Full Stack Development with MERN

Database Design and Development Report

Date	12th July 2024
Team ID	SWTID1719933594
Project Name	SHOPEZ -E-COMMERCE APP
Maximum Marks	

Project Title: SHOPEZ -E-COMMERCE APP

Date: 12th July 2024

Prepared by: Nandini J Nair

Objective

The objective of this report is to outline the database design and implementation details for the SHOPEZ -E-COMMERCE APP project, including schema design and database management system (DBMS) integration.

Technologies Used

- **Database Management System (DBMS):** MongoDB
- **Object-Document Mapper (ODM):** Mongoose

Design the Database Schema

The database schema is designed to accommodate the following entities and relationships:

1. Users

- Attributes: [

name,

email,

password,

phone,

address,

answer,

role]

2. Product

- Attributes: [name,
Slug,
Description,
Price,
Category,
Quantity,
Photo,
shipping]

3. orders

-Attributes: [products,
Payment{buyer},
Status]

4. category

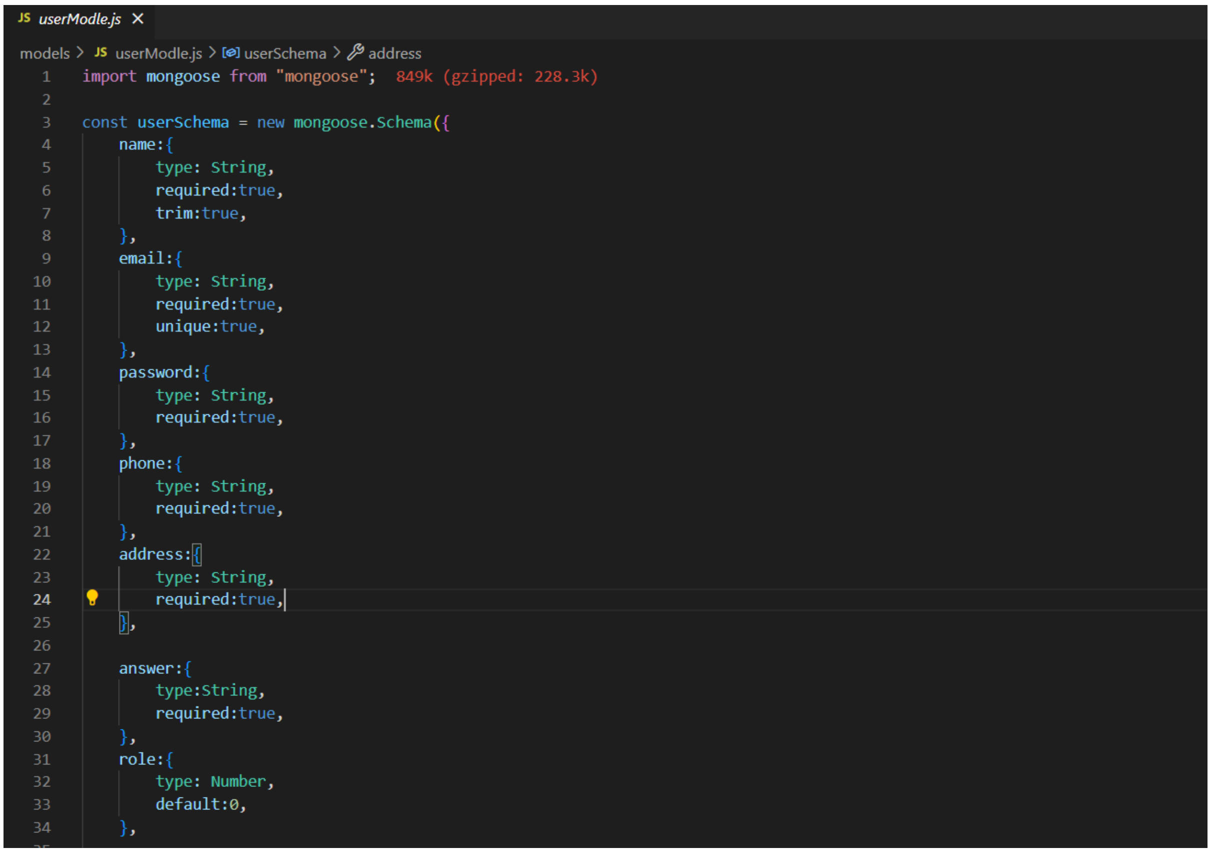
Attributes: [name,
Slug]

Implement the Database using MongoDB

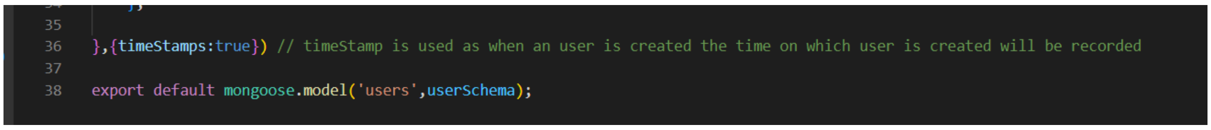
The MongoDB database is implemented with the following collections and structures:

Database Name: shopez

1. Collection: users

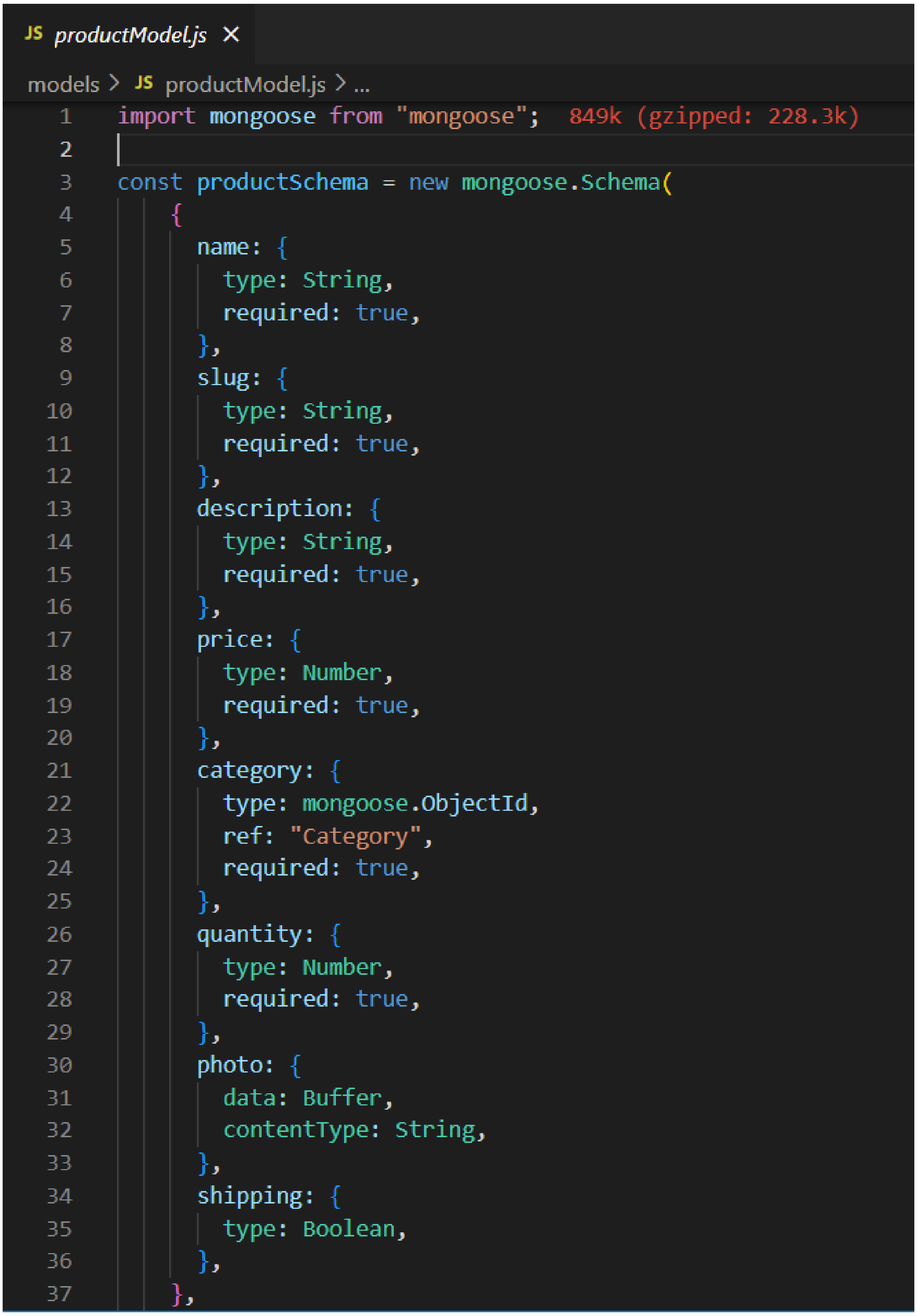


```
import mongoose from "mongoose";  849k (gzipped: 228.3k)

const userSchema = new mongoose.Schema({
    name:{
        type: String,
        required:true,
        trim:true,
    },
    email:{
        type: String,
        required:true,
        unique:true,
    },
    password:{
        type: String,
        required:true,
    },
    phone:{
        type: String,
        required:true,
    },
    address:{
        type: String,
        required:true,
    },

    answer:{
        type:String,
        required:true,
    },
    role:{
        type: Number,
        default:0,
    },
```



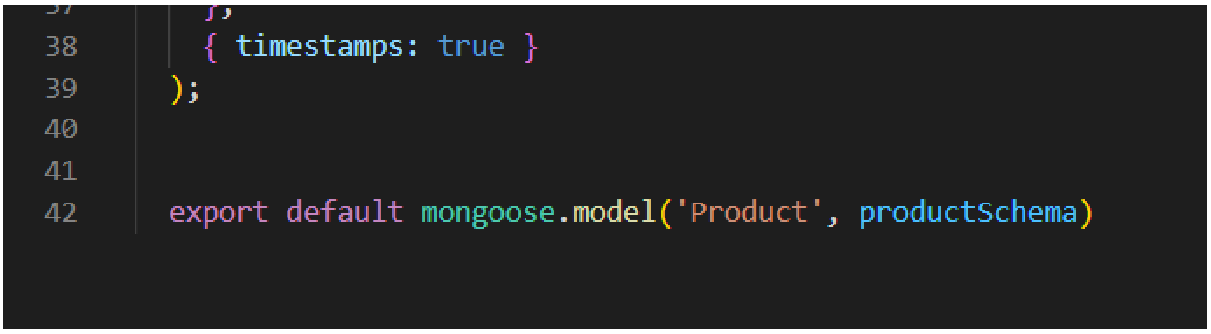
```

},{timeStamps:true}) // timeStamp is used as when an user is created the time on which user is created will be recorded

export default mongoose.model('users',userSchema);
```

2. Collection: Product


productModel.js ×

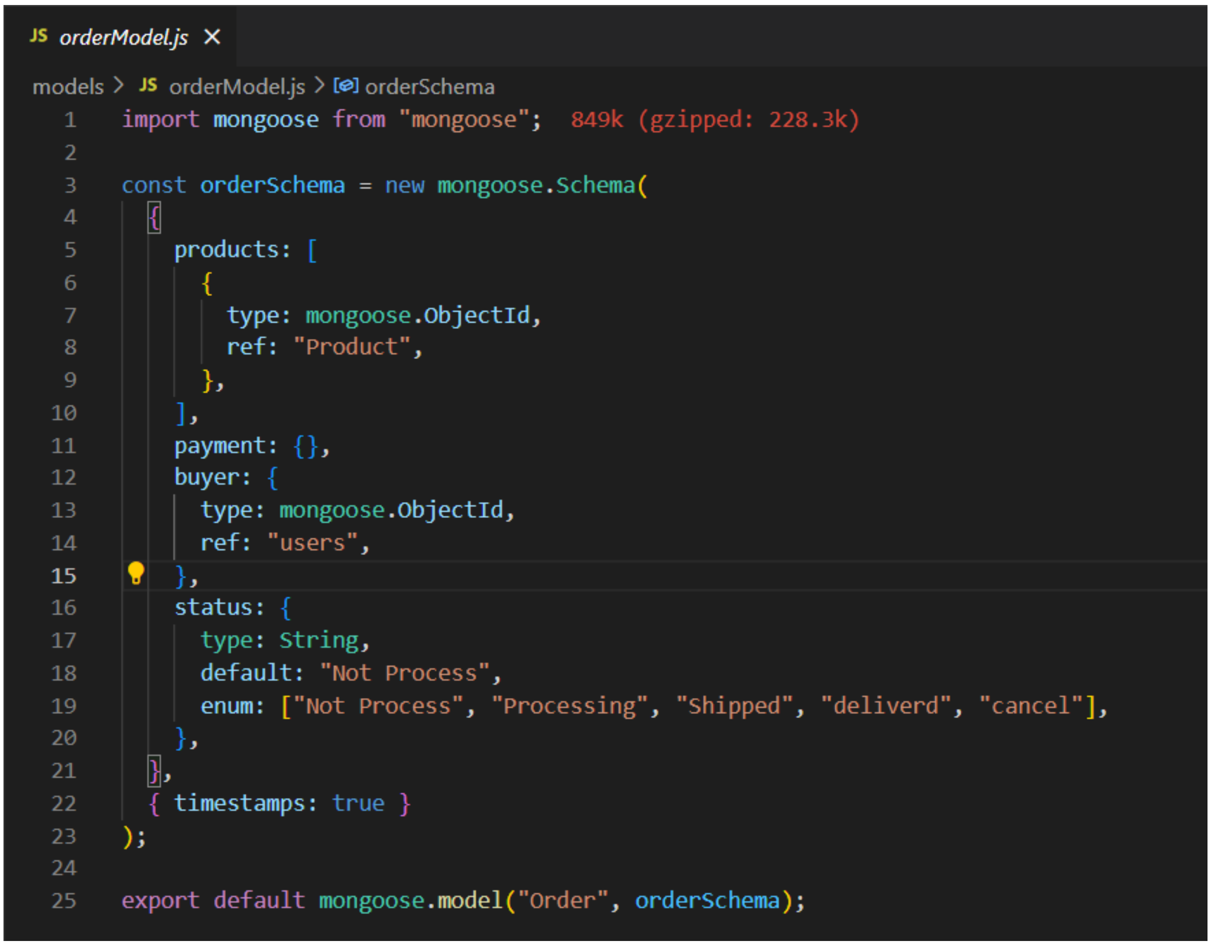
models > productModel.js > ...

```
import mongoose from "mongoose";  849k (gzipped: 228.3k)

const productSchema = new mongoose.Schema(
  {
    name: {
      type: String,
      required: true,
    },
    slug: {
      type: String,
      required: true,
    },
    description: {
      type: String,
      required: true,
    },
    price: {
      type: Number,
      required: true,
    },
    category: {
      type: mongoose.ObjectId,
      ref: "Category",
      required: true,
    },
    quantity: {
      type: Number,
      required: true,
    },
    photo: {
      data: Buffer,
      contentType: String,
    },
    shipping: {
      type: Boolean,
    },
  },
```



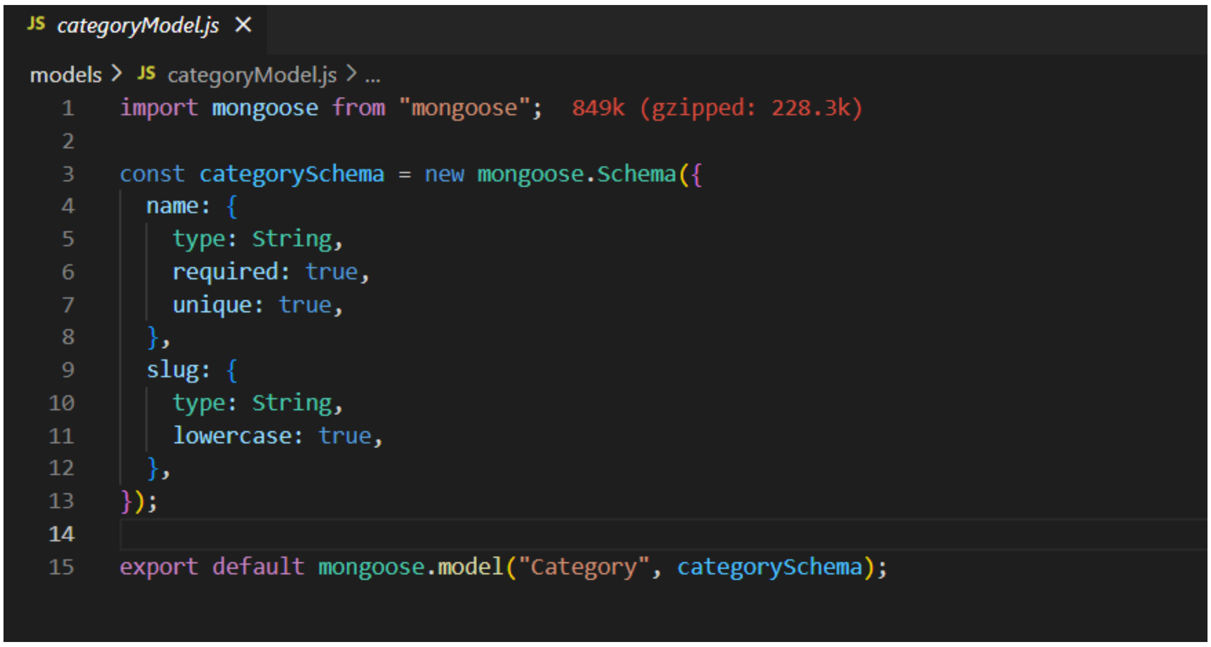
```
    { timestamps: true }
);


export default mongoose.model('Product', productSchema)
```

3. Collection: Order



```
import mongoose from "mongoose";  849k (gzipped: 228.3k)

const orderSchema = new mongoose.Schema(
  {
    products: [
      {
        type: mongoose.ObjectId,
        ref: "Product",
      },
    ],
    payment: {},
    buyer: {
      type: mongoose.ObjectId,
      ref: "users",
    },
    status: {
      type: String,
      default: "Not Process",
      enum: ["Not Process", "Processing", "Shipped", "deliverd", "cancel"],
    },
  },
  { timestamps: true }
);

export default mongoose.model("Order", orderSchema);
```

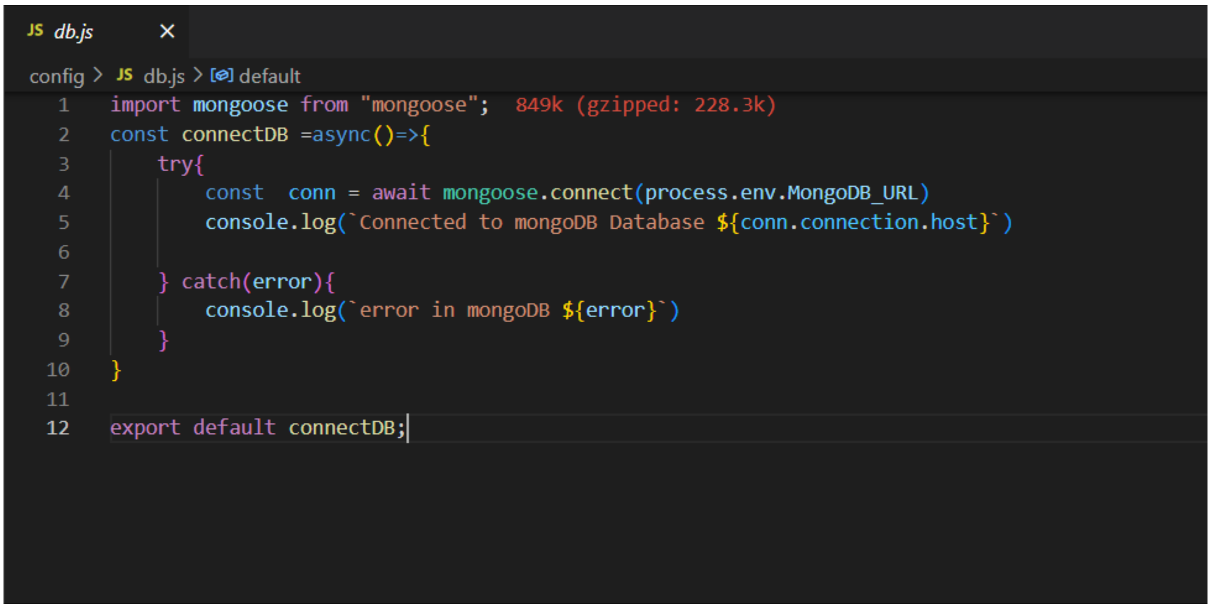
4 collection:Category



```js
import mongoose from "mongoose";  849k (gzipped: 228.3k)

const categorySchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    unique: true,
  },
  slug: {
    type: String,
    lowercase: true,
  },
});

export default mongoose.model("Category", categorySchema);
```

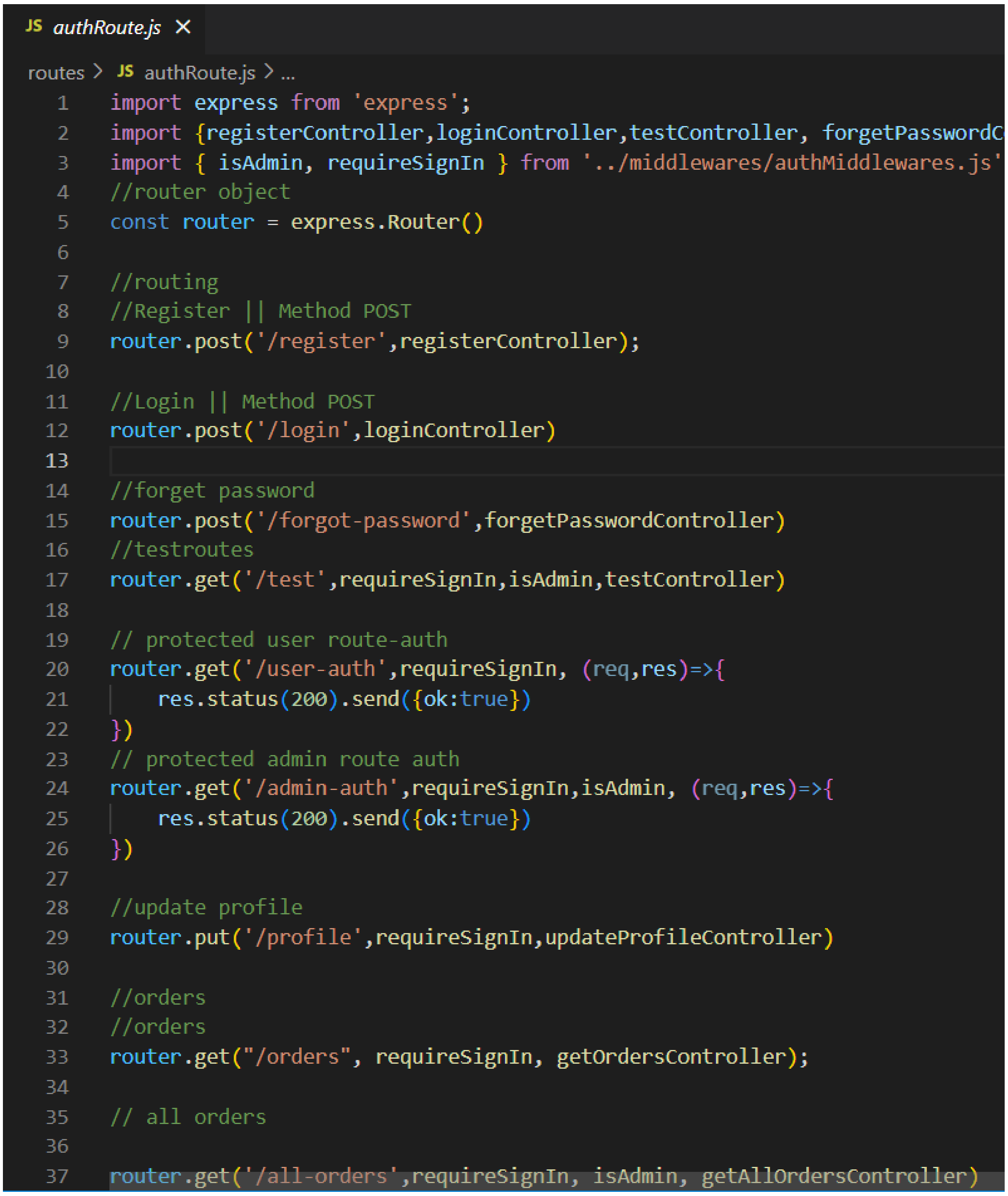
Integration with Backend

- Database connection: Database connection done using Mongoose

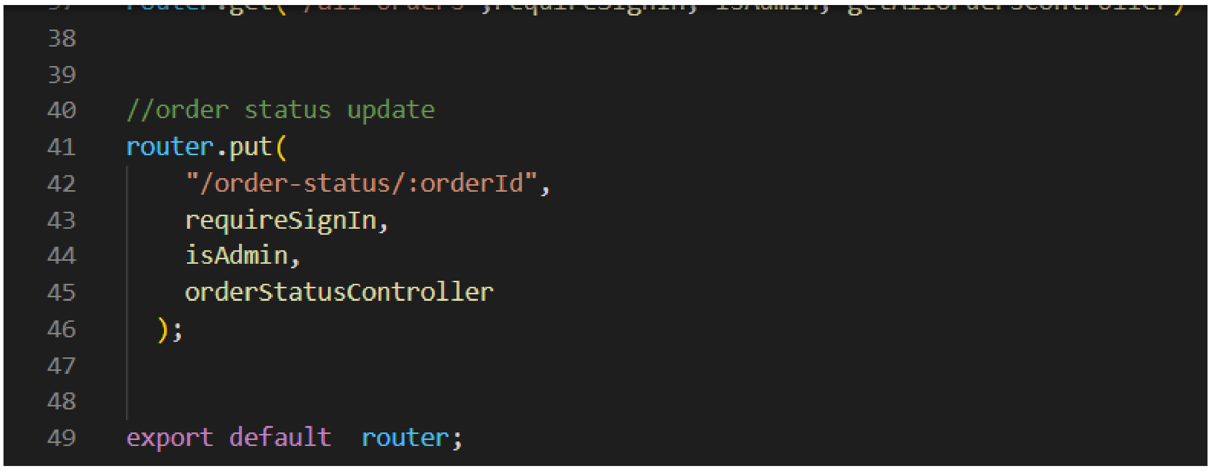


```js
import mongoose from "mongoose";  849k (gzipped: 228.3k)
const connectDB =async()=>{
    try{
        const  conn = await mongoose.connect(process.env.MongoDB_URL)
        console.log(`Connected to mongoDB Database ${conn.connection.host}`)

    } catch(error){
        console.log(`error in mongoDB ${error}`)
    }
}

export default connectDB;
```

- The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
  - User Management: CRUD operations for users.

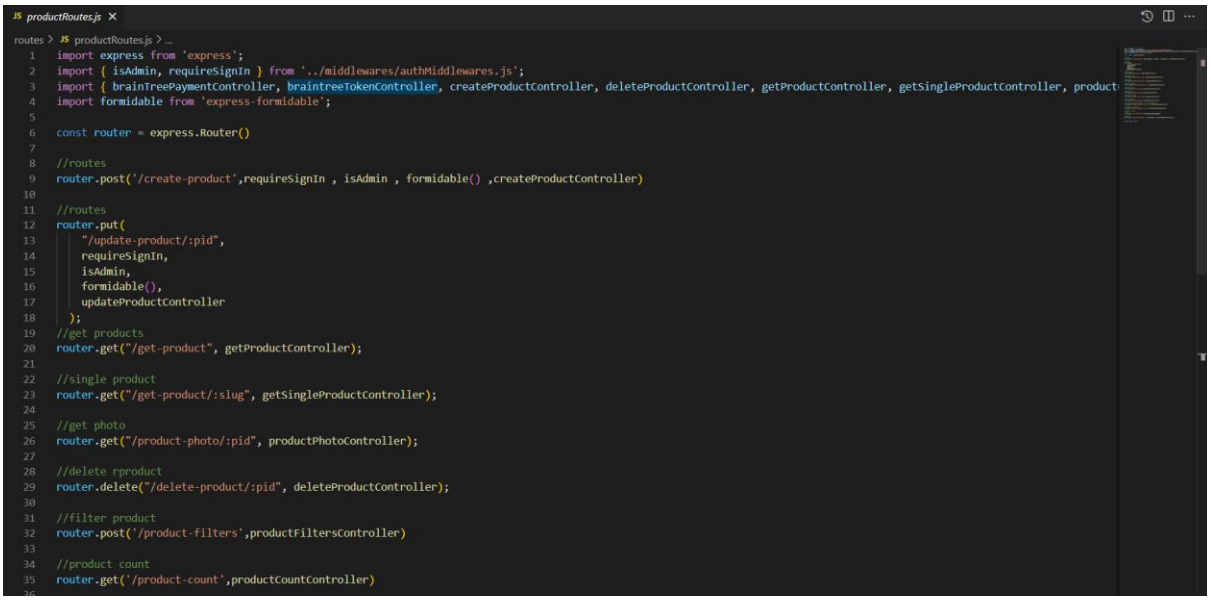


```
import express from 'express';
import {registerController,loginController,testController, forgetPasswordC
import { isAdmin, requireSignIn } from '../middlewares/authMiddlewares.js'
//router object
const router = express.Router()

//routing
//Register || Method POST
router.post('/register',registerController);

//Login || Method POST
router.post('/login',loginController)

//forget password
router.post('/forgot-password',forgetPasswordController)
//testroutes
router.get('/test',requireSignIn,isAdmin,testController)

// protected user route-auth
router.get('/user-auth',requireSignIn, (req,res)=>{
    res.status(200).send({ok:true})
})
// protected admin route auth
router.get('/admin-auth',requireSignIn,isAdmin, (req,res)=>{
    res.status(200).send({ok:true})
})

//update profile
router.put('/profile',requireSignIn,updateProfileController)

//orders
//orders
router.get("/orders", requireSignIn, getOrdersController);

// all orders

router.get('/all-orders',requireSignIn, isAdmin, getAllOrdersController)
```

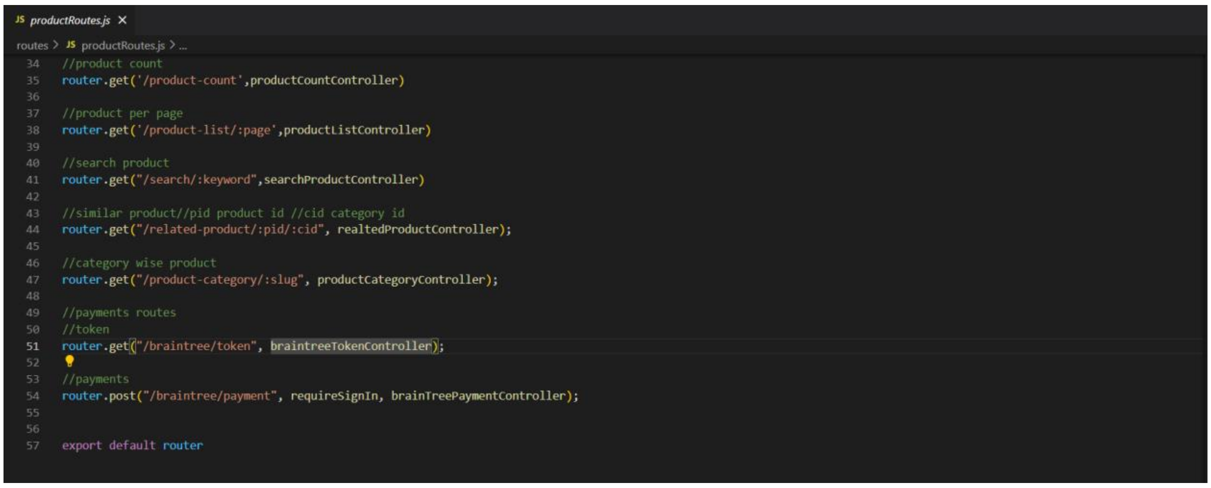


```
//order status update
router.put(
    "/order-status/:orderId",
    requireSignIn,
    isAdmin,
    orderStatusController
  );


export default  router;
```

- Product Management: CRUD operations for products

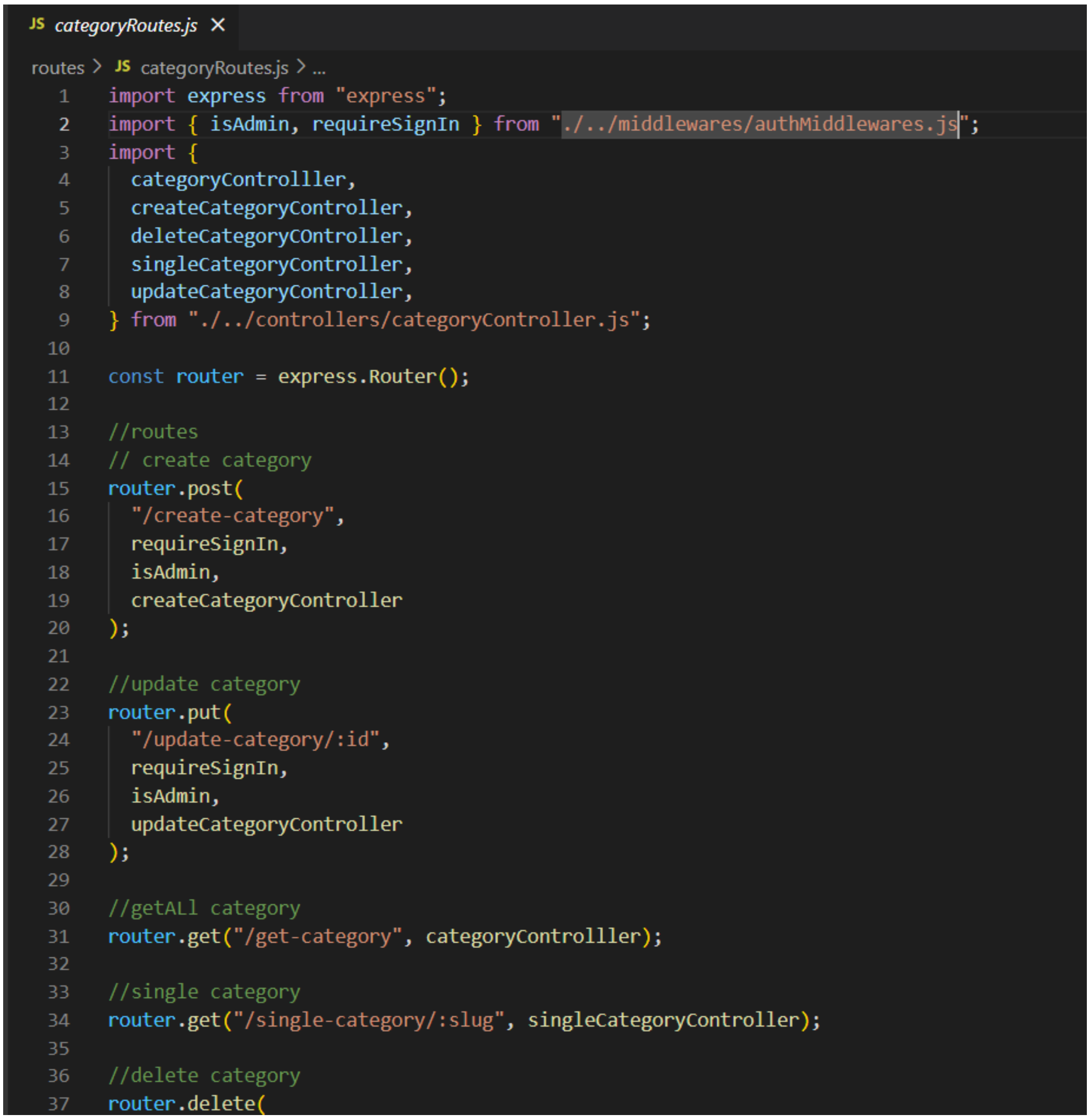


```
import express from 'express';
import { isAdmin, requireSignIn } from '../middlewares/authMiddlewares.js';
import { brainTreePaymentController, braintreeTokenController, createProductController, deleteProductController, getProductController, getSingleProductController, product
import formidable from 'express-formidable';

const router = express.Router()

//routes
router.post('/create-product',requireSignIn , isAdmin , formidable() ,createProductController)

//routes
router.put(
    "/update-product/:pid",
    requireSignIn,
    isAdmin,
    formidable(),
    updateProductController
  );
//get products
router.get("/get-product", getProductController);

//single product
router.get("/get-product/:slug", getSingleProductController);

//get photo
router.get("/product-photo/:pid", productPhotoController);

//delete rproduct
router.delete("/delete-product/:pid", deleteProductController);

//filter product
router.post('/product-filters',productFiltersController)

//product count
router.get('/product-count',productCountController)
```

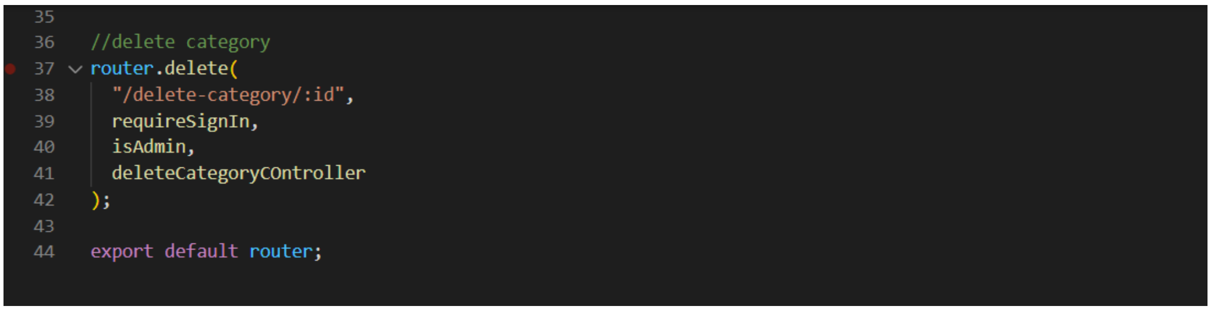


```
//product count
router.get('/product-count',productCountController)

//product per page
router.get('/product-list/:page',productListController)

//search product
router.get("/search/:keyword",searchProductController)

//similar product//pid product id //cid category id
router.get("/related-product/:pid/:cid", realtedProductController);

//category wise product
router.get("/product-category/:slug", productCategoryController);

//payments routes
//token
router.get("/braintree/token", braintreeTokenController);

//payments
router.post("/braintree/payment", requireSignIn, brainTreePaymentController);


export default router
```

- Categories Management: CRUD operations for categories



```js
import express from "express";
import { isAdmin, requireSignIn } from "./../middlewares/authMiddlewares.js";
import {
  categoryControlller,
  createCategoryController,
  deleteCategoryCOntroller,
  singleCategoryController,
  updateCategoryController,
} from "./../controllers/categoryController.js";

const router = express.Router();

//routes
// create category
router.post(
  "/create-category",
  requireSignIn,
  isAdmin,
  createCategoryController
);

//update category
router.put(
  "/update-category/:id",
  requireSignIn,
  isAdmin,
  updateCategoryController
);

//getALl category
router.get("/get-category", categoryControlller);

//single category
router.get("/single-category/:slug", singleCategoryController);

//delete category
router.delete(
```



```js

//delete category
router.delete(
  "/delete-category/:id",
  requireSignIn,
  isAdmin,
  deleteCategoryCOntroller
);

export default router;
```