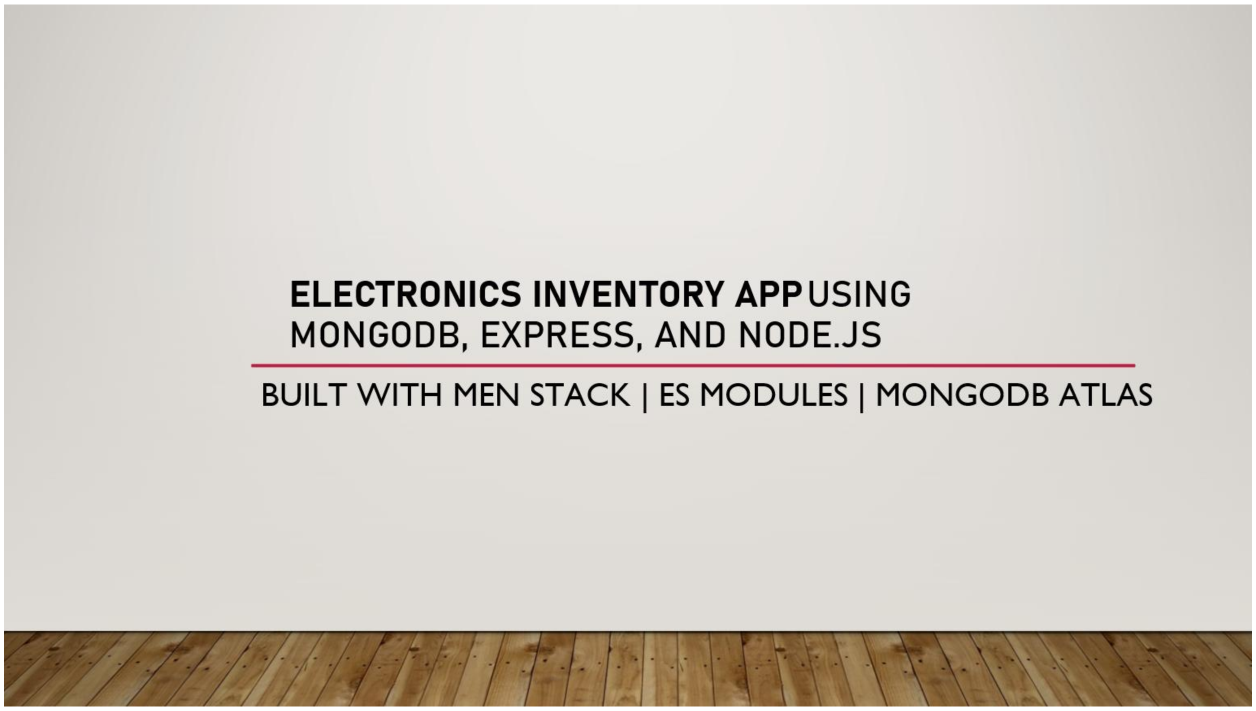


ELECTRONICS INVENTORY APP USING MONGODB, EXPRESS, AND NODE.JS

BUILT WITH MEN STACK | ES MODULES | MONGODB ATLAS

Tools and Technologies Used

Node.js – JavaScript runtime for backend

Express – Web framework for building REST APIs

MongoDB Atlas – Cloud NoSQL database

Mongoose – ODM for MongoDB

Dotenv – Secure environment variable management

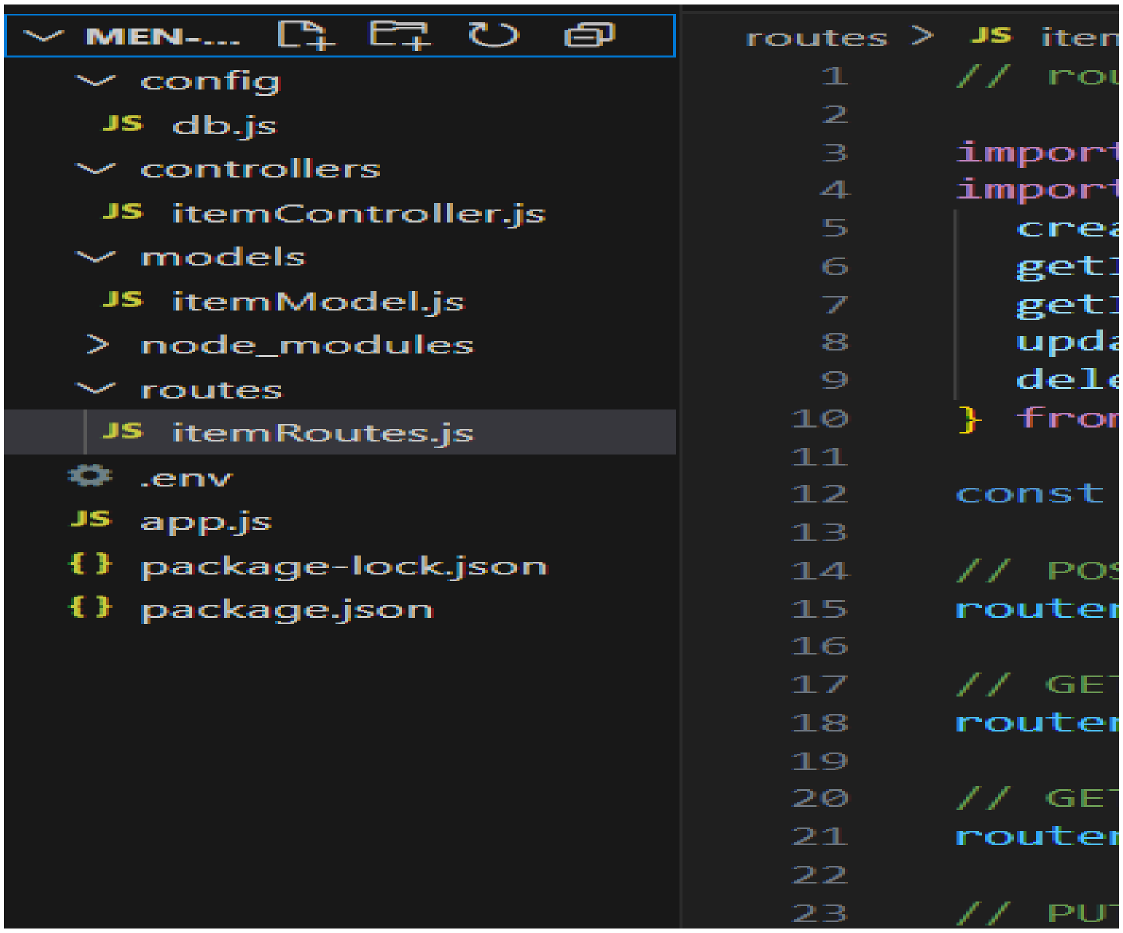
Nodemon – Dev tool for auto-reloading server

