Project Name : Swap Shop

Date : 21 July 2022

Tutorial Video : <https://www.youtube.com/watch?v=ldGl6L4Vktk&list=PLj-4DlPRT48lXaz5YLvbLC38m25W9Kmqy&index=1>

**PHASE I : REST API with MongoDb**

**\***Creating package.json -> npm init

Libraries Installation

\*npm add express mongoose dotenv helmet morgan nodemon

- mongoose documentation : <https://mongoosejs.com/docs/connections.html>

- dotenv docs : <https://www.npmjs.com/package/dotenv>

: After creating mongoDb its going on to give us some secrete url which includes our database , server name user names and our passwords.

-helmet docs : <https://www.npmjs.com/package/helmet>.

: Secure request made to the server, because some request hearder contains some valuable properties inside.

-morgan docs : <https://www.npmjs.com/package/morgan>

: Request logger middleware. It tells us which request has been made and what was the results and how long it took.

-nodemon docs : <https://www.npmjs.com/package/nodemon>

: Whenever we make changes inside the files we must go back to restart the console to record the new updates/changes. Nodemon allows us to start the server once anytime we make changes it will refresh the application automatically.so it is listening to the files.

: code => package.json line 7 : delete test and add start nodemon mainfile.js

\*npm start => initiating the server

\* Express is a **web application framework for Node.js** , developers use Express to simplify the task of writing server code. There’s no need to repeat the same code over and over, as you would with the Node.js HTTP module.

\*Configure dotenv to enable usage and write all the secrete keys in the .env file

\*secrete key the data is a string of this format mongodb+srv://katlego:19740526@cluster0.8ckxv.mongodb.net/social?retryWrites=true&w=majority

-we obtained it on the mongoDb cloud <https://cloud.mongodb.com/v2/62d9475dad72501b87f43a19#clusters>

\*morgan library is working because we are getting this string

1 --- [date] “request type / address” status(404) duration of the response(150)

: ::1 - - [21/Jul/2022:17:11:17 +0000] "GET /favicon.ico HTTP/1.1" 404 150

Web page connection // path :” /routes”

Inserted after middleware lines of code

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

    res.send("Congratulations on making hosting your first ever simple wep page Mighty!!!");

});

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

    res.send("Welcome to the user page!");

});

\*we not using this lines of code since we are using the rest api.

-> Modules

Users.js

\*Create the attributes value each user will have , each attribute must have properties

\*property “required: true” -> the attribute is not optional ,every user must have it.

\*property “type:Array,default:[] “ in follwers/following -> this will all the users that are following you or that you are following.

\*” module.exports = mongoose.model("User",UserSchema);

” this line of code makes the module (“User”) be exported as library which can be used in other files/classes. Object oriented programming.

\*timestamps: whenever a create or update the user , the time stamp will automatically be updated. Helps to keep track when did each even take place. <-

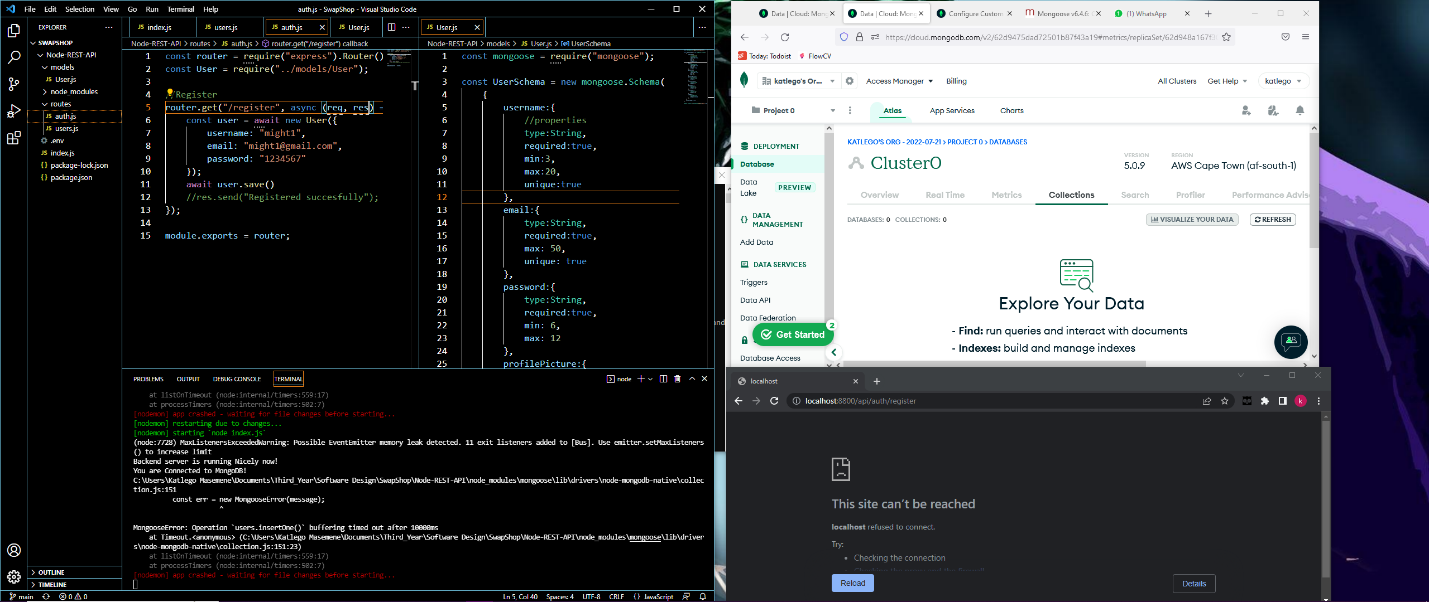
Auth.js

Registration testing :

\*await: waits for the data to be loaded into the database

\*npm add bcrypt => library used to hash the password:

Error: Mongo database connection error



FIX: Log the connection and expoliot the error

const db = mongoose.connection

db.on('error',(err)=> {

    console.log('Database connection has failed my G!...error is',err);

})

db.once('open',()=>{

    console.log('Database Connection succesful,Good job!');

})

**Postman app**

\*Error :”cloud error cannot send requests” => you must install the postman desktop,double click on it (it won’t open) then re-send the request on cloud.

\*Note that the variable name accomapanied with “await” corresposnds to the name of the database on postman . i.e

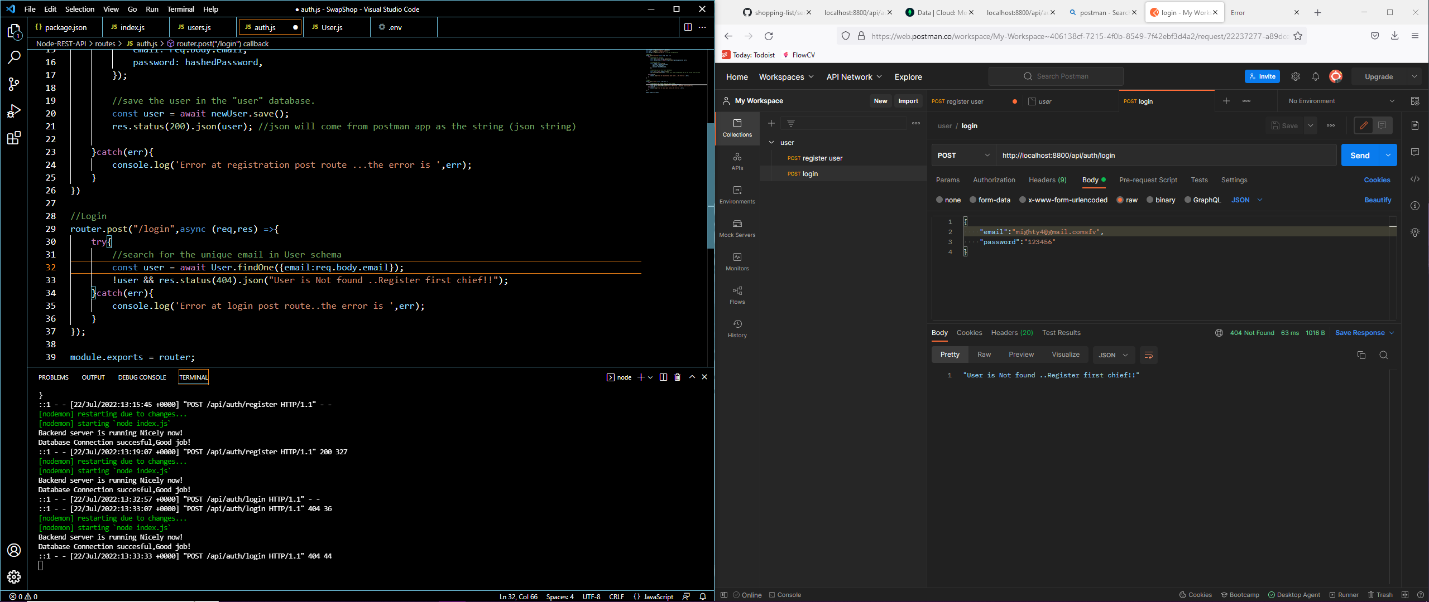
Saves the user into the database

const user = await newUser.save();

finds the corresponding name in the “user” database

const user = await User.findOne({email:req.body.email});

database name(user) on postman must correspond to the object name(user) of the class User(which was imported)



Route/users.js line 6

\*Note that we use triple equal signs instead of double equal sign to check if two parameters are equal

\* //update user put route :

Line 6 - 14

-authenticate the users credentials (correct profile /is admin)

- updating the password: we get the password from the body, generate new one and update the body.password with the new hashed

//Line 18 – 20

const user = await User.findByIdAndUpdate(req.params.id,{

                $set: req.body,

            });

Finds the user in the database by the provided id and it changes whichever parameter specified on postman..(wherether it is city relation,from etc) the code “{$set:req.body,}”handles all those generala properties (those without required in them)

\*//get user

- //omit passwaord in the retrieval

        const {password,updatedAt,createdAt, ...other} = user.\_doc

        res.status(200).json(other);

include other properties you don’t wan to be retrievd in the curly brackets

and note that “json.(other)” that other is not optional without it we get an empty json string

//Follow and unfollow

\*Note that the the Id taken from the body is the one the currently logged in user and the id from the parama (i.e the id on the URL) is the one of the person we want to follow or unfollow.

//Database creation Logic

\*Note that auth.js and users.js they both make use of the same parent class / mongoose schema “User”. So hence why we getting one database for both files on mongoDb atlas.(so basically putiing all the info of auth.js in user,js wouldn’t have causes any issues.)

\*However post.js make use of a different schema named Post.js ,hence why we are getting a new different database for the posts.

\*In conclusion, each model/mongoose schema/parent class will have it’s own designated database.

**Phase II : React Front end**

\*After creating the folder:type in “ npx create-react-app” note that the name of the folder must not be have caps.

\*Delete them all and be left with App.js and index.js

\*Downloading the appropriate fonts : <https://fonts.google.com/specimen/Roboto>

-Selected fonts: Thin 100 , light 300, Regular 400, Medium 500 italic, Bold 700, Black 900