CRUD - OPERATION

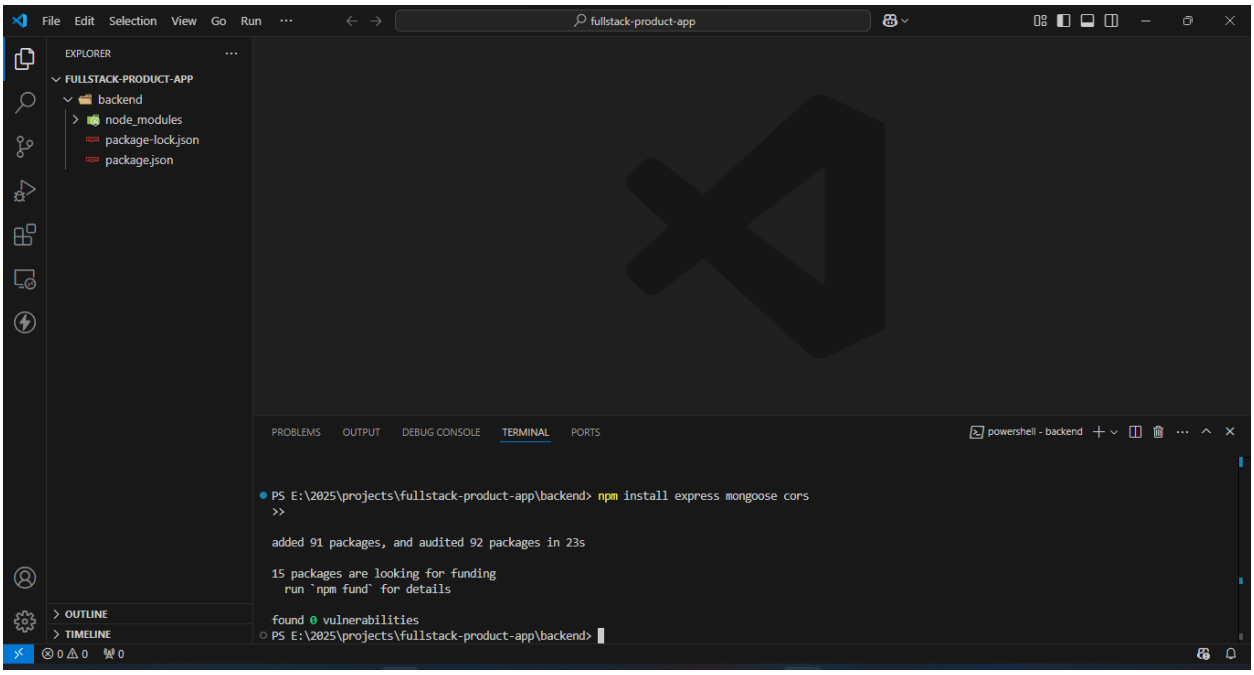
Steps:

1. *Open CMD:*

- *mkdir fullstack-product-app*
- *cd fullstack-product-app*

- *mkdir backend* ***(run the commands)***
  - *cd backend*
  - *npm init -y*
  - *npm install express mongoose cors*
  - *npm install config*
  - *npm install dotenv*

Output:



2. Create a Folders:

- mkdir controllers
  - touch productControllers.js

```
// Code: productControllers.js:
const Product = require('../models/productModels');

// Create a new product
exports.createProduct = async (req, res) => {
  try {
    const product = await Product.create(req.body);
    res.status(201).json({
      success: true,
      data: product
    });
  } catch (error) {
    res.status(400).json({
      success: false,
      error: error.message
    });
  }
};

// Get all products
exports.getAllProducts = async (req, res) => {
  try {
    const products = await Product.find();
    res.status(200).json({
      success: true,
      count: products.length,
      data: products
    });
  } catch (error) {
    res.status(500).json({
      success: false,
      error: error.message
    });
  }
};

// Get single product by ID
exports.getProductById = async (req, res) => {
  try {
```

```
    const product = await Product.findById(req.params.id);
    if (!product) {
      return res.status(404).json({
        success: false,
        error: 'Product not found'
      });
    }
    res.status(200).json({
      success: true,
      data: product
    });
  } catch (error) {
    res.status(500).json({
      success: false,
      error: error.message
    });
  }
};

// Update product by ID
exports.updateProduct = 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({
        success: false,
        error: 'Product not found'
      });
    }
    res.status(200).json({
      success: true,
      data: product
    });
  } catch (error) {
    res.status(400).json({
      success: false,
```

```
      error: error.message
    });
  }
};

// Delete product by ID
exports.deleteProduct = async (req, res) => {
  try {
    const product = await Product.findByIdAndDelete(req.params.id);
    if (!product) {
      return res.status(404).json({
        success: false,
        error: 'Product not found'
      });
    }
    res.status(200).json({
      success: true,
      data: {}
    });
  } catch (error) {
    res.status(500).json({
      success: false,
      error: error.message
    });
  }
};
```

