|  |  |
| --- | --- |
| **Name**: KONIKA KHURANA | **UID**:23BCS12925 |
| **Branch**: CSE | **Section**: 23 BCS\_FS 622-A |
| **Semester**: 5 | **Date of Performance**: 09/10/2025 |
| **Subject**: FULL STACK-LAB | **Subject Code**: 23CSP-339 |

**Mongo db 1:**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.json());

mongoose.connect('mongodb://localhost:27017/productsDB', {

useNewUrlParser: true,

useUnifiedTopology: true

})

.then(() => console.log('MongoDB connected'))

.catch(err => console.log(err));

const productSchema = new mongoose.Schema({

name: { type: String, required: true },

price: { type: Number, required: true },

category: { type: String, required: true }

});

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

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

try {

const product = new Product(req.body);

const savedProduct = await product.save();

res.status(201).json(savedProduct);

} catch (err) {

res.status(400).json({ message: err.message });

}

});

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

try {

const products = await Product.find();

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

} catch (err) {

res.status(500).json({ message: err.message });

}

});

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

try {

const updatedProduct = await Product.findByIdAndUpdate(

req.params.id,

req.body,

{ new: true }

);

if (!updatedProduct)

return res.status(404).json({ message: 'Product not found' });

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

} catch (err) {

res.status(400).json({ message: err.message });

}

});

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

try {

const deletedProduct = await Product.findByIdAndDelete(req.params.id);

if (!deletedProduct)

return res.status(404).json({ message: 'Product not found' });

res.status(200).json({

message: 'Product deleted',

product: deletedProduct

});

} catch (err) {

res.status(500).json({ message: err.message });

}

});

const PORT = 3000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

Expected output:

A screen shot of a computer program

AI-generated content may be incorrect.

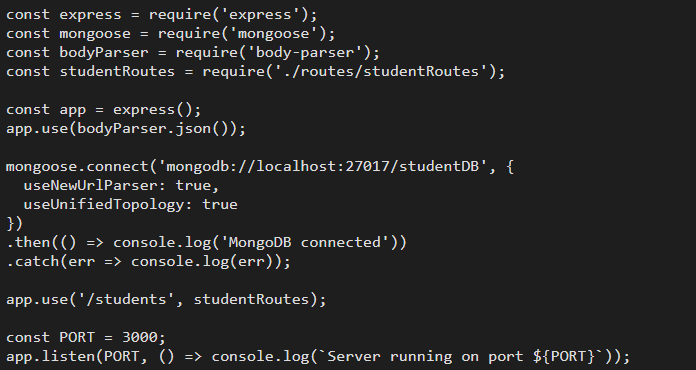
Mongo db 2:

Project Structure

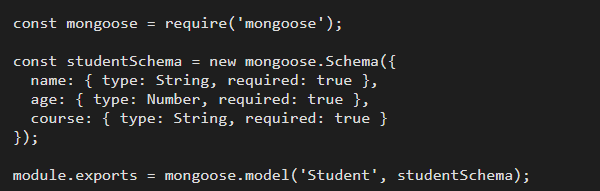
A screen shot of a computer code

AI-generated content may be incorrect.

**server.js**



**Models/studentModel.js**

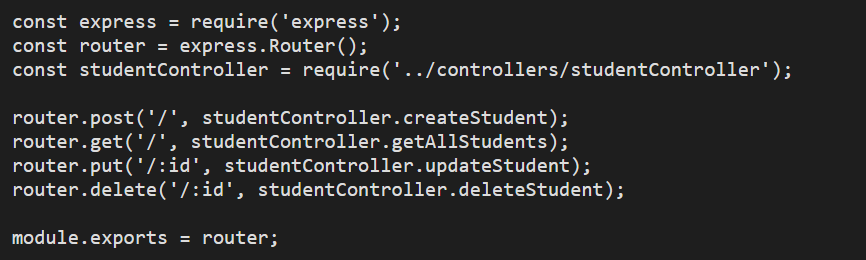
****

**Controllers/studentController.js**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Routes/studentRoutes.js**

****

**Expected Output (Postman Style):**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Mongo db 3:**

**use ecommerceDB;**

**db.createCollection("products");**

**db.products.insertMany([**

**{**

**name: "Smartphone",**

**price: 699,**

**category: "Electronics",**

**variants: []**

**},**

**{**

**name: "Winter Jacket",**

**price: 200,**

**category: "Apparel",**

**variants: [**

**{ color: "Black", size: "S", stock: 8 },**

**{ color: "Gray", size: "M", stock: 12 }**

**]**

**},**

**{**

**name: "Running Shoes",**

**price: 120,**

**category: "Footwear",**

**variants: [**

**{ color: "Red", size: "M", stock: 10 },**

**{ color: "Blue", size: "L", stock: 5 }**

**]**

**}**

**]);**

**db.products.find().pretty();**

**db.products.find({ category: "Electronics" }).pretty();**

**db.products.find({ "variants.color": "Blue" }).pretty();**

**db.products.find({}, { name: 1, "variants.color": 1, \_id: 0 }).pretty();**

**db.products.updateOne(**

**{ name: "Running Shoes", "variants.color": "Blue" },**

**{ $set: { "variants.$.stock": 8 } }**

**);**

**db.products.deleteOne({ name: "Smartphone" });**