**Restful API using Express.js and**

**Database and Index in MongoDB**

**Name:** Pandu Ranga Prudvi Jerripothula

**E-mail:** prudvijerripothula@gmail.com

**Phone:** 6301021361

**Roll No:**20VV1A0429

**College Name:** JNTU-GV College of Engineering Vizianagaram

**Source Code:**

**Server.js:**

const express = require('express')

const mongoose = require('mongoose')

const Product = require('./productModel')

const app = express()

app.use(express.json())

app.use(express.urlencoded({extended: false}))

//routes

app.get('/',(req, res) => {

    res.send('Hello NODE API')

})

app.get('/blog',(req, res) => {

    res.send('My name is Prudvi')

})

app.get('/products',async(req,res) =>{

    try {

        const products = await Product.find({});

        res.status(200).json(products)

    } catch (error) {

        res.status(500).json({message:error.message})

    }

})

app.get('/products/:id',async(req,res) => {

    try {

        const {id}=req.params;

        const product = await Product.findById(id);

        res.status(200).json(product)

    } catch (error) {

        res.status(500).json({message:error.message})

    }

} )

app.post('/products',async(req,res) =>{

    try {

        const product = await Product.create(req.body)

        res.status(200).json(product);

    } catch (error) {

        console.log(error.message);

        res.status(500).json({message: error.message})

    }

})

//update a product

app.put('/products/:id',async(req,res) => {

    try {

        const{id} = req.params;

        const product = await Product.findByIdAndUpdate(id, req.body);

        if(!product){

            return res.status(404).json({message: 'cannot find any product with ID ${id}'})

        }

        const updatedProduct = await Product.findById(id);

        res.status(200).json(updatedProduct);

    } catch (error) {

        res.status(500).json({message: error.message})

    }

})

//delete a product

app.delete('/products/:id',async(req, res) => {

    try {

        const{id}= req.params;

        const product = await Product.findByIdAndDelete(id);

        if(!product){

            return res.status(404).json({message: 'cannot find any product with ID ${id}'})

        }

        res.status(200).json(product);

    } catch (error) {

        res.status(500).json({message: error.message})

    }

})

mongoose.set("strictQuery",false)

mongoose.

connect('mongodb+srv://admin:Admin-1234@prudviapi.hnussfn.mongodb.net/NODEAPI?retryWrites=true&w=majority&appName=PrudviAPI')

.then(()=>{

    app.listen(3000,()=> {

    console.log('Node API app is running on port 3000')

})

    console.log('connected to MongoDB')

}).catch((error) =>{

    console.log(error)

})

**Package.json:**

const mongoose = require('mongoose')

const productSchema = mongoose.Schema(

    {

        name:{

            type: String,

            required:[true,"Please enter a product name"]

        },

        quantity:{

            type: Number,

            required: true,

            default: 0

        },

        price:{

            type:Number,

            required:true,

        },

        image:{

            type: String,

            required: false

        }

    },

    {

        timestamps:true

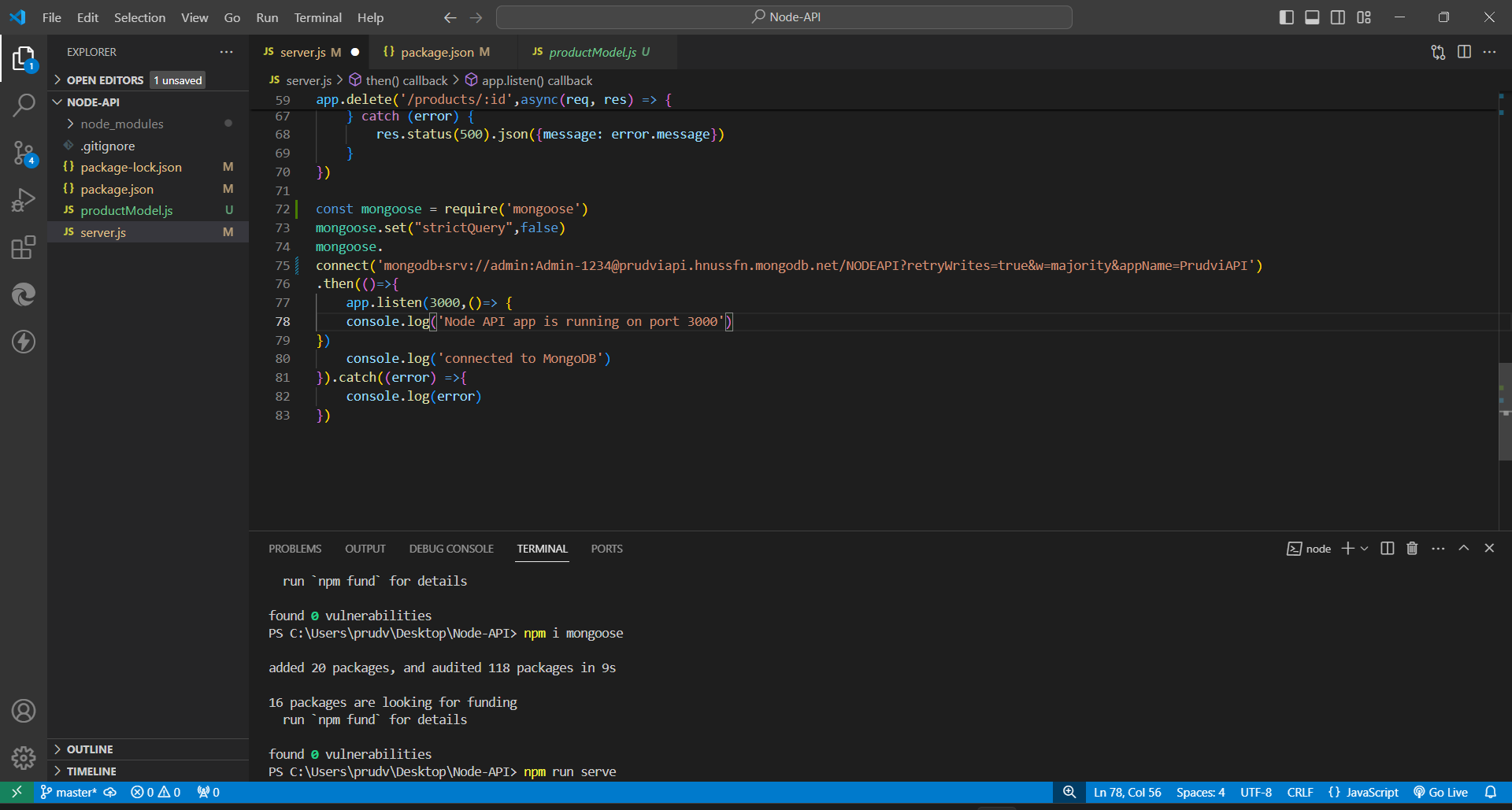
    }

)

const Product = mongoose.model('Product',productSchema);

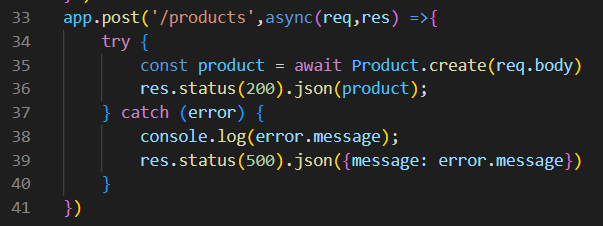
module.exports= Product;

**MongoDB Connection:**

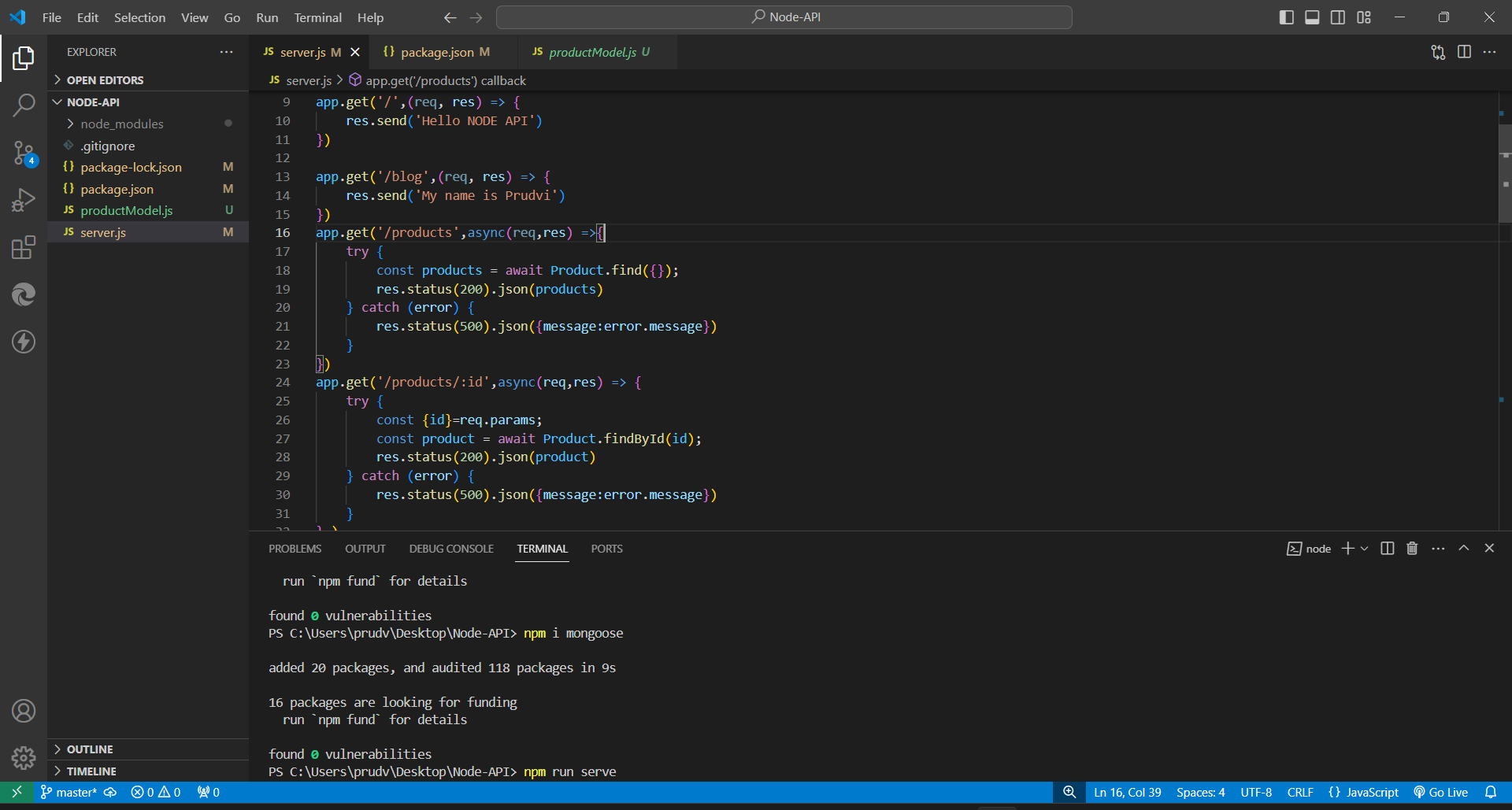
****

**CRUD Operations:**

**Create Operation:**

****

**Read/Read by ID Operation:**



**Update by ID Operation:**

****

**Delete by ID Operation:**

****

**Method of Running:**

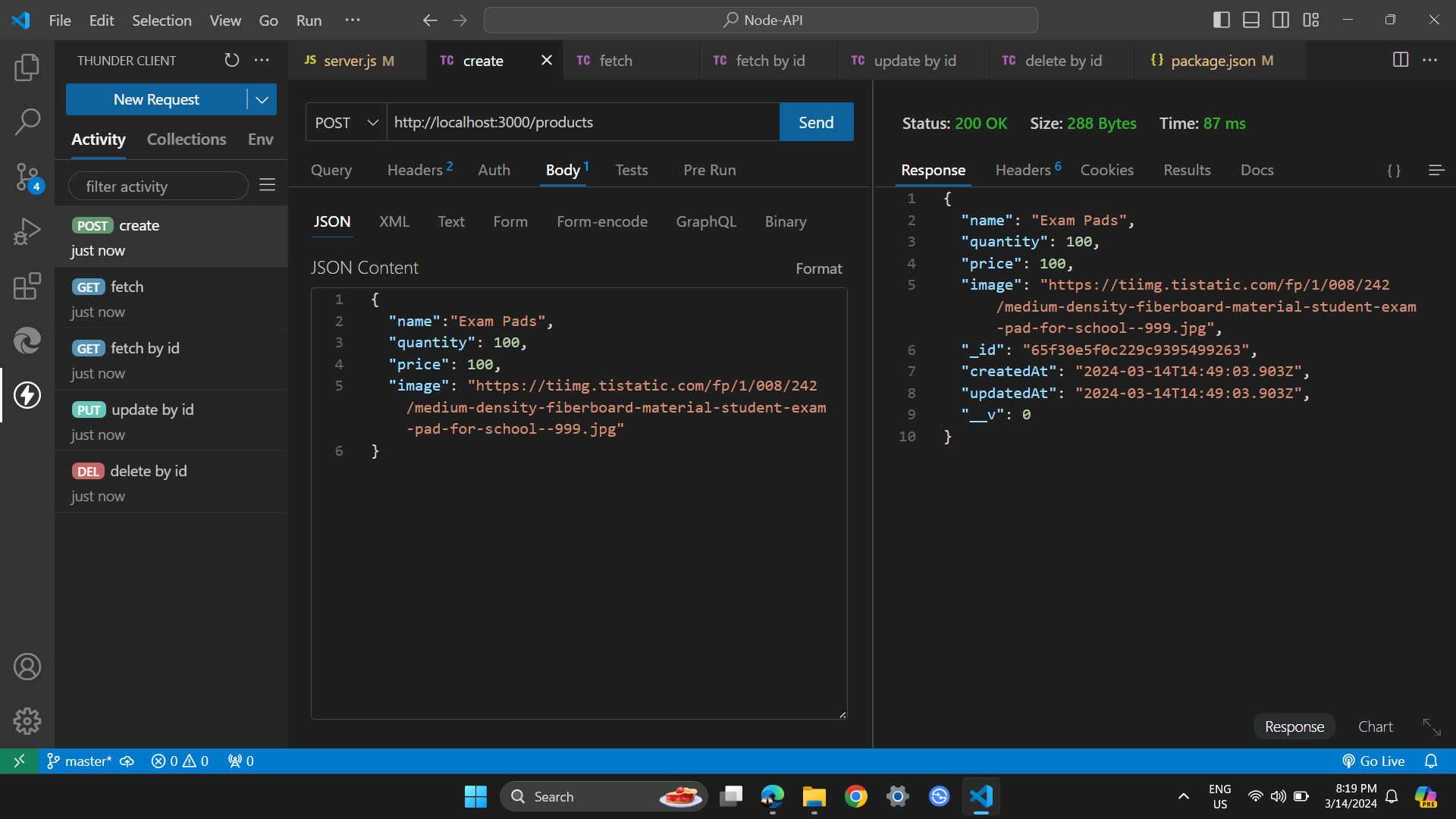
1. Create package.json using the code “**npm init -y**” in the terminal.
2. Add empty git using the code “**git init**”, “**git add .**”, “**git commit -m “project”** ”.
3. Install the required dependencies in terminal respectively as
   1. Express.js – “**npm i express**”
   2. Nodemon – “**npm i nodemon -D**”
   3. Mongoose – “**npm i mongoose**”
4. Run the API using “**node server.js**” in the terminal.

**Outputs:**

**CRUD Operations:**

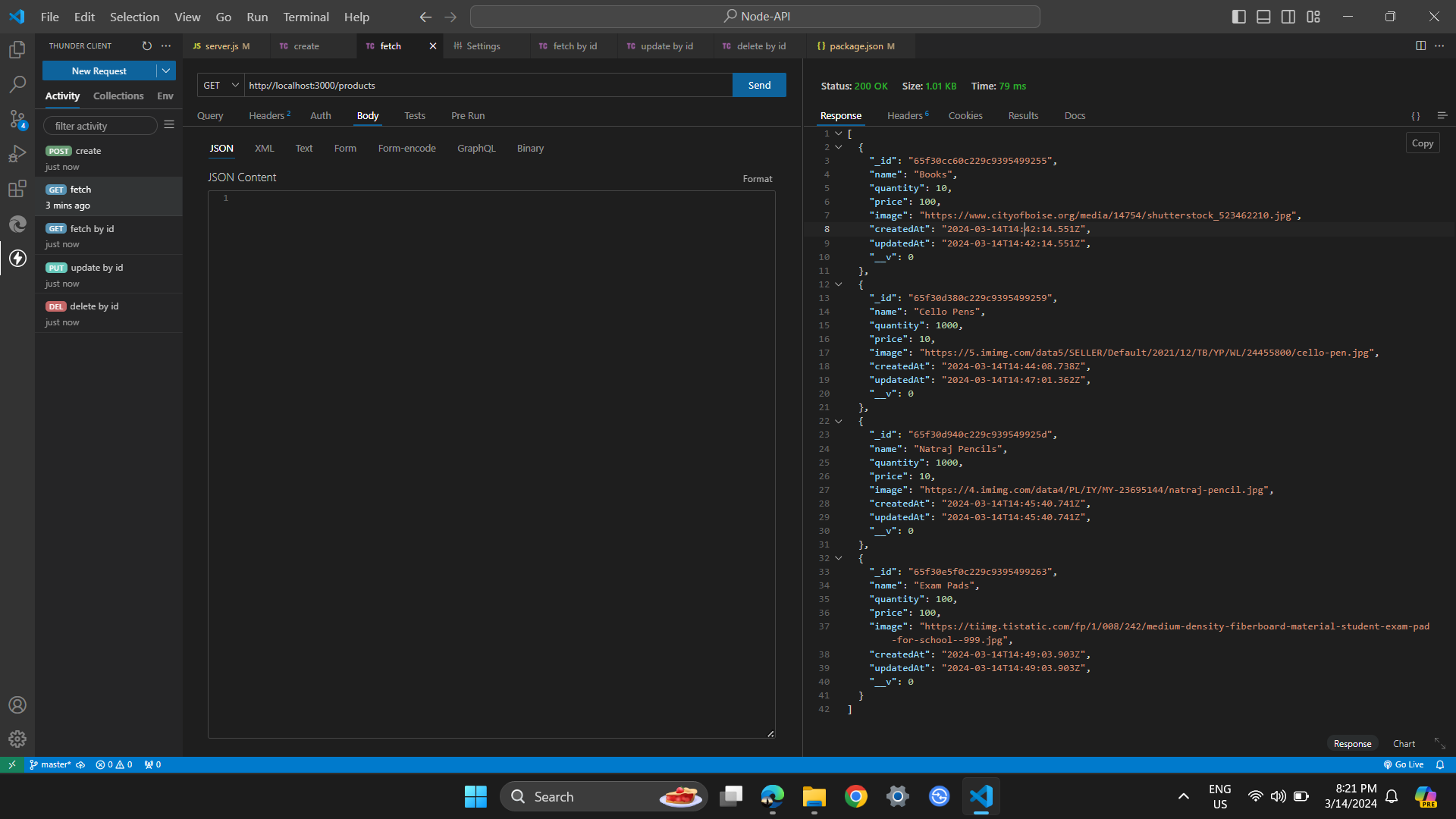
1. **Create Operation:**

The below is operation of creating an item/product in the database:

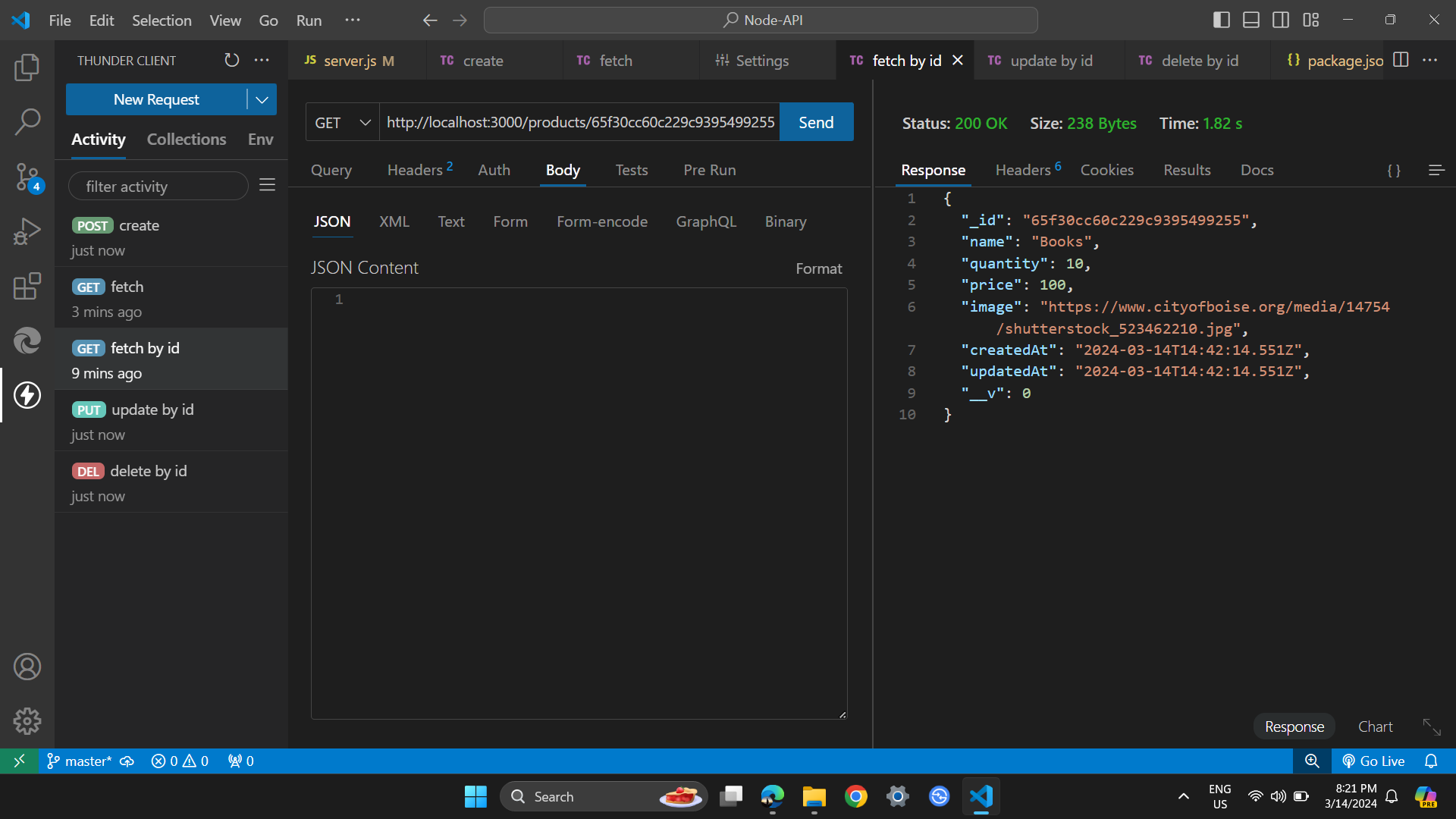


1. **Read/Read by ID Operation:**

The below output is an operation of read all items/products:

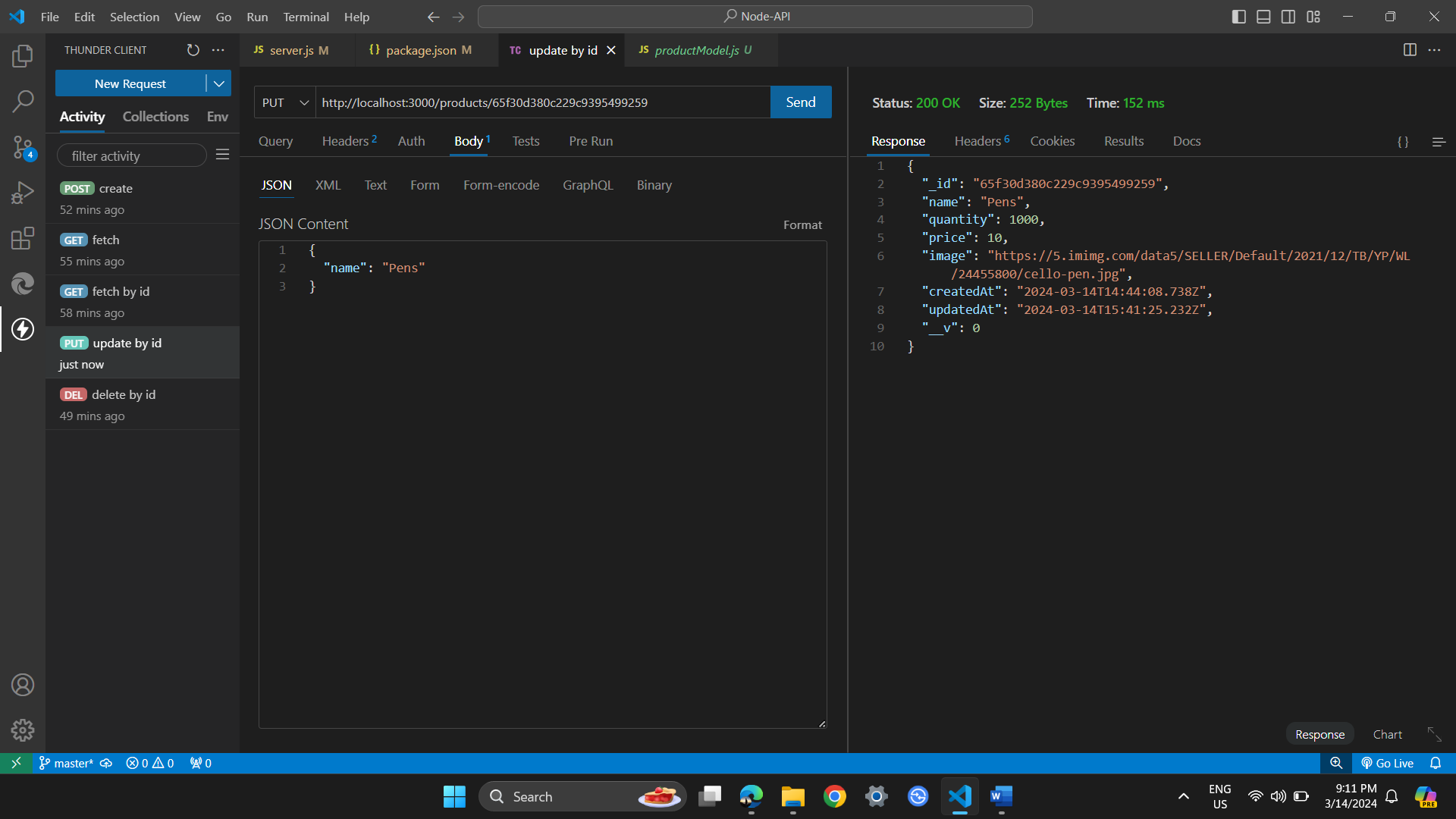


The below is an operation of read by id:



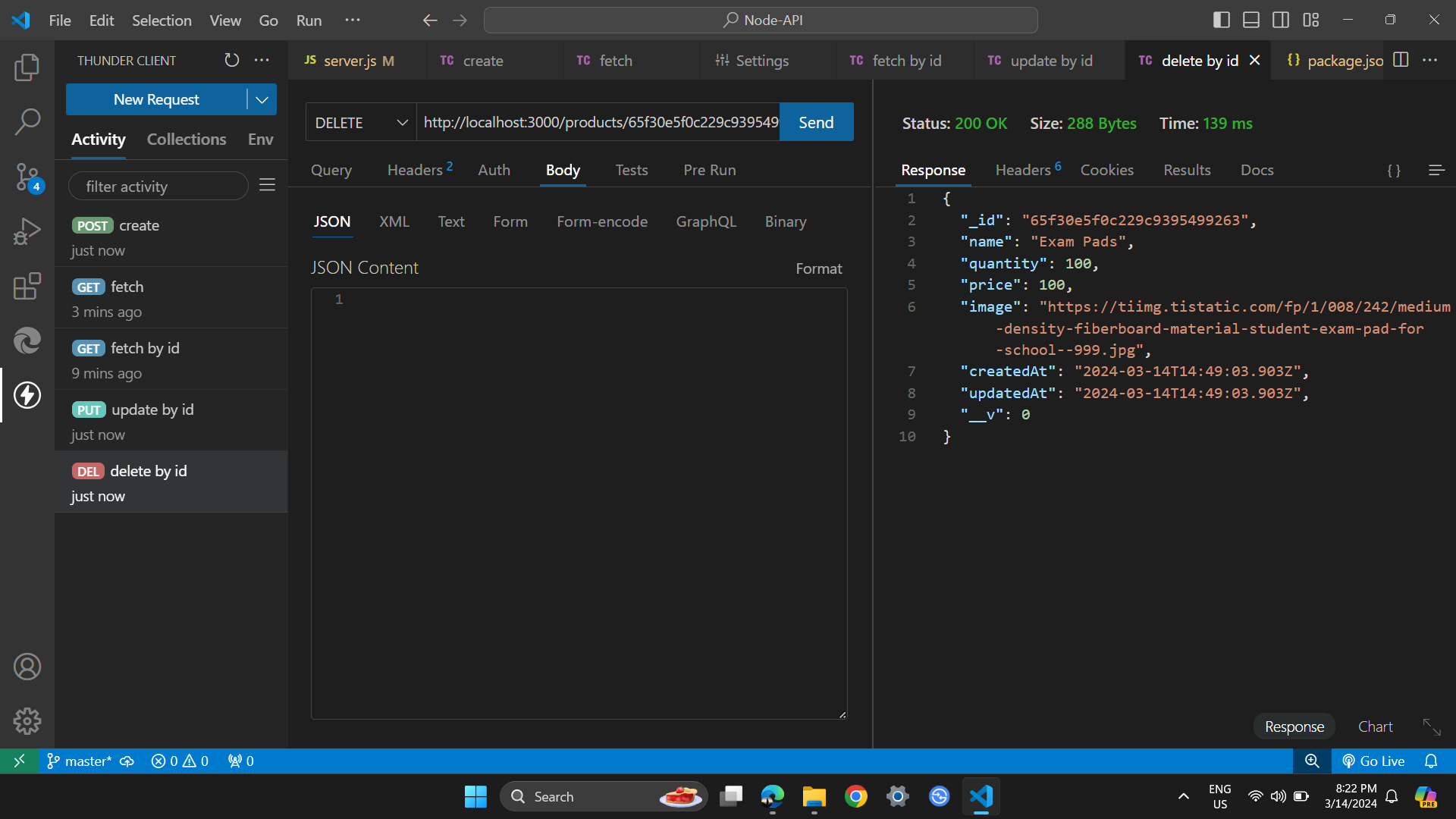
1. **Update by ID Operation:**

Changing the name: Cello Pens to Pens is identified in the below output.

****

1. **Delete by ID Operation:**

The below is an operation for deleting an item where the object is first identified before deleting:



And below is the operation that depicts the successful deletion of the item:

