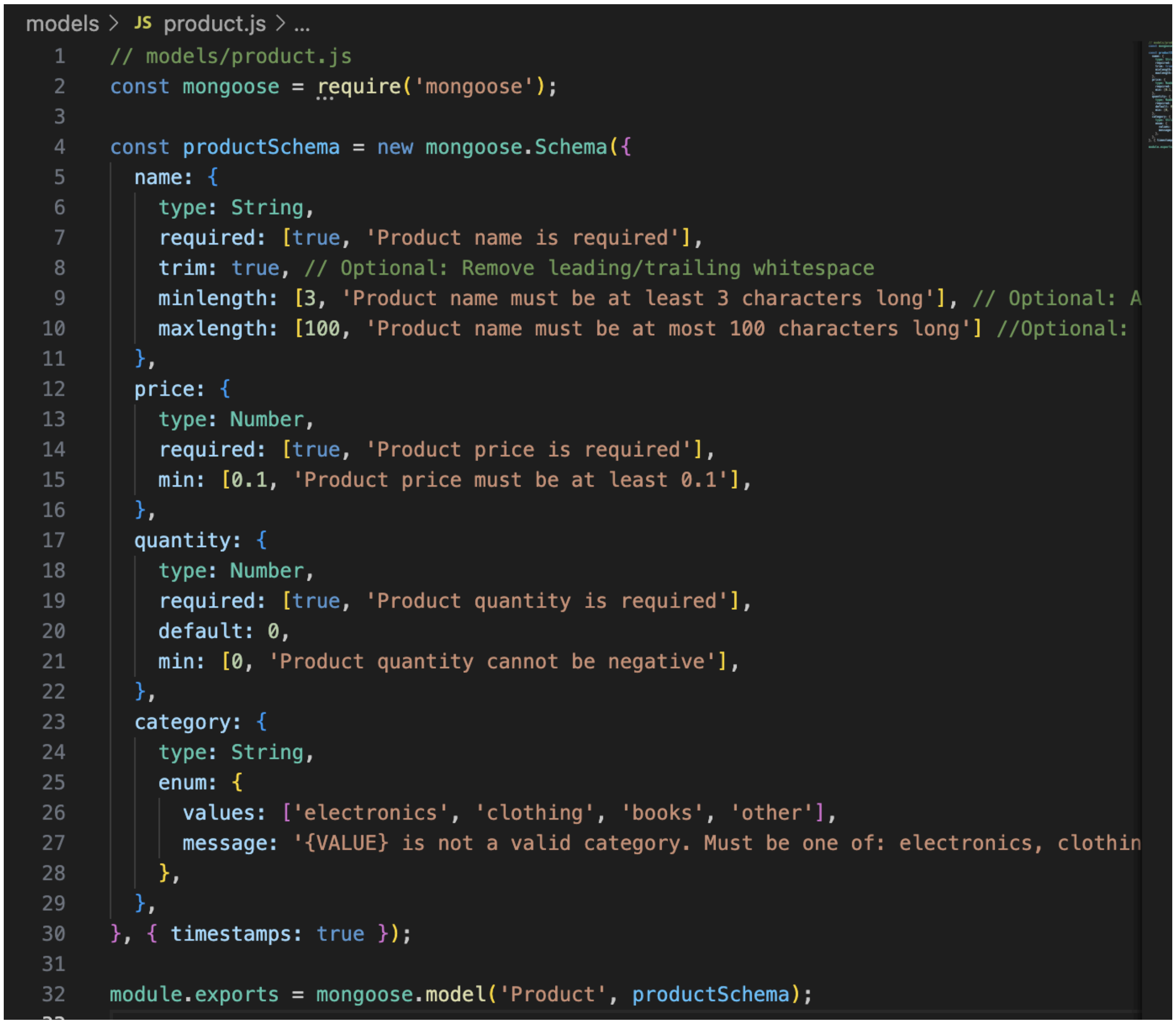
# DATA-236 Sec 12 - Distributed Systems for Data Engineering
# HOMEWORK 5

**Nandhakumar Apparsamy**
**018190003**

Objective Build a RESTful API using Node.js and MongoDB (with Mongoose) to manage a product inventory. The API should perform basic CRUD operations and include data validation.

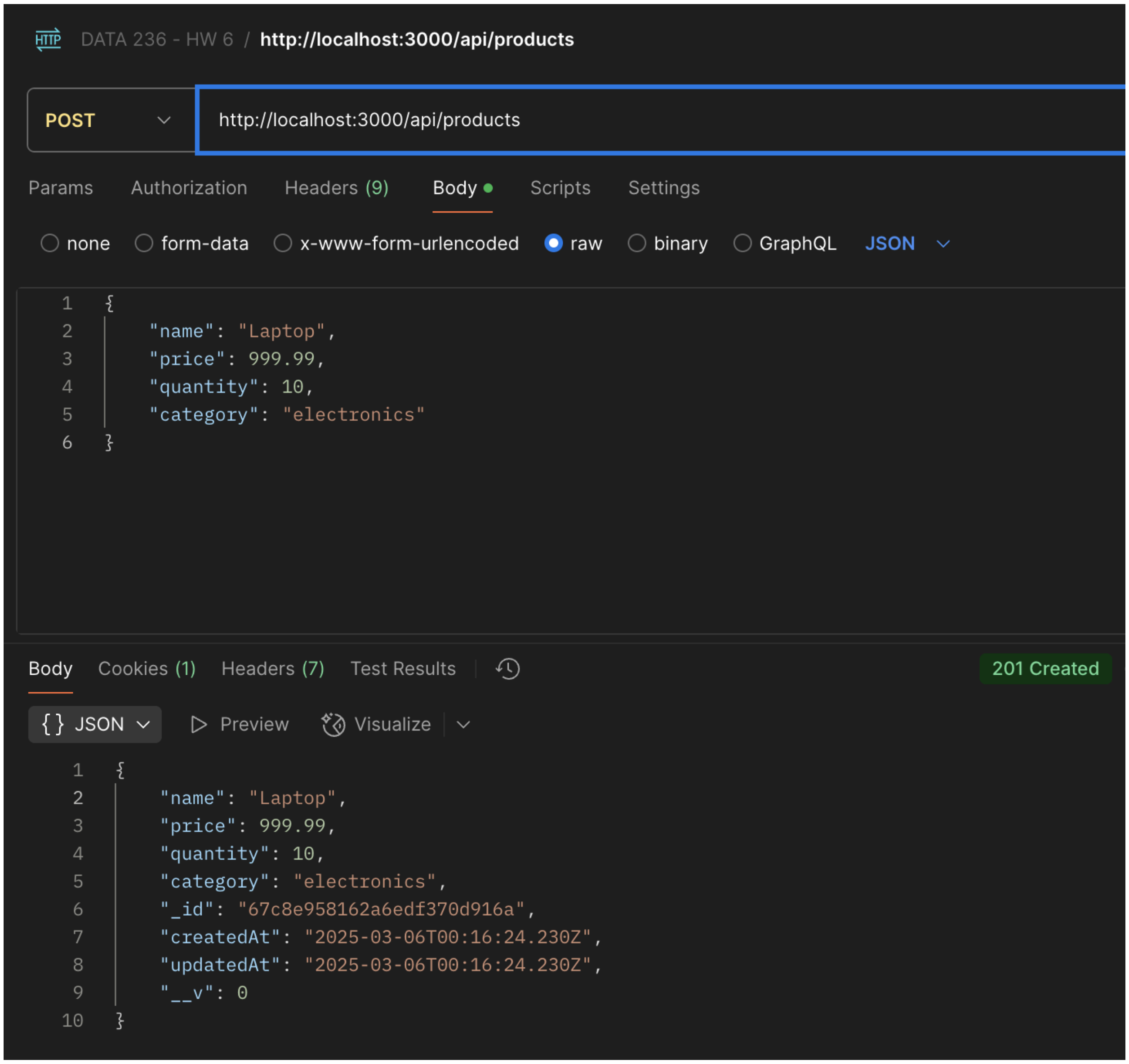
**Data Model (submit models code)**
Create a Product model with these fields:
● name (String, required)
● price (Number, required, minimum value: 0.1)
● quantity (Number, required, default: 0, minimum: 0)
● category (String, enum: ['electronics', 'clothing', 'books', 'other'])
● createdAt and updatedAt (Timestamps, auto-generated)



models > JS product.js > ...

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

const productSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, 'Product name is required'],
    trim: true, // Optional: Remove leading/trailing whitespace
    minlength: [3, 'Product name must be at least 3 characters long'], // Optional: A
    maxlength: [100, 'Product name must be at most 100 characters long'] //Optional:
  },
  price: {
    type: Number,
    required: [true, 'Product price is required'],
    min: [0.1, 'Product price must be at least 0.1'],
  },
  quantity: {
    type: Number,
    required: [true, 'Product quantity is required'],
    default: 0,
    min: [0, 'Product quantity cannot be negative'],
  },
  category: {
    type: String,
    enum: {
      values: ['electronics', 'clothing', 'books', 'other'],
      message: '{VALUE} is not a valid category. Must be one of: electronics, clothin
    },
  },
}, { timestamps: true });

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

**API Endpoints (1 screenshot per action)**
Implement these endpoints:
1. POST `/api/products` Create a new product

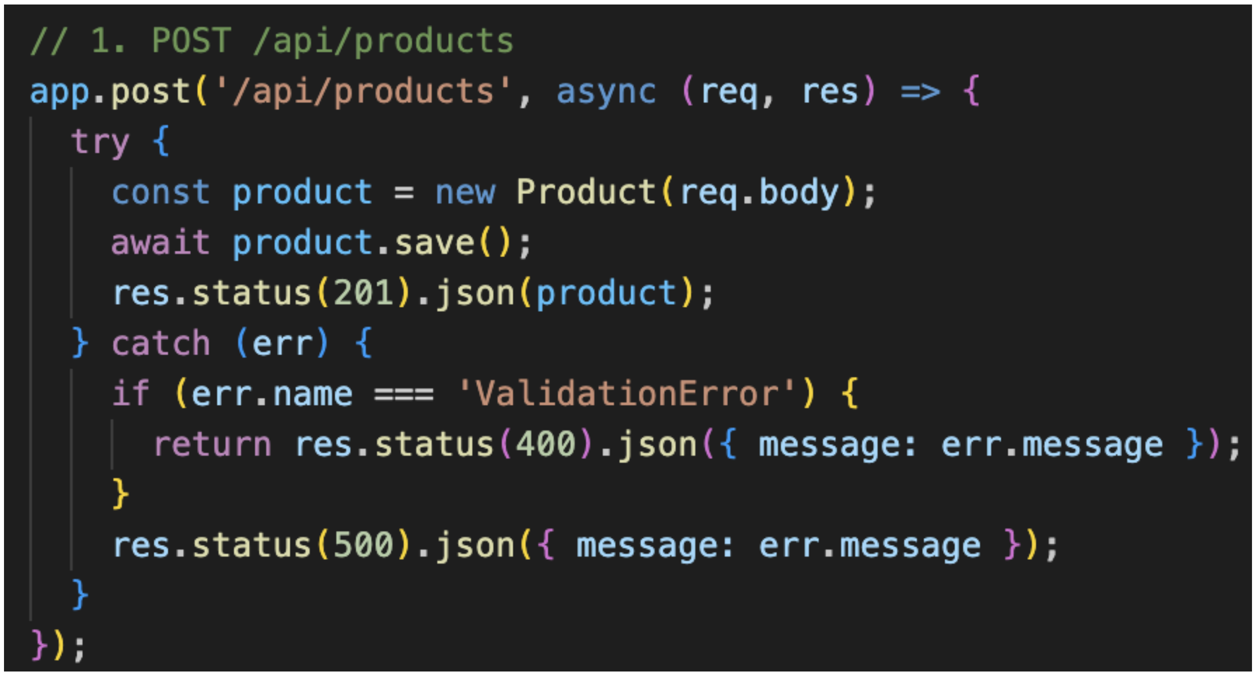

DATA 236 - HW 6 / http://localhost:3000/api/products

POST http://localhost:3000/api/products

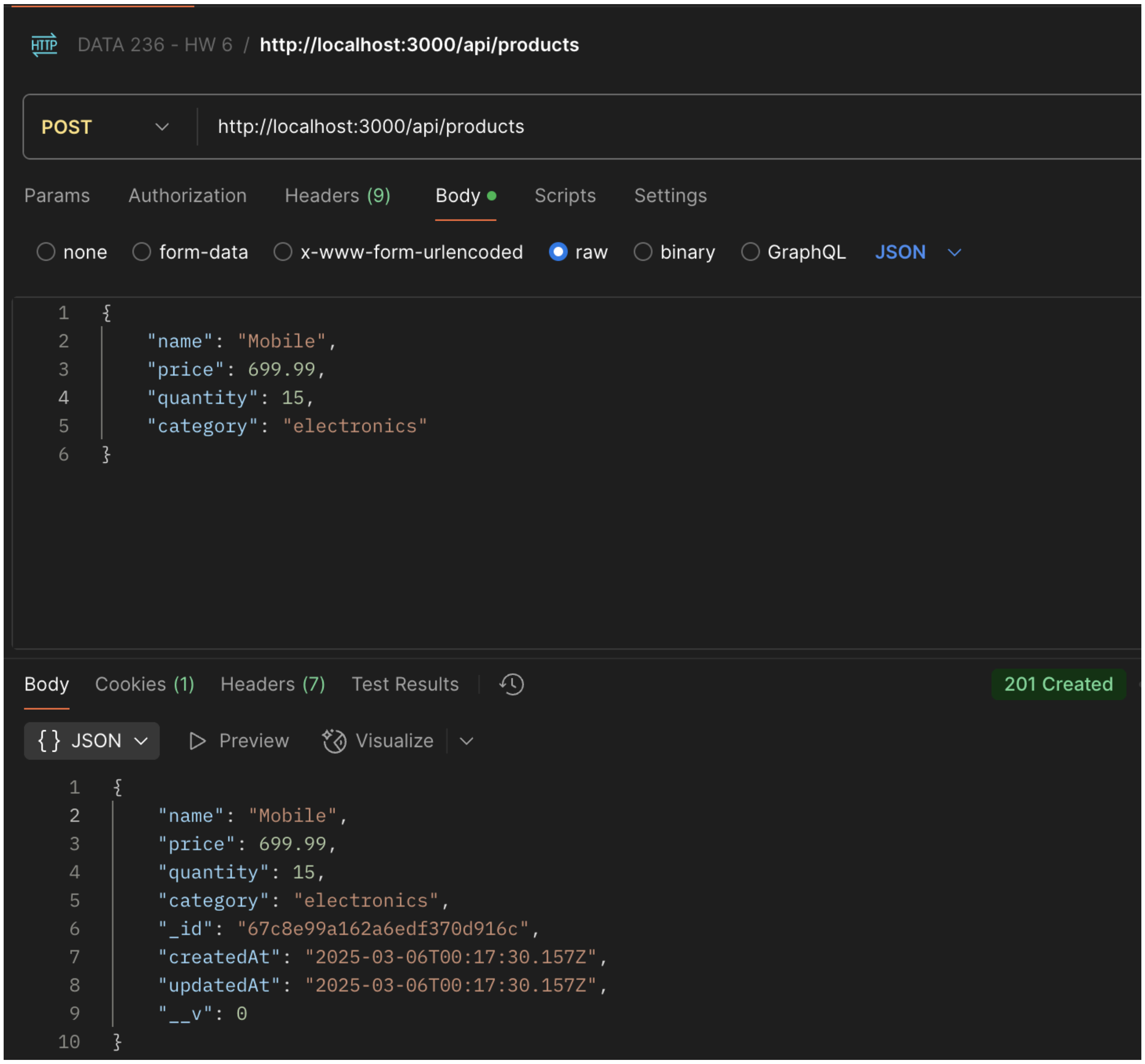
none · form-data · x-www-form-urlencoded · [✓ raw] · binary · GraphQL · JSON

```
{
    "name": "Laptop",
    "price": 999.99,
    "quantity": 10,
    "category": "electronics"
}
```

```
{
    "name": "Laptop",
    "price": 999.99,
    "quantity": 10,
    "category": "electronics",
    "_id": "67c8e958162a6edf370d916a",
    "createdAt": "2025-03-06T00:16:24.230Z",
    "updatedAt": "2025-03-06T00:16:24.230Z",
    "__v": 0
}
```



```javascript
// 1. POST /api/products
app.post('/api/products', async (req, res) => {
  try {
    const product = new Product(req.body);
    await product.save();
    res.status(201).json(product);
  } catch (err) {
    if (err.name === 'ValidationError') {
      return res.status(400).json({ message: err.message });
    }
    res.status(500).json({ message: err.message });
  }
});
```


DATA 236 - HW 6 / http://localhost:3000/api/products

POST http://localhost:3000/api/products

Params | Authorization | Headers (9) | Body ● | Scripts | Settings

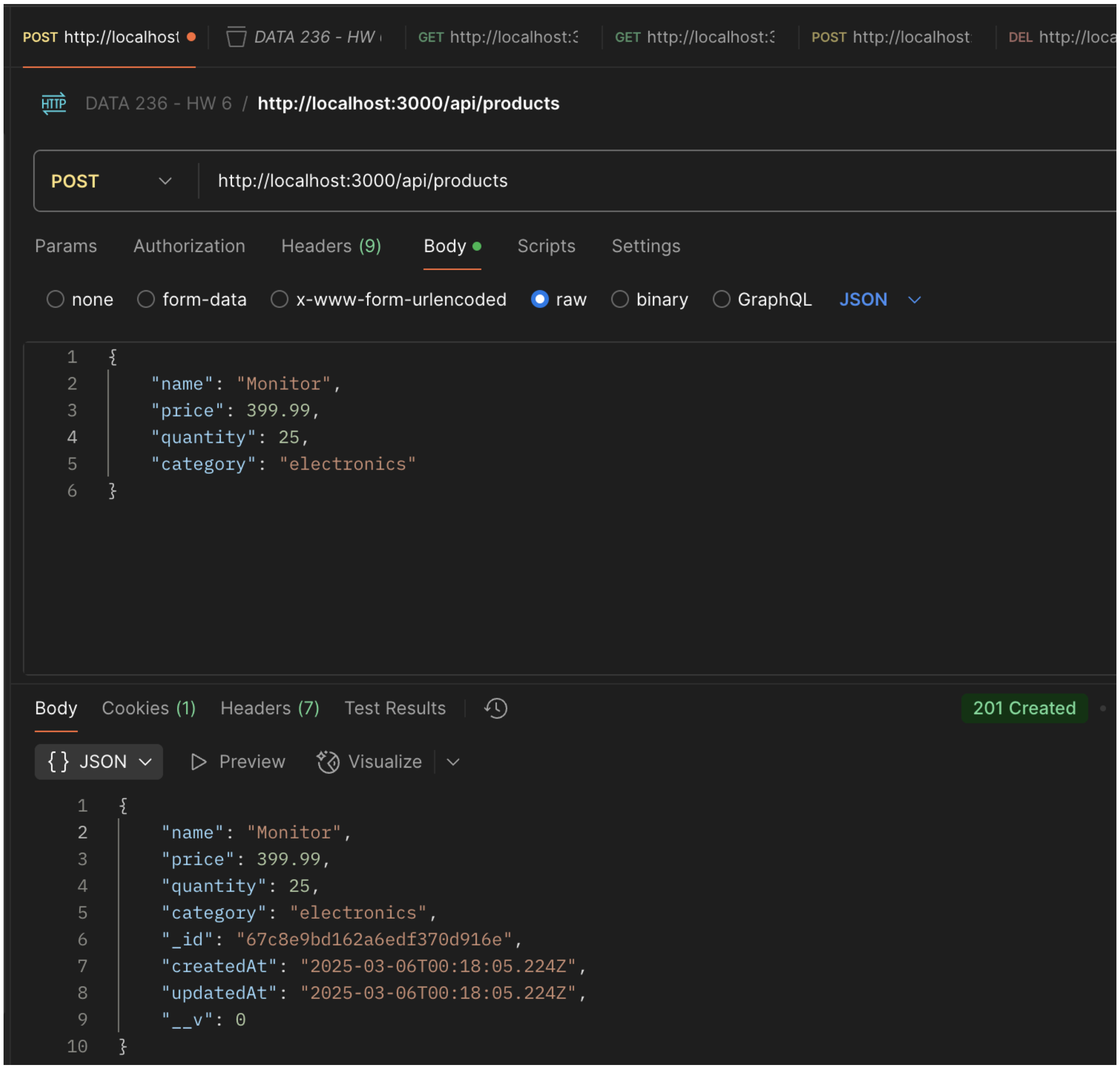
none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "name": "Mobile",
    "price": 699.99,
    "quantity": 15,
    "category": "electronics"
}
```

Body | Cookies (1) | Headers (7) | Test Results — 201 Created

{} JSON | Preview | Visualize

```
{
    "name": "Mobile",
    "price": 699.99,
    "quantity": 15,
    "category": "electronics",
    "_id": "67c8e99a162a6edf370d916c",
    "createdAt": "2025-03-06T00:17:30.157Z",
    "updatedAt": "2025-03-06T00:17:30.157Z",
    "__v": 0
}
```



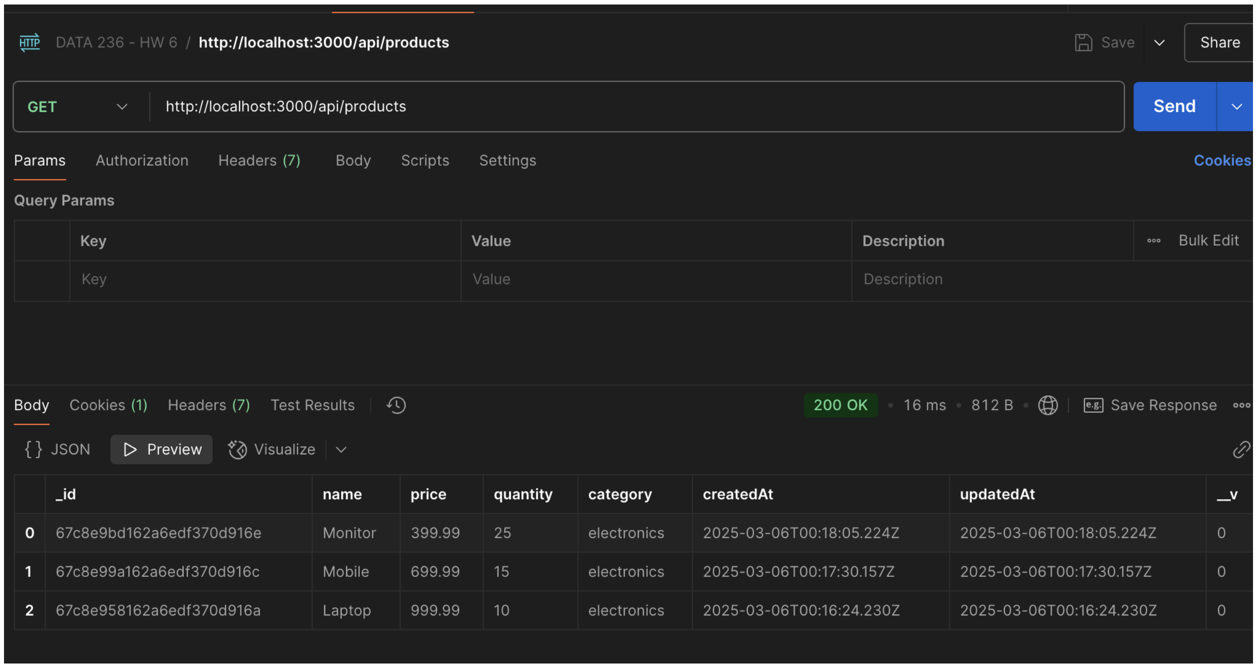
POST http://localhost:3000/api/products

```
{
    "name": "Monitor",
    "price": 399.99,
    "quantity": 25,
    "category": "electronics"
}
```

201 Created