Dependencies

Installing the required packages:

- npm init -y
- npm install express dotenv mongoose
- npm install nodemon --save-dev

Project Structure



```
MEN-...
  config
    db.js
  controllers
    itemController.js
  models
    itemModel.js
  node_modules
  routes
    itemRoutes.js
  .env
  app.js
  package-lock.json
  package.json
```

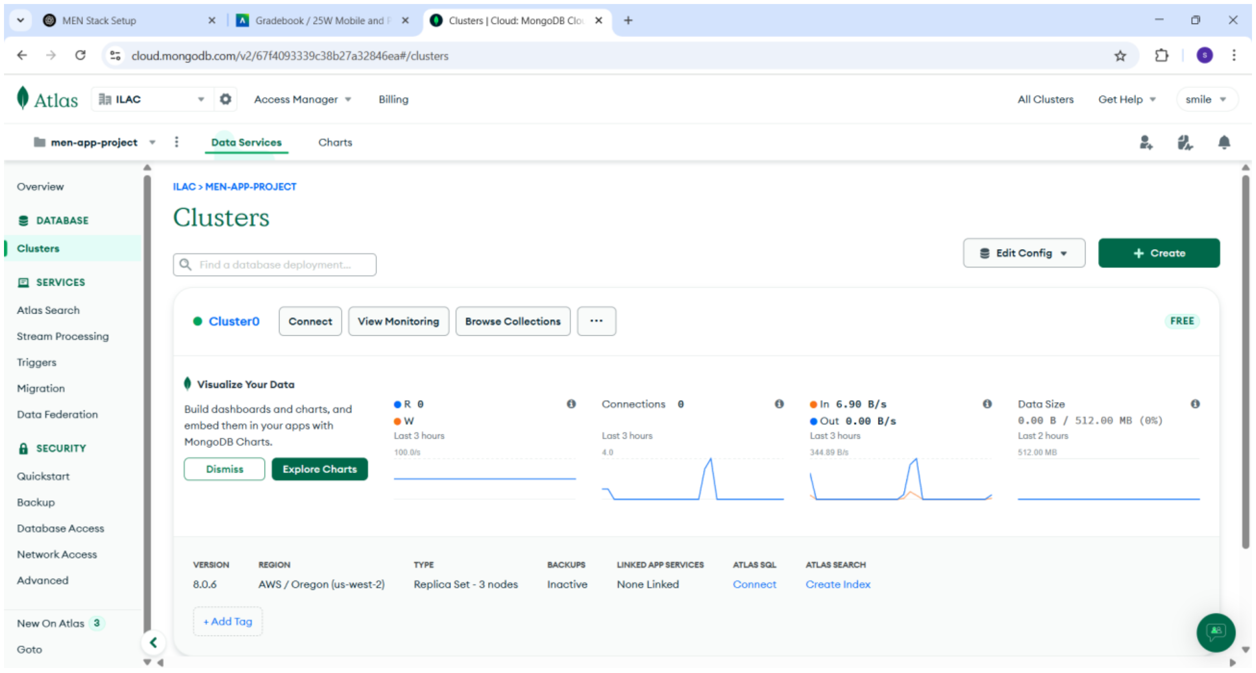
```
routes > JS iten
// rou

import
import
  crea
  getI
  getI
  upda
  dele
} fron

const

// POS
router

// GET
router

// GET
router

// PUT
```

```
Men-app/
├──── config/        # DB config
├──── controllers/   # Route logic
├──── models/        # Mongoose schema
├──── routes/        #Contains API route definitions
├──── app.js         # Main entry
├──── .env           # Mongo URI stored here
```

Creating MongoDB Atlas setup

- Go to https://www.mongodb.com/cloud/atlas and log in.
- Create a free cluster (select latest version).
- Create a new database and a collection (e.g., items).
- Get connection string and place it in the .env file:

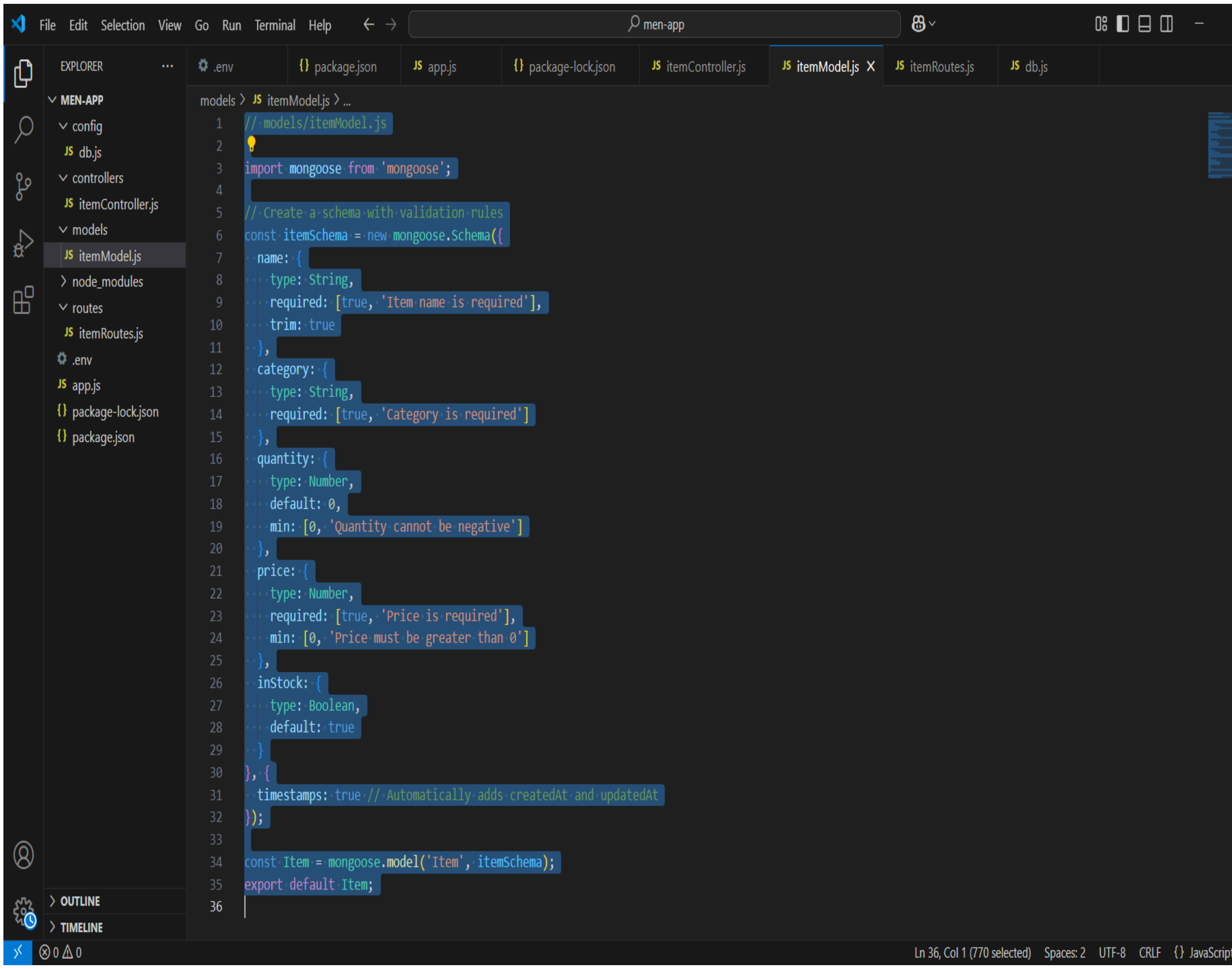
  MONGO_URI=mongodb+srv://myuser27:mypassword27@cluster0.yz3rag3.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0