3. Create a Folders:

- mkdir models
  - touch productModels.js

```
// Code: productModels.js:
const mongoose = require('mongoose');

const productSchema = new mongoose.Schema({
 productName: {
  type: String,
  required: [true, 'Product name is required'],
  trim: true
 },
 description: {
  type: String,
  required: [true, 'Product description is required'],
  trim: true
 },
 category: {
  type: String,
  required: [true, 'Product category is required'],
  enum: ['Mobile', 'Laptop', 'Tablet'],
  trim: true
 },
 quantity: {
  type: Number,
  required: [true, 'Product quantity is required'],
  min: [0, 'Quantity cannot be negative']
 },
 price: {
  type: Number,
  required: [true, 'Product price is required'],
  min: [0, 'Price cannot be negative']
 }
}, {
 timestamps: true
});

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

4. Create a Folders:

- mkdir routes
  - touch productRoutes.js

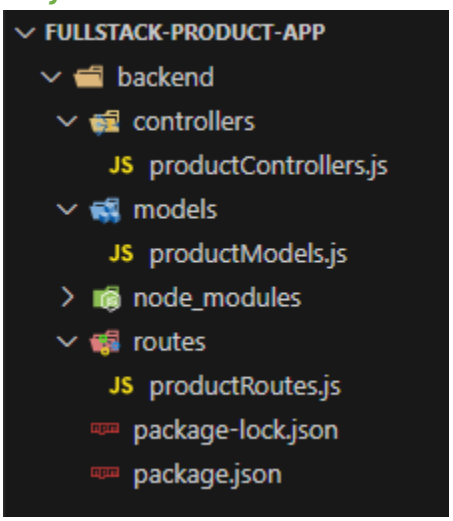
```
// Code: productRoutes.js:
        const express = require('express');
        const router = express.Router();
        const {
         createProduct,
         getAllProducts,
         getProductById,
         updateProduct,
         deleteProduct
        } = require('../controllers/productControllers');

        router.route('/')
         .get(getAllProducts)
         .post(createProduct);

        router.route('/:id')
         .get(getProductById)
         .put(updateProduct)
         .delete(deleteProduct);

module.exports = router;
```

Project Structure:



5. **Create .env file:**

```
touch .env
```

```
// .env file:
PORT=3000
MONGODB_URI=mongodb://localhost:27017/productDB
```

6. **Create Three Files in backend:**

- config.js
- index.js
- .gitignore

7. **Codes:**

```
//config.js

        require('dotenv').config();

        module.exports = {
         port: process.env.PORT || 3000,
         mongoUri: process.env.MONGODB_URI ||
        'mongodb://localhost:27017/productDB'
};
```

```
//index.js

// run `node index.js`  in the terminal
console.log(` Hello Node.js v${process.versions.node}!` );
```

```
//.gitignore

.node_modules
.bolt
```

8. Create server.js file:

```
//server.js
        const express = require('express');
        const mongoose = require('mongoose');
        const cors = require('cors');
        const config = require('./config');
        const productRoutes = require('./routes/productRoutes');

        const app = express();

        // Middleware
        app.use(cors());
        app.use(express.json());
        // Connect to MongoDB
        mongoose.connect('mongodb://127.0.0.1:27017/productDB', {
        useNewUrlParser: true,
          useUnifiedTopology: true,
          serverSelectionTimeoutMS: 15000
        })
        .then(() => console.log('MongoDB connected successfully'))
        .catch(err => console.log('MongoDB connection error:', err));

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

        // Basic route
        app.get('/', (req, res) => {
          res.json({ message: 'Welcome to Product Management API' });
        });