```
{
    "name": "Monitor",
    "price": 399.99,
    "quantity": 25,
    "category": "electronics",
    "_id": "67c8e9bd162a6edf370d916e",
    "createdAt": "2025-03-06T00:18:05.224Z",
    "updatedAt": "2025-03-06T00:18:05.224Z",
    "__v": 0
}
```

2. GET `/api/products` Get all products (sorted by newest first)


DATA 236 - HW 6 / http://localhost:3000/api/products

GET http://localhost:3000/api/products

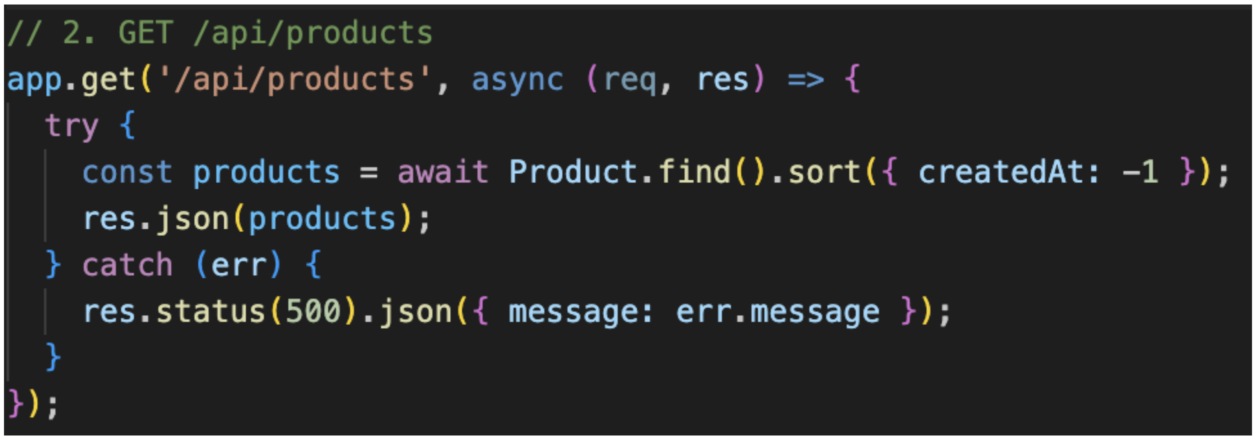
Params | Authorization | Headers (7) | Body | Scripts | Settings

Query Params

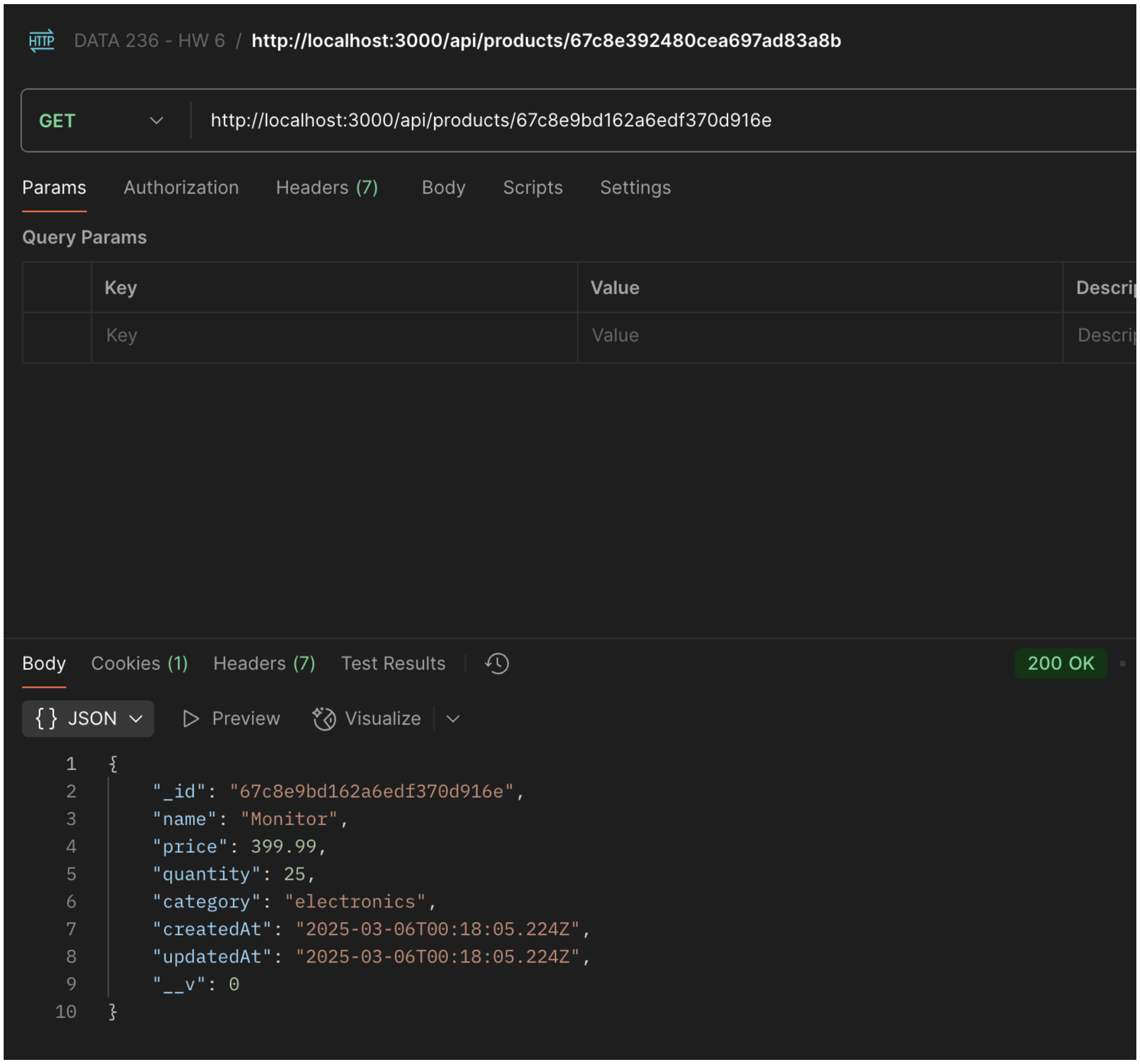
| Key | Value | Description |
|---|---|---|
| Key | Value | Description |

Body | Cookies (1) | Headers (7) | Test Results — 200 OK • 16 ms • 812 B

|  | _id | name | price | quantity | category | createdAt | updatedAt | __v |
|---|---|---|---|---|---|---|---|---|
| 0 | 67c8e9bd162a6edf370d916e | Monitor | 399.99 | 25 | electronics | 2025-03-06T00:18:05.224Z | 2025-03-06T00:18:05.224Z | 0 |
| 1 | 67c8e99a162a6edf370d916c | Mobile | 699.99 | 15 | electronics | 2025-03-06T00:17:30.157Z | 2025-03-06T00:17:30.157Z | 0 |
| 2 | 67c8e958162a6edf370d916a | Laptop | 999.99 | 10 | electronics | 2025-03-06T00:16:24.230Z | 2025-03-06T00:16:24.230Z | 0 |

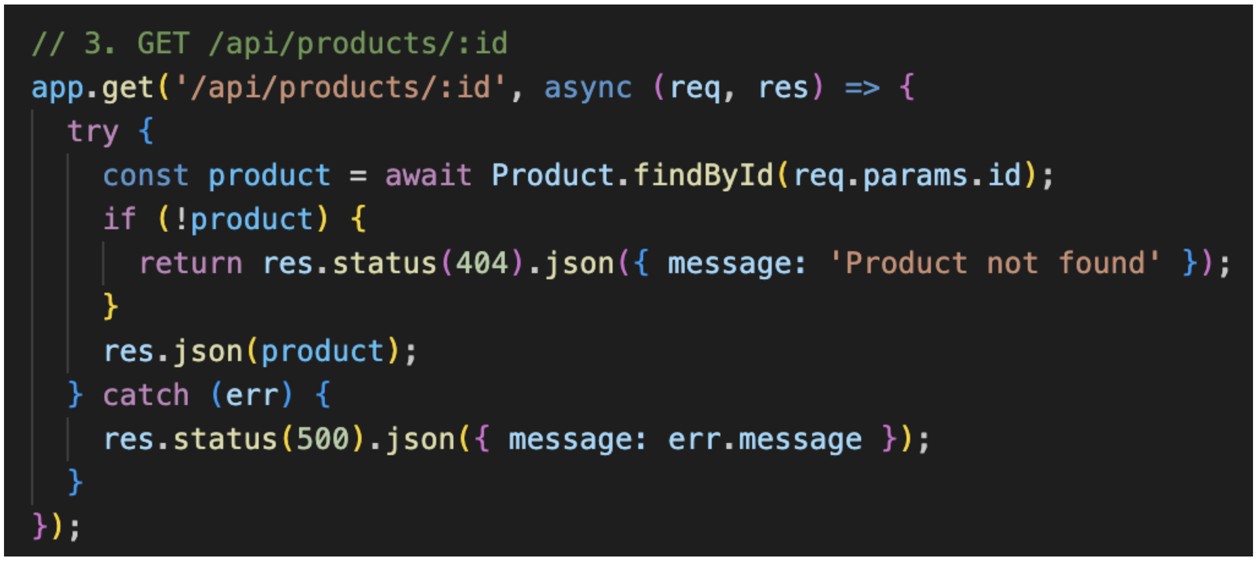

```
// 2. GET /api/products
app.get('/api/products', async (req, res) => {
  try {
    const products = await Product.find().sort({ createdAt: -1 });
    res.json(products);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});
```

3. GET `/api/products/:id` Get a single product by ID


DATA 236 - HW 6 / http://localhost:3000/api/products/67c8e392480cea697ad83a8b

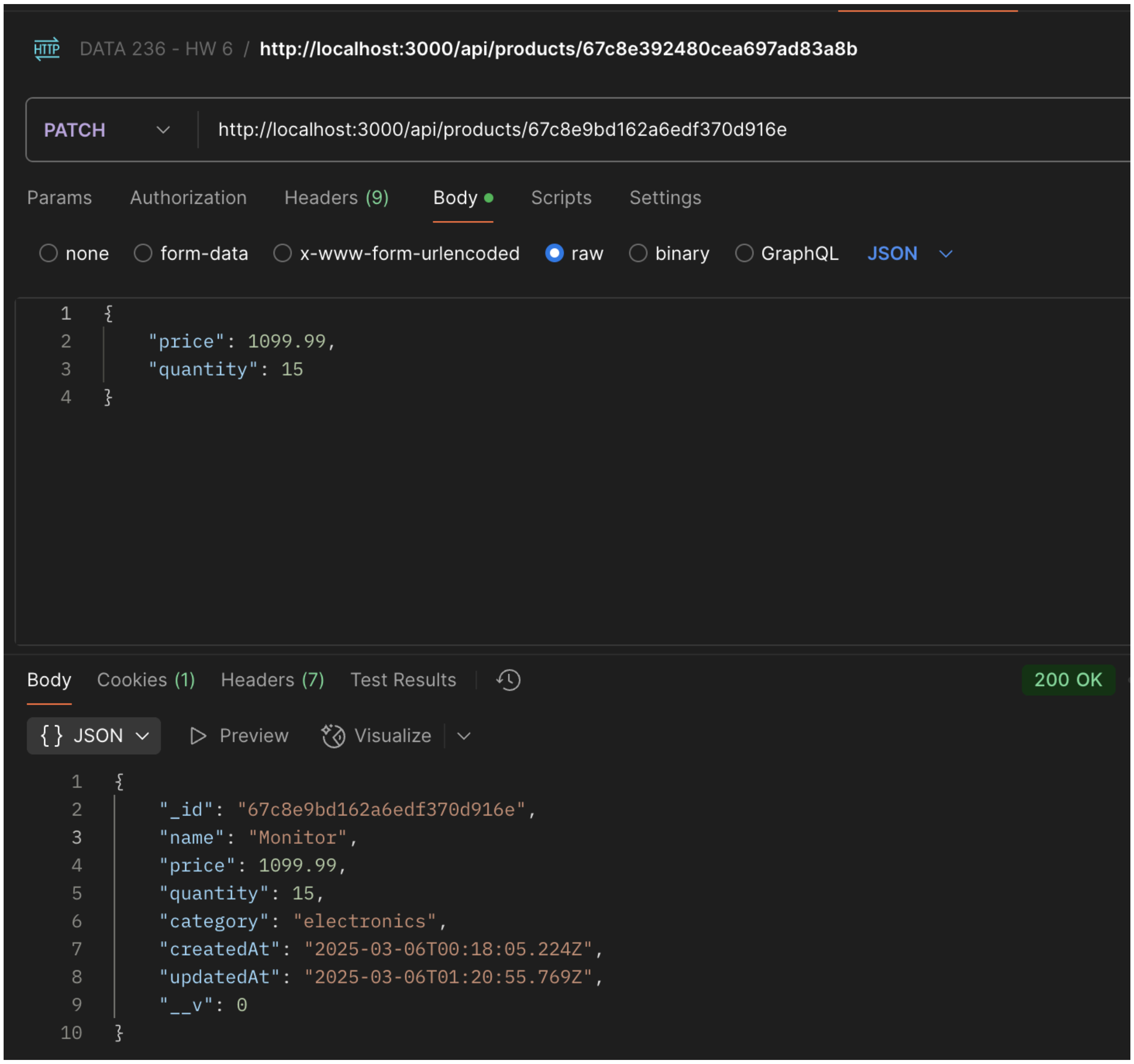
GET http://localhost:3000/api/products/67c8e9bd162a6edf370d916e

```
{
    "_id": "67c8e9bd162a6edf370d916e",
    "name": "Monitor",
    "price": 399.99,
    "quantity": 25,
    "category": "electronics",
    "createdAt": "2025-03-06T00:18:05.224Z",
    "updatedAt": "2025-03-06T00:18:05.224Z",
    "__v": 0
}
```



```
// 3. GET /api/products/:id
app.get('/api/products/:id', async (req, res) => {
  try {
    const product = await Product.findById(req.params.id);
    if (!product) {
      return res.status(404).json({ message: 'Product not found' });
    }
    res.json(product);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});
```

4. PATCH `/api/products/:id` Update product details (allow partial updates)



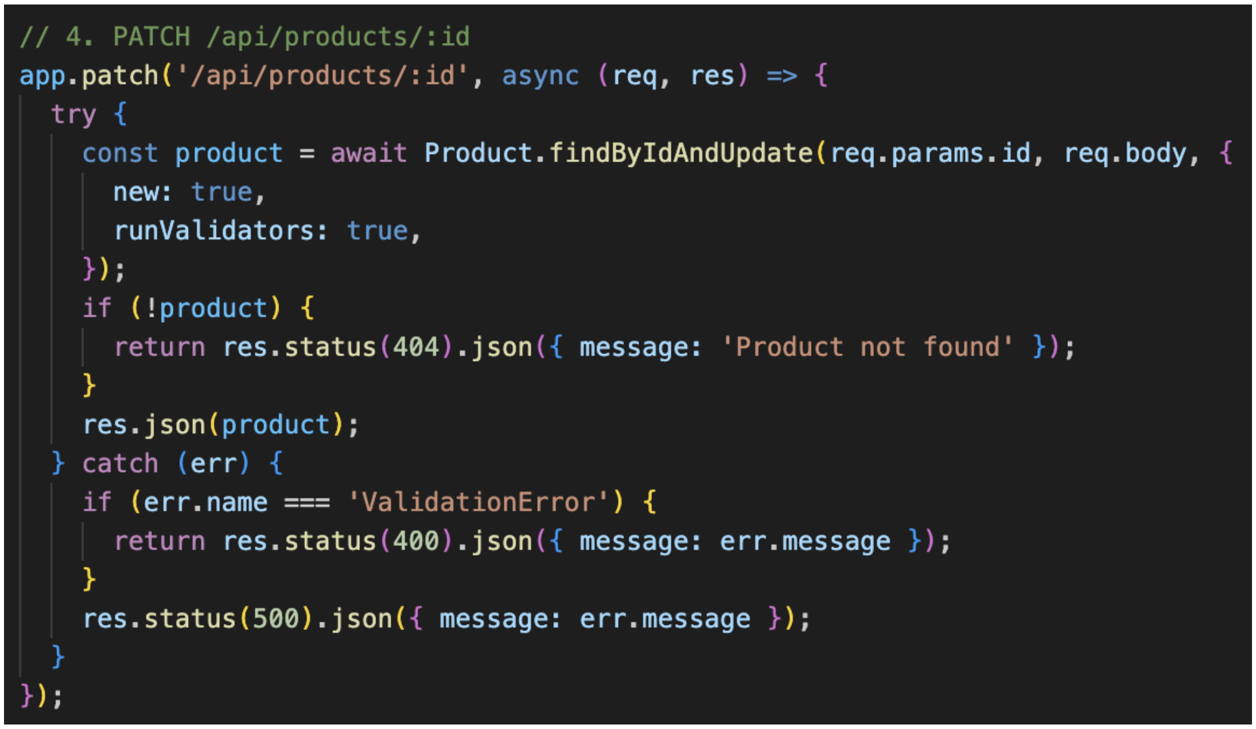
DATA 236 - HW 6 / http://localhost:3000/api/products/67c8e392480cea697ad83a8b

PATCH http://localhost:3000/api/products/67c8e9bd162a6edf370d916e

```
{
    "price": 1099.99,
    "quantity": 15
}
```

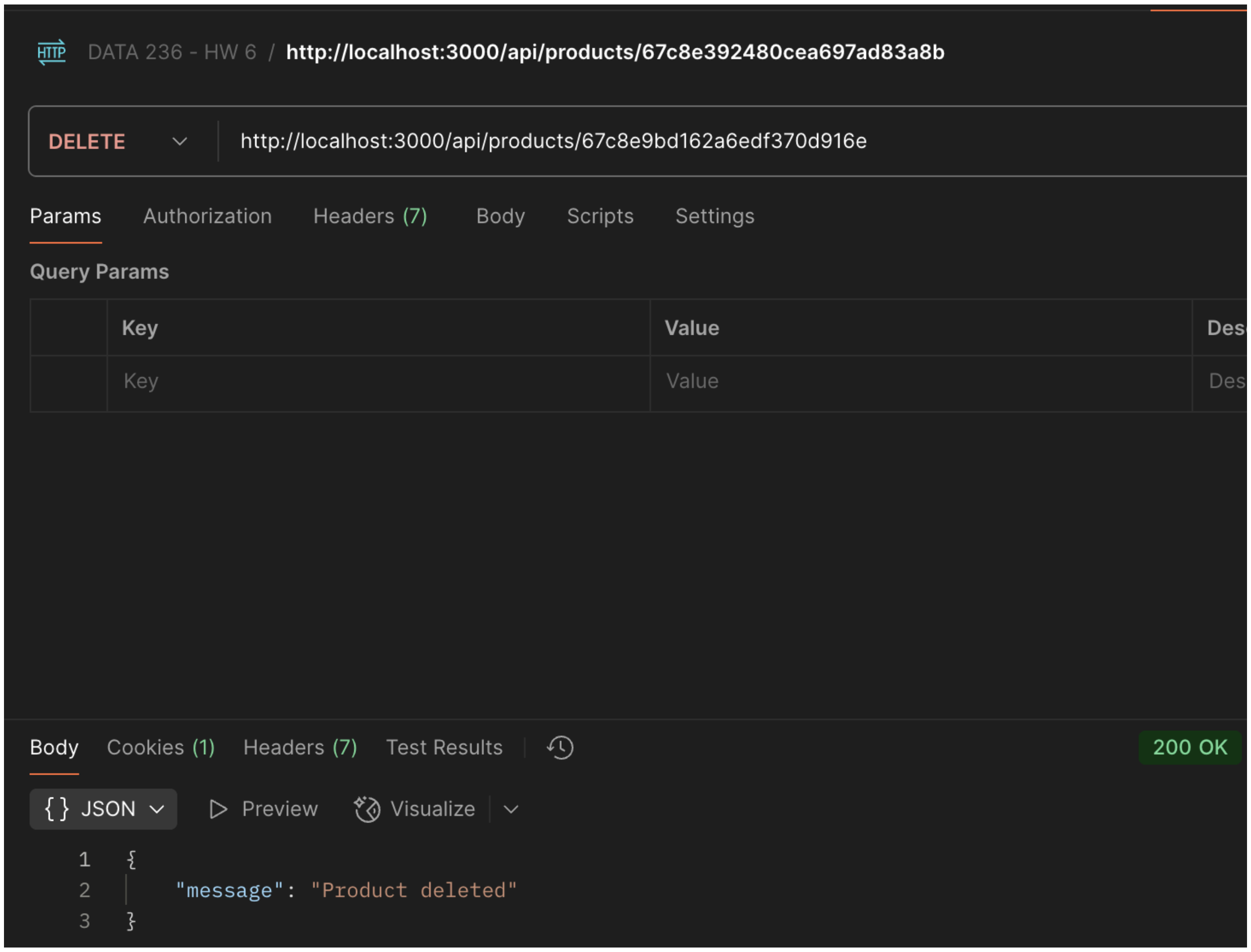
200 OK

```
{
    "_id": "67c8e9bd162a6edf370d916e",
    "name": "Monitor",
    "price": 1099.99,
    "quantity": 15,
    "category": "electronics",
    "createdAt": "2025-03-06T00:18:05.224Z",
    "updatedAt": "2025-03-06T01:20:55.769Z",
    "__v": 0
}
```



```javascript
// 4. PATCH /api/products/:id
app.patch('/api/products/:id', async (req, res) => {
  try {
    const product = await Product.findByIdAndUpdate(req.params.id, req.body, {
      new: true,
      runValidators: true,
    });
    if (!product) {
      return res.status(404).json({ message: 'Product not found' });
    }
    res.json(product);
  } catch (err) {
    if (err.name === 'ValidationError') {
      return res.status(400).json({ message: err.message });
    }
    res.status(500).json({ message: err.message });
  }
});
```

5. DELETE `/api/products/:id` Delete a product



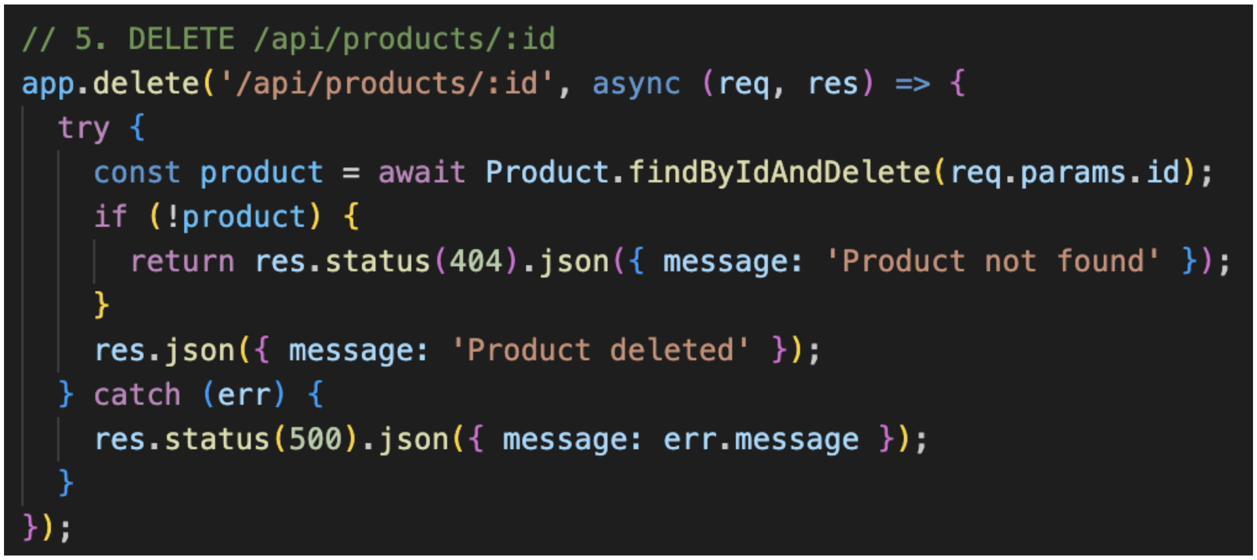
DATA 236 - HW 6 / http://localhost:3000/api/products/67c8e392480cea697ad83a8b

DELETE http://localhost:3000/api/products/67c8e9bd162a6edf370d916e

Query Params

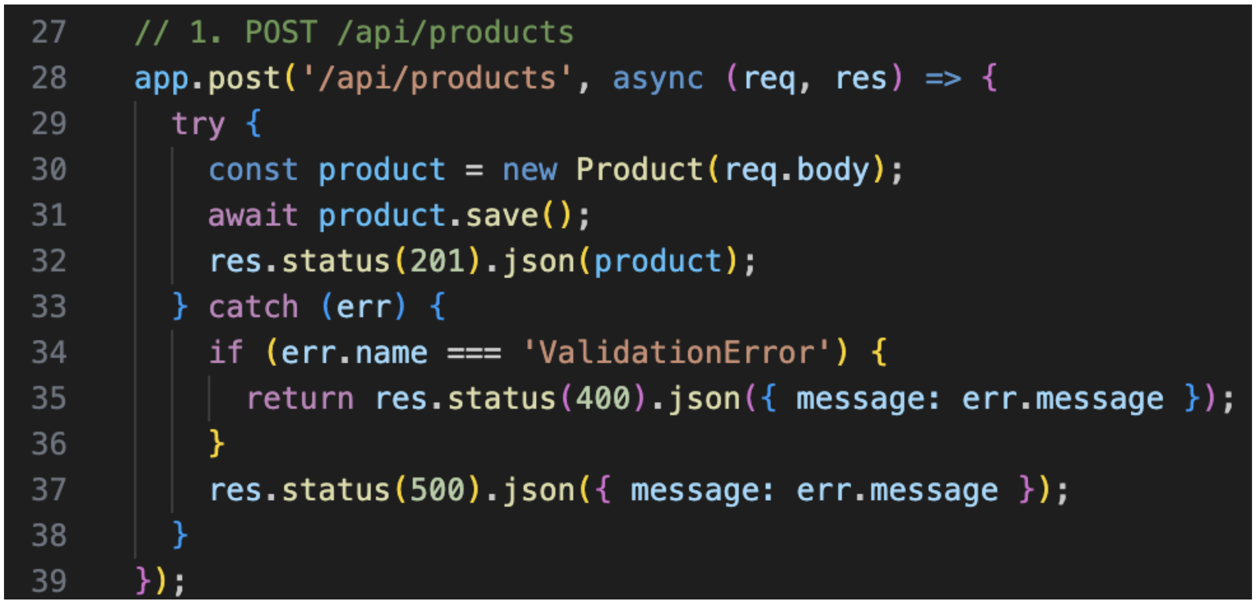
| | Key | Value | Des |
|---|---|---|---|
| | Key | Value | Des |

```
{
    "message": "Product deleted"
}
```



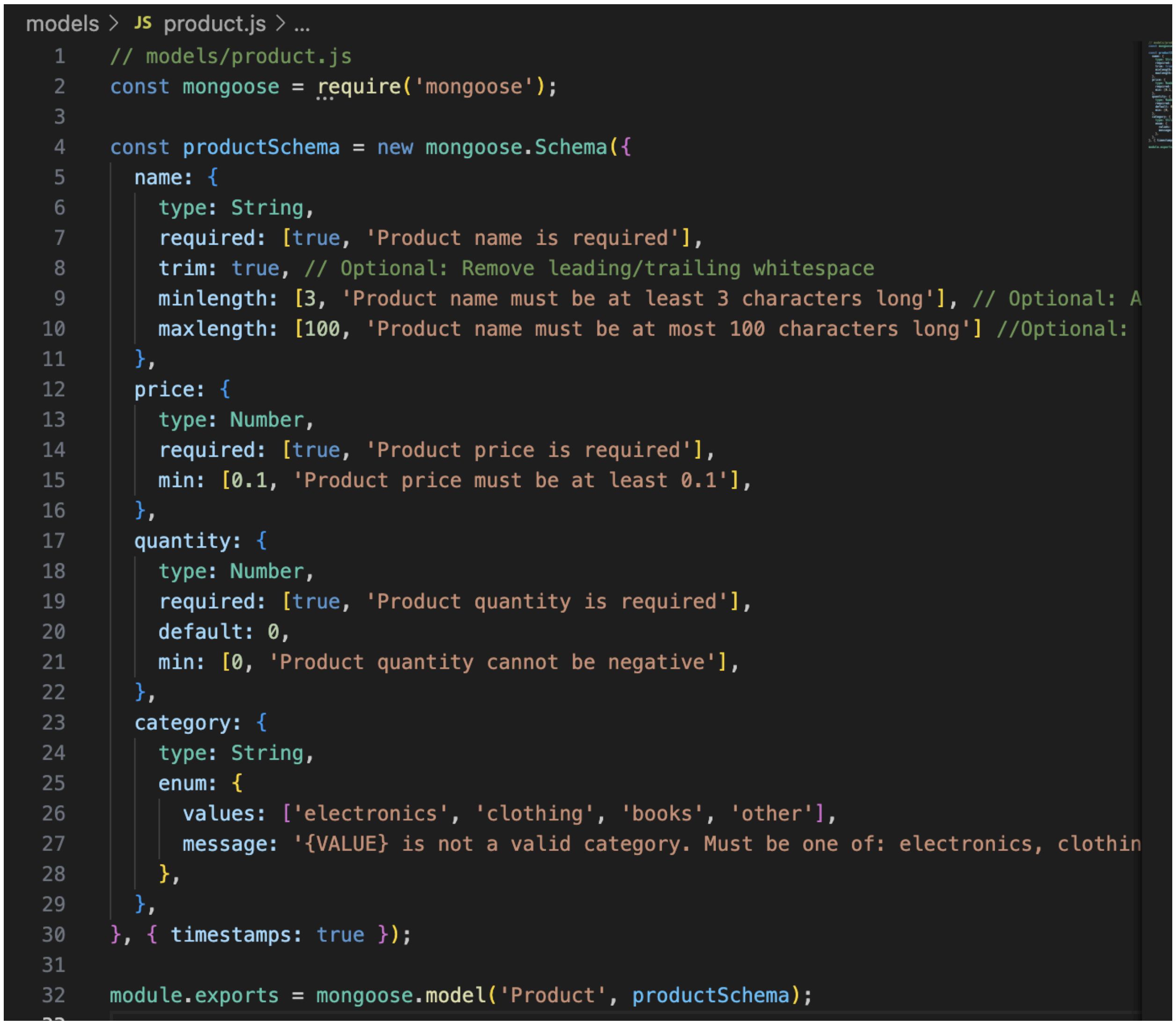
```
// 5. DELETE /api/products/:id
app.delete('/api/products/:id', async (req, res) => {
  try {
    const product = await Product.findByIdAndDelete(req.params.id);
    if (!product) {
      return res.status(404).json({ message: 'Product not found' });
    }
    res.json({ message: 'Product deleted' });
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});
```

**Validation (submit code snippet of how validation and errors are handled)**
Return proper error responses for:
● Missing required fields



```javascript
// 1. POST /api/products
app.post('/api/products', async (req, res) => {
  try {
    const product = new Product(req.body);
    await product.save();
    res.status(201).json(product);
  } catch (err) {
    if (err.name === 'ValidationError') {
      return res.status(400).json({ message: err.message });
    }
    res.status(500).json({ message: err.message });
  }
});
```

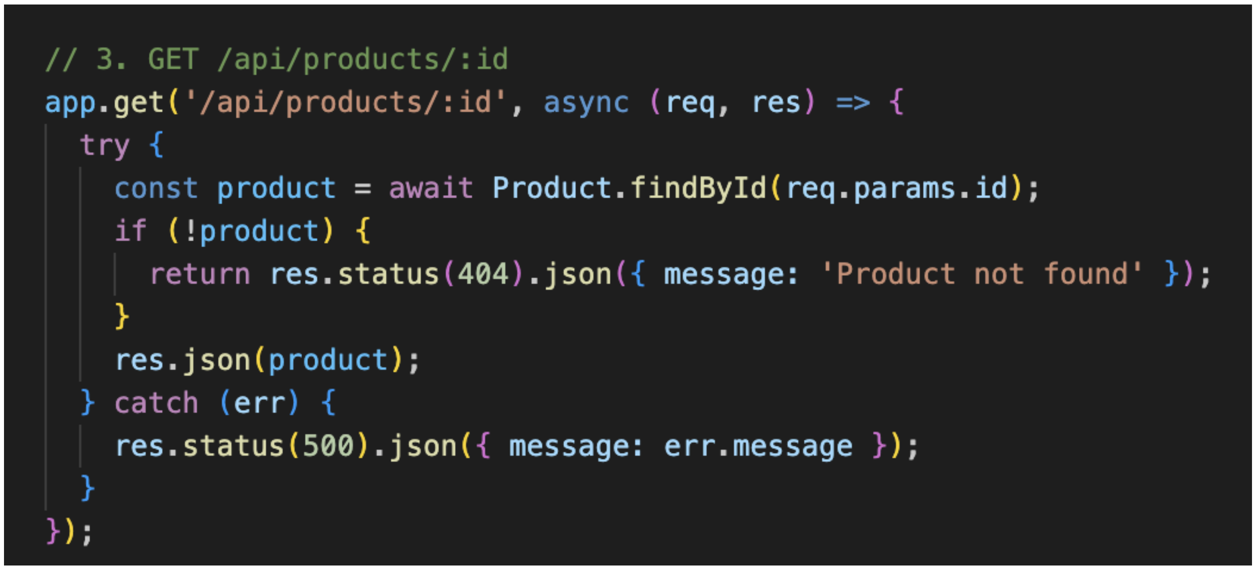
● Invalid data types



models > JS product.js > ...

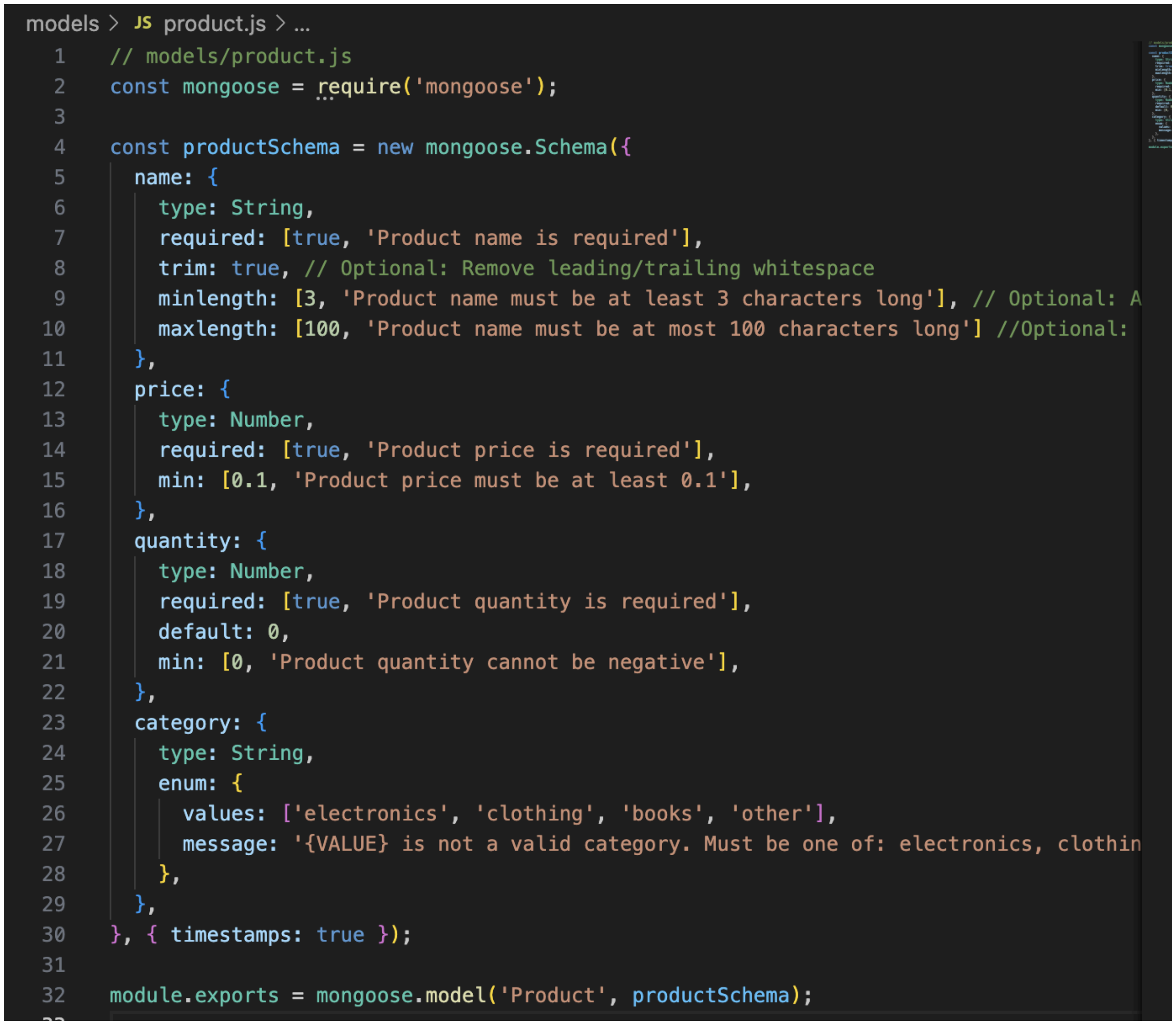
```javascript
// models/product.js
const mongoose = require('mongoose');

const productSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, 'Product name is required'],
    trim: true, // Optional: Remove leading/trailing whitespace
    minlength: [3, 'Product name must be at least 3 characters long'], // Optional: A
    maxlength: [100, 'Product name must be at most 100 characters long'] //Optional:
  },
  price: {
    type: Number,
    required: [true, 'Product price is required'],
    min: [0.1, 'Product price must be at least 0.1'],
  },
  quantity: {
    type: Number,
    required: [true, 'Product quantity is required'],
    default: 0,
    min: [0, 'Product quantity cannot be negative'],
  },
  category: {
    type: String,
    enum: {
      values: ['electronics', 'clothing', 'books', 'other'],
      message: '{VALUE} is not a valid category. Must be one of: electronics, clothin
    },
  },
}, { timestamps: true });

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

- Non-existent product ID



```
// 3. GET /api/products/:id
app.get('/api/products/:id', async (req, res) => {
  try {
    const product = await Product.findById(req.params.id);
    if (!product) {
      return res.status(404).json({ message: 'Product not found' });
    }
    res.json(product);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});
```

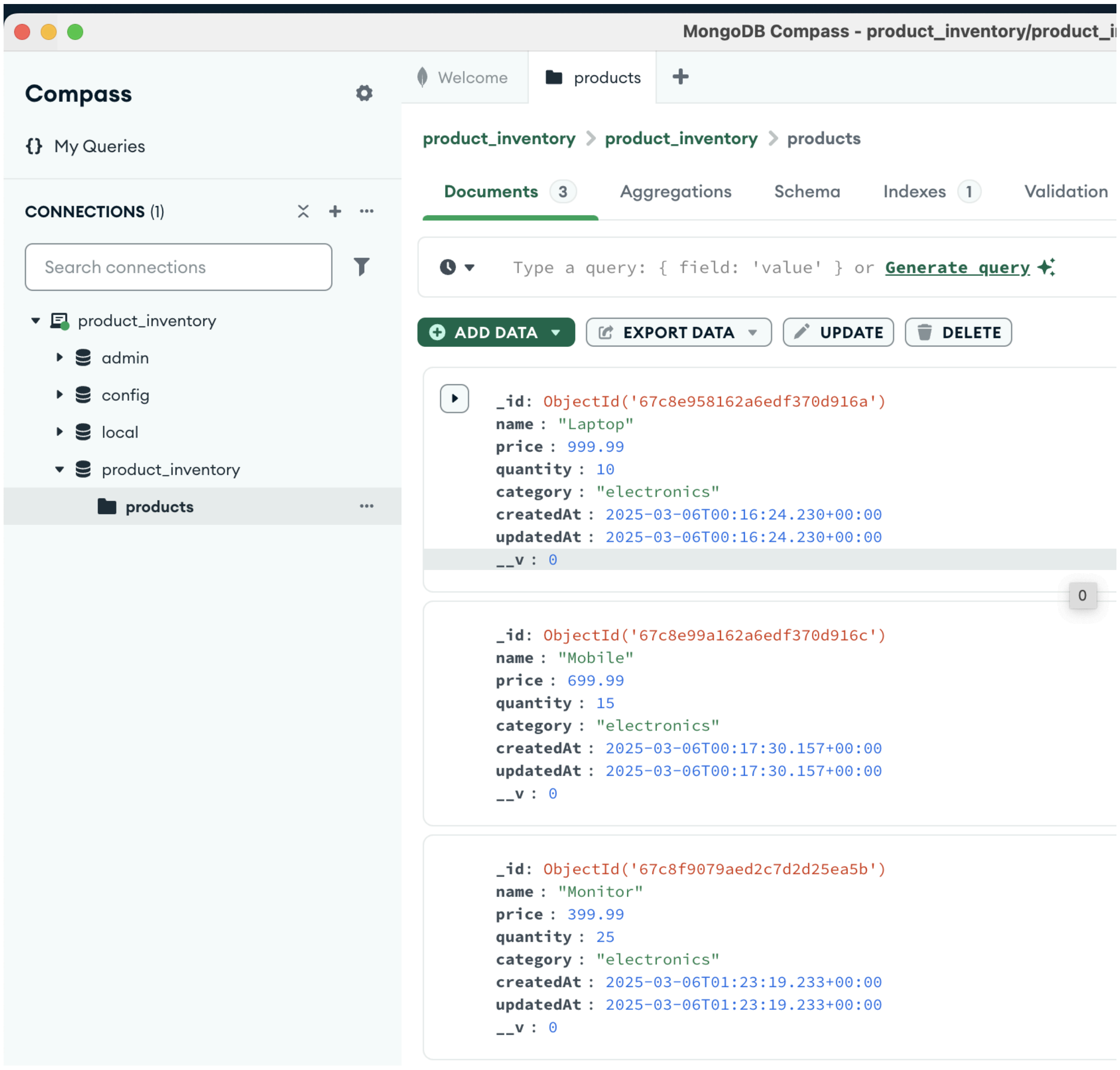
● Use Mongoose validators (required, min, enum, etc.)



models > JS product.js > ...

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

const productSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, 'Product name is required'],
    trim: true, // Optional: Remove leading/trailing whitespace
    minlength: [3, 'Product name must be at least 3 characters long'], // Optional: A
    maxlength: [100, 'Product name must be at most 100 characters long'] //Optional:
  },
  price: {
    type: Number,
    required: [true, 'Product price is required'],
    min: [0.1, 'Product price must be at least 0.1'],
  },
  quantity: {
    type: Number,
    required: [true, 'Product quantity is required'],
    default: 0,
    min: [0, 'Product quantity cannot be negative'],
  },
  category: {
    type: String,
    enum: {
      values: ['electronics', 'clothing', 'books', 'other'],
      message: '{VALUE} is not a valid category. Must be one of: electronics, clothin
    },
  },
}, { timestamps: true });

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

**MongoDB Compass Screenshot**
Include a screenshot of MongoDB Compass showing the `products` collection with at least 3 documents inserted.



```
_id: ObjectId('67c8e958162a6edf370d916a')
name : "Laptop"
price : 999.99
quantity : 10
category : "electronics"
createdAt : 2025-03-06T00:16:24.230+00:00
updatedAt : 2025-03-06T00:16:24.230+00:00
__v : 0

_id: ObjectId('67c8e99a162a6edf370d916c')
name : "Mobile"
price : 699.99
quantity : 15
category : "electronics"
createdAt : 2025-03-06T00:17:30.157+00:00
updatedAt : 2025-03-06T00:17:30.157+00:00
__v : 0

_id: ObjectId('67c8f9079aed2c7d2d25ea5b')
name : "Monitor"
price : 399.99
quantity : 25
category : "electronics"
createdAt : 2025-03-06T01:23:19.233+00:00
updatedAt : 2025-03-06T01:23:19.233+00:00
__v : 0
```

**Grading: 10 points**
Model Creation - 2 points
API calls - 5 points
Error Handling & Validation - 2 points
Compass Screenshot - 1 point

Submit a single PDF named LastName_HW5.pdf, add only the necessary screenshots, and try to keep the file size small.