**Back-end Setup and Testing with Postman**

**\*Note:** if this is a new folder where you are creating a project in, run the command ***git init*** first after creating the project folder to initialize an *empty git repository*

Create the project folder: mkdir full\_stack\_project

**cd full\_stack\_project**

git init – to initialize the folder as a git repository

create file .gitignore in the full\_stack\_project

add line to file:

node\_modules/

save file

**Client**

npx create-react-app client

cd client

install axios in the client folder: npm i axios

install react router dom: npm install react-router-dom

install socket.io: npm install socket.io-client

**Server**

**cd full\_stack\_project**

mkdir server

cd server

mkdir config controllers models routes

create a server.js file in the server folder

npm init -y // creates package.json

npm i express mongoose cors

npm i dotenv bcrypt cookie-parser jsonwebtoken

install socket.io: npm install socket.io

Create .env file in the server directory.

**Create these files in each folder:**

**config folder**

create jwt.config.js

create mongoose.config.js

**controllers folder**

create user.controller.js

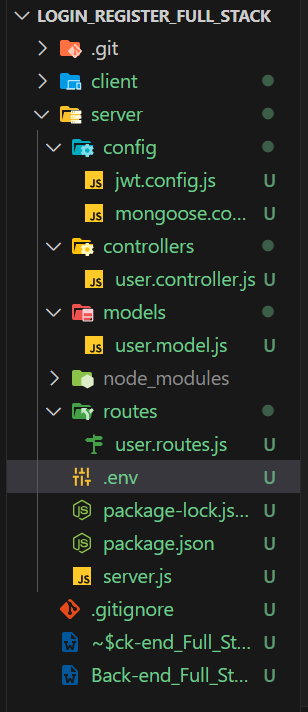
**models folder**

create user.model.js

**routes folder**

create user.routes.js

**Folder Structure for Server:**



.env

MY\_PORT = 8000

DB\_NAME = "testUserLogRegDB"

JWT\_SECRET = "lk;srjfkljsdfaklfgjklfgdsajkhgdfjkhgefrj"

server.js

require("dotenv").config();

const express = require("express");

const cors = require("cors");

const cookieParser = require("cookie-parser");

const app = express();

app.use(express.json())

app.use(express.urlencoded({ extended: true }))

app.use(cors({

    origin: "http://localhost:3000",

    credentials: true

}))

app.use(cookieParser())

require("./config/mongoose.config")

// require('./routes/bbqblog.routes')(app)

require("./routes/user.routes")(app)

app.listen(process.env.MY\_PORT, () => console.log(`You are connected to port ${process.env.MY\_PORT}`))

jwt.config.js

const jwt = require("jsonwebtoken")

module.exports = {

    authenticate(req, res, next) {

        jwt.verify(req.cookies.usertoken,

            process.env.JWT\_SECRET,

            (err, payload) => {

                if (err) {

                    console.log(err);

                    res.status(401).json({ verified: false })

                }

                else {

                    console.log(payload);

                    req.jwtpayload = payload

                    next()

                }

            }

        )

    }

}

mongoose.config.js

const mongoose = require("mongoose");

mongoose.connect(`mongodb://localhost/${process.env.DB\_NAME}`, {

    useNewUrlParser: true,

    useUnifiedTopology: true,

})

    .then(() => {

        console.log(`You are connected to the database called ${process.env.DB\_NAME}`)

    })

    .catch((err) => {

        console.log(`you had a problem connecting the ${process.env.DB\_NAME}. Here is your error:`, err)

    })

user.controller.js

const User = require('../models/user.model')

const bcrypt = require('bcrypt')

const jwt = require('jsonwebtoken')

module.exports = {

    register: (req, res) => {

        const user = new User(req.body)

        user

            .save()

            .then((newUser) => {

                console.log(newUser)

                console.log('Successfull Registered!')

                res.json({

                    successMessage: 'Thank you for registering',

                    user: newUser,

                })

            })

            .catch((err) => {

                console.log('register not successful!')

                res.status(400).json(err)

            })

    },

    login: (req, res) => {

        User.findOne({ email: req.body.email })

            .then((userRecord) => {

                if (userRecord === null) {

                    res.status(400).json({ message: 'Invalid Login Attempt' })

                } else {

                    bcrypt

                        .compare(req.body.password, userRecord.password)

                        .then((isPasswordValid) => {

                            if (isPasswordValid) {

                                console.log('password is valid')

                                res

                                    .cookie(

                                        'usertoken',

                                        jwt.sign(

                                            {

                                                id: userRecord.\_id,

                                                email: userRecord.email,

                                                username: userRecord.username,

                                            },

                                            process.env.JWT\_SECRET

                                        ),

                                        {

                                            httpOnly: true,

                                            expires: new Date(Date.now() + 9000000),

                                        }

                                    )

                                    .json({

                                        message: 'Succesfully',

                                        userLoggedIn: userRecord.username,

                                    })

                            } else {

                                res.status(400).json({ message: 'Invalid Attempt' })

                            }

                        })

                        .catch((err) => {

                            console.log(err)

                            res.status(400).json({ message: 'Invalid Attempt' })

                        })

                }

            })

            .catch((err) => {

                console.log(err)

                res.status(400).json({ message: 'Invalid Attempt' })

            })

    },

    logout: (req, res) => {

        console.log('logging out')

        res.clearCookie('usertoken')

        res.json({

            message: 'You have successfully logged out!',

        })

    },

    getLoggedInUser: (req, res) => {

        User.findOne({ \_id: req.jwtpayload.id })

            .then((user) => {

                console.log(user)

                res.json(user)

            })

            .catch((err) => {

                console.log(err)

            })

    },

    findAllUsers: (req, res) => {

        User.find()

            .then((allUsers) => {

                res.json(allUsers)

            })

            .catch((err) => {

                console.log('Find All Users failed')

                res.json({ message: 'Something went wrong in findAll', error: err })

            })

    },

}

user.model.js

const mongoose = require("mongoose");

const bcrypt = require("bcrypt");

const UserSchema = new mongoose.Schema({

    username: {

        type: String,

        required: [true, "Username is required"]

    },

    email: {

        type: String,

        required: [true, "Email address is required"]

    },

    password: {

        type: String,

        required: [true, "Password is required"],

        minLength: [8, "Passwords MUST be at least 8 characters"]

    }

}, { timestamps: true })

UserSchema.virtual("confirmPassword")

    .get(() => this.\_confirmPassword)

    .set((value) => this.\_confirmPassword = value)

UserSchema.pre("validate", function (next) {

    if (this.password !== this.confirmPassword) {

        this.invalidate("confirmPassword", "Passwords must match!!!")

        console.log("Passwords don't match!")

    }

    next()

})

UserSchema.pre("save", function (next) {

    console.log("in pre save");

    bcrypt.hash(this.password, 10)

        .then((hashedPassword) => {

            this.password = hashedPassword;

            next();

        })

})

const User = mongoose.model("User", UserSchema);

module.exports = User;

user.routes.js

const UserController = require("../controllers/user.controller");

const { authenticate } = require("../config/jwt.config")

module.exports = (app) => {

    app.get("/api/allUsers", UserController.findAllUsers);

    app.post("/api/users/register", UserController.register)

    app.post("/api/users/login", UserController.login);

    app.post("/api/users/logout", UserController.logout);

    app.get("/api/users", authenticate, UserController.getLoggedInUser);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Start mongodb in a bash terminal:**

mongod --ipv6

**Start the server in a bash terminal:**

Change into the server directory: cd server

Start the server: npx nodemon server.js

You should see the messages:

You are connected to port 8000

You are connected to the database called {databaseName}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Postman**

Get all users: http://localhost:8000/api/allUsers

A screenshot of a computer

Description automatically generated

Register: http://localhost:8000/api/users/register

A screenshot of a computer

Description automatically generated

Login: http://localhost:8000/api/users/login

A screenshot of a computer

Description automatically generated

Logout: http://localhost:8000/api/users/logout

A screenshot of a computer

Description automatically generated

Get user that is logged in: http://localhost:8000/api/users

A screenshot of a computer

Description automatically generated

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*