

EXPERIMENT 4.3

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

Folder Structure

```
ecommerce-catalog/  
├── .env  
├── package.json  
├── server.js  
├── config/  
│   └── db.js  
├── models/  
│   └── product.model.js  
├── controllers/  
│   └── product.controller.js  
├── routes/  
│   └── product.routes.js  
├── seed/  
│   └── seedProducts.js  
└── ui/  
    └── index.html
```

Steps Performed

1. Initialize Node Project

2. `mkdir ecommerce-catalog`
3. `cd ecommerce-catalog`
4. `npm init -y`
5. `npm install express mongoose dotenv cors`
6. `npm install -D nodemon`

7. Create .env file

8. `PORT=4000`
9. `MONGO_URI=mongodb://127.0.0.1:27017/ecommerce`

10. Database Connection (`config/db.js`)

Connected Node.js to local MongoDB using Mongoose.

11. Model (`models/product.model.js`)

Defined **nested schema**:

- `category` → object with path and leaf
- `variants[]` → array of subdocuments
 - each variant has `attrs`, `price`, `inventory[]`, `images[]`
- `rating`, `tags`, `attributes`

12. Controller (`controllers/product.controller.js`)

Wrote functions for:

- create product
- list products with filters
- get product by slug
- update nested fields like price or inventory

13. Routes (**routes/product.routes.js**)

Exposed REST endpoints under `/api/products`.

14. Server (**server.js**)

Setup Express app with routes, JSON parsing, and Mongo connection.

15. Seed Script (**seed/seedProducts.js**)

Inserted sample products (Zen Cotton Tee, Neo Sneaker, Acme Runner Shoe) with nested variants, price, inventory, and images.

16. `npm run seed`

17. Run Server

18. `npm run dev`

API served at `http://localhost:4000/api/products`.

19. Frontend (**ui/index.html**)

- Simple HTML + JS file
- Fetches product list from API
- Displays cards with brand, price, rating, category, and tags
- Filters by search, category, and price

20. Browser Output

- Opened `index.html` in browser (via Live Server or `npx serve ui`).
- Products rendered as cards.
- Filters applied dynamically.
- Nested structure visible through variants, price, and inventory.

FILE: config/db.js

```
// config/db.js
const mongoose = require('mongoose');

module.exports = async function connectDB() {
  const uri = process.env.MONGO_URI;
  if (!uri) throw new Error('MONGO_URI missing in .env');
  mongoose.set('strictQuery', true);
  await mongoose.connect(uri, { autoIndex: true });
  console.log('✅ MongoDB connected');
};
```

FILE: controllers/product.controller.js

```
// controllers/product.controller.js
```

```

const Product = require('../models/product.model');

exports.createProduct = async (req, res) => {
  try { const doc = new Product(req.body); await doc.save();
    res.status(201).json(doc); }
  catch (err) { res.status(400).json({ error: err.message }); }
};

exports.getProducts = async (req, res) => {
  try {
    const { q, leaf, brand, active, page = 1, limit = 12, min = 0, max =
      Number.MAX_SAFE_INTEGER } = req.query;
    const match = {};
    if (leaf) match['category.leaf'] = leaf;
    if (brand) match.brand = brand;
    if (active !== undefined) match.isActive = active === 'true';
    match.$or = [
      { 'variants.price.sale': { $gte: Number(min), $lte: Number(max) } },
      { 'variants.price.list': { $gte: Number(min), $lte: Number(max) } }
    ];
    if (q) match.$text = { $search: q };
    const skip = (Number(page) - 1) * Number(limit);
    const [items, total] = await Promise.all([
      Product.find(match).sort(q ? { score: { $meta: 'textScore' } } :
        { minPrice: 1 }).skip(skip).limit(Number(limit)),
      Product.countDocuments(match)
    ]);
    res.json({ page: Number(page), limit: Number(limit), total, items });
  } catch (err) { res.status(500).json({ error: err.message }); }
};

exports.getBySlug = async (req, res) => {
  try { const doc = await Product.findOne({ slug: req.params.slug }); if (!
    doc) return res.status(404).json({ error: 'Not found' }); res.json(doc); }
  catch (err) { res.status(500).json({ error: err.message }); }
};

exports.updatePriceBySku = async (req, res) => {
  try {
    const { sku } = req.params; const { sale } = req.body;
    const doc = await Product.findOneAndUpdate(
      { 'variants.sku': sku },
      { $set: { 'variants.$[v].price.sale': sale, updatedAt: new
        Date() } },
      { new: true, arrayFilters: [{ 'v.sku': sku }] }
    );
    if (!doc) return res.status(404).json({ error: 'SKU not found' });
    await doc.save(); res.json(doc);
  } catch (err) { res.status(400).json({ error: err.message }); }
};

exports.bumpInventory = async (req, res) => {
  try {
    const { sku } = req.params; const { warehouse, delta } = req.body;
    const doc = await Product.findOneAndUpdate(
      { 'variants.sku': sku },
      { $inc: { 'variants.$[v].inventory.$[w].qty': Number(delta) }, $set:
        { updatedAt: new Date() } },
      { new: true, arrayFilters: [{ 'v.sku': sku }, { 'w.warehouse':
        warehouse }] }
    );
    if (!doc) return res.status(404).json({ error: 'SKU/warehouse not
      found' });
  }
};

