MongoDB Document Data Model: Hands-On Lab (30–60 minutes)

This lab introduces the Document Data Model using MongoDB. You'll install MongoDB via Docker, perform CRUD operations with realistic sample data, and model an applied scenario (a product catalog with reviews and categories). The lab is suitable for beginners and is fully reproducible.

Prerequisites

i) Docker (version 24+ recommended) and Docker Compose (v2+)
ii) A terminal (macOS/Linux/WSL/PowerShell)
iii) Curl or a REST client (e.g., Postman)
iv) Optional: Node.js (v18+) if you want to run the sample API

What you'll do

i) Set up MongoDB in Docker and seed sample data
ii) Explore the Document Data Model basics
iii) Perform CRUD operations using Mongo Shell and API routes
iv) Apply the model to a product catalog scenario
v) View outputs and respond to practical questions
vi) Summarize team contributions

1. Setup Instructions

We'll use Docker Compose to start:

i) MongoDB Community Server 7.0
ii) Mongo Express (simple web UI) pinned to 1.0.2

Files:

i) docker-compose.yml`- services and networking
ii) seed/ - JSON data to pre-populate the database
iii) scripts/ - helper scripts for seeding and testing

Steps

1. Clone this repository

```
$ git clone https://github.com/estherndegwa/110544_111530_Cat1
Cloning into '110544_111530_Cat1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

2. Start services

```
$ docker compose up -d
[+] Running 9/9
 ✔ mongo Pulled                                                    59.5s
   ✔ 049450a11439 Pull complete                                    16.5s
   ✔ 9985892e237f Pull complete                                     1.7s
   ✔ 897d7bc84cb2 Pull complete                                    52.2s
   ✔ 9d2241e4f0ab Pull complete                                     1.7s
   ✔ 211ac2b07d3c Pull complete                                     1.6s
   ✔ 4afb2761a021 Pull complete                                    52.1s
   ✔ a901f9c6e182 Pull complete                                    16.4s
   ✔ 7e49dc6156b0 Pull complete                                    15.7s
[+] Running 4/4
 ✔ Network 110544_111530_cat1_default    Crea...                   0.4s
 ✔ Volume 110544_111530_cat1_mongo-data  Cr...                     0.0s
```

```
 ✔ Container mongo          Started                                1.0s
 ✔ Container mongo-express  Started                                0.7s
```

Verifies:

i) MongoDB listening on `mongodb://localhost:27017`
ii) Mongo Express UI at `http://localhost:8081`

3. Check container status

docker compose ps