        // Error handling middleware
        app.use((err, req, res, next) => {
          console.error(err.stack);
          res.status(500).json({
            success: false,
            error: 'Something went wrong!'
          });
        });
        // Start server
        app.listen(config.port, () => {
          console.log(` Server is running on port ${config.port}` );
});
```

9. *Open MongoDB Compass:*

- ***Connect*** *the localhost 127.0.0.1:27017*

10. *Open Terminal Run Command:*

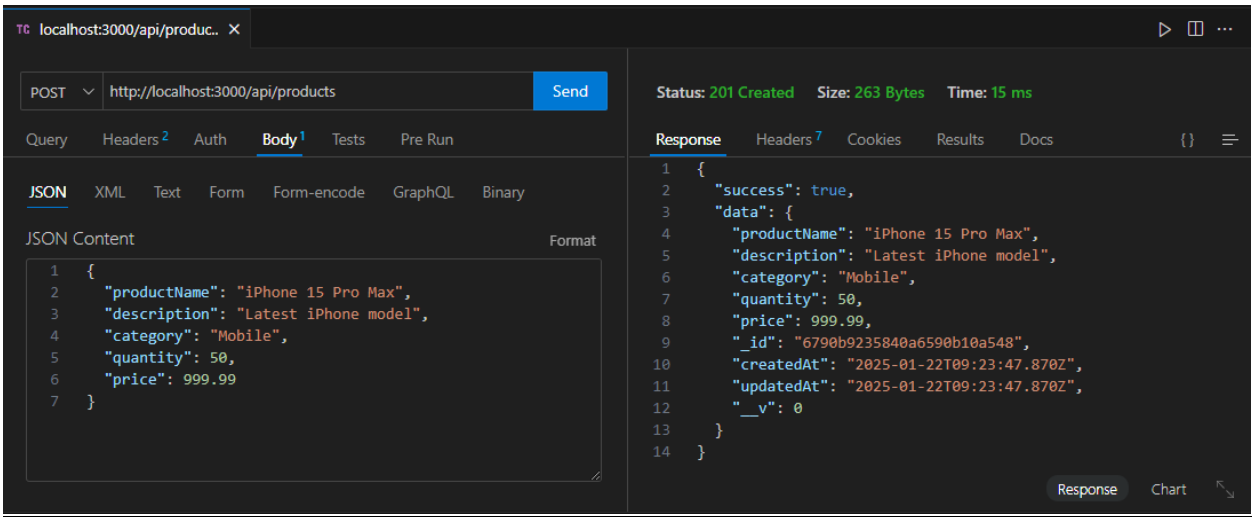
node server.js

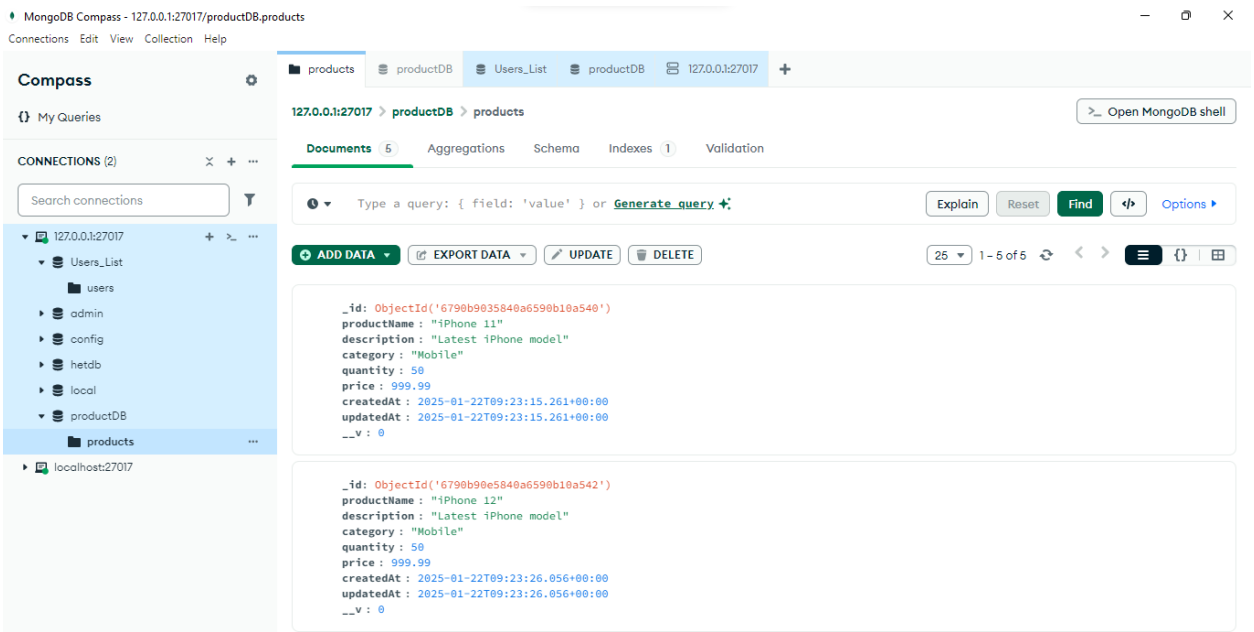
```
PS E:\2025\projects\fullstack-product-app\backend> node server.js
(node:2828) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: us
rsion
(Use `node --trace-warnings ...` to show where the warning was created)
(node:2828) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option:
jor version
Server is running on port 3000
MongoDB connected successfully
```

11. Open ThunderClient:

✓ **Create new product:**

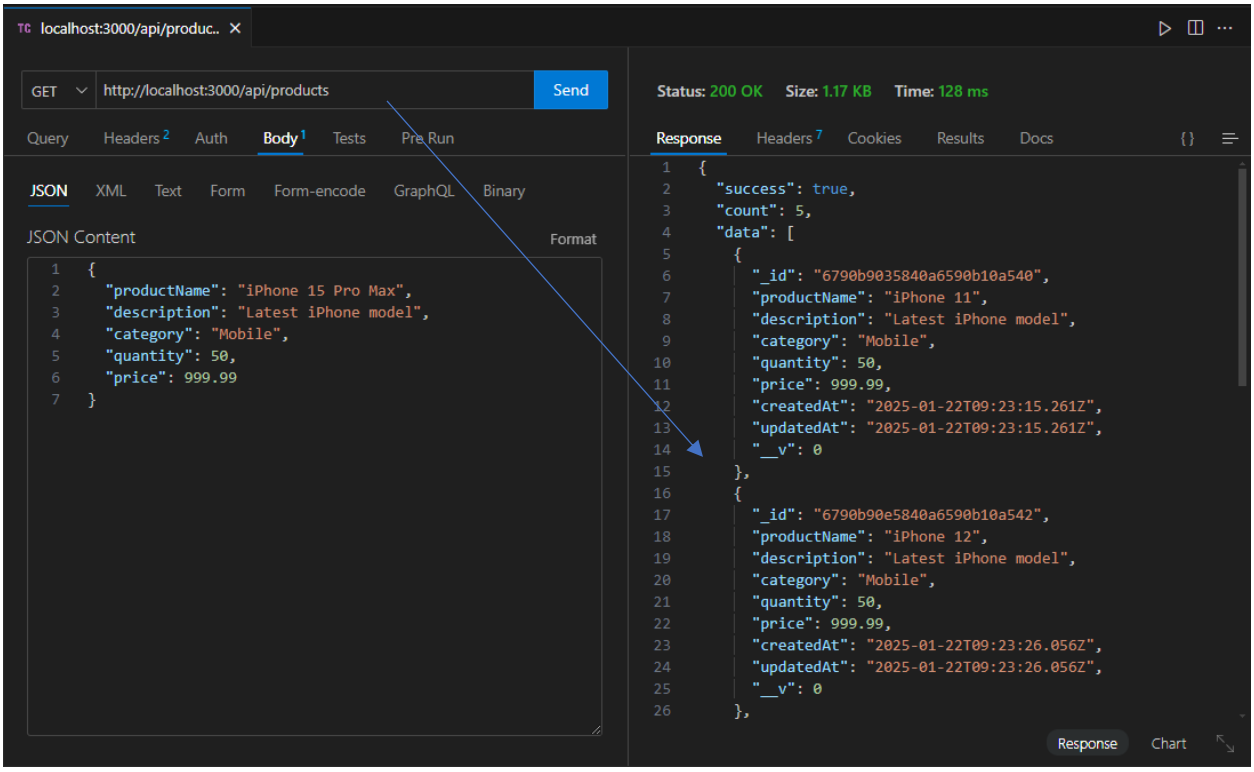
POST	http://localhost:3000/api/products

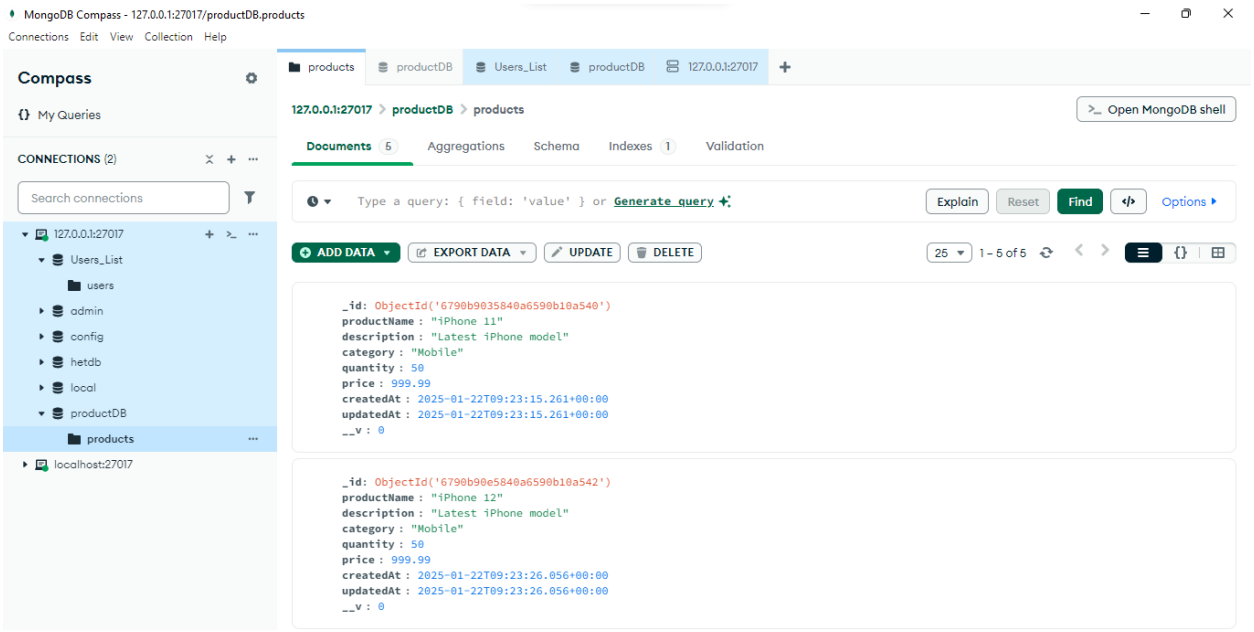




✓ ***Get all products:***

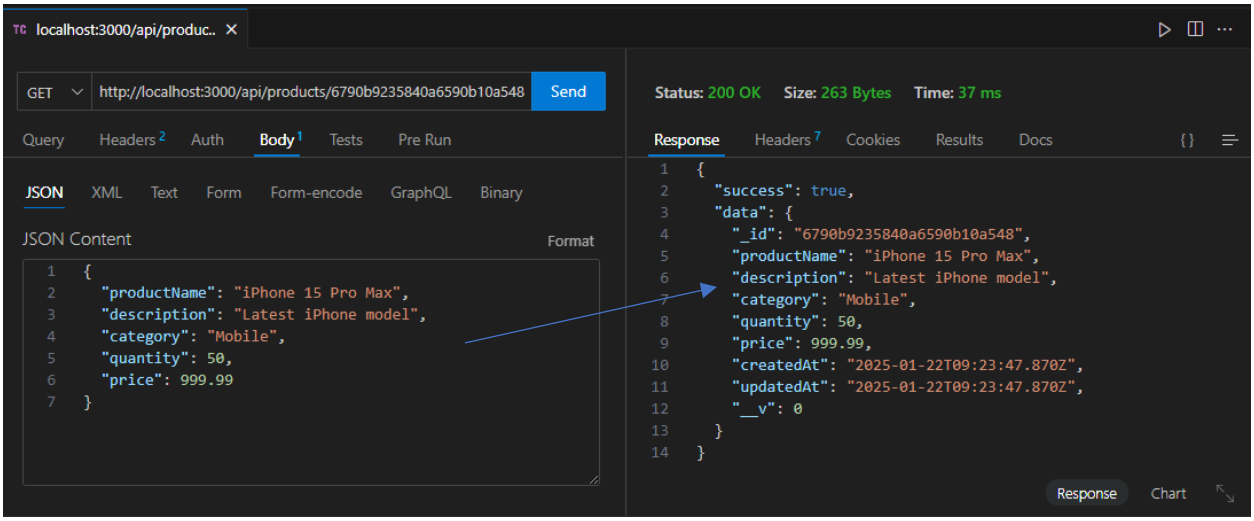
GET	***http://localhost:3000/api/products***

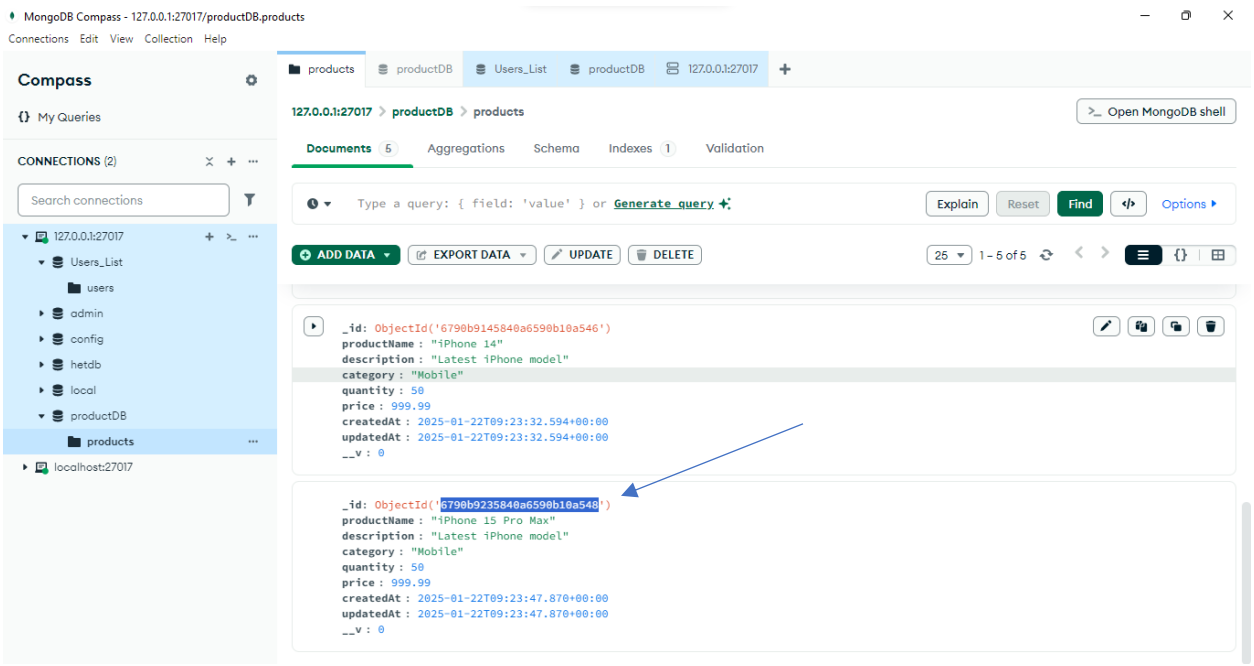




✓ Get product by ID:

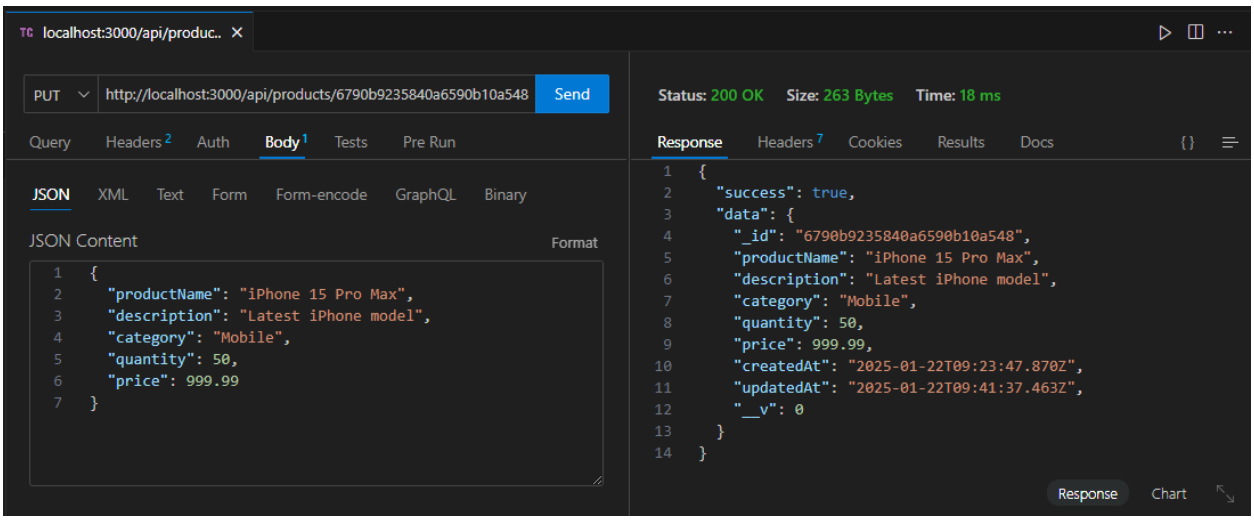
GET	http://localhost:3000/api/products/6790b9235840a6590b10a548