```

```

    await doc.save(); res.json(doc);
  } catch (err) { res.status(400).json({ error: err.message }); }
};

```

FILE: models/product.model.js

```

// models/product.model.js
const mongoose = require('mongoose');

const ImageSchema = new mongoose.Schema({ url: { type: String, required:
true }, alt: String }, { _id: false });
const InventorySchema = new mongoose.Schema({ warehouse: { type: String,
required: true }, qty: { type: Number, required: true, min: 0 } }, { _id:
false });
const PriceSchema = new mongoose.Schema({ list: { type: Number, required:
true, min: 0 }, sale: { type: Number, default: null, min: 0 }, currency:
{ type: String, default: 'INR' } }, { _id: false });

const VariantSchema = new mongoose.Schema({
  sku: { type: String, required: true },
  attrs: { type: mongoose.Schema.Types.Mixed, default: {} },
  price: { type: PriceSchema, required: true },
  inventory: { type: [InventorySchema], default: [] },
  images: { type: [ImageSchema], default: [] }
}, { _id: false });

const ProductSchema = new mongoose.Schema({
  slug: { type: String, required: true, unique: true, index: true },
  title: { type: String, required: true, text: true },
  description: { type: String, default: '', text: true },
  category: { path: { type: [String], required: true, index: true }, leaf:
{ type: String, required: true, index: true } },
  brand: { type: String, index: true, text: true },
  tags: { type: [String], default: [], index: true },
  attributes: { type: mongoose.Schema.Types.Mixed, default: {} },
  variants: { type: [VariantSchema], validate: v => Array.isArray(v) &&
v.length > 0 },
  rating: { avg: { type: Number, default: 0 }, count: { type: Number,
default: 0 } },
  isActive: { type: Boolean, default: true },
  createdAt: { type: Date, default: Date.now },
  updatedAt: { type: Date, default: Date.now },
  minPrice: { type: Number, default: null }
});

ProductSchema.pre('save', function(next) {
  if (this.variants?.length) {
    const prices = this.variants.map(v => (v.price.sale ?? v.price.list));
    this.minPrice = Math.min(...prices);
  }
  this.updatedAt = new Date();
  next();
});

ProductSchema.index({ 'variants.attrs.color': 1 });
ProductSchema.index({ 'variants.price.list': 1 });
ProductSchema.index({ 'variants.price.sale': 1 });
ProductSchema.index({ 'attributes.$**': 1 });

module.exports = mongoose.model('Product', ProductSchema);

```

FILE: routes/product.routes.js

```
// routes/product.routes.js
const express = require('express');
const router = express.Router();
const ctl = require('../controllers/product.controller');

router.post('/', ctl.createProduct);
router.get('/', ctl.getProducts);
router.get('/:slug', ctl.getBySlug);
router.patch('/sku/:sku/price', ctl.updatePriceBySku);
router.patch('/sku/:sku/inventory', ctl.bumpInventory);

module.exports = router;
```

