subcategories: [
          { name: 'Shoes', products: [{ name: 'Nike Air', price: 120,
stock: 20 }] }
        ]
      }
    ]
  });

  await catalog.save();
  console.log('Seeded ✅');
  process.exit();
};

seed().catch(err => { console.error(err); process.exit(1); });
```

Frontend (React) — Simple UI to interact with nested structure

frontend/package.json

```
{
  "name": "catalog-frontend",
  "version": "1.0.0",
  "private": true,
  "dependencies": {
    "axios": "^1.4.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build"
  }
}
```

frontend/src/index.js

```
import React from 'react';
import { createRoot } from 'react-dom/client';
import App from './App';

createRoot(document.getElementById('root')).render(<App />);
```

frontend/src/App.js

```
import React from 'react';
import CatalogView from './components/CatalogView';

export default function App(){
  return (
    <div style={{ padding: 20, fontFamily: 'Arial, sans-serif' }}>
      <h1>🛍️ E-commerce Catalog (Nested Documents)</h1>
      <CatalogView />
    </div>
  );
}
```

frontend/src/api.js

```
import axios from 'axios';
const API = axios.create({ baseURL: 'http://localhost:5000/api' });
export default API;
```

frontend/src/components/CatalogView.js

```
import React, { useEffect, useState } from 'react';
import API from '../api';
import CategoryForm from './CategoryForm';
import SubcategoryForm from './SubcategoryForm';
import ProductForm from './ProductForm';

export default function CatalogView(){
  const [catalog, setCatalog] = useState(null);
  const [catalogId, setCatalogId] = useState(null);
  const [showCatForm, setShowCatForm] = useState(false);
  const [editing, setEditing] = useState(null); // {type, ids...}

  useEffect(() => {
    // Fetch first catalog if exists or create one
    API.get('/catalogs')
      .then(res => {
        if (res.data.length) {
          setCatalog(res.data[0]);
          setCatalogId(res.data[0]._id);
        } else {
          // create default
          API.post('/catalogs', { name: 'Main Catalog', description: 'Auto
created' })
```

```
            .then(r => { setCatalog(r.data); setCatalogId(r.data._id); });
        }
      })
      .catch(console.error);
  }, []);

  const refresh = () => {
    if (!catalogId) return;
    API.get(`/catalogs/${catalogId}`).then(r =>
setCatalog(r.data)).catch(console.error);
  };

  if (!catalog) return <div>Loading…</div>;

  return (
    <div>
      <h2>{catalog.name}</h2>
      <p>{catalog.description}</p>

      <button onClick={() => setShowCatForm(s => !s)}>+ Add
Category</button>
      {showCatForm && (
        <CategoryForm catalogId={catalog._id} onDone={() => {
setShowCatForm(false); refresh(); }} />
      )}

      <div style={{ marginTop: 20 }}>
        {catalog.categories.map(cat => (
          <div key={cat._id} style={{ border: '1px solid #ddd', padding:
12, marginBottom: 10 }}>
            <h3>{cat.name}</h3>
            <button onClick={() => {
API.delete(`/catalogs/${catalog._id}/categories/${cat._id}`).then(refresh);
}}>Delete Category</button>
            <button onClick={() => setEditing({ type: 'subcategory',
catalogId: catalog._id, categoryId: cat._id })}>+ Add Subcategory</button>

            <div style={{ marginLeft: 12, marginTop: 10 }}>
              {cat.subcategories.map(sub => (
                <div key={sub._id} style={{ border: '1px dashed #ccc',
padding: 10, marginBottom: 8 }}>
                  <strong>{sub.name}</strong>
                  <div>
                    <button onClick={() => {
API.delete(`/catalogs/${catalog._id}/categories/${cat._id}/subcategories/${
sub._id}`).then(refresh); }}>Delete Subcategory</button>
                    <button onClick={() => setEditing({ type: 'product',
catalogId: catalog._id, categoryId: cat._id, subcatId: sub._id })}>+ Add
Product</button>
                  </div>

                  <div style={{ marginTop: 8 }}>
                    {sub.products.map(prod => (
                      <div key={prod._id} style={{ display: 'flex',
justifyContent: 'space-between', padding: 6, borderTop: '1px solid #eee'
}}>
                        <div>
                          <strong>{prod.name}</strong> — ${prod.price}
                          <div style={{ fontSize: 12, color: '#555'
}}>{prod.description}</div>
                        </div>
                        <div>
                          <button onClick={() =>
API.delete(`/catalogs/${catalog._id}/categories/${cat._id}/subcategories/${
sub._id}/products/${prod._id}`).then(refresh)}>Delete</button>
                          <button onClick={() => setEditing({ type:
'editProduct', catalogId: catalog._id, categoryId: cat._id, subcatId:
sub._id, productId: prod._id })}>Edit</button>
                        </div>
                      </div>
                    ))}
                  </div>
                </div>
              ))}
            </div>
          </div>
        ))}
      </div>

      {editing?.type === 'subcategory' && (
        <SubcategoryForm
          catalogId={editing.catalogId}
          categoryId={editing.categoryId}
          onDone={() => { setEditing(null); refresh(); }}
        />
      )}

      {editing?.type === 'product' && (
        <ProductForm
          catalogId={editing.catalogId}
          categoryId={editing.categoryId}
          subcatId={editing.subcatId}
          onDone={() => { setEditing(null); refresh(); }}
        />
      )}

      {editing?.type === 'editProduct' && (
        <ProductForm
          catalogId={editing.catalogId}
          categoryId={editing.categoryId}
          subcatId={editing.subcatId}
          productId={editing.productId}
          onDone={() => { setEditing(null); refresh(); }}
        />
      )}
    </div>
  );
}
```

frontend/src/components/CategoryForm.js

```
import React, { useState } from 'react';
import API from '../api';

export default function CategoryForm({ catalogId, onDone }){
  const [name, setName] = useState('');
  const submit = async (e) => {
    e.preventDefault();
    await API.post(`/catalogs/${catalogId}/categories`, { name });
    setName('');
    onDone();
  };
```

```
return (
  <form onSubmit={submit} style={{ marginTop: 10 }}>
    <input placeholder="Category name" value={name} onChange={e =>
```