```
$ docker compose ps
NAME            IMAGE                  COMMAND                  SERVICE         CREATED
        STATUS          PORTS
mongo           mongo:7.0              "docker-entrypoint.s…"   mongo           3 minutes ag
o       Up 36 seconds   0.0.0.0:27017->27017/tcp, [::]:27017->27017/tcp
mongo-express   mongo-express:1.0.2    "/sbin/tini -- /dock…"   mongo-express   About a minu
te ago   Up 35 seconds   0.0.0.0:8081->8081/tcp, [::]:8081->8081/tcp
```

Expected:

i) mongo` Up
ii) `mongo-express` Up

4. Seed sample data

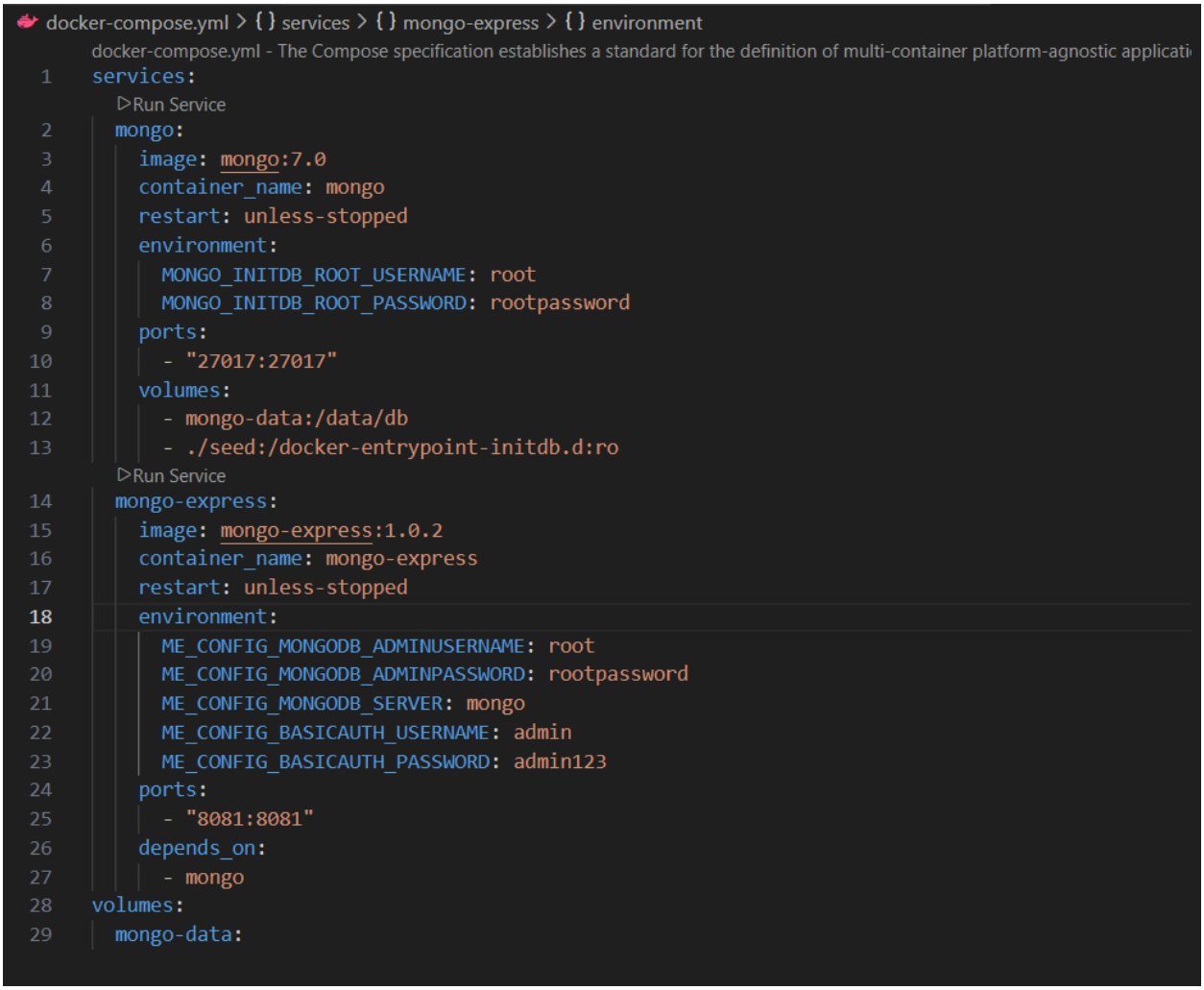
./scripts/seed.sh

This loads `products`, `categories`, and `reviews` into a `shop` database.

```
$ ./scripts/seed.sh
Seeding MongoDB...
Current Mongosh Log ID: 6929b6b470b51ad3ab9dc29c
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serv
erSelectionTimeoutMS=2000&authSource=admin&appName=mongosh+2.5.9
Using MongoDB:          7.0.26
Using Mongosh:          2.5.9

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodi
cally (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2025-11-28T14:45:14.754+00:00: Using the XFS filesystem is strongly recommended with the
 WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2025-11-28T14:45:15.641+00:00: vm.max_map_count is too low
------

test>
test>
test>
test>
shop>
shop>
shop> true
shop> true
shop> true
shop>
shop> ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... .
.. ... ... ... ... ... ... ... ... ... ... ... ... {
  acknowledged: true,
  insertedIds: { '0': 'SKU-1001', '1': 'SKU-1002', '2': 'SKU-2001' }
}
shop>
shop> ... ... ... ... {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6929b6b570b51ad3ab9dc29d'),
    '1': ObjectId('6929b6b570b51ad3ab9dc29e'),
    '2': ObjectId('6929b6b570b51ad3ab9dc29f')
  }
}
shop>
shop> ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... .
.. {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6929b6b570b51ad3ab9dc2a0'),
    '1': ObjectId('6929b6b570b51ad3ab9dc2a1'),
    '2': ObjectId('6929b6b570b51ad3ab9dc2a2')
  }
}
shop>
shop>
shop> categories_1
shop> price_1_in_stock_1
shop> product_id_1_created_at_-1
shop>
shop> Seed completed for 'shop' database.
```

Notes:

i) Configuration is set in `docker-compose.yml`. Auth is enabled with `MONGO_INITDB_ROOT_USERNAME` and `MONGO_INITDB_ROOT_PASSWORD`.
ii) For Windows, run scripts via PowerShell equivalents or use Git Bash.



Document Data Model Basics

MongoDB stores data as BSON (binary JSON). Collections hold documents (like JSON objects) with flexible schemas.

Key strengths:

i) Flexible schema: evolve fields without migrations
ii) Nested documents: embed related data (e.g., specs, ratings)
iii) Rich querying on nested fields, arrays, and text

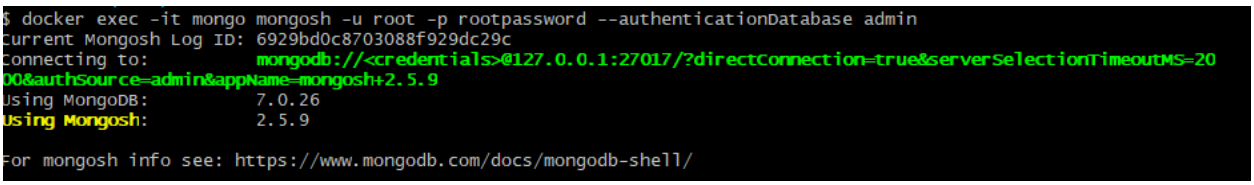
CRUD Operations

You can use either:

i) MongoDB Shell (`mongosh` inside the container)
ii) HTTP API endpoints (optional Node.js Express service)
iii) Mongo Express UI for quick visibility

Connect to Mongo Shell

docker exec -it mongo mongosh -u root -p rootpassword --authenticationDatabase admin



Switch to the `shop` database:

use shop

i) Create

Insert a new product:

```
shop> db.products.insertOne({
..    _id: "SKU-2013",
..    name: "USB-C Charger 65W",
..    brand: "ChargePro",
..    price: 39.99,
..    in_stock: true,
..    categories: ["power", "accessories"],
..    specs: { color: "white", wattage: 65, ports: ["USB-C"] },
..    tags: ["fast-charge", "compact"],
..    ratings: { average: 4.3, count: 56 }
.. })
{ acknowledged: true, insertedId: 'SKU-2013' }
```

Insert a review:

```
shop> db.reviews.insertOne({
...    product_id: "SKU-2013",
...    user: { id: "U-100", name: "Jane Doe" },
...    rating: 5,
...    comment: "Charges my laptop and phone fast!",
...    created_at: new Date()
... })
{
  acknowledged: true,
  insertedId: ObjectId('6929be3f4bd92de2919dc29d')
}
shop>
```

ii) Read

Find by ID: db.products.findOne({ id: "SKU-2001" })

```
shop> db.products.findOne({ _id: "SKU-2001" })
{
  _id: 'SKU-2001',
  name: 'USB-C Charger 65W',
  brand: 'ChargePro',
  price: 39.99,
  in_stock: true,
  categories: [ 'power', 'accessories' ],
  specs: { color: 'white', wattage: 65, ports: [ 'USB-C' ] },
  tags: [ 'fast-charge', 'compact' ],
  ratings: { average: 4.3, count: 56 }
}
```

Filter and projection:

```
shop> db.products.find(
...    { price: { $lt: 100 }, in_stock: true, categories: "accessories" },
...    { name: 1, brand: 1, price: 1, _id: 0 }
... )
[
  { name: 'USB-C Charger 65W', brand: 'ChargePro', price: 39.99 },
  { name: 'USB-C Charger 65W', brand: 'ChargePro', price: 39.99 }
]
```

Nested field query:

```
shop> db.products.find({ "specs.wattage": { $gte: 60 } })
[
  {
    _id: 'SKU-2001',
    name: 'USB-C Charger 65W',
    brand: 'ChargePro',
    price: 39.99,
    in_stock: true,
    categories: [ 'power', 'accessories' ],
    specs: { color: 'white', wattage: 65, ports: [ 'USB-C' ] },
    tags: [ 'fast-charge', 'compact' ],
    ratings: { average: 4.3, count: 56 }
  },
  {
    _id: 'SKU-2013',
    name: 'USB-C Charger 65W',
    brand: 'ChargePro',
    price: 39.99,
    in_stock: true,
    categories: [ 'power', 'accessories' ],
    specs: { color: 'white', wattage: 65, ports: [ 'USB-C' ] },
    tags: [ 'fast-charge', 'compact' ],
    ratings: { average: 4.3, count: 56 }
  }
]
```

iii) Update

Update a price and add a tag:

```
shop> db.products.updateOne(
...   { _id: "SKU-2001" },
...   { $set: { price: 34.99 }, $addToSet: { tags: "travel" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Upsert (insert if missing):

```
shop> db.products.updateOne(
...   { _id: "SKU-9999" },
...   { $set: { name: "Demo Product", price: 9.99, in_stock: false } },
...   { upsert: true }
... )
{
  acknowledged: true,
  insertedId: 'SKU-9999',
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
```

iv) Delete

Remove a product:

```
shop> db.products.deleteOne({ _id: "SKU-9999" })
{ acknowledged: true, deletedCount: 1 }
```

Cascade-like cleanup (manual):

```
shop> db.reviews.deleteMany({ product_id: "SKU-9999" })
{ acknowledged: true, deletedCount: 0 }
```

APPLIED SCENARIO: PRODUCT CATALOG WITH REVIEWS

Problem

An online store needs to manage:

i) Products with flexible attributes (specs vary by category)
ii) Categories and tags for navigation
iii) Customer reviews with ratings and comments
iv) Efficient reads of product details and related reviews

Why Document Model?

i) Products are naturally document-shaped with nested attributes
ii) Schema flexibility allows adding new specs without migrations
iii) Embedding summary fields (like ratings, average) supports fast reads
iv) Separate `reviews` collection keeps large lists manageable

Data Model

Collections:

i) `products`: product documents with nested `specs`, `ratings`, arrays `categories`, `tags`
ii) `categories`: top-level category definitions and metadata
iii) `reviews`: customer reviews referencing `product_id`

a) Read product with recent reviews (aggregation):

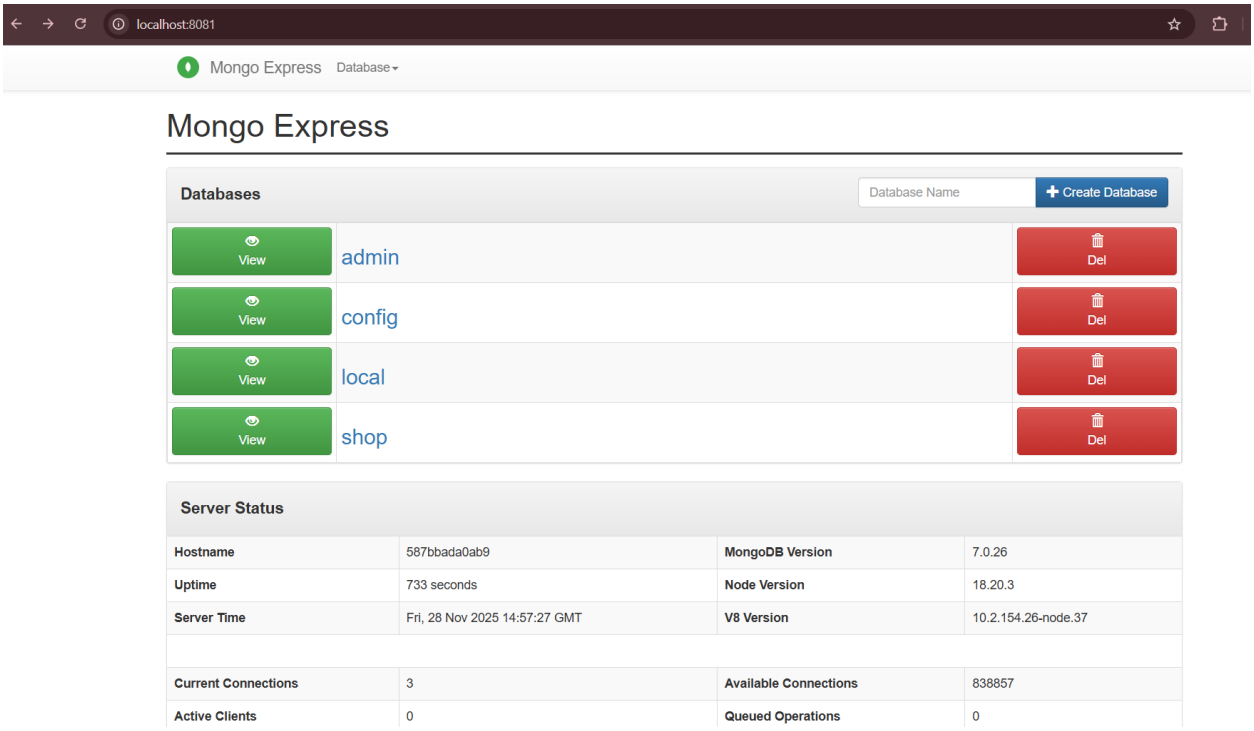
```
[
  {
    _id: 'SKU-2001',
    name: 'USB-C Charger 65W',
    brand: 'ChargePro',
    price: 34.99,
    in_stock: true,
    categories: [ 'power', 'accessories' ],
    specs: { color: 'white', wattage: 65, ports: [ 'USB-C' ] },
    tags: [ 'fast-charge', 'compact', 'travel' ],
    ratings: { average: 4.3, count: 56 },
    recent_reviews: [
      {
        _id: ObjectId('6929b6b570b51ad3ab9dc2a2'),
        product_id: 'SKU-2001',
        user: { id: 'U-003', name: 'Charlie' },
        rating: 5,
        comment: 'Charges my laptop fast.',
        created_at: ISODate('2025-11-28T14:50:29.825Z')
      }
    ]
  }
]
```

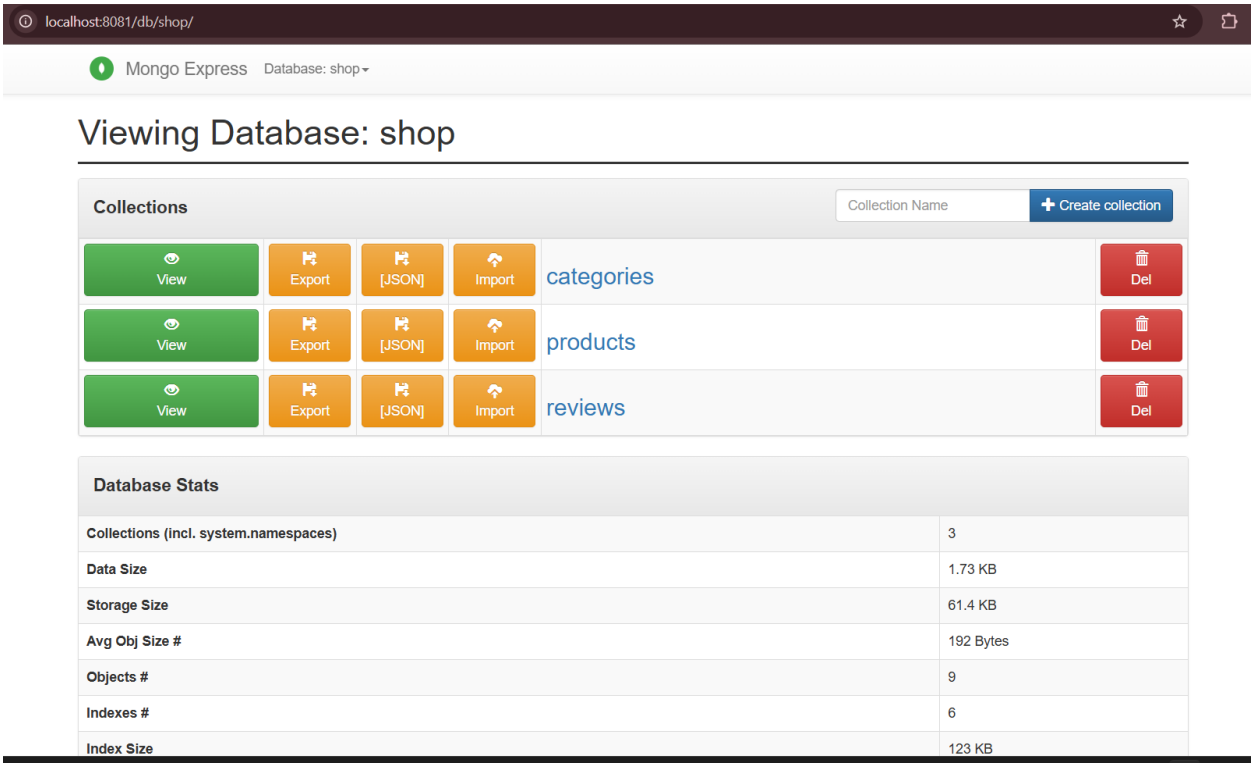
b) Update rating summary after new review:

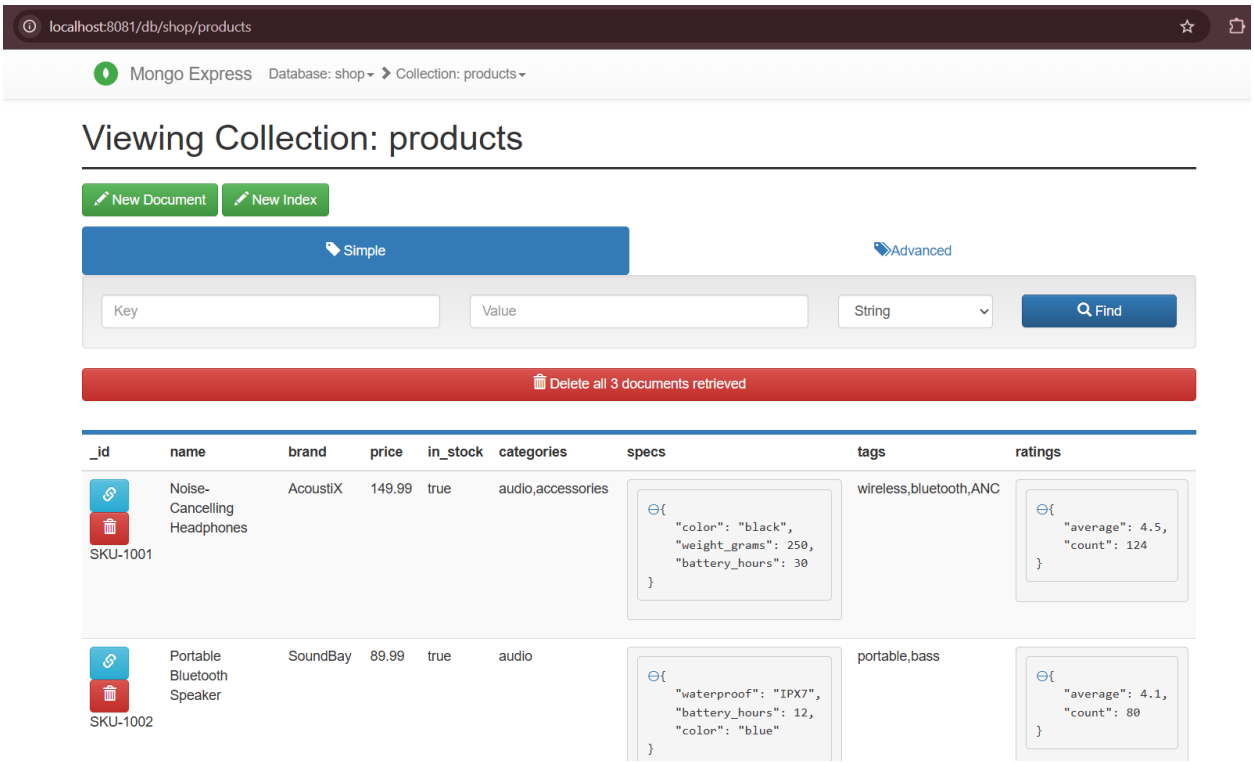
```
shop> const pid = "SKU-2001";
... const summary = db.reviews.aggregate([
...   { $match: { product_id: pid } },
...   { $group: { _id: "$product_id", count: { $sum: 1 }, avg: { $avg: "$rating" } } }
... ]).toArray()[0];
...
... db.products.updateOne(
...   { _id: pid },
...   { $set: { "ratings.count": summary.count, "ratings.average": summary.avg } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

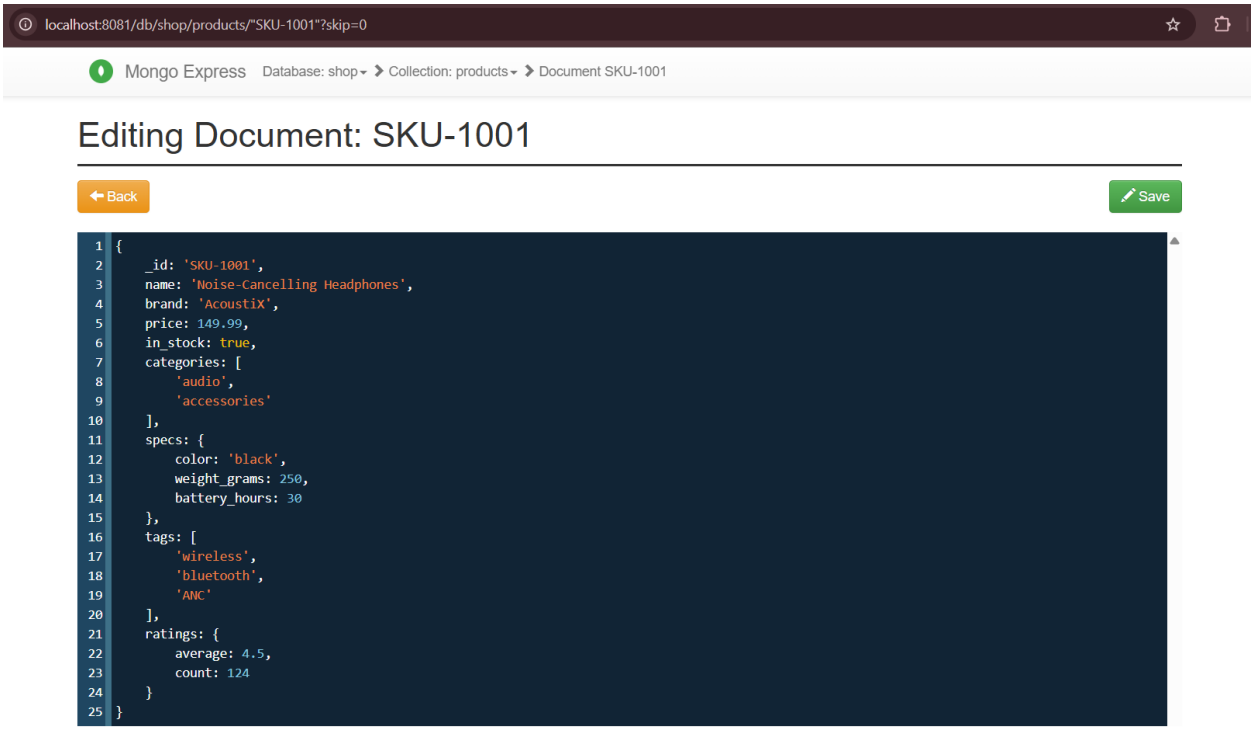
VISUALS AND OUTPUT

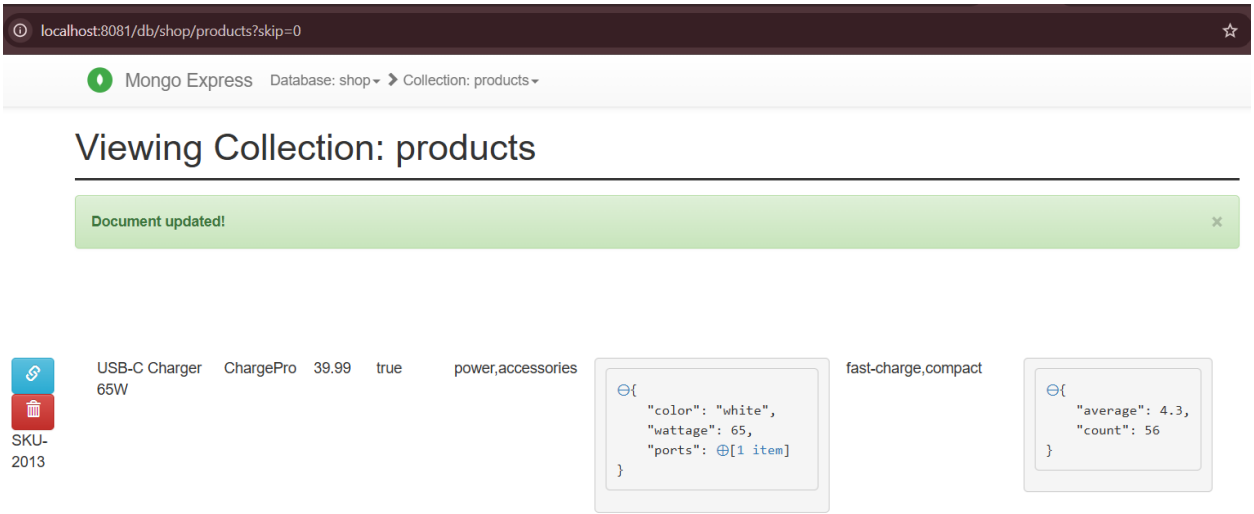
a) Use Mongo Express at `http://localhost:8081`:
   i) View `shop` database and browse `products` and `reviews`.
   ii) Confirms inserts, updates, and deletes.




localhost:8081/db/shop/
Mongo Express
Database: shop
Viewing Database: shop
Collections
Collection Name
Create collection
View
Export
[JSON]
Import
categories
Del
products
reviews
Database Stats
Collections (incl. system.namespaces) 3
Data Size 1.73 KB
Storage Size 61.4 KB
Avg Obj Size # 192 Bytes
Objects # 9
Indexes # 6
Index Size 123 KB


localhost:8081/db/shop/products
Mongo Express
Database: shop
Collection: products
Viewing Collection: products
New Document
New Index
Simple
Advanced
Key
Value
String
Find
Delete all 3 documents retrieved
_id
name
brand
price
in_stock
categories
specs
tags
ratings
SKU-1001
Noise-Cancelling Headphones
AcoustiX
149.99
true
audio,accessories
{ "color": "black", "weight_grams": 250, "battery_hours": 30 }
wireless,bluetooth,ANC
{ "average": 4.5, "count": 124 }
SKU-1002
Portable Bluetooth Speaker
SoundBay
89.99
true
audio
{ "waterproof": "IPX7", "battery_hours": 12, "color": "blue" }
portable,bass
{ "average": 4.1, "count": 80 }


localhost:8081/db/shop/products/"SKU-1001"?skip=0
Mongo Express
Database: shop
Collection: products
Document SKU-1001
Editing Document: SKU-1001
Back
Save
{
_id: 'SKU-1001',
name: 'Noise-Cancelling Headphones',
brand: 'AcoustiX',
price: 149.99,
in_stock: true,
categories: [
'audio',
'accessories'
],
specs: {
color: 'black',
weight_grams: 250,
battery_hours: 30
},
tags: [
'wireless',
'bluetooth',
'ANC'
],
ratings: {
average: 4.5,
count: 124
}
}


localhost:8081/db/shop/products?skip=0
Mongo Express
Database: shop
Collection: products
Viewing Collection: products
Document updated!
SKU-2013
USB-C Charger 65W
ChargePro
39.99
true
power,accessories
{
"color": "white",
"wattage": 65,
"ports": [1 item]
}
fast-charge,compact
{
"average": 4.3,
"count": 56
}