FILE: server.js

```
// server.js
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const connectDB = require('./config/db');
const productRoutes = require('./routes/product.routes');

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

// health check
app.get('/', (_, res) => res.json({ ok: true, service: 'ecommerce-catalog-api' }));

// routes
app.use('/api/products', productRoutes);

// start
const PORT = process.env.PORT || 4000;
connectDB().then(() => {
  app.listen(PORT, () => console.log(`✅ Server running on http://localhost:${PORT}`));
}).catch(err => {
  console.error('DB connection failed:', err);
  process.exit(1);
});
```

FILE: seed/seedProducts.js

```
// seed/seedProducts.js
require('dotenv').config();
const connectDB = require('../config/db');
const Product = require('../models/product.model');

(async () => {
  try {
    await connectDB();
    await Product.deleteMany({}); // reset

    const data = [
      {
        slug: "acme-runner-shoe",
        title: "Acme Runner Shoe",
        description: "Lightweight running shoe.",
        category: { path: ["Men", "Footwear", "Running Shoes"], leaf: "Running Shoes" },
      },
    ];
```

```

    brand: "Acme",
    tags: ["sports", "new"],
    attributes: { material: "mesh", waterproof: false },
    variants: [
      {
        sku: "AC-RUN-BLK-42",
        attrs: { color: "Black", size: 42 },
        price: { list: 4999, sale: 4499, currency: "INR" },
        inventory: [{ warehouse: "IN-DEL", qty: 12 }, { warehouse: "IN-
MUM", qty: 8 }],
        images: [{ url: "https://img/runner-black-1.jpg", alt: "Black
side" }]
      },
      {
        sku: "AC-RUN-BLU-43",
        attrs: { color: "Blue", size: 43 },
        price: { list: 4999, sale: null, currency: "INR" },
        inventory: [{ warehouse: "IN-DEL", qty: 5 }],
        images: [{ url: "https://img/runner-blue-1.jpg", alt: "Blue
side" }]
      }
    ],
    rating: { avg: 4.4, count: 281 },
    isActive: true
  },
  {
    slug: "zen-cotton-tee",
    title: "Zen Cotton Tee",
    description: "100% organic cotton T-shirt.",
    category: { path: ["Men", "Apparel", "T-Shirts"], leaf: "T-Shirts" },
    brand: "Zenwear",
    tags: ["casual"],
    attributes: { material: "cotton", gsm: 180 },
    variants: [
      {
        sku: "ZN-TEE-WHT-M",
        attrs: { color: "White", size: "M" },
        price: { list: 999, sale: 799, currency: "INR" },
        inventory: [{ warehouse: "IN-DEL", qty: 40 }],
        images: [{ url: "https://img/tee-white-1.jpg", alt: "White tee"
}]
      }
    ],
    rating: { avg: 4.1, count: 73 },
    isActive: true
  }
];

await Product.insertMany(data);

console.log('✅ Seeded products:', await Product.countDocuments());
process.exit(0);
} catch (err) {
  console.error('Seed error:', err);
  process.exit(1);
}
})();

```

FILE: package.json

```

{
  "name": "ecommerce-catalog",
  "version": "1.0.0",

```

```

"main": "server.js",
"type": "commonjs",
"scripts": {
  "dev": "nodemon server.js",
  "start": "node server.js",
  "seed": "node seed/seedProducts.js"
},
"dependencies": {
  "cors": "^2.8.5",
  "dotenv": "^17.2.2",
  "express": "^5.1.0",
  "mongoose": "^8.18.1"
},
"devDependencies": {
  "nodemon": "^3.1.10"
}
}

