This my project I created to be e-commerce app

Support me to rewrite this code with best Practices

App.router.js

import authRouter from './modules/auth/auth.router.js'

import userRouter from './modules/user/user.router.js'

import productRouter from './modules/product/product.router.js'

import categoryRouter from './modules/category/category.router.js'

import cartRouter from './modules/cart/cart.router.js'

import connectDB from '../DB/connection.js';

import { globalErrHandling } from './utils/errorHandling.js';

import OTPRouter from './modules/otp/otp.router.js'

import cors from 'cors'

const initApp = (app,express) => {

    // convert buffer data

    app.use(express.json({}));

    // app.use(cors())

    app.use(cors());

    //app Routing

    app.get('/', (req, res) => res.send('Hello World!'));

    app.use('/auth',authRouter)

    app.use('/otp',OTPRouter)

    app.use('/user',userRouter)

    app.use('/product',productRouter)

    app.use('/category',categoryRouter)

    app.use('/cart',cartRouter)

    app.all('\*', (req, res) => res.status(404).json({error:'Page not found!'}));

    // globalErrHandling

    app.use(globalErrHandling)

    // DB connection

    connectDB()

};

export default initApp;

modules:

* auth
* cart
* category
* order
* otp
* product
* review
* user

start with auth modules :

auth.router.js

import { Router } from "express";

import auth from "../../middleware/auth.middleware.js";

import {validation} from "../../middleware/validation.js";

import { loginSchema, signupSchema, updatePassword } from "./auth.validation.js";

import \* as authController from './controller/auth.controller.js'

const router = Router();

router.get('/',authController.getAuthModule)

router.post('/signup',authController.signup)

router.post('/login',validation(loginSchema), authController.login)

router.get('/confirmEmail/:token',authController.confirmEmail)

router.patch('/logOut',auth,authController.logOut)

router.patch('/password',validation(updatePassword),auth,authController.restPassword)

export default router

auth.validation.js

import joi from 'joi';

import { generalFields } from '../../middleware/validation.js';

export const signupSchema = joi.object({

        firstName:generalFields.name.required(),

        lastName:generalFields.name.required(),

        email:generalFields.email,

        password:generalFields.password,

        confirmPassword:generalFields.confirmPassword.valid(joi.ref('password')),

        age:joi.number().min(15).max(70).required()

    }).required()

export const loginSchema = joi.object({

        email:generalFields.email,

        password:generalFields.password

    }).required()

export const updatePassword = joi.object({

        oldPassword:generalFields.password,

        newPassword:generalFields.password.invalid(joi.ref('oldPassword')),

        confirmPassword:generalFields.confirmPassword.valid(joi.ref('newPassword'))

    }).required()

Auth.router.js

import userModel from "../../../../DB/models/User.model.js";

import { asyncHandler } from "../../../utils/errorHandling.js";

import sendEmail from "../../../utils/email.js";

import { generateToken, verifyToken } from "../../../utils/generateAndVerifyToken.js";

import { compare, hash } from "../../../utils/hashAndCompare.js";

export const getAuthModule = (req, res) => res.json({message:'Auth module'})

export const signup = asyncHandler(

        async(req, res,next) =>{

            const {userName,email,password,age} = req.body;

            if(!req.body) return next(new Error(`Data is required`,{cause:400}));

            const user = await userModel.findOne({email});

            console.log(user);

            if(user)return next(new Error('Email already  exist',{cause:409}));

            const hashPassword = hash({plaintext:password,});

            const token = generateToken({payload:{email},expiresIn:60\*5})

            const link = `http://localhost:5000/auth/confirmEmail/${token}`

            const html = `<a href=${link}>Click me to confirm your email</a>`

            if(! await sendEmail({to:email,subject:'confirm\_Email',html}))return next(new Error('Email rejected',{cause:400}))

            const createdUser = await userModel.create({

                userName,

                email,

                password:hashPassword,

                age

            });

            return res.status(201).json({message:'success',user:createdUser.\_id});

        }

    )

    export const login = asyncHandler(

        async(req, res,next) =>{

            const {email,password} = req.body;

            if(!(email && password)) return next(new Error('Data is required',{cause:400}));

            const user = await userModel.findOne({email});

            if(!user) return next(new Error('Email not exist',{cause:400}));

            if(!user.confirmEmail)return next(new Error('you have to confirm your email first',{cause: 400}))

            if(user.status === 'online') return next(new Error('In-valid email your are online',{cause:409}));

            const match = compare({plaintext:password,hashValue:user.password});

            if(!match) return next(new Error('In-valid Password',{cause:400}));

            const token = generateToken({

                payload:{id:user.id,isLoggedIn:true,role:user.role},

                expiresIn:60\*60\*24

            })

            user.status = 'online';

            await user.save();

            return res.status(200).json({message:'done',access\_token:token});

        }

    )

export const confirmEmail = asyncHandler(

    async(req, res,next) =>{

        const {token} = req.params

        const decoded = verifyToken({token})

        const user = await userModel.updateOne({email:decoded.email,confirmEmail:false},{confirmEmail:true})

        if(!user) return next(new Error('In\_valid account',{cause:404}));

        return res.status(200).redirect(`https://www.google.com/`)

    }

)

export const logOut = asyncHandler(async (req, res, next) => {

    const user = await userModel.findOneAndUpdate({\_id:req.user.id,status:'online'},{status:'offline'})

    if(!user) return next(new Error('In-valid user'));

    return res.status(200).json({message:'Done'});

});

export const restPassword = asyncHandler( async (req, res, next) =>{

    const {oldPassword,newPassword,confirmPassword} = req.body;

    console.log({oldPassword,newPassword,confirmPassword});

    const user = await userModel.findById(req.user.id);

    if(!compare({plaintext:oldPassword,hashValue:user.password})) return next(new Error('In-valid oldPassword',{cause:400}))

    const hashPassword = hash({plaintext:newPassword})

    user.password = hashPassword;

    await user.save();

    return res.status(200).json({message:'Done'})

});

User.model.js

import mongoose, { model, Schema } from "mongoose";

const userSchema =  new Schema({

    // firstName:{type:String,required:true},

    // lastName:{type:String,required:true},

    userName:{

        type:String,

        required:true

    },

    bio:String,

    profilePic:String,

    profilePicId:String,

    coverPic: [],

    address:{

        type:String,

        require:true

    },

    age:Number,

    email:{

        type:String,

        required:true,

        unique:true,

        lowercase:true,

        match:/^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$/

    },

    password:{

        type:String,

        required:true

    },

    phone:{

        type:String,

        require:true,

    },

    gender:{

        type:String,

        default:'male',

        enum:['male','female']

    },

    status:{

        type:String,

        default:'offline',

        enum:['offline','online','blocked']

    },

    role:{

        type:String,

        default:'customer',

        enum:['admin','customer']

    },

    confirmEmail:{

        type:Boolean,

        default:false,

    },

    softDelete:{

        type:Boolean,

        default:false

    },

},{ timestamps:true });

// ,{

//     toObject:{virtuals:true},

//     toJSON:{virtuals:true},

//     timestamps:true

// }

// );

// userSchema.virtual('post',{

//     ref:'Post',

//     localField:'\_id',

//     foreignField:'userId'

// })

const userModel = mongoose.models.User || model('User',userSchema)

export default userModel

and user model :

user.router.js

import { Router } from "express";

import auth from "../../middleware/auth.middleware.js";

import { authorization } from "../../middleware/authorization.middleware.js";

import { validation } from "../../middleware/validation.js";

import \* as userController from './controller/user.controller.js'

import { updateProfile } from "./user.validation.js";

import { fileUpload, fileValidation } from "../../utils/cloudMulter.js";

const router = Router();

router.get('/',auth,authorization,userController.getUserModule)

router.get('/profile',auth,userController.profile)

router.put('/profile/update',auth,validation(updateProfile),userController.updateProfile)

router.patch('/profile/softDelete',auth,userController.softDelete)

router.patch('/profile/unSoftDelete',auth,userController.unSoftDelete)

router.delete('/profile/delete',auth,userController.deleteUser)

router.patch('/profilePic',auth,fileUpload(fileValidation.image).single('image'),userController.profilePic)

router.patch('/profileCovPic',auth,fileUpload(fileValidation.image).array('image',5),userController.profileCovPic)

export default router

user.validation.js

import joi from 'joi';

import { generalFields } from '../../middleware/validation.js';

export const updateProfile = joi.object({

        firstName:generalFields.name.required(),

        lastName:generalFields.name.required(),

        phone:generalFields.phone.required()

    }).required()

User.controller.js

import userModel from "../../../../DB/models/User.model.js"

import cloudinary from "../../../utils/cloudinary.js"

import { asyncHandler } from "../../../utils/errorHandling.js"

export const getUserModule = async(req, res) =>{

    const users = await userModel.find().select('\_id userName image ')

    return res.json({message:'User module',users})

}

export const profile = asyncHandler(async(req, res,next) =>{

    const user = await userModel.findById(req.user.\_id).select('\_id timestamps firstName lastName phone email confirmEmail status').populate([

        {

            path:'post'

        },

    ]);

    return res.json({message:'Done',user});

})

export const updateProfile = asyncHandler(async(req,res,next)=>{

    // const {userName, bio, } = req.body;

    const user = await userModel.findByIdAndUpdate({\_id:req.user.\_id},req.body);

    if(!user)return next(new Error('In-valid user',{cause:400}))

    return res.json({message:'Done',user});

})

export const softDelete = asyncHandler(async(req,res,next)=>{

    const user = await userModel.findByIdAndUpdate({\_id:req.user.\_id},{softDelete:true},{new:true});

    if(!user)return next(new Error('In-valid user',{cause:400}))

    return res.json({message:'Done',user});

})

export const unSoftDelete = asyncHandler(async(req,res,next)=>{

    const user = await userModel.findByIdAndUpdate({\_id:req.user.\_id},{softDelete:false},{new:true});

    if(!user)return next(new Error('In-valid user',{cause:400}))

    return res.json({message:'Done',user});

})

export const deleteUser = asyncHandler(async(req,res,next)=>{

    const user = await userModel.findByIdAndDelete({\_id:req.user.\_id})

    return res.json({message:'Done',user});

})

// User pic

export const profilePic = asyncHandler(async(req,res,next) =>{

    console.log(req.file);

    const {secure\_url,public\_id} = await cloudinary.uploader.upload(req.file.path,{folder:`user/${req.user.\_id}/profile`})

    console.log({secure\_url,public\_id});

    const user = await userModel.findByIdAndUpdate(req.user.\_id,{profilePic:secure\_url,profilePicId:public\_id},{new:false})

    await cloudinary.uploader.destroy(user.profilePicId)

    return res.json({message:'Done',user})

});

export const profileCovPic = asyncHandler(async(req,res,next) =>{

    console.log(req.files);

    console.log(req.user.\_id);

        const coverPic = []

    for (const image of req.files) {

        if(!(image.size < 52428800 )) return next(new Error('it is big size file',{cause:400}));

        const {secure\_url,public\_id} = await cloudinary.uploader.upload(image.path,{folder:`user/${req.user.\_id}/profile/cover`})

        console.log({secure\_url,public\_id});

        coverPic.push({secure\_url,public\_id})

    }

    console.log(coverPic);

    const user =await userModel.findByIdAndUpdate(req.user.\_id,{coverPic},{new:true})

    return res.json({message:'Done',user})

});

And his model . user.model.js

And product moduels :-

Product.router.js

import { Router } from "express";

import auth from "../../middleware/auth.middleware.js";

import \* as productController from './controller/product.controller.js'

import { fileUpload, fileValidation } from "../../utils/cloudMulter.js";

import { validation } from "../../middleware/validation.js";

import { add,checkId, update } from "./product.validation.js";

const router = Router();

router.get('/', productController.getAllProducts)

//==================================== products====================================

//---------------------------owner-------------------------------------

router.get('/myProducts',auth,productController.getAllProducts)

router.product('/add',auth,fileUpload(fileValidation.image).single('image'),validation(add),productController.newProduct);

router.put('/:productId/',auth,validation(update),productController.updateProduct)

router.delete('/:productId/',auth,validation(checkId),productController.deleteproduct)

//---------------------------visitor-------------------------------------

router.patch('/:productId/like',auth,validation(checkId),productController.likeProduct)

router.patch('/:productId/unLike',auth,validation(checkId),productController.unLikeProduct)

export default router

products.validation.js

import Joi from "joi";

import { generalFields } from "../../middleware/validation.js";

export const add = Joi.object({

    title:generalFields.title.required(),

    caption:generalFields.caption.required(),

    file:generalFields.file.required()

}).required();

export const checkId = Joi.object({

    postId:generalFields.id.required()

}).required();

export const update = Joi.object({

    id:generalFields.id.required(),

    title:generalFields.title,

    caption:generalFields.caption,

    file:generalFields.file

});

//============================ comment ====================================

export const createComment = Joi.object({

    postId:generalFields.id.required(),

    text:generalFields.text.required(),

    file:generalFields.file

});

Product.controller.js

// import moment from "moment";

import productModel from "../../../../DB/models/product.model.js";

import cloudinary from "../../../utils/cloudinary.js";

import { asyncHandler } from "../../../utils/errorHandling.js";

// import { nanoid } from "nanoid";

export const getAllProducts = asyncHandler(async(req,res,next)=>{

    const products = await productModel.find().populate([

        {

            path:'userId',

            select:'userName profilePic'

        },

        {

            path:'like',

            select:'userName profilePic'

        },

        {

            path:'unlike',

            select:'userName profilePic'

        }

    ])

    return res.status(200).json({message:'Done',products})

})

// ----------------  my crud of my product------------------------------------------------

export const getUProducts = asyncHandler(async(req,res,next)=>{

    const products = await productModel.find({userId:req.user.\_id});

    if(!products) return next(new Error('In-valid y',{cause:404}));

    return res.status(200).json({message:'Done',products})

});

export const newProduct = asyncHandler(async(req,res,next)=>{

    const {name, description, category, image, price, stock, like, unLike, isDelete, totalVote, status} = req.body

    if(!req.body && !req.file)return next(new Error('In-valid data',{cause:400}));

    const {public\_id,secure\_url} = await cloudinary.uploader.upload(req.file.path,{folder:`product/${req.user.\_id}`})

    const product = await productModel.create({name, description, category, price, stock, like, unLike, isDelete, totalVote, status, userId:req.user.\_id,image:{public\_id,secure\_url}})

    return res.status(201).json({message:'Done',product})

});

export const updateProduct = asyncHandler(async(req,res,next)=>{

    const {id} = req.params;

    const product = await productModel.findById(id)

    if(!product) next(new Error('product not found',{cause:404}));

    if(req.file){

        const {public\_id,secure\_url} = await cloudinary.uploader.upload(req.file.path,{folder:`product/${req.user.\_id}`})

        const updatePicProduct = await productModel.findByIdAndUpdate({userId:req.user.\_id},{image:{public\_id,secure\_url}},{new:false})

        await cloudinary.uploader.destroy(product.image.public\_id)

    }

    const updateProduct = await productModel.findByIdAndUpdate({userId:req.user.\_id},req.body)

    return res.status(200).json({message:'Done',updateProduct})

});

export const deleteProduct = asyncHandler(async (req,res,next) => {

    const {id} = req.params;

    const deleteProduct = await productModel.findOneAndDelete({\_id:id,userId:req.user.\_id},{new:false})

    await cloudinary.uploader.destroy(book.image.public\_id)

    return deleteBook ? res.status(200).json({message:'done', deleteProduct}):next(new Error('In-valid product',{cause:404}))

})

//---------------------------- visitor  CURD------------------------------------

export const likeProduct = asyncHandler(async(req, res, next)=>{

    const {productId} = req.params;

    const product = await productModel.findByIdAndUpdate(productId,{ $addToSet: { like: req.user.\_id},$pull:{unlike:req.user.\_id}},{new:true});

    if(!product) next(new Error('product not found',{cause:404}));

    product.totalVote = product.like.length - product.unlike.length

    await product.save();

    return res.status(200).json({message:'Done',product});

})

export const unLikeProduct = asyncHandler(async(req, res, next)=>{

    const {productId} = req.params;

    const product = await productModel.findByIdAndUpdate(productId,{ $addToSet: { unlike: req.user.\_id},$pull:{like: req.user.\_id}},{new:true})

    if(!product) next(new Error('product not found',{cause:404}));

    product.totalVote = product.like.length - product.unlike.length

    await product.save();

    return res.status(200).json({message:'Done',product});

})

Product.model.js

import mongoose, { model, Schema, Types } from "mongoose";

const productSchema =  new Schema({

    name:{type:String,required:true},

    description:String,

    category: { type: Types.ObjectId, ref: 'Category' },

    image:[{type: Object,required:true}],

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

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

    userId:{type: Types.ObjectId,ref:'User',required:true},

    like:[{type: Types.ObjectId,ref:'User'}],

    unlike:[{type: Types.ObjectId,ref:'User'}],

    isDelete:{type: Boolean,default:false},

    totalVote:{type: Number,default:0},

    status:{type: String,default:'public',enum:['public','private']}

},{

    toObject:{virtuals:true},

    toJSON:{virtuals:true},

    timestamps:true

});

// productSchema.virtual('Cat',{

//     ref:'Cat',

//     localField:'\_id',

//     foreignField:'productId'

// })

const productModel = mongoose.models.Product || model('Product',productSchema)

export default productModel

And cart moduels :-

cart.router.js

import { Router } from "express";

import auth from "../../middleware/auth.middleware.js";

import \* as cartController from './controller/cart.controller.js'

// import commentRouter from '../comment/comment.router.js'

import { fileUpload, fileValidation } from "../../utils/cloudMulter.js";

import { validation } from "../../middleware/validation.js";

import { add, checkId, update } from "./cart.validation.js";

const router = Router();

// router.use('/:cartId/comment/', commentRouter)

router.get('/', cartController.getAllCarts)

//==================================== categories====================================

//---------------------------owner-------------------------------------

router.get('myCart',auth,cartController.getAllCarts)

router.cart('/add',auth,fileUpload(fileValidation.image).single('image'),validation(add),cartController.newCart);

router.put('/:cartId/',auth,validation(update),cartController.updateCart)

router.delete('/:cartId/',auth,validation(checkId),cartController.deleteCart)

export default router

cart.validation.js

import Joi from "joi";

import { generalFields } from "../../middleware/validation.js";

export const add = Joi.object({

    title:generalFields.title.required(),

    caption:generalFields.caption.required(),

    file:generalFields.file.required()

}).required();

export const checkId = Joi.object({

    postId:generalFields.id.required()

}).required();

export const update = Joi.object({

    id:generalFields.id.required(),

    title:generalFields.title,

    caption:generalFields.caption,

    file:generalFields.file

});

//============================ comment ====================================

export const createComment = Joi.object({

    postId:generalFields.id.required(),

    text:generalFields.text.required(),

    file:generalFields.file

});

Cart.controller.js

import cartModel from "../../../../DB/models/Cart.model.js";

import { asyncHandler } from "../../../utils/errorHandling.js";

export const getAllCarts = asyncHandler(async(req,res,next)=>{

    const categories = await cartModel.find()

    // .populate([

    //     {

    //         path:'userId',

    //         select:'userName profilePic'

    //     },

    //     {

    //         path:'like',

    //         select:'userName profilePic'

    //     },

    //     {

    //         path:'unlike',

    //         select:'userName profilePic'

    //     }

    // ])

    return res.status(200).json({message:'Done',categories: categories})

})

// ----------------  my crud of my cart------------------------------------------------

export const getUCategories = asyncHandler(async(req,res,next)=>{

    const categories = await cartModel.find({userId:req.user.\_id});

    if(!categories) return next(new Error('In-valid y',{cause:404}));

    return res.status(200).json({message:'Done',categories: categories})

});

export const newCart = asyncHandler(async(req,res,next)=>{

    const { name, description } = req.body

    if(!req.body)return next(new Error('In-valid data',{cause:400}));

    const cart = await cartModel.create({name, description})

    return res.status(201).json({message:'Done',cart})

});

export const updateCart = asyncHandler(async(req,res,next)=>{

    const {id} = req.params;

    const cart = await cartModel.findById(id)

    if(!cart) next(new Error('cart not found',{cause:404}));

    const updateCart = await cartModel.findByIdAndUpdate({userId:req.user.\_id},req.body)

    return res.status(200).json({message:'Done',updateCart})

});

export const deleteCart = asyncHandler(async (req,res,next) => {

    const {id} = req.params;

    const deleteCart = await cartModel.findOneAndDelete({\_id:id,userId:req.user.\_id},{new:false})

    return deleteBook ? res.status(200).json({message:'done', deleteCart}):next(new Error('In-valid cart',{cause:404}))

})

Cart.model.js

import mongoose, { model, Schema, Types } from "mongoose";

const cartSchema = new mongoose.Schema({

    user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },

    items: [{

        productId: { type: mongoose.Schema.Types.ObjectId, ref: 'Product' },

        quantity: Number

    }],

}, { timestamps: true });

const cartModel = mongoose.models.Cart || model('Cart',cartSchema)

export default cartModel

And category moduels :-

category.router.js

import { Router } from "express";

import auth from "../../middleware/auth.middleware.js";

import \* as categoryController from './controller/category.controller.js'

// import commentRouter from '../comment/comment.router.js'

import { fileUpload, fileValidation } from "../../utils/cloudMulter.js";

import { validation } from "../../middleware/validation.js";

import { add,checkId, update } from "./category.validation.js";

const router = Router();

// router.use('/:categoryId/comment/', commentRouter)

router.get('/', categoryController.getAllCategories)

//==================================== categories====================================

//---------------------------owner-------------------------------------

router.get('myCategories',auth,categoryController.getAllCategories)

router.category('/add',auth,fileUpload(fileValidation.image).single('image'),validation(add),categoryController.newCategory);

router.put('/:categoryId/',auth,validation(update),categoryController.updateCategory)

router.delete('/:categoryId/',auth,validation(checkId),categoryController.deleteCategory)

export default router

category.validation.js

import Joi from "joi";

import { generalFields } from "../../middleware/validation.js";

export const add = Joi.object({

    title:generalFields.title.required(),

    caption:generalFields.caption.required(),

    file:generalFields.file.required()

}).required();

export const checkId = Joi.object({

    postId:generalFields.id.required()

}).required();

export const update = Joi.object({

    id:generalFields.id.required(),

    title:generalFields.title,

    caption:generalFields.caption,

    file:generalFields.file

});

//============================ comment ====================================

export const createComment = Joi.object({

    postId:generalFields.id.required(),

    text:generalFields.text.required(),

    file:generalFields.file

});

category.controller.js

import categoryModel from "../../../../DB/models/Category.model.js";

import { asyncHandler } from "../../../utils/errorHandling.js";

export const getAllCategories = asyncHandler(async(req,res,next)=>{

    const categories = await categoryModel.find()

    // .populate([

    //     {

    //         path:'userId',

    //         select:'userName profilePic'

    //     },

    //     {

    //         path:'like',

    //         select:'userName profilePic'

    //     },

    //     {

    //         path:'unlike',

    //         select:'userName profilePic'

    //     }

    // ])

    return res.status(200).json({message:'Done',categories: categories})

})

// ----------------  my crud of my category------------------------------------------------

export const getUCategories = asyncHandler(async(req,res,next)=>{

    const categories = await categoryModel.find({userId:req.user.\_id});

    if(!categories) return next(new Error('In-valid y',{cause:404}));

    return res.status(200).json({message:'Done',categories: categories})

});

export const newCategory = asyncHandler(async(req,res,next)=>{

    const { name, description } = req.body

    if(!req.body)return next(new Error('In-valid data',{cause:400}));

    const category = await categoryModel.create({name, description})

    return res.status(201).json({message:'Done',category})

});

export const updateCategory = asyncHandler(async(req,res,next)=>{

    const {id} = req.params;

    const category = await categoryModel.findById(id)

    if(!category) next(new Error('category not found',{cause:404}));

    const updateCategory = await categoryModel.findByIdAndUpdate({userId:req.user.\_id},req.body)

    return res.status(200).json({message:'Done',updateCategory})

});

export const deleteCategory = asyncHandler(async (req,res,next) => {

    const {id} = req.params;

    const deleteCategory = await categoryModel.findOneAndDelete({\_id:id,userId:req.user.\_id},{new:false})

    return deleteBook ? res.status(200).json({message:'done', deleteCategory}):next(new Error('In-valid category',{cause:404}))

})

Category.model.js

import mongoose, { model, Schema, Types } from "mongoose";

const categorySchema =  new Schema({

    name:{type:String,required:true},

    description: String,

},{ timestamps:true });

const categoryModel = mongoose.models.Category || model('Category',categorySchema)

export default categoryModel

And order moduels :-

order.router.js

import { Router } from "express";

import auth from "../../middleware/auth.middleware.js";

import \* as orderController from './controller/order.controller.js'

// import commentRouter from '../comment/comment.router.js'

import { fileUpload, fileValidation } from "../../utils/cloudMulter.js";

import { validation } from "../../middleware/validation.js";

import { add, checkId, update } from "./order.validation.js";

const router = Router();

// router.use('/:orderId/comment/', commentRouter)

router.get('/', orderController.getAllOrders)

//==================================== categories====================================

//---------------------------owner-------------------------------------

router.get('myOrder',auth,orderController.getAllOrders)

router.order('/add',auth,fileUpload(fileValidation.image).single('image'),validation(add),orderController.newOrder);

router.put('/:orderId/',auth,validation(update),orderController.updateOrder)

router.delete('/:orderId/',auth,validation(checkId),orderController.deleteOrder)

export default router

order.validation.js

import Joi from "joi";

import { generalFields } from "../../middleware/validation.js";

export const add = Joi.object({

    title:generalFields.title.required(),

    caption:generalFields.caption.required(),

    file:generalFields.file.required()

}).required();

export const checkId = Joi.object({

    postId:generalFields.id.required()

}).required();

export const update = Joi.object({

    id:generalFields.id.required(),

    title:generalFields.title,

    caption:generalFields.caption,

    file:generalFields.file

});

//============================ comment ====================================

export const createComment = Joi.object({

    postId:generalFields.id.required(),

    text:generalFields.text.required(),

    file:generalFields.file

});

order.controller.js

import orderModel from "../../../../DB/models/order.model.js";

import { asyncHandler } from "../../../utils/errorHandling.js";

export const getAllOrders = asyncHandler(async(req,res,next)=>{

    const categories = await orderModel.find()

    // .populate([

    //     {

    //         path:'userId',

    //         select:'userName profilePic'

    //     },

    //     {

    //         path:'like',

    //         select:'userName profilePic'

    //     },

    //     {

    //         path:'unlike',

    //         select:'userName profilePic'

    //     }

    // ])

    return res.status(200).json({message:'Done',categories: categories})

})

// ----------------  my crud of my order------------------------------------------------

export const getUCategories = asyncHandler(async(req,res,next)=>{

    const categories = await orderModel.find({userId:req.user.\_id});

    if(!categories) return next(new Error('In-valid y',{cause:404}));

    return res.status(200).json({message:'Done',categories: categories})

});

export const newOrder = asyncHandler(async(req,res,next)=>{

    const { name, description } = req.body

    if(!req.body)return next(new Error('In-valid data',{cause:400}));

    const order = await orderModel.create({name, description})

    return res.status(201).json({message:'Done',order})

});

export const updateOrder = asyncHandler(async(req,res,next)=>{

    const {id} = req.params;

    const order = await orderModel.findById(id)

    if(!order) next(new Error('order not found',{cause:404}));

    const updateOrder = await orderModel.findByIdAndUpdate({userId:req.user.\_id},req.body)

    return res.status(200).json({message:'Done',updateOrder})

});

export const deleteOrder = asyncHandler(async (req,res,next) => {

    const {id} = req.params;

    const deleteOrder = await orderModel.findOneAndDelete({\_id:id,userId:req.user.\_id},{new:false})

    return deleteBook ? res.status(200).json({message:'done', deleteOrder}):next(new Error('In-valid order',{cause:404}))

})

Order.model.js

import mongoose, { model, Schema, Types } from "mongoose";

const orderSchema = new mongoose.Schema({

    user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },

    products: [{

        productId: { type: mongoose.Schema.Types.ObjectId, ref: 'Product' },

        quantity: Number

    }],

    totalAmount: Number,

    status: String,

}, { timestamps: true });

const orderModel = mongoose.models.Order || model('Order',orderSchema)

export default orderModel

And review moduels :-

review.router.js

import { Router } from "express";

import auth from "../../middleware/auth.middleware.js";

import \* as cartController from './controller/cart.controller.js'

import commentRouter from '../comment/comment.router.js'

import { fileUpload, fileValidation } from "../../utils/cloudMulter.js";

import { validation } from "../../middleware/validation.js";

import { add, checkId, update } from "./cart.validation.js";

const router = Router();

router.use('/:cartId/comment/', commentRouter)

router.get('/', cartController.getAllCarts)

//==================================== categories====================================

//---------------------------owner-------------------------------------

router.get('myCart',auth,cartController.getAllCarts)

router.cart('/add',auth,fileUpload(fileValidation.image).single('image'),validation(add),cartController.newCart);

router.put('/:cartId/',auth,validation(update),cartController.updateCart)

router.delete('/:cartId/',auth,validation(checkId),cartController.deleteCart)

export default router

review.validation.js

import Joi from "joi";

import { generalFields } from "../../middleware/validation.js";

export const add = Joi.object({

    title:generalFields.title.required(),

    caption:generalFields.caption.required(),

    file:generalFields.file.required()

}).required();

export const checkId = Joi.object({

    postId:generalFields.id.required()

}).required();

export const update = Joi.object({

    id:generalFields.id.required(),

    title:generalFields.title,

    caption:generalFields.caption,

    file:generalFields.file

});

//============================ comment ====================================

export const createComment = Joi.object({

    postId:generalFields.id.required(),

    text:generalFields.text.required(),

    file:generalFields.file

});

review.controller.js

import cartModel from "../../../../DB/models/Cart.model.js";

import { asyncHandler } from "../../../utils/errorHandling.js";

export const getAllCarts = asyncHandler(async(req,res,next)=>{

    const categories = await cartModel.find()

    // .populate([

    //     {

    //         path:'userId',

    //         select:'userName profilePic'

    //     },

    //     {

    //         path:'like',

    //         select:'userName profilePic'

    //     },

    //     {

    //         path:'unlike',

    //         select:'userName profilePic'

    //     }

    // ])

    return res.status(200).json({message:'Done',categories: categories})

})

// ----------------  my crud of my cart------------------------------------------------

export const getUCategories = asyncHandler(async(req,res,next)=>{

    const categories = await cartModel.find({userId:req.user.\_id});

    if(!categories) return next(new Error('In-valid y',{cause:404}));

    return res.status(200).json({message:'Done',categories: categories})

});

export const newCart = asyncHandler(async(req,res,next)=>{

    const { name, description } = req.body

    if(!req.body)return next(new Error('In-valid data',{cause:400}));

    const cart = await cartModel.create({name, description})

    return res.status(201).json({message:'Done',cart})

});

export const updateCart = asyncHandler(async(req,res,next)=>{

    const {id} = req.params;

    const cart = await cartModel.findById(id)

    if(!cart) next(new Error('cart not found',{cause:404}));

    const updateCart = await cartModel.findByIdAndUpdate({userId:req.user.\_id},req.body)

    return res.status(200).json({message:'Done',updateCart})

});

export const deleteCart = asyncHandler(async (req,res,next) => {

    const {id} = req.params;

    const deleteCart = await cartModel.findOneAndDelete({\_id:id,userId:req.user.\_id},{new:false})

    return deleteBook ? res.status(200).json({message:'done', deleteCart}):next(new Error('In-valid cart',{cause:404}))

})

Review.model.js

import mongoose, { model, Schema, Types } from "mongoose";

const reviewSchema = new mongoose.Schema({

    user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },

    product: { type: mongoose.Schema.Types.ObjectId, ref: 'Product' },

    rating: Number,

    comment: String,

}, { timestamps: true });

const reviewModel = mongoose.models.Review || model('Review',reviewSchema)

export default reviewModel

auth.middleware.js

import userModel from "../../DB/models/User.model.js";

import { asyncHandler } from "../utils/errorHandling.js";

import { verifyToken } from "../utils/generateAndVerifyToken.js";

const auth = asyncHandler(async(req,res,next)=>{

    const{authorization} = req.headers;

    if(!authorization?.startsWith(process.env.BEARER\_KEY))return next(new Error('In-valid authorization or bearer token',{cause:'500'}))

    const token = authorization.split(process.env.BEARER\_KEY)[1];

    if(!token)return next(new Error('Token is required',{cause:'403'}))

    const decoded = verifyToken({token});

    if(!decoded?.id)return next(new Error('In\_valid token payload',{cause:'500'}))

    const authUser = await userModel.findById(decoded.id).select('userName email role status');

    if(!authUser)return next(new Error('Not register account',{cause:'500'}))

    req.user = authUser;

    return next();

})

export default auth;

authorization.middleware.js

import { asyncHandler } from "../utils/errorHandling.js";

export const authorization = asyncHandler(async(req,res,next)=>{

    // const authUser = await userModel.findById(req.user.\_id).select('userName email role status');

    if(!req.user.role === 'admin') return next(new Error('Not authentication',{cause:400}))

    return next();

})

validation.js

import joi from 'joi';

import { Types } from 'mongoose';

const dataMethods= ['body','query','params']

const validateObjectId = (value,helper) =>{

    return Types.ObjectId.isValid(value)? true :helper.message('In-valid objectId')

}

export const generalFields = {

    id:joi.string().custom(validateObjectId),

    name:joi.string().min(2).max(50),

    email:joi.string().email({minDomainSegments:2,maxDomainSegments:4,tlds:{allow:['com','net']}}),

    password:joi.string()/\*.pattern(new RegExp(/^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[a-zA-Z]).{8,}$/))\*/,

    confirmPassword:joi.string(),

    age:joi.number().min(15).max(70),

    phone:joi.string().min(11).max(11),

    //-------------------------book fields---------------------------------------

    title:joi.string().min(2).max(500),

    caption:joi.string().min(2).max(500),

    file:joi.object(),

    text:joi.string()

}

export const validation = (schema) => {

    return (req, res,next) => {

        const inputs = {...req.body,...req.query,...req.params}

        console.log(inputs ,'in validate');

        // if(req.headers?.authorization){inputs.authorization = req.headers.authorization}

        if(req.file || req.files){inputs.file = req.file || req.files}

        const validationResult = schema.validate(inputs,{abortEarly:false})

        console.log(validationResult.error , 'befor error validation');

        if(validationResult.error)return res.status(400).json({message: 'Validation error',validationResult:validationResult.error.details})

        console.log('done validation');

        return next();

    }

}

// export const validation =(schema)=>{

//     return (req, res,next) => {

//         const validationErr = []

//         dataMethods.forEach(key => {

//             if(schema[key]){

//                 const validationResult = schema[key].validate(req[key],{abortEarly:false});

//                 if(validationResult.error)validationErr.push(validationResult.error.details);

//             }

//         });

//         if(validationErr.length > 0)return res.json({message:'validation Error',validationErr});

//         return next();

//     }

// };

1. rewrite code by best Practices
2. rewrite code of
   1. auth.validation.js
   2. user.validation.js
   3. product.validation.js
   4. order.validation.js
   5. cart.validation.js
   6. category.validation.js
   7. review.validation.js

and I have this error and don’t not why this error please tell what and how to solve

Node.js v20.11.0

[nodemon] app crashed - waiting for file changes before starting...

[nodemon] restarting due to changes...

[nodemon] starting `node index.js`

file:///D:/Ruot%20c39/backEnd/assignmets/suplira/be/src/modules/product/product.router.js:15

router.product('/add',auth,fileUpload(fileValidation.image).single('image'),validation(add),productController.newProduct);

^

TypeError: router.product is not a function

at file:///D:/Ruot%20c39/backEnd/assignmets/suplira/be/src/modules/product/product.router.js:15:8

at ModuleJob.run (node:internal/modules/esm/module\_job:218:25)

at async ModuleLoader.import (node:internal/modules/esm/loader:329:24)

at async loadESM (node:internal/process/esm\_loader:28:7)

at async handleMainPromise (node:internal/modules/run\_main:113:12)

Node.js v20.11.0

[nodemon] app crashed - waiting for file changes before starting...

[nodemon] restarting due to changes...

[nodemon] starting `node index.js`

file:///D:/Ruot%20c39/backEnd/assignmets/suplira/be/src/modules/product/product.router.js:15

router.product('/add',auth,fileUpload(fileValidation.image).single('image'),validation(add),productController.newProduct);

^

TypeError: router.product is not a function

at file:///D:/Ruot%20c39/backEnd/assignmets/suplira/be/src/modules/product/product.router.js:15:8

at ModuleJob.run (node:internal/modules/esm/module\_job:218:25)

at async ModuleLoader.import (node:internal/modules/esm/loader:329:24)

at async loadESM (node:internal/process/esm\_loader:28:7)

at async handleMainPromise (node:internal/modules/run\_main:113:12)

Node.js v20.11.0

[nodemon] app crashed - waiting for file changes before starting...