VERSION	REGION	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SQL	ATLAS SEARCH
8.0.6	AWS / Oregon (us-west-2)	Replica Set - 3 nodes	Inactive	None Linked	Connect	Create Index

Creating Mongoose Schema

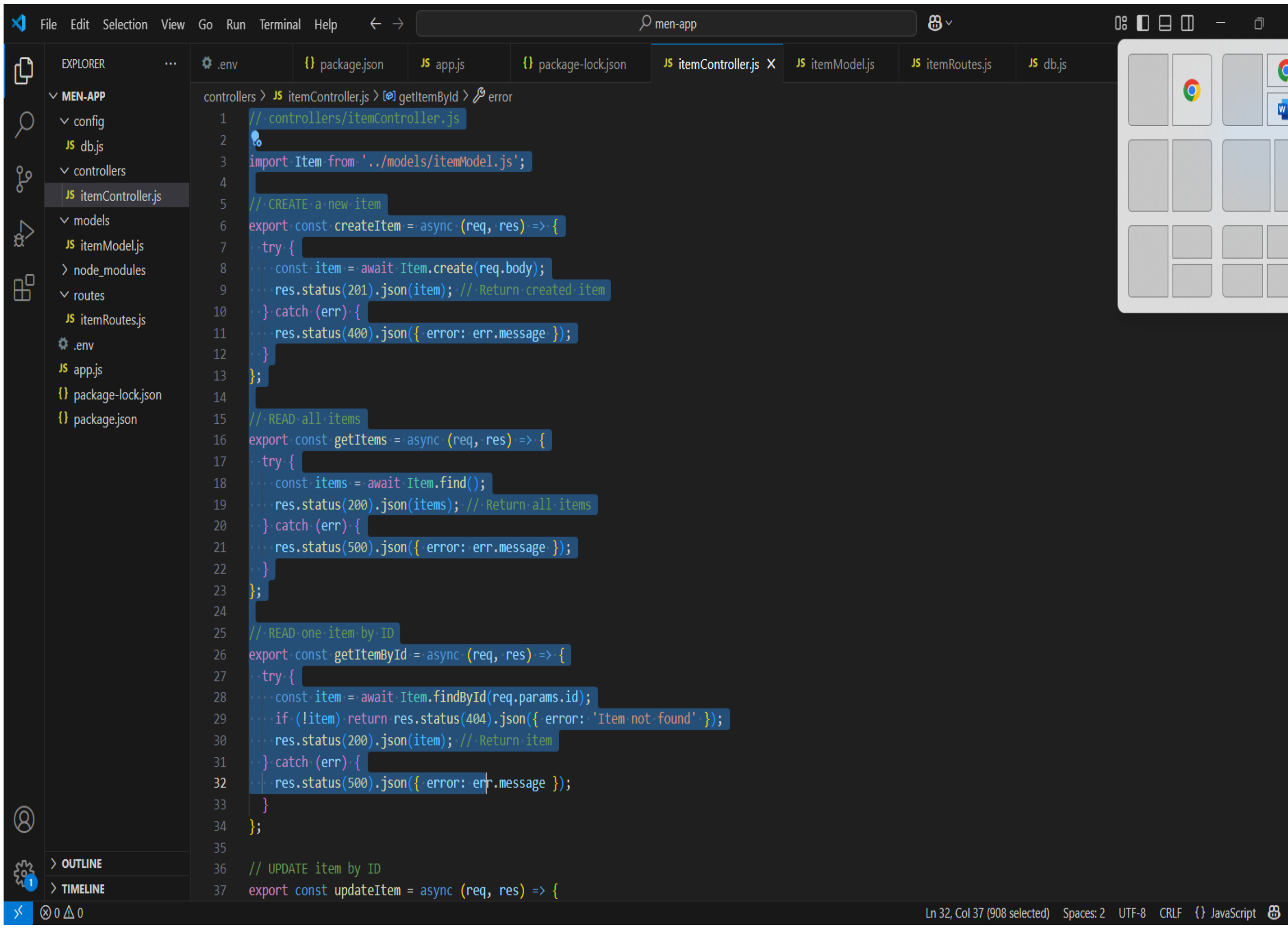
Inside models/itemModel.js:



```
// models/itemModel.js

import mongoose from 'mongoose';

// Create a schema with validation rules
const itemSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, 'Item name is required'],
    trim: true
  },
  category: {
    type: String,
    required: [true, 'Category is required']
  },
  quantity: {
    type: Number,
    default: 0,
    min: [0, 'Quantity cannot be negative']
  },
  price: {
    type: Number,
    required: [true, 'Price is required'],
    min: [0, 'Price must be greater than 0']
  },
  inStock: {
    type: Boolean,
    default: true
  }
}, {
  timestamps: true // Automatically adds createdAt and updatedAt
});

const Item = mongoose.model('Item', itemSchema);
export default Item;
```