```

FILE: ui/index.html

```

<!-- ui/index.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>E-commerce Catalog • Nested Explorer</title>
  <style>
    :root{
      --bg:#0f172a; --panel:#0b1221; --card:#0c1326; --muted:#94a3b8; --
text:#e5e7eb;
      --accent:#22d3ee; --accent2:#a78bfa; --border:#1f2a44; --
chip:#23314f; --ring:rgba(34,211,238,.4)
    }
    *{box-sizing:border-box}
    body{margin:0;background:linear-gradient(180deg,#0b1226,#0f172a
35%);color:var(--text);font-family:ui-sans-serif,system-ui,-apple-
system,Segoe UI,Roboto,Helvetica,Arial}
    .container{max-width:1200px;margin:28px auto;padding:0 16px}
    header{display:flex;align-items:center;justify-content:space-
between;margin-bottom:14px}
    h1{margin:0;font-
size:clamp(20px,2.5vw,30px);display:flex;gap:10px;align-items:center}
    h1 .dot{height:10px;width:10px;border-radius:50%;background:var(--
accent);box-shadow:0 0 0 6px rgba(34,211,238,.12)}
    .muted{color:var(--muted);font-size:12px}

    .controls{display:grid;grid-template-columns:1.6fr .8fr .8fr .9fr auto
auto;gap:10px;margin:18px 0 22px}
    .control input,.control select{width:100%;padding:12px 14px;border-
radius:12px;border:1px solid var(--border);background:var(--
panel);color:var(--text);outline:none}
    .control input:focus,.control select:focus{box-shadow:0 0 0 4px var(--
ring)}
    .btn{padding:12px 14px;border-radius:12px;border:0;background:linear-
gradient(90deg,var(--accent),var(--accent2));color:#051018;font-
weight:800;cursor:pointer}

    .grid{display:grid;grid-template-
columns:repeat(1,minmax(0,1fr));gap:16px}
    @media(min-width:860px){.grid{grid-template-
columns:repeat(2,minmax(0,1fr))}}

```

```

    .card{background:linear-gradient(180deg,#0c1326,#0d162a);border:1px
solid var(--border);border-radius:18px;overflow:hidden;box-shadow:0 10px
36px rgba(2,8,23,.3)}
    .card-head{display:flex;gap:14px;padding:14px;border-bottom:1px solid
var(--border)}
    .thumb{flex:0 0 120px;height:84px;background:#0a0f1c;border-
radius:12px;overflow:hidden}
    .thumb img{width:100%;height:100%;object-fit:cover}
    .title{margin:0 0 4px;font-size:18px;font-weight:800}
    .brand{font-size:12px;color:var(--muted)}
    .price{font-weight:900}
    .price s{color:#9ca3af;font-weight:500;margin-left:8px}
    .chips{display:flex;flex-wrap:wrap;gap:6px;margin-top:6px}
    .chip{font-size:11px;padding:4px 8px;border-radius:999px;border:1px
solid var(--chip);background:rgba(67,56,202,.12);color:#c7d2fe}

    details{border-top:1px solid var(--border)}
    details[open] summary{border-bottom:1px dashed var(--border)}
    summary{list-style:none;padding:12px
16px;cursor:pointer;display:flex;align-items:center;justify-content:space-
between}
    summary::-webkit-details-marker{display:none}
    .caret{margin-right:8px;transition:transform .2s}
    details[open] .caret{transform:rotate(90deg)}

    .section{padding:10px 16px 16px}
    .kv{display:grid;grid-template-columns:140px 1fr;gap:10px;margin:6px
0;color:var(--muted)}
    .mono{font-family: ui-monospace,SFMono-
Regular,Menlo,Monaco,Consolas,monospace; background:#0b1221;border:1px
solid var(--border);border-radius:10px;padding:10px;overflow:auto}

    table{width:100%;border-collapse:separate;border-spacing:0 6px}
    th,td{text-align:left;padding:10px 12px}
    tbody tr{background:#0b1221;border:1px solid var(--border)}
    tbody td:first-child, thead th:first-child{border-top-left-
radius:10px;border-bottom-left-radius:10px}
    tbody td:last-child, thead th:last-child{border-top-right-
radius:10px;border-bottom-right-radius:10px}
    .badge{font-size:11px;border:1px solid var(--chip);padding:2px
8px;border-radius:999px}

    .pagination{display:flex;align-items:center;justify-
content:center;gap:10px;margin:20px 0}
    .page-btn{padding:10px 14px;border-radius:12px;border:1px solid var(--
border);background:var(--panel);color:var(--text);cursor:pointer}
    .page-btn[disabled]{opacity:.4;cursor:not-allowed}
</style>
</head>
<body>
  <div class="container">
    <header>
      <h1><span class="dot"></span> Nested Explorer</h1>
      <div class="muted">Shows actual nested structure from <code>GET /api/
products</code></div>
    </header>

    <div class="controls">
      <div class="control"><input id="q" placeholder="Search (title, desc,
brand, tags)"/></div>
      <div class="control"><input id="min" type="number" placeholder="Min
₹"/></div>