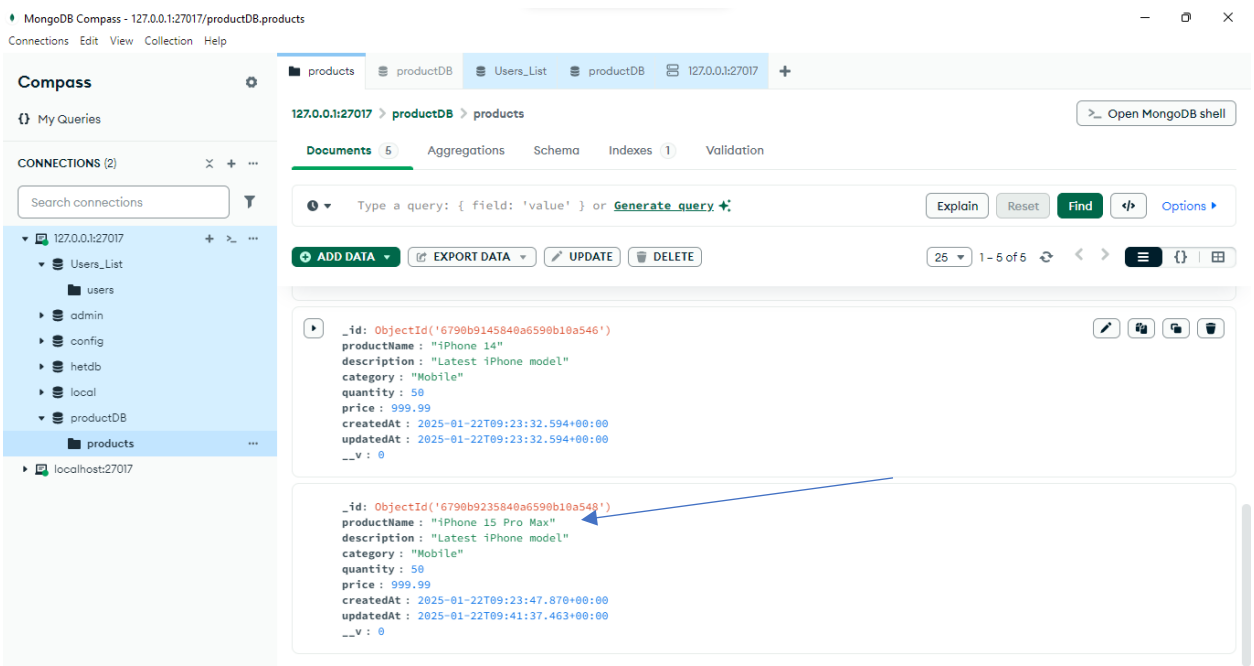


✓ ***Update product:***

PUT	***http://localhost:3000/api/products/6790b9235840a6590b10a548***



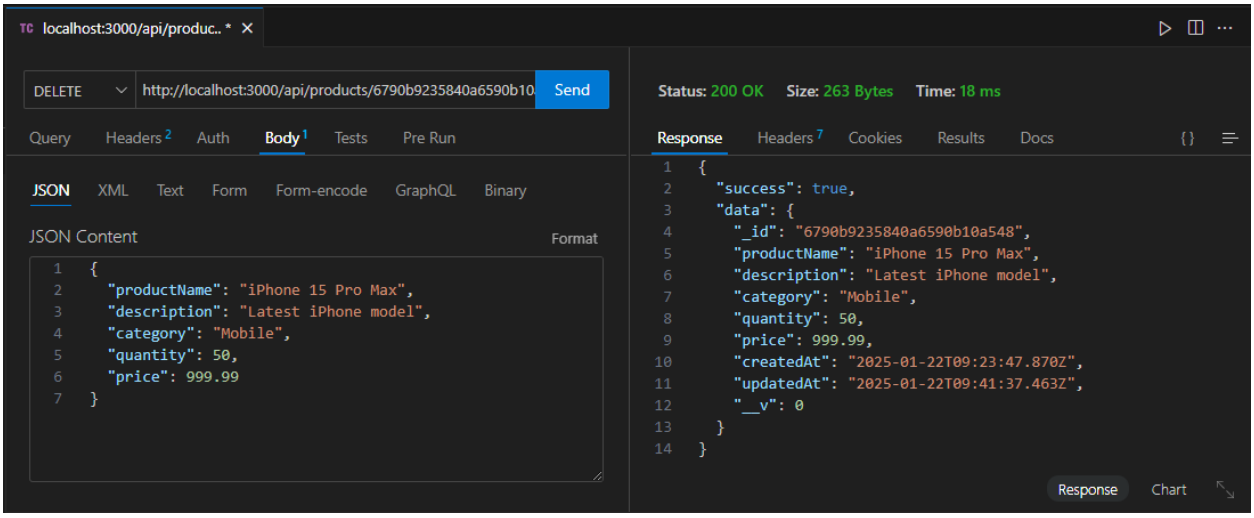
After:



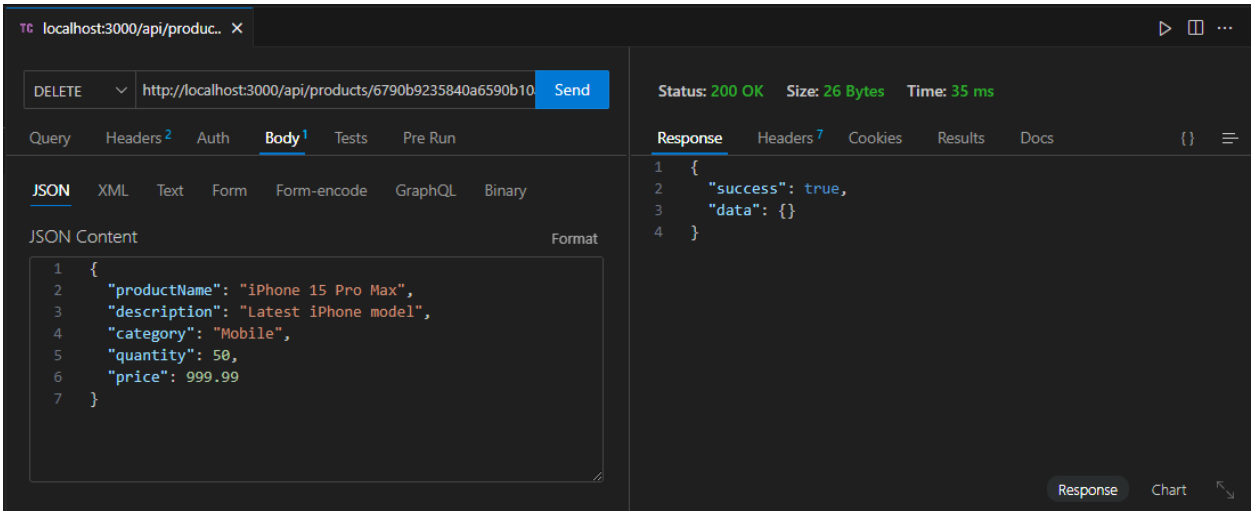
✓ Delete product:

DELETE	http://localhost:3000/api/products/6790b9235840a6590b10a548

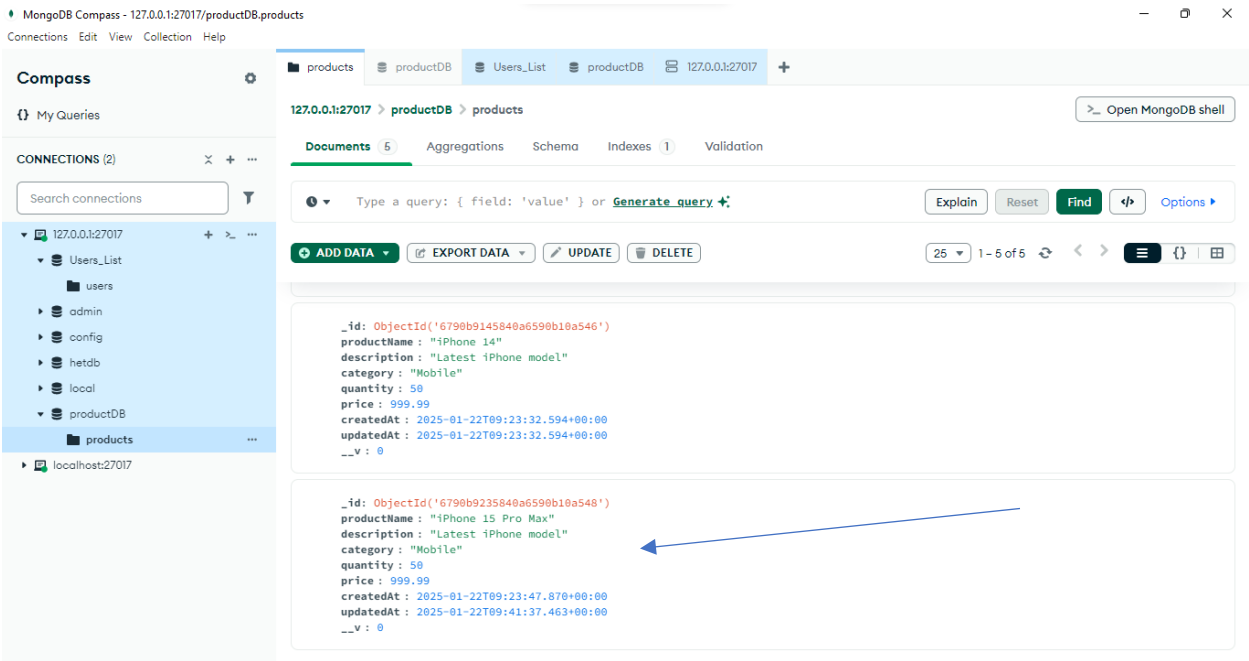
Before Delete:



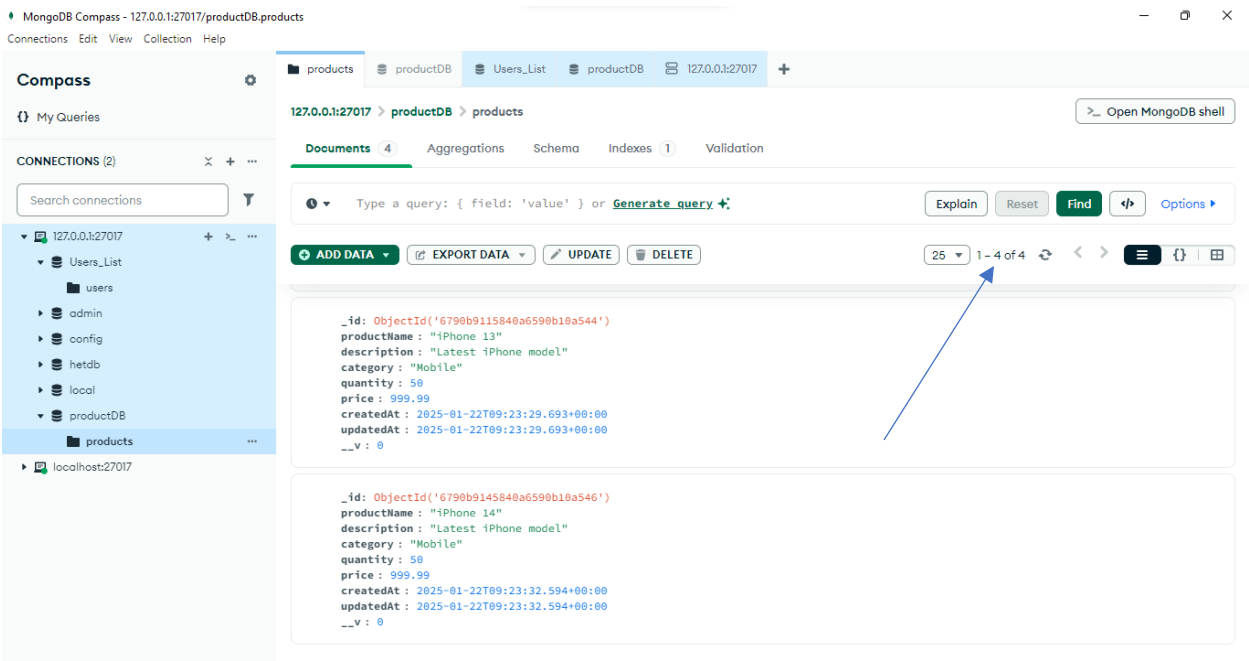
After Delete:



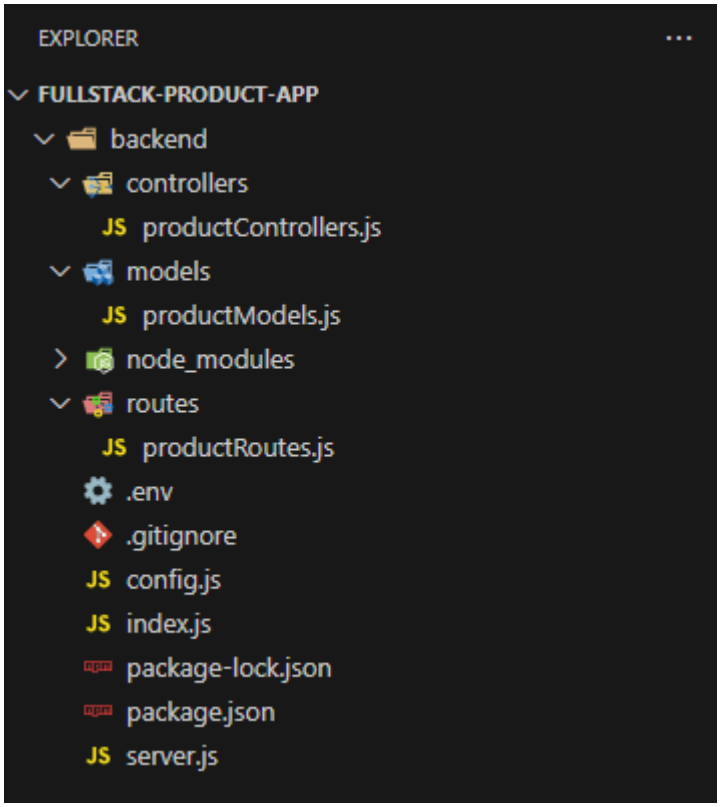
Before:



After:



Project Structure:



Author:

- *HET DABHI*

"Stay connected with me for more tech updates and projects!"

📘 ***GitHub:*** ***hetdabhi***

🔗 ***LinkedIn:*** ***hetdabhi***