Creating Controller Logic

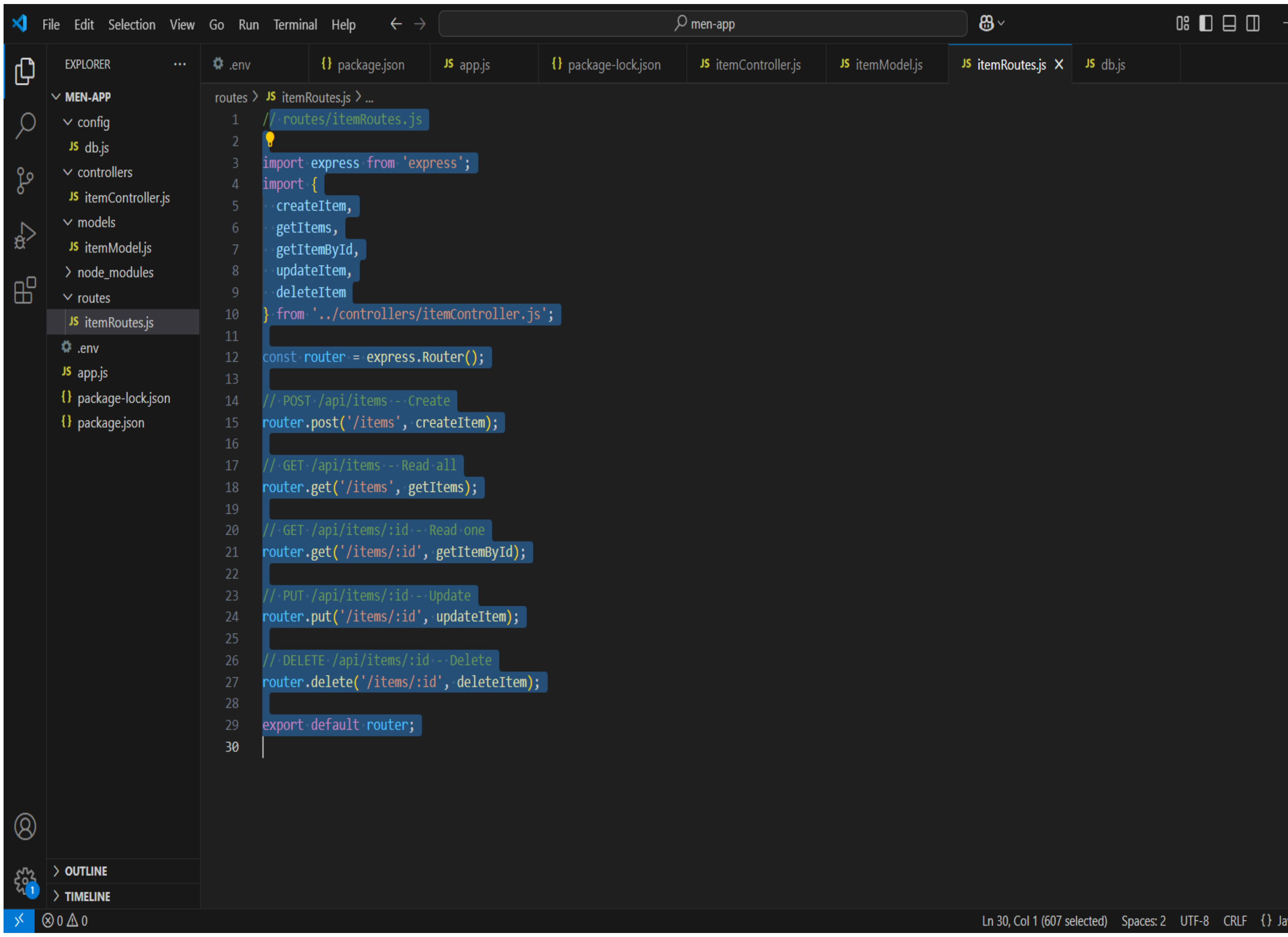
Inside controllers/itemController.js:



```javascript
// controllers/itemController.js

import Item from '../models/itemModel.js';

// CREATE a new item
export const createItem = async (req, res) => {
  try {
    const item = await Item.create(req.body);
    res.status(201).json(item); // Return created item
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
};

// READ all items
export const getItems = async (req, res) => {
  try {
    const items = await Item.find();
    res.status(200).json(items); // Return all items
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// READ one item by ID
export const getItemById = async (req, res) => {
  try {
    const item = await Item.findById(req.params.id);
    if (!item) return res.status(404).json({ error: 'Item not found' });
    res.status(200).json(item); // Return item
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// UPDATE item by ID
export const updateItem = async (req, res) => {
```

Define API Routes

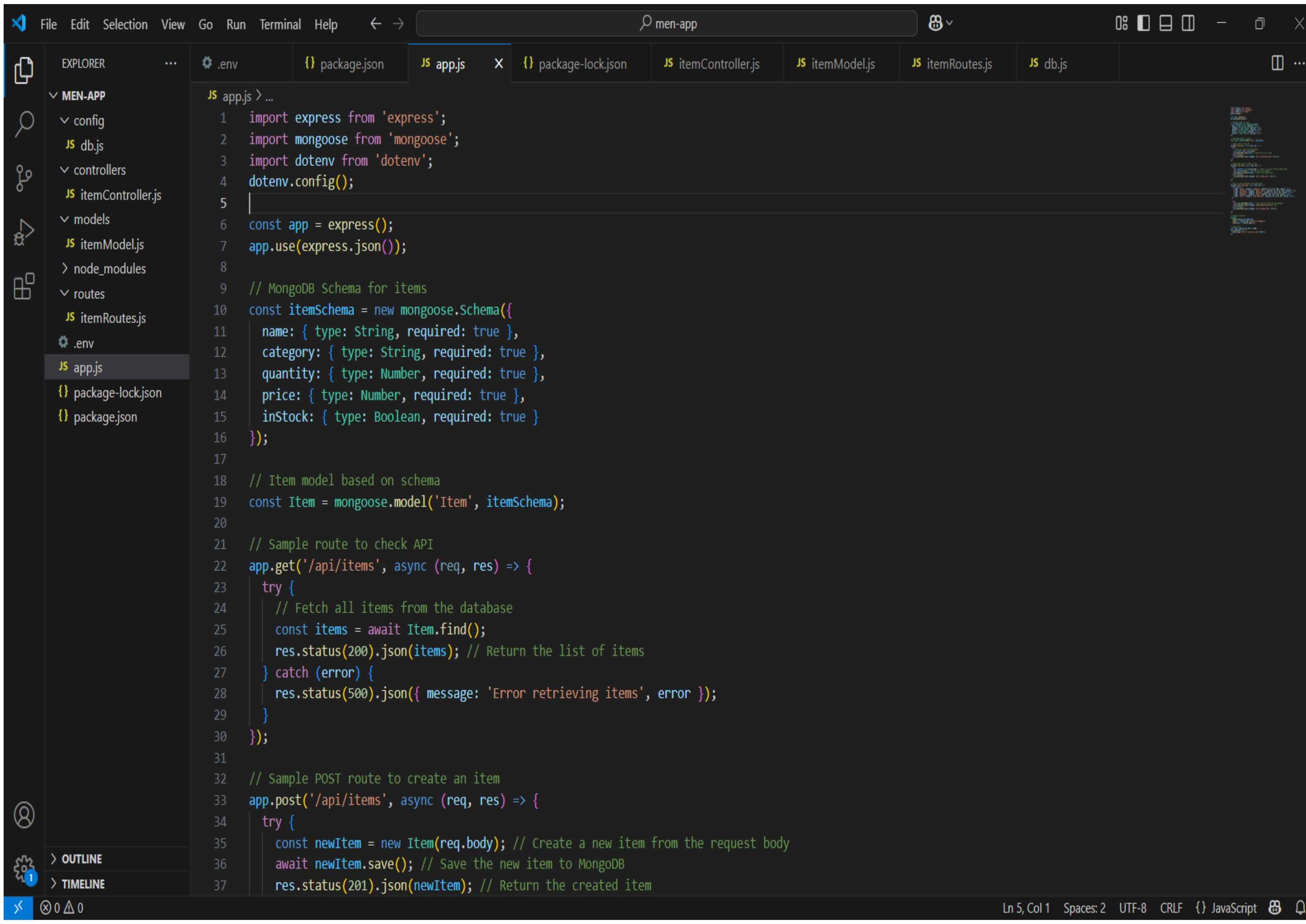
Inside routes/itemRoutes.js:



```
// routes/itemRoutes.js

import express from 'express';
import {
  createItem,
  getItems,
  getItemById,
  updateItem,
  deleteItem
} from '../controllers/itemController.js';

const router = express.Router();

// POST /api/items - Create
router.post('/items', createItem);

// GET /api/items - Read all
router.get('/items', getItems);

// GET /api/items/:id - Read one
router.get('/items/:id', getItemById);

// PUT /api/items/:id - Update
router.put('/items/:id', updateItem);

// DELETE /api/items/:id - Delete
router.delete('/items/:id', deleteItem);

export default router;
```