```



```

        <div class="control"><input id="max" type="number" placeholder="Max
₹"/></div>
        <div class="control"><select id="leaf"><option value="">All
Categories</option></select></div>
        <button class="btn" id="apply">Apply</button>
        <button class="btn" id="toggleRaw">Raw JSON: Off</button>
    </div>

    <div id="grid" class="grid"></div>

    <div class="pagination">
        <button class="page-btn" id="prev">Prev</button>
        <span id="pageInfo" class="muted"></span>
        <button class="page-btn" id="next">Next</button>
    </div>
</div>

<script>
const API_BASE = 'http://localhost:4000/api/products';
let state = { page: 1, limit: 6, total: 0, items: [], filters: { q:'',
min:'', max:'', leaf:'' }, showRaw:false };

const grid = document.getElementById('grid');
const q = document.getElementById('q');
const min = document.getElementById('min');
const max = document.getElementById('max');
const leaf = document.getElementById('leaf');
const apply = document.getElementById('apply');
const prev = document.getElementById('prev');
const next = document.getElementById('next');
const pageInfo = document.getElementById('pageInfo');
const toggleRaw = document.getElementById('toggleRaw');

function fmtINR(n){ try{ return new Intl.NumberFormat('en-IN',
{style:'currency',currency:'INR',maximumFractionDigits:0}).format(n);}
catch{ return '₹'+n; } }

function pickThumb(item){
    const v = item.variants?.[0];
    const url = v?.images?.[0]?.url || '';
    return url ? `` : '';
}

function pricePair(v){
    const list = v?.price?.list ?? null;
    const sale = (v?.price?.sale ?? list);
    const main = sale ? fmtINR(sale) : (list? fmtINR(list): '');
    const strike = (sale && list && sale < list) ? `<s>${fmtINR(list)}</s>` :
    '';
    return `${main} ${strike}`;
}

function kv(obj){
    return Object.entries(obj||{}).map(([k,v])=>`<div class="kv"><div
class="muted">${k}</div><div>${v}</div></div>`).join('');
}

function renderCard(item){
    const leafPath = (item.category?.path||[]).join(' > ');
    const rating = `★ ${item.rating?.avg ?? 0} · ${item.rating?.count ?? 0}
`;

```

```

const variantsTable = `
  <div class="section">
    <h4 style="margin:6px 0 10px">Variants (${item.variants?.length||
0})</h4>
    <table>
      <thead><tr><th>SKU</th><th>Attrs</th><th>Price</th><th>Inventory
(warehouse→qty)</th><th>Images</th></tr></thead>
      <tbody>
        ${ (item.variants||[]).map(v=>`
          <tr>
            <td><span class="badge">${v.sku}</span></td>
            <td>${kv(v.attrs)}</td>
            <td>${pricePair(v)}</td>
            <td>${(v.inventory||[]).map(w=>`<span class="badge">${
{w.warehouse}: ${w.qty}</span>`).join(' ')}</td>
            <td>${(v.images||[]).map(img=>`<span class="badge">img</
span>`).join(' ')}</td>
          </tr>
        `).join('')}
      </tbody>
    </table>
  </div>`;

const raw = `<pre class="mono">${escapeHtml(JSON.stringify(item,null,2))}
</pre>`;

return `
  <article class="card">
    <div class="card-head">
      <div class="thumb">${pickThumb(item)}</div>
      <div style="flex:1">
        <h3 class="title">${item.title} <span class="chip">${
{item.brand||''}</span></h3>
        <div class="brand">${leafPath}</div>
        <div class="chips">${(item.tags||[]).map(t=>`<span class="chip">${
{t}</span>`).join('')} <span class="chip">${rating}</span></div>
      </div>
      <div class="price" style="align-self:center">${priceSummary(item)}
    </div>
  </div>

  <details>
    <summary><span class="caret">▶</span>Show nested fields (variants,
price, inventory, images, attributes)</summary>
    ${variantsTable}
    <div class="section">
      <h4 style="margin:10px 0 6px">Product Attributes</h4>
      ${kv(item.attributes)}
    </div>
    ${state.showRaw ? `<div class="section"><h4>Raw JSON</h4>${raw}</
div>` : ''}
  </details>
</article>
`;
}

function priceSummary(item){
  let minSale = Infinity, minList = Infinity;
  for(const v of (item.variants||[])){
    const sale = (v.price?.sale ?? v.price?.list);
    const list = v.price?.list ?? sale;
    if (sale < minSale) minSale = sale;
  }
}

```

```

    if (list < minList) minList = list;
  }
  const main = isFinite(minSale) ? fmtINR(minSale) : (isFinite(minList)?
fmtINR(minList): '');
  const strike = (isFinite(minSale) && isFinite(minList) && minSale <
minList) ? `${fmtINR(minList)}` : '';
  return `${main} ${strike}`;
}

function escapeHtml(s){return s.replace(/[\<>"']/
g,c=>({"&":"&","<":"&lt;",">":"&gt;","\"":"&quot;","'":"&#39;"}[c]))}

function render(){
  const { items, page, limit, total } = state;
  pageInfo.textContent = `Page ${page} of ${Math.max(1, Math.ceil(total/
limit))}`;
  prev.disabled = page <= 1; next.disabled = page >= Math.ceil(total/
limit);
  grid.innerHTML = items.map(renderCard).join('');
}

async function load(){
  const params = new URLSearchParams();
  params.set('page', state.page);
  params.set('limit', state.limit);
  if (state.filters.q) params.set('q', state.filters.q);
  if (state.filters.min) params.set('min', state.filters.min);
  if (state.filters.max) params.set('max', state.filters.max);
  if (state.filters.leaf) params.set('leaf', state.filters.leaf);

  const res = await fetch(`${API_BASE}?${params.toString()}`);
  const data = await res.json();
  state.items = data.items || []; state.total = data.total || 0;

  const leaves = [...new
Set(state.items.map(i=>i.category?.leaf).filter(Boolean))];
  if (leaf.options.length === 1 && leaves.length){
    for(const l of leaves){ const opt=document.createElement('option');
opt.value=l; opt.textContent=l; leaf.appendChild(opt); }
  }
  render();
}

apply.addEventListener('click', ()=>{ state.page=1;
state.filters={ q:q.value.trim(), min:min.value, max:max.value,
leaf:leaf.value }; load(); });
prev.addEventListener('click', ()=>{ state.page=Math.max(1,state.page-1);
load(); });
next.addEventListener('click', ()=>{ state.page=state.page+1; load(); });

toggleRaw.addEventListener('click', ()=>{ state.showRaw=!state.showRaw;
toggleRaw.textContent = `Raw JSON: ${state.showRaw? 'On':'Off'}`; render();
});

load();
</script>
</body>
</html>

```

FILE: .env (example)

