packages

"bcrypt": "^6.0.0",

"body-parser": "^2.2.0",

"cors": "^2.8.5",

"dotenv": "^16.6.0",

"express": "^5.1.0",

"jsonwebtoken": "^9.0.2",

"mongodb": "^6.17.0",

"multer": "^2.0.1",

"swagger-ui-express": "^5.0.1",

"urlencode": "^2.0.0",

"winston": "^3.17.0"

Data Model

you have 4 apis

server.use('/api/products',jwtAuth,productRouter);

server.use('/api/cartItems',jwtAuth,cartRouter);

server.use('/api/orders',jwtAuth,orderRouter);

server.use('/api/users',userRouter)

User Table

{

"\_id": {

"$oid": "685c13d424a3eab5d46e7e07"

},

"name": "John Doe",

"email": "john.doe@gmail.com",

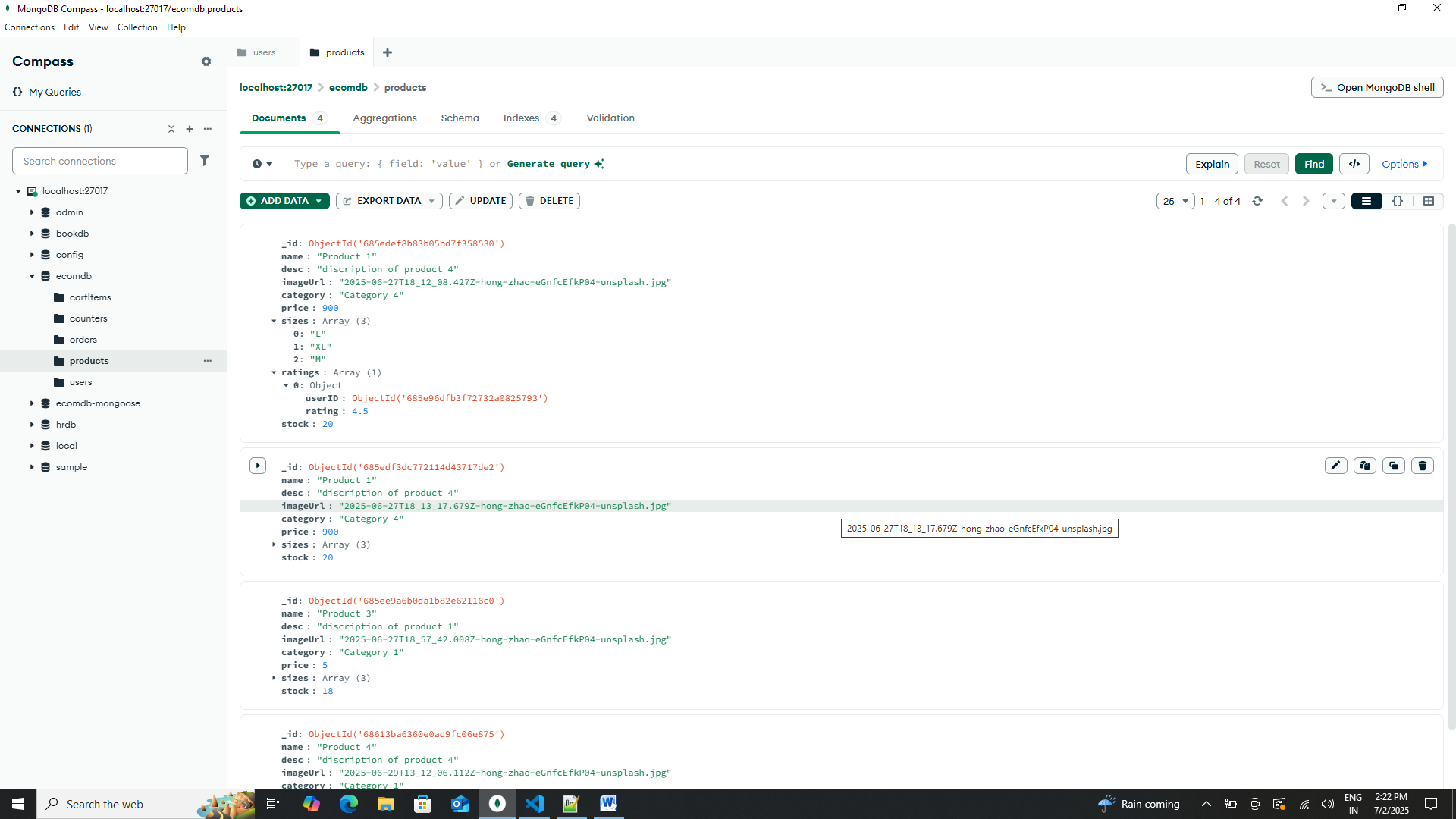
"password": "Password1",

"type": "Customer",

"id": null

}

Product Table



CartItems

**\_id**: 1

**productID**: ObjectId('685edf3dc772114d43717de2')

**userID**: ObjectId('685e96dfb3f72732a0825793')

**quantity**: 4

Orders

**\_id**: ObjectId('686277e348aea360bc9902b1')

**userId** : ObjectId('68615c92ad9fd1017c61df73')

**totalAmount** : 2510

**timestamp**: 2025-06-30T11:41:23.604+00:00

Counters

**\_id** : "cartItemId"

**Value** : 5

Codeing steps

User signup

Take the name emil etc from req.body

Hash the password

Create a object with it and then send that object to repository

And save that in db

User sign in

Get the details from database by using request email and then you will get the object of user

If user is not there send the message incorrect credentials or if user is found then compare the password

If password is matched create a token with userid and email and send the token

Add a Product

Get all the details from req.body

Make a object of it and use split with sizes

Send the object to data base and save it

Get One product

Get id from params

Send id to database and get the record

Get all products

Get all products from database and get the record

Filter Products

Get minprice , maxprice and categories

Send this to repository function in repository create one empty object

Use or operator functionality also

{$or:[{category:{$in:categories}},filterExpression]};

Or take an array and which is true that object will return

After creating object the put that object in find

And use project function also and toArray

Add rating in product object

Get user id , productid , rating

Put these values in repository function

So here are two ways we will try those two ways also

First way

Get the product by giving product id and check that product has the rating of that user

If user does not have the ratings array we will add the array by using push operator

Update the product with product id and add the array

If there is already a value of same user id

Then update that rating

Second way

Add the array by uising push and to update just pull the array element and then again add that array by using push

Average Price

Directly go to repository function

Use aggregate pipline grup by category to show fields id and average price

Cart Items

1. Add the cart item

Take user id and product id and quantity

Use update One operator with upsert match with productid and user id if matching increment the quantity else add id with custom value and then increment the quantity

1. Get Cart Items

Get user id and pass that user id to repository and use that id and return the documents

1. Delete Cart Items

Get user id and cart item id send that in repository function

And use deleteone function and if deletedcount is greater that 0 then send the response of this comparission

Order

Get the user id and send that user id in repository function and get the total amount

And after that take userid totalmount and timestamp and insert in orders table

Now reduce the stock of product

Then delete the cart

Get the client

const client = getClient();

            const session = client.startSession();

session.startTransaction();

 await db.collection(this.collection).insertOne(newOrder,{session});

await session.commitTransaction();

            session.endSession();

await session.abortTransaction();

            session.endSession();

([

            // 1. Get Cart items for the user

            {

                $match:{userID : new ObjectId(userId)}

            },

            // 2. Get the products form products collection.

            {

                $lookup:{

                    from:"products",

                    localField:"productID",

                    foreignField:"\_id",

                    as : "productInfo"

                }

            },

            // 3. Unwind the productinfo

            {

                $unwind:"$productInfo"

            },

            // 4. Calculate totalAmount for each cartitems.

            {

                $addFields:{

                    "totalAmount":{

                        $multiply:["$productInfo.price", "$quantity"]

                    }

                }

            }

        ],{session}).toArray();