Set Up Main App File

In app.js:

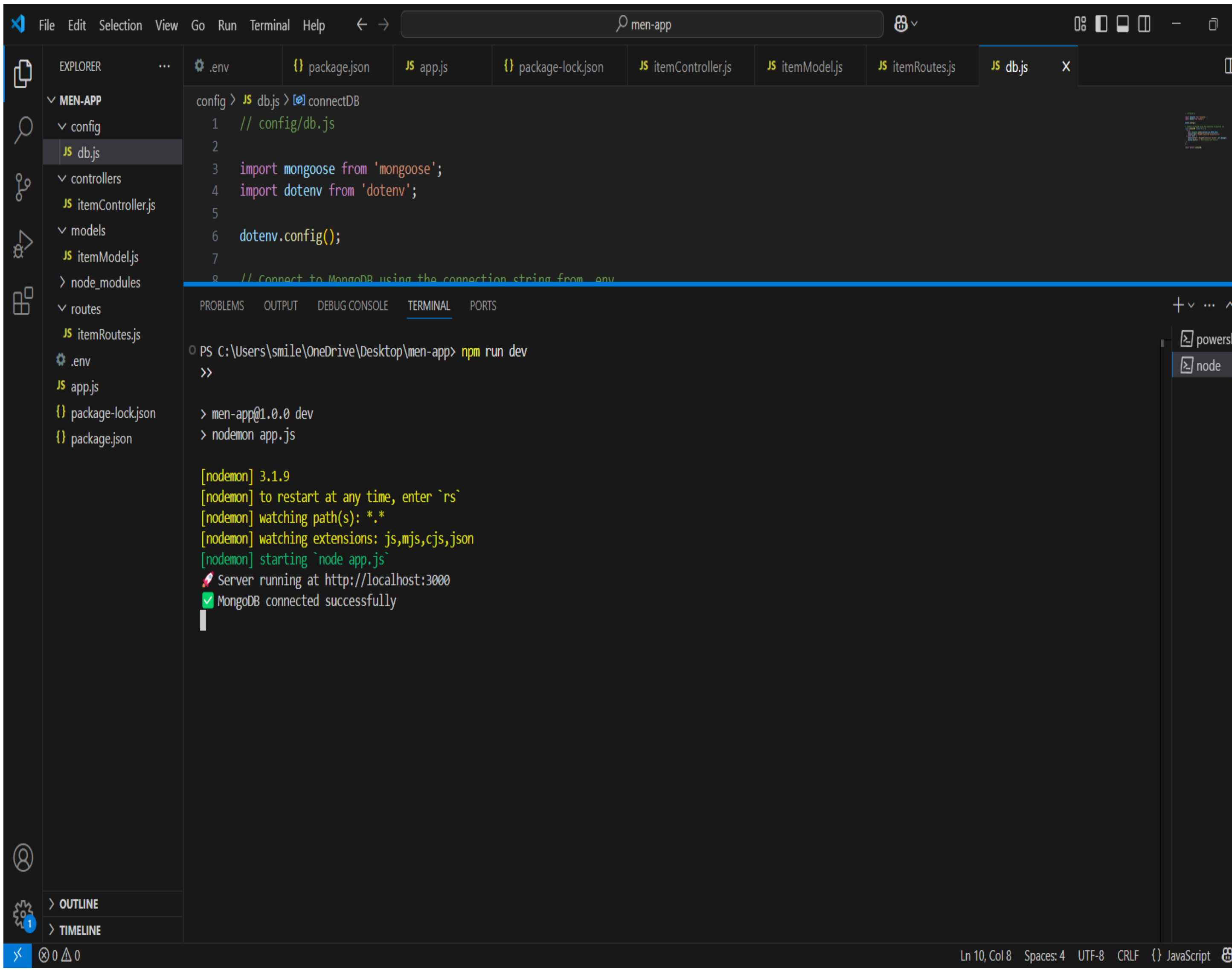


```javascript
import express from 'express';
import mongoose from 'mongoose';
import dotenv from 'dotenv';
dotenv.config();

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

// MongoDB Schema for items
const itemSchema = new mongoose.Schema({
  name: { type: String, required: true },
  category: { type: String, required: true },
  quantity: { type: Number, required: true },
  price: { type: Number, required: true },
  inStock: { type: Boolean, required: true }
});

// Item model based on schema
const Item = mongoose.model('Item', itemSchema);

// Sample route to check API
app.get('/api/items', async (req, res) => {
  try {
    // Fetch all items from the database
    const items = await Item.find();
    res.status(200).json(items); // Return the list of items
  } catch (error) {
    res.status(500).json({ message: 'Error retrieving items', error });
  }
});

// Sample POST route to create an item
app.post('/api/items', async (req, res) => {
  try {
    const newItem = new Item(req.body); // Create a new item from the request body
    await newItem.save(); // Save the new item to MongoDB
    res.status(201).json(newItem); // Return the created item
```