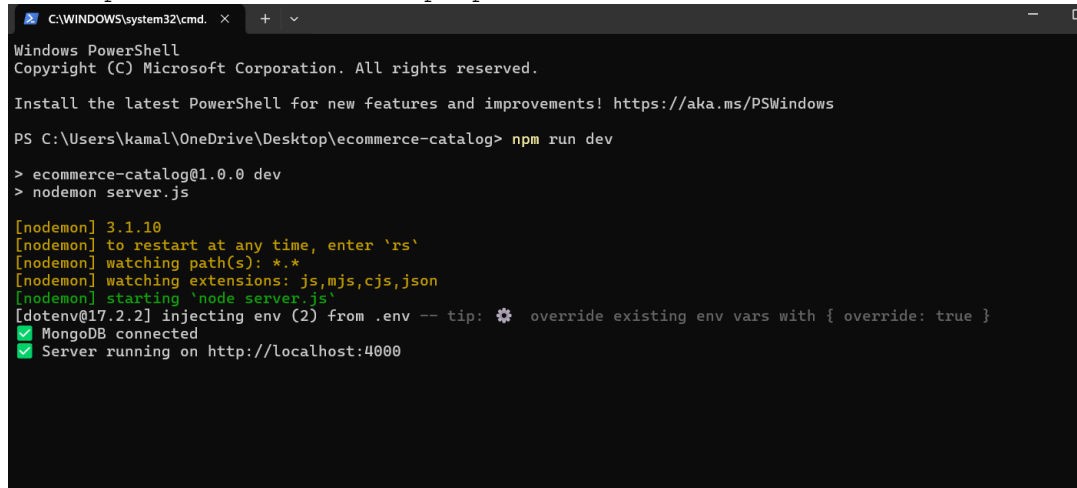
```

PORT=4000
MONGO_URI=mongodb://127.0.0.1:27017/ecommerce

```

QUICK TEST COMMAND

curl "http://localhost:4000/api/products"



```
C:\WINDOWS\system32\cmd. x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\kama1\OneDrive\Desktop\ecommerce-catalog> npm run dev

> ecommerce-catalog@1.0.0 dev
> nodemon server.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node server.js'
[dotenv@17.2.2] injecting env (2) from .env -- tip: ⚙ override existing env vars with { override: true }
✔ MongoDB connected
✔ Server running on http://localhost:4000
```

Figure 1: The cmd showing the server is running in mongoDB.

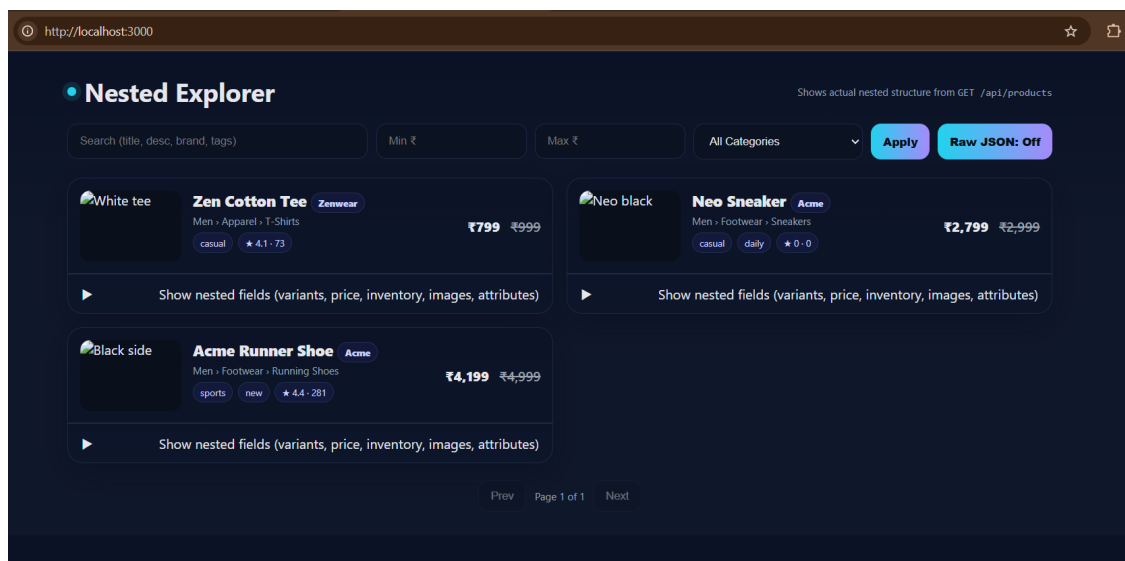


Figure 2 : System running on browser