Test with Postman

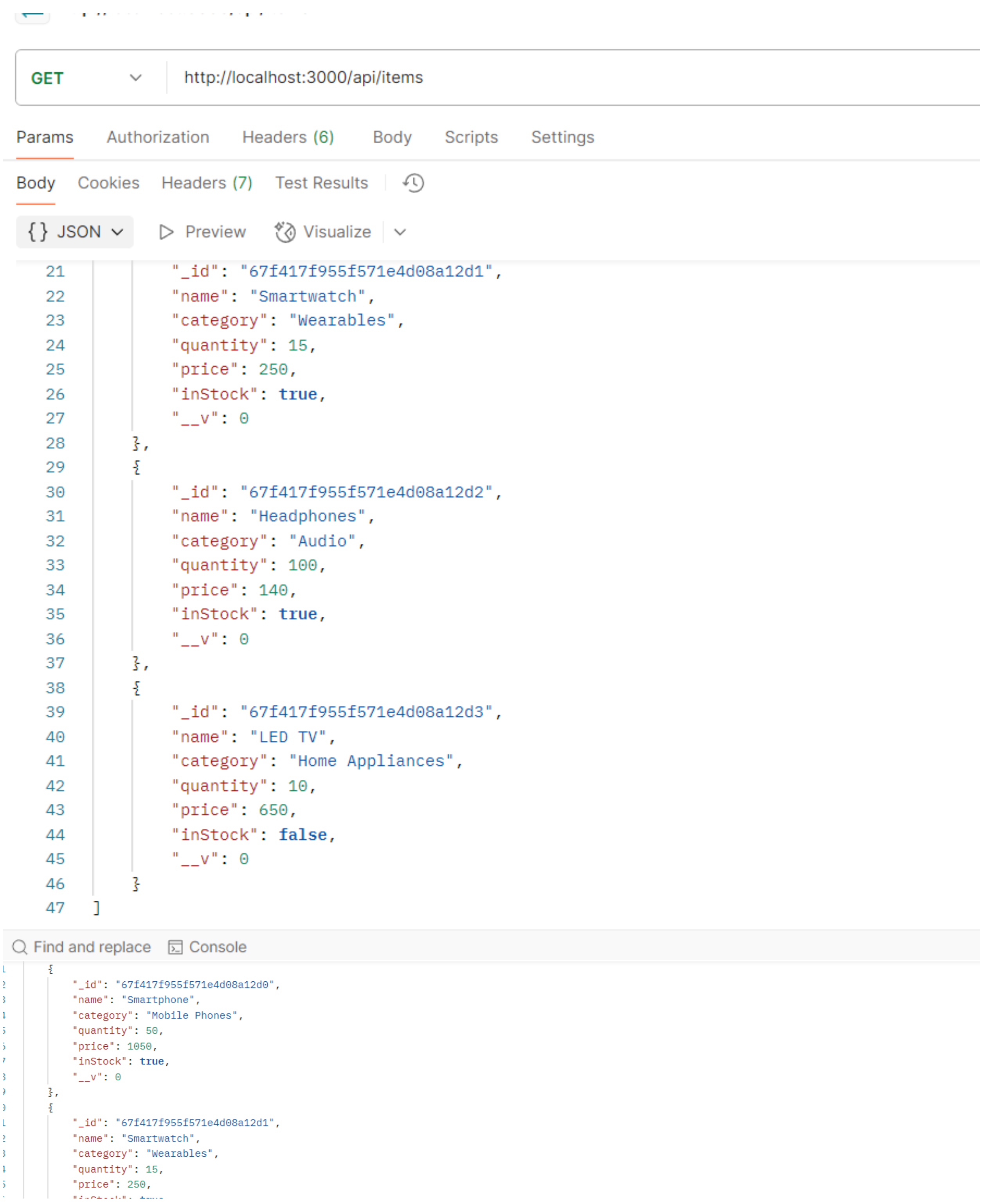
- Open Postman.
- Send a POST request to:

http://localhost:3000/api/items



```
// config/db.js

import mongoose from 'mongoose';
import dotenv from 'dotenv';

dotenv.config();

// Connect to MongoDB using the connection string from .env
```

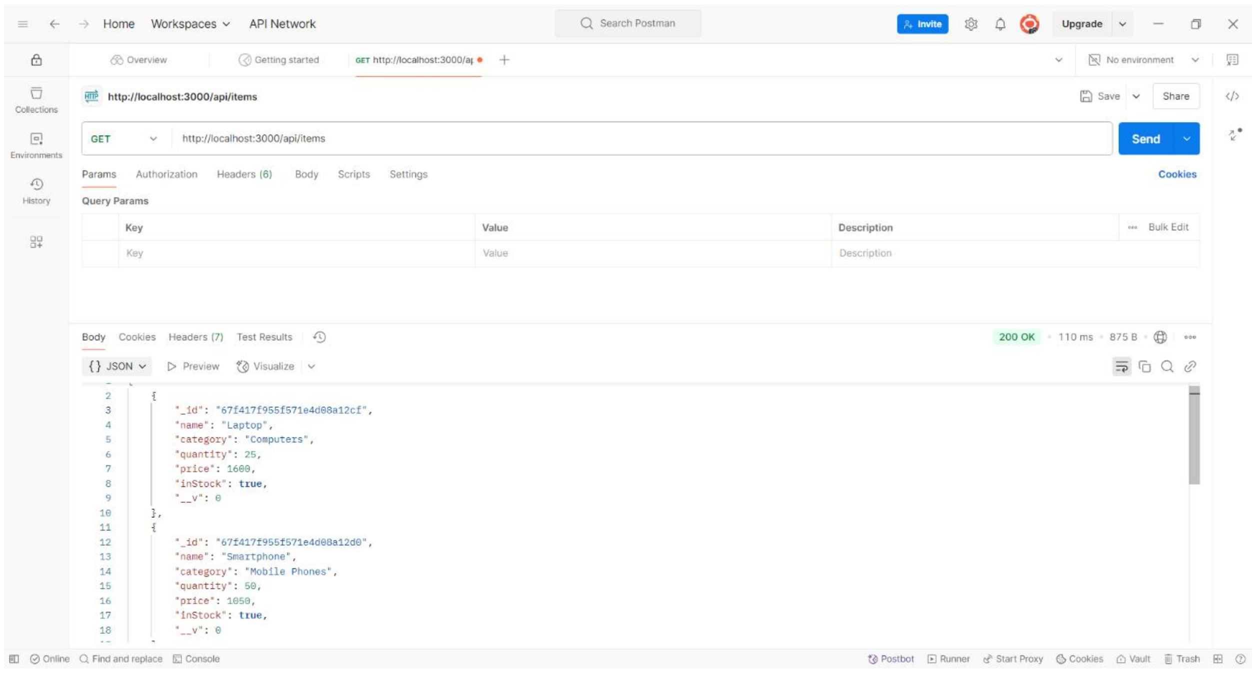
```
PS C:\Users\smile\OneDrive\Desktop\men-app> npm run dev
>>

> men-app@1.0.0 dev
> nodemon app.js

[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
🚀 Server running at http://localhost:3000
✅ MongoDB connected successfully
```



```
http://localhost:3000/api/items
```

```
        "_id": "67f417f955f571e4d08a12d1",
        "name": "Smartwatch",
        "category": "Wearables",
        "quantity": 15,
        "price": 250,
        "inStock": true,
        "__v": 0
    },
    {
        "_id": "67f417f955f571e4d08a12d2",
        "name": "Headphones",
        "category": "Audio",
        "quantity": 100,
        "price": 140,
        "inStock": true,
        "__v": 0
    },
    {
        "_id": "67f417f955f571e4d08a12d3",
        "name": "LED TV",
        "category": "Home Appliances",
        "quantity": 10,
        "price": 650,
        "inStock": false,
        "__v": 0
    }
]
```

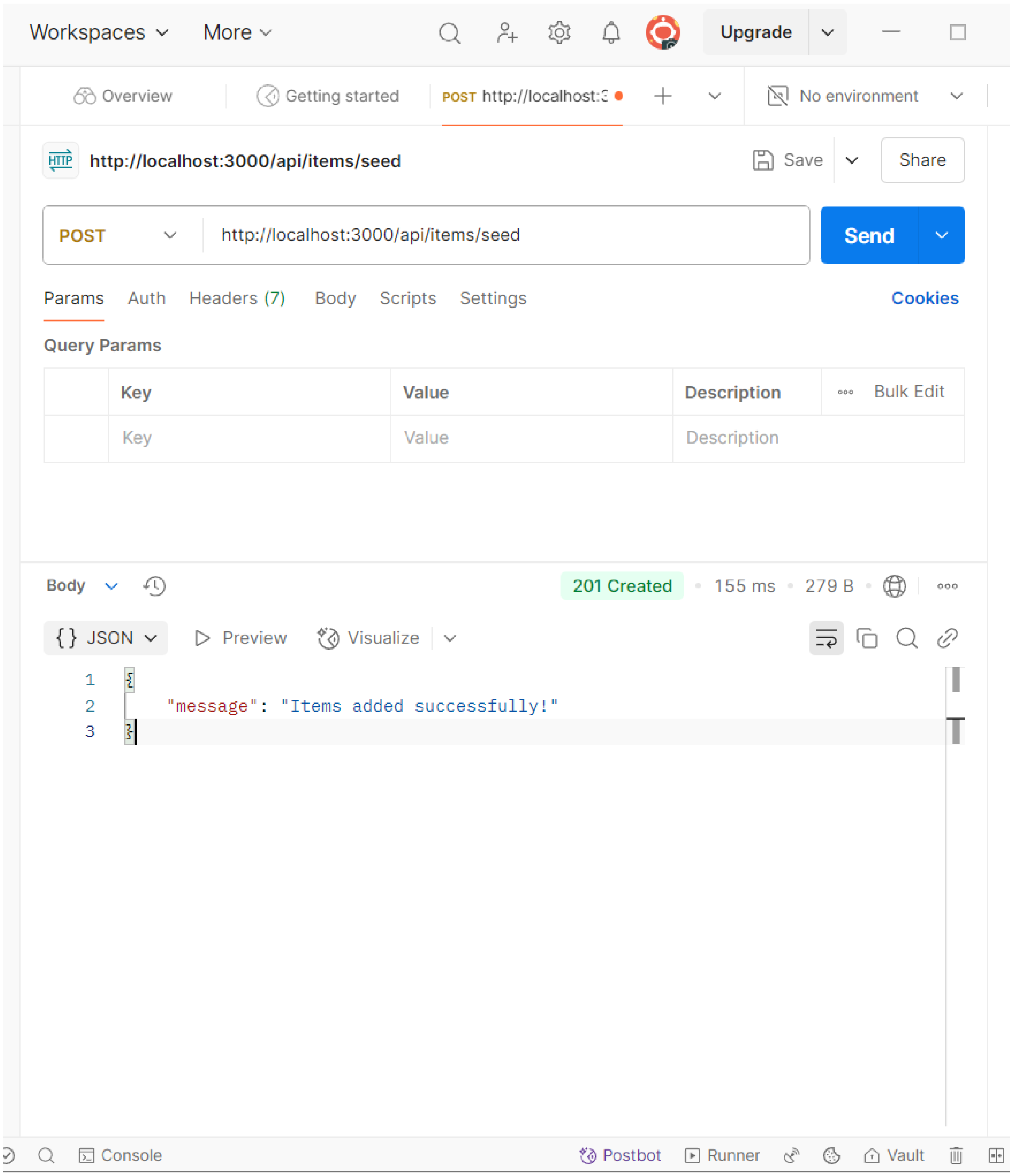
```
    {
        "_id": "67f417f955f571e4d08a12d0",
        "name": "Smartphone",
        "category": "Mobile Phones",
        "quantity": 50,
        "price": 1050,
        "inStock": true,
        "__v": 0
    },
    {
        "_id": "67f417f955f571e4d08a12d1",
        "name": "Smartwatch",
        "category": "Wearables",
        "quantity": 15,
        "price": 250,
```



GET http://localhost:3000/api/items

200 OK · 110 ms · 875 B

```
    {
        "_id": "67f417f955f571e4d08a12cf",
        "name": "Laptop",
        "category": "Computers",
        "quantity": 25,
        "price": 1600,
        "inStock": true,
        "__v": 0
    },
    {
        "_id": "67f417f955f571e4d08a12d0",
        "name": "Smartphone",
        "category": "Mobile Phones",
        "quantity": 50,
        "price": 1050,
        "inStock": true,
        "__v": 0
```



POST http://localhost:3000/api/items/seed

Params | Auth | Headers (7) | Body | Scripts | Settings | Cookies

Query Params

	Key	Value	Description
	Key	Value	Description

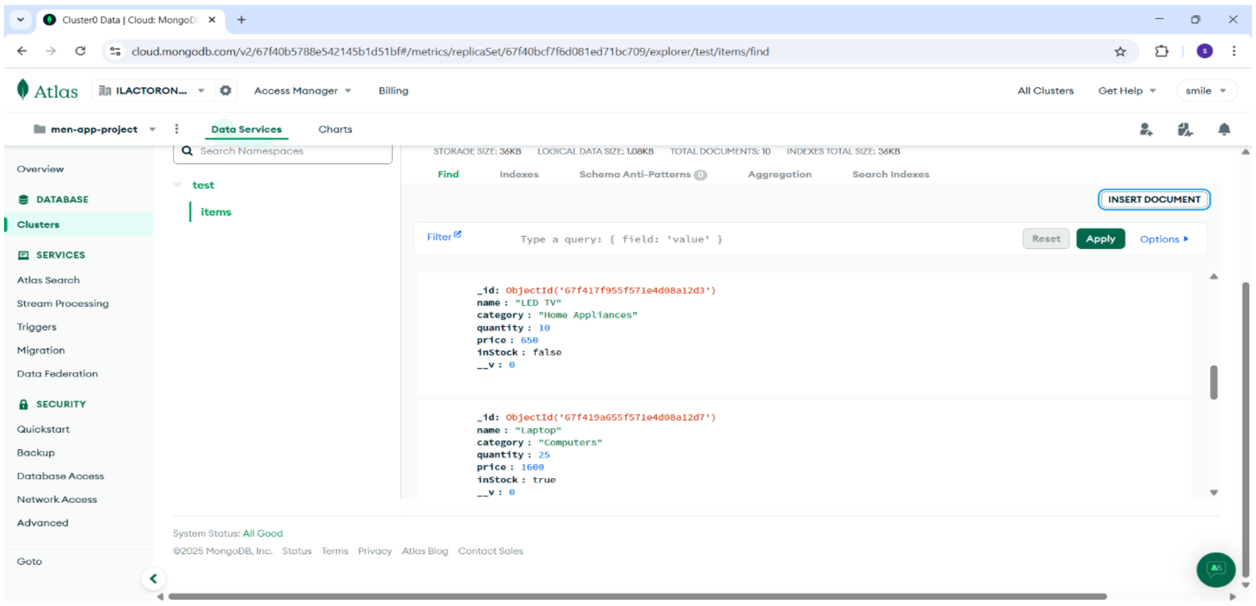
Body — 201 Created • 155 ms • 279 B

JSON

```
{
    "message": "Items added successfully!"
}
```

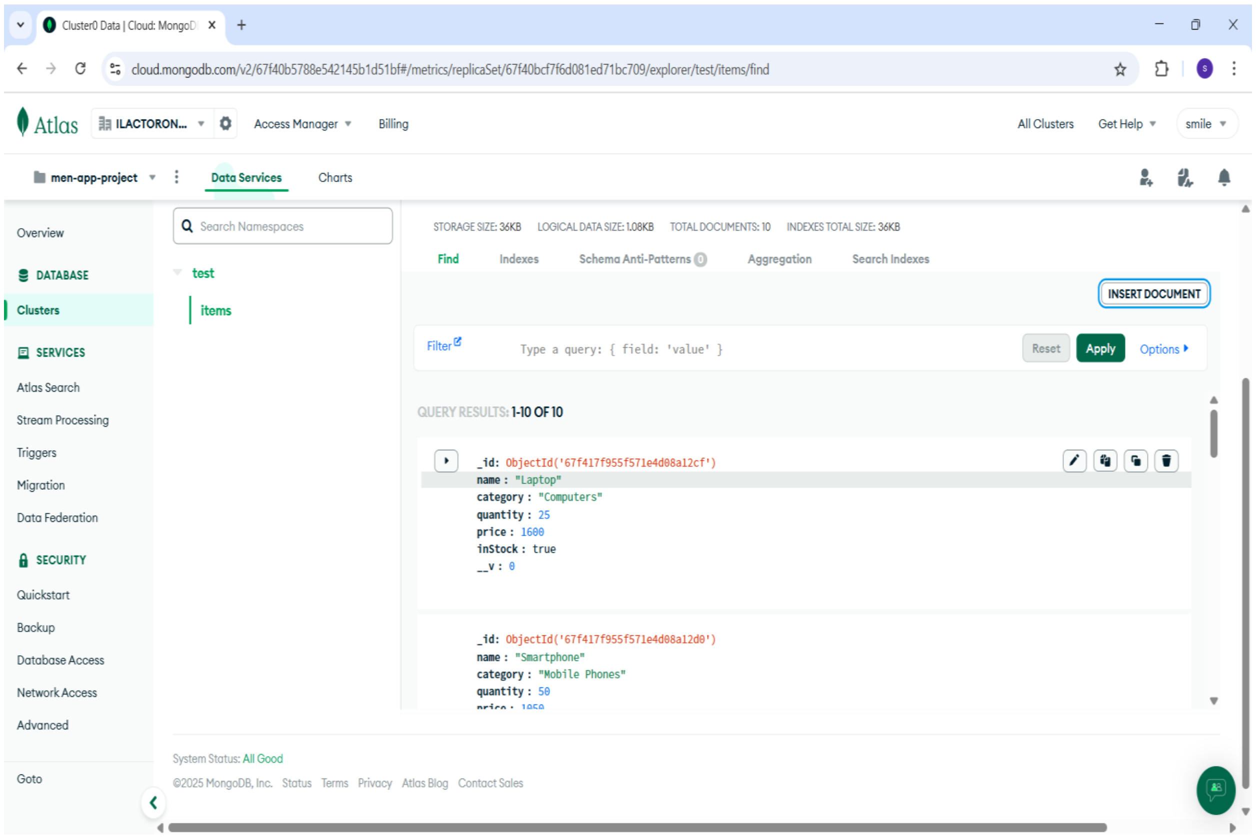
MongoDB Atlas Collection with One Document

The items collection is visible in my MongoDB Atlas database. Electronic things are represented by the documents in this collection. The five essential fields in every document are name, category, quantity, price, and inStock. To guarantee data quality, validation procedures are used and the data is kept in a professional schema.



STORAGE SIZE: 36KB LOGICAL DATA SIZE: 1.08KB TOTAL DOCUMENTS: 10 INDEXES TOTAL SIZE: 36KB

```
_id: ObjectId('67f417f955f571e4d08a12d3')
name : "LED TV"
category : "Home Appliances"
quantity : 10
price : 650
inStock : false
__v : 0

_id: ObjectId('67f419a655f571e4d08a12d7')
name : "Laptop"
category : "Computers"
quantity : 25
price : 1600
inStock : true
__v : 0
```



STORAGE SIZE: 36KB LOGICAL DATA SIZE: 1.08KB TOTAL DOCUMENTS: 10 INDEXES TOTAL SIZE: 36KB

QUERY RESULTS: 1-10 OF 10

```
_id: ObjectId('67f417f955f571e4d08a12cf')
name : "Laptop"
category : "Computers"
quantity : 25
price : 1600
inStock : true
__v : 0

_id: ObjectId('67f417f955f571e4d08a12d0')
name : "Smartphone"
category : "Mobile Phones"
quantity : 50
price : 1050
```

System Status: All Good

©2025 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales