**Backend Tutorial:**

cmd > node

U get nodejs cmd line.

Ejs:

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

    res.render('index',{ name : "zeenath"})

})

Views/Index.ejs

<body>

    <h1>Home</h1>

    <!-- <p>Hello <%= locals.nambo %></p> -->

        <!-- <p>Hello <%= locals.name %></p> -->

        <p>Hello <%= name %></p>

</body>

**Serving public folder:**

Public/ index.html

Keeping index.html in public folder=> we must access it directly like localhost:3000/index.html

Setting absolute path of static folder

app.use(express.static(\_\_dirname + 'public')); //Serves resources from public folder

refer staticserving files

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Home</title>

    <link rel="stylesheet" href="css/style1.css">

</head>

<body>

    <!-- <h1>Home</h1>

    <p>Hello <%= locals.nambo %></p> -->

        <!-- <p>Hello <%= locals.name %></p> -->

        <!-- <p>Hello <%= name %></p> -->

        <!-- <p>hello 2</p> -->

        <h1>contact form</h1>

        <form action="/" method="post">

            <input type="text" name="name" placeholder="name"/><br>

            <input type="email" name="email" placeholder="email"/><br>

            <button type="submit">send message</button>

        </form>

</body>

</html>

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

    res.render('index',{ name : "zeenath"})

    // res.sendFile('index.html')

})

app.post('/',(req,res)=>{

    console.log(req.body)

console.log(req.body.name)

    res.send('success login')

})

On submitting form, it will go to post (‘/’)

{ name: 'zee', email: 'zeenathzeenath2710@gmail.com' }

Whatever we give name to input field it will get stored into it.

Creating an array and pushing into it. Can use mongodb later.

**In index.js:**

Const users=[];

app.post('/',(req,res)=>{

    console.log(req.body)

    array.push({username : req.body.name, email: req.body.email})

    console.log('array is' ,array)

    res.send('success login')

})

can even create success.ejs another file and render it here on post , on submitting form

app.post('/',(req,res)=>{

    console.log(req.body)

    array.push({username : req.body.name, email: req.body.email})

    console.log('array is' ,array)

    // res.send('success login')

    // res.render('success')

    res.redirect('/success') // get redirect to / endpoint

})

We can also use redirect like this.

app.get(‘/success’,(req,res)=>{

res.render(‘success’);

}

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

    res.json({ array})

})

Run localhost:3000/users - > u get empty array

Now go to localhost:3000 and submit form

Then come to localhost:3000/users – u get that detail into array

Default action for form =’/’

Mongodb:

Now instead of array using mongodb

mongoose.connect('mongodb://localhost:27017',{dbname: "backend"})

.then(()=>console.log('database connected'))

.catch((e)=> console.log(e))

const messageSchema= new mongoose.Schema({

    name: String ,

    email: String,

});

const Message =mongoose.model('Message',messageSchema)

here, instead of localhost – take 127.0.0.1

app.post(‘/contact’,async (req,res)=>{

const userData={ username: req.body.name, email: req.body.email}

console.log(userData)

await Message.create(userData);

res.redirect(‘/success’)

}

Email get stored in db, name no as username is not matching with name in schema

Await Message.create({ name: req.body.name, email: req.body.email});

Or

Const {name,email}=req.body // destructuring

Await Message.create({name ,email});

**Authentication:**

**Login.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Home</title>

    <link rel="stylesheet" href="css/style1.css">

</head>

<body>

    <!-- <h1>Home</h1>

    <p>Hello <%= locals.nambo %></p> -->

        <!-- <p>Hello <%= locals.name %></p> -->

        <!-- <p>Hello <%= name %></p> -->

        <!-- <p>hello 2</p> -->

        <h1>contact form</h1>

        <form action="/login" method="post">

            <!-- default action='/' -->

            <button type="submit">login</button>

        </form>

</body>

</html>

app.post('/login',(req,res)=>{s

    res.cookie('token','im in ')

    res.redirect('/')

})

**On clicking login button cookie get stored.**

**Adding expiry date**

app.post('/login',(req,res)=>{

    res.cookie('token','im in ',{

        httpOnly:true, expires:new Date(Date.now()+60\*1000)

    })

    res.redirect('/')

})

**To show logout button when login , and to show login button when logout**

**Check is there cookie in ‘/’ get endpoint**

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

    console.log(req.cookies)

    res.render('login',{ name : "zeenath"})

    // res.sendFile('index.html')

})

**U get undefined when u do req.cookie , on cliking login button , token is stored in cookie but got undefined**

**So install cookie-parser and import it and then see u get cookie.**

 const cookieParser=require('cookie-parser')

 app.use(cookieParser())

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

    console.log(req.cookies)

    console.log(req.cookies.token)

    res.render('login',{ name : "zeenath"})

    // res.sendFile('index.html')

})

**Req.cookies- get all cookie values**

**Req.cookies.token – get only token**

    const token=req.cookies.token

or destrcturing as

    const {token}= req.cookies

**console.log(token)**

**logout.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Home</title>

    <link rel="stylesheet" href="css/style1.css">

</head>

<body>

        <form action="/logout" method="post">

            <!-- default action='/' -->

            <button type="submit">Logout</button>

        </form>

</body>

</html>

app.post('/login',(req,res)=>{

    console.log('login post req')

    res.cookie('token','im in ',{

        httpOnly:true, expires:new Date(Date.now()+60\*1000)

    })

    res.redirect('/')

})

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

    // console.log(req.cookies)

    // console.log(req.cookies.token)

    // const token=req.cookies.token

    const {token}= req.cookies

    console.log(token)

    if(token){

        res.render('login')

    }

    else{

        res.render('logout')

    }

    // res.sendFile('index.html')

})

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

    res.cookie('token',null,{

        httpOnly:true, expires:new Date(Date.now())

    });

    res.redirect('/')

})

**So initially we get login button , on click , token generate when token get generate , got logout button.**

**Using authentication function as middleware:**

const isAuthenticated=(req,res,next)=>{

    const {token}= req.cookies

    console.log(token)

    if(token){

        next()

    }

    else{

        res.render('login')

    }

}

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

    // // console.log(req.cookies)

    // // console.log(req.cookies.token)

    // // const token=req.cookies.token

    // const {token}= req.cookies

    // console.log(token)

    // if(token){

    //     res.render('logout')

    // }

    // else{

    //     res.render('login')

    // }

    // // res.render('login')

    // // res.sendFile('index.html')

    res.render( 'logout')

})

Now creating new user

**Modifying user schema name**

const userSchema= new mongoose.Schema({

    name: String ,

    email: String,

});

const User =mongoose.model('User',userSchema)

remove /contact,/users, / success endpoints

app.post('/login',(req,res)=>{

    console.log(req.body)

    // console.log('login post req')

    res.cookie('token','im in ',{

        httpOnly:true, expires:new Date(Date.now()+6\*1000)

    })

    res.redirect('/')

})

Console.log(req.body) in post ‘/login’ endpoint to know entered details of login page getting or not.

**Login.ejs**

body>

    <form action="/login" method="post">

        <h1>Login now</h1>

        <input type="text" name="name" placeholder="name"/><br>

        <input type="email" name="email" placeholder="email"/><br>

        <button type="submit">Login</button>

    </form>

</body>

</htm

app.post('/login',async(req,res)=>{

    console.log(req.body)

    const {name, email}= req.body

    const user =await User.create({

        name,

        email

    });

    // console.log('login post req')

    res.cookie('token',user.\_id,{

        httpOnly:true, expires:new Date(Date.now()+6\*1000)

    })

    res.redirect('/')

})

Saving user data using create(). Then storing its id into token

app.post('/login',async(req,res)=>{

    console.log(req.body)

    const {name, email}= req.body

    const user =await User.create({

        name,

        email

    });

    // console.log('login post req')

    res.cookie('token',user.\_id,{

        httpOnly:true, expires:new Date(Date.now()+6\*1000)

    })

    res.redirect('/')

})

Aise hi kisiku user.\_id phek ni sakte , so use jwt authentication

Npm I jsonwebtoken

Import jsonwebtoken as const jwt=require(‘jsonwentoken’)

Creating token in ‘/login’ post endpoint .

Before storing user id directly(encoded) form into cookie.

app.post('/login',async(req,res)=>{

    console.log(req.body)

    const {name, email}= req.body

    const user =await User.create({

        name,

        email

    });

    // console.log('login post req')

    const token=jwt.sign({\_id: user.\_id},"zeenathsecret123");

    console.log(token)

    //res.cookie('token',user.\_id,{

       // httpOnly:true, expires:new Date(Date.now()+6\*1000)

    //})

    res.redirect('/')

})

Now on login , u got token in console.log(token);

Copy and paste that token in jwt.io , u got the id. Bcoz we passed as a payload a id only.

Now store token in cookie.

Now token getting stored into cookie, but in encoded form

So if we decrypt using jwt.io also .

Verify token in isauthentication function before calling next();

const isAuthenticated=(req,res,next)=>{

    const {token}= req.cookies

    console.log(token)

    if(token){

        const decoded=jwt.verify(token,"zeenathsecret123");

        console.log(decoded)

        next()

    }

    else{

        res.render('login')

    }

}

So jwt.verify () – give decoded data .

Console.log(decoded) – will print { \_id: '657add1e77f66487989e4748', iat: 1702550814 }

Saving whole user information in req.user

Saving in req.user so that user can be accessible in its next function. App.get(‘/’,isauthenticated,(req,res)=>{ }

Console.log (req.user) in isauthenticated function

Now check req.user is accessible all over.

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

    // // console.log(req.cookies)

    // // console.log(req.cookies.token)

    // // const token=req.cookies.token

    // const {token}= req.cookies

    // console.log(token)

    // if(token){

    //     res.render('logout')

    // }

    // else{

    //     res.render('login')

    // }

    // // res.render('login')

    // // res.sendFile('index.html')

    console.log(req.user)

    res.render( 'logout')

})

**Logout.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Home</title>

    <link rel="stylesheet" href="css/style1.css">

</head>

<body>

        <h1>Hi ! <%=name %></h1>

        <form action="/logout" method="get">

            <!-- default action='/' -->

            <button type="submit">Logout</button>

        </form>

</body>

</html>

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

    console.log(req.user)

    res.render( 'logout',{name:req.user.name})

})

In login we can check whether user exists,

app.post('/login',async(req,res)=>{

    console.log(req.body)

    const {name, email}= req.body;

    let user=await User.findOne({email});

    if(!user){

        return console.log('register first')

    }

     user =await User.create({

        name,

        email

    });

    // console.log('login post req')

    const token=jwt.sign({\_id: user.\_id},"zeenathsecret123");

    console.log(token)

    res.cookie('token',token,{

        httpOnly:true, expires:new Date(Date.now()+6\*1000)

    })

    res.redirect('/')

})

So if we enter email and name which is already existing in mongodb – then we get the remaining code as it is ,

If we enter new name and email then we get as register first in console.

So if I enter already existing email and name then only, im able to login, otherwise got console as register first.

So if user doesn’t already exists, we can redirect to register

    if(!user){

        // return console.log('register first')

        return res.redirect('/register')

    }

**Register.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Register</title>

</head>

<body>

    <form  method="post">

        <!-- default action='/' -->

        <h1>Register now</h1>

        <input type="text" name="name" placeholder="name"/><br>

        <input type="email" name="email" placeholder="email"/><br>

        <input type="password" placeholder="password"><br/>

        <button type="submit">Sign Up</button>

    </form>

</body>

</html>

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

    res.render('register')

})

In app.post(‘/login’)

If (!user) redirect(‘/register’)

app.post('/register',async(req,res)=>{

    console.log(req.body)

    const {name, email,password}= req.body;

    let user=await User.findOne({email});

    if(user){ // if user already exits , redirect to login

        // return console.log('register first')

        return res.redirect('/login')

    }

     user =await User.create({

        name,

        email,

        password

    });

    // console.log('login post req')

    const token=jwt.sign({\_id: user.\_id},"zeenathsecret123");

    console.log(token)

    res.cookie('token',token,{

        httpOnly:true, expires:new Date(Date.now()+6\*1000)

    })

    res.redirect('/')

})

Changing ‘/login’ to’/register’ and added password also.

In schema , added password

const userSchema= new mongoose.Schema({

    name: String ,

    email: String,

    password:String

});

When signup using new user details got’ hi name ‘ and token got stored in cookie

If signup using existing user details, then got login page. , and no cookie token exits here in login page.

Unable to access ‘/’ endpoint bcoz it is logout page . it will be accessible only on login.

**Login.ejs- have email and password**

    <form action="/login" method="post">

        <h1>Login now</h1>

        <input type="email" name="email" placeholder="email"/><br>

        <input type="password" placeholder="password" name="password"><br/>

        <button type="submit">Login</button>

    </form>

app.post('/login',async(req,res)=>{

    const {email,password}=req.body;

    let user=await User.findOne({email});

    if(!user) return res.redirect('/register');

})

On giving new email and password- should gets redirect to register page

app.post('/login',async(req,res)=>{

    const {email,password}=req.body;

    let user=await User.findOne({email});

    if(!user) return res.redirect('/register');

    const isMatch=user.password===password;

    if(!isMatch) return res.render('/login',{message: Incorrect Password})

})

If user exists and password is not match, then print incorrect password.

If matches password, same token gen

app.post('/login',async(req,res)=>{

    const {email,password}=req.body;

    let user=await User.findOne({email});

    if(!user) return res.redirect('/register');

    const isMatch=user.password===password;

    if(!isMatch) return res.render('login',{message: 'Incorrect Password'})

    const token=jwt.sign({\_id: user.\_id},"zeenathsecret123");

    console.log(token)

    res.cookie('token',token,{

        httpOnly:true, expires:new Date(Date.now()+6\*1000)

    })

    res.redirect('/')

})

On logout, clear cookie

**Login.ejs**

    <form action="/login" method="post">

        <h1>Login now</h1>

        <input type="email" name="email" placeholder="email"/><br>

        <input type="password" placeholder="password" name="password"><br/>

        <p class="error"><%= message %></p>

        <button type="submit">Login</button>

    </form>

Passing email also

    if(!isMatch) return res.render('login',{email ,message: 'Incorrect Password'})

login.ejs

        <input type="email" name="email" placeholder="email" value="<%= locals.email %>"/><br>

Hashing password: bcrypt

Npm I bcrypt

Import bcrypt from

In post(‘/register’)

app.post('/register',async(req,res)=>{

    console.log(req.body)

    const {name, email,password}= req.body;

    let user=await User.findOne({email});

    if(user){

        // return console.log('register first')

        return res.redirect('/login')

    }

    const hashedpassword= await bcrypt.hash(password,10);

     user =await User.create({

        name,

        email,

        password: hashedpassword,

    });

    // console.log('login post req')

    const token=jwt.sign({\_id: user.\_id},"zeenathsecret123");

    console.log(token)

    res.cookie('token',token,{

        httpOnly:true, expires:new Date(Date.now()+6\*1000)

    })

    res.redirect('/')

})

Compare bcrypted password in ‘/login’ post using bcrypt.compare

app.post('/login',async(req,res)=>{

    const {email,password}=req.body;

    let user=await User.findOne({email});

    if(!user) return res.redirect('/register');

    const isMatch=await bcrypt.compare(password,user.password)

    if(!isMatch) return res.render('login',{email ,message: 'Incorrect Password'})

    const token=jwt.sign({\_id: user.\_id},"zeenathsecret123");

    console.log(token)

    res.cookie('token',token,{

        httpOnly:true, expires:new Date(Date.now()+6\*1000)

    })

    res.redirect('/')

})

Now creating apis in structured way.

**Api and params:**

**Npm init**

**PS C:\Users\Dell\OneDrive\edureka class\NodeApi> npm i express dotenv cookie-parser**

In package.json : add type:module – then import and exports works ,

Or else , type: COmmonjs – then require and module.exports

In scripts: can add “start” : “node server.js”

“dev” : “ nodemon server.js”

Create app.js

const express=require('express');

const app=express();

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

    res.send('nice working')

})

app.listen(3000,()=>{

    console.log('server started at 3000')

})

To get all users data at localhost:3000/users/all

Connecting to mongodb

const express=require('express');

const app=express();

const mongoose=require('mongoose')

mongoose.connect('mongodb://127.0.0.1:27017',{ dbName: "backendapi"})

.then(()=> console.log('database connected'))

.catch((e)=> console.log(e))

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

    res.send('nice working')

})

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

    res.json({

        success: true,

        users:[],

    })

})

app.listen(3000,()=>{

    console.log('server started at 3000')

})

**creating schema:**

const schema=new mongoose.Schema({

    name: String,

    email:String,

    password:String,

});

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

app.get('/users/all',async(req,res)=>{

    const users=await User.find({})

    res.json({

        success: true,

        users

    })

})

**Creating new dummy user , giving value here in post**

app.post('/users/new',async(req,res)=>{

    await User.create({

        name:"zeenath",

        email:"zee@123gmail.com",

        password:"fhdjf"

    })

    res.json({

        success: true,

        message:"Registered successfully"

    })

})

app.post('/users/new',async(req,res)=>{

    const {name,email,password}= req.body

    await User.create({

        name,

        email,

        password,

    })

    res.json({

        success: true,

        message:"Registered successfully"

    })

})

**As key value pais are same written as it is in create instaed of name:name …**

app.use(express.json())

    res.status(201).json({

        success: true,

        message:"Registered successfully"

    })

**To pass cookie-**     res.status(201).cookie("tempi","lol").json({

        success: true,

        message:"Registered successfully"

    })

**At /users/all – got all users data from mongodb**

**After ? mark what u entered will be in url of postman**

**Req.query – get all wt we entered**

**If I give in query**

app.get('/users/all',async(req,res)=>{

    const users=await User.find({})

    console.log(req.query)

    res.json({

        success: true,

        users

    })

})

**Pass any query u get in console.log**

<http://localhost:3000/users/all?zeekou=kou&category=cat1>

to add more queries can use & here

    const var1=req.query.zeekou;

    console.log(var1)

app.get('/userid',async(req,res)=>{

    const {id}=req.body;

    const user=await User.findById(id);

    res.json({

        success:true,

        user

    })

})

**Can pass one id in body**

app.get('/userid',async(req,res)=>{

    const id=req.body.id;

    console.log(req.body)

    const user=await User.findById(id);

    console.log(user)

    res.json({

        success:true,

        user

    })

})

**But when we wants to send data in body,use post req**

**Or get me bhejna tho send in params/query**

**But in get, query is optional**

**Dynamic id**

app.get('/userid/:id',async(req,res)=>{

    // const id=req.body.id;

    // console.log(req.body)

    const {id}=req.query

    console.log(req.params)

    const user=await User.findById(id);

    console.log(user)

    res.json({

        success:true,

        user:{}

    })

})

**So what ever we give inplace of :id get stored into req.params**

**Write route as /:userid then u get in output userid:{}**

**Always keep dynamic route at the end**

app.get('/userid/special',(req,res)=>{

    res.json({

        success: true,

        message:"Just joking"

    })

})

app.get('/userid/:id',async(req,res)=>{

    // const id=req.body.id;

    // console.log(req.body)

    const {id}=req.query

    console.log(req.params)

    const user=await User.findById(id);

    console.log(user)

    res.json({

        success:true,

        user:{}

    })

})

**The above endpoints if u keep below one up then get error on accessing /userid/special , if kept in this order**

**Mvc architecture**

**In app.js**

**Const router=express.Router()**

**Now can do router.get , router.post ….**

**Previous file :**

const express=require('express');

const app=express();

app.use(express.json())

const mongoose=require('mongoose')

mongoose.connect('mongodb://127.0.0.1:27017',{ dbName: "backendapi"})

.then(()=> console.log('database connected'))

.catch((e)=> console.log(e))

const schema=new mongoose.Schema({

    name: String,

    email:String,

    password:String,

});

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

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

    res.send('nice working')

})

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

//     res.json({

//         success:true,

//         users:[],

//     })

// })

app.get('/users/all',async(req,res)=>{

    const users=await User.find({})

    console.log(req.query)

    const var1=req.query.zeekou;

    console.log(var1)

    res.json({

        success: true,

        users

    })

})

// app.post('/users/new',async(req,res)=>{

//     await User.create({

//         name:"zeenath",

//         email:"zee@123gmail.com",

//         password:"fhdjf"

//     })

//     res.json({

//         success: true,

//         message:"Registered successfully"

//     })

// })

app.get('/userid/special',(req,res)=>{

    res.json({

        success: true,

        message:"Just joking"

    })

})

app.post('/users/new',async(req,res)=>{

    const {name,email,password}= req.body

    await User.create({

        name,

        email,

        password,

    })

    res.status(201).cookie("tempi","lol").json({

        success: true,

        message:"Registered successfully"

    })

})

//dynamic routing

app.get('/userid/:id',async(req,res)=>{

    // const id=req.body.id;

    // console.log(req.body)

    const {id}=req.query

    console.log(req.params)

    const user=await User.findById(id);

    console.log(user)

    res.json({

        success:true,

        user:{}

    })

})

app.listen(3000,(req,res)=>{

    console.log('server started')

})

**Create folder routes**

**Routes/User.js**

const express =require('express');

const router=express.Router();

export default router;

const express =require('express');

const router=express.Router();

const {user}  =require('../models/user.js')

router.get('/',(req,res)=>{

    res.send('nice working')

})

router.get('/users/all',async(req,res)=>{

    const users=await User.find({})

    console.log(req.query)

    const var1=req.query.zeekou;

    console.log(var1)

    res.json({

        success: true,

        users

    })

})

router.post('/users/new',async(req,res)=>{

    const {name,email,password}= req.body

    await User.create({

        name,

        email,

        password,

    })

    res.status(201).cookie("tempi","lol").json({

        success: true,

        message:"Registered successfully"

    })

})

router.get('/userid/special',(req,res)=>{

    res.json({

        success: true,

        message:"Just joking"

    })

})

router.get('/userid/:id',async(req,res)=>{

    // const id=req.body.id;

    // console.log(req.body)

    const {id}=req.query

    console.log(req.params)

    const user=await User.findById(id);

    console.log(user)

    res.json({

        success:true,

        user:{}

    })

})

export default router;

**can add same prefix /users in app.js**

**create models folder**

**move model and schema into model/user.js and export it**

const mongoose=require('mongoose')

const schema=new mongoose.Schema({

    name: String,

    email:String,

    password:String,

});

export const User = mongoose.model("User",schema);

**import router in app.js**

const userRouter=require('./routes/user')

app.use(userRouter)

**adding prefix**

app.use('/users',userRouter)

**remove ‘/users’ in all routes of routes/user.js**

**mvc architecture reference:**

database.js:

const mongoose=require('mongoose')

const connenctDB=async()=>{

    await mongoose.connect('mongodb://127.0.0.1:27017',{ dbName: "backendapi"})

.then(()=> console.log('database connected'))

.catch((e)=> console.log(e))

}

// connectDB()

module.exports=connenctDB

models/user.js – user schema

const mongoose=require('mongoose')

const schema=new mongoose.Schema({

    name: String,

    email:String,

    password:String,

});

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

 module.exports=User;

models/task.js- task schema

import schema in main file where u use User.find()

const mongovar=require('./data/database')

mongovar();

**app.js:**

const express=require('express');

const app=express();

const mongoose=require('mongoose')

const mongovar=require('./data/database')

const userRouter=require('./routes/user')

// console.log(userRouter)

app.use(express.json())

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

app.use('/users',userRouter)

mongovar();

app.listen(3000,(req,res)=>{

    console.log('server started')

})

**In routes/controllers import model**

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

const express =require('express');

const router=express.Router();

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

router.get('/',(req,res)=>{

    res.send('nice working')

})

router.get('/all',async(req,res)=>{

    const users=await User.find({})

    console.log(req.query)

    const var1=req.query.zeekou;

    console.log(var1)

    res.json({

        success: true,

        users

    })

})

router.post('/new',async(req,res)=>{

    const {name,email,password}= req.body

    await User.create({

        name,

        email,

        password,

    })

    res.status(201).cookie("tempi","lol").json({

        success: true,

        message:"Registered successfully"

    })

})

router.get('/userid/special',(req,res)=>{

    res.json({

        success: true,

        message:"Just joking"

    })

})

router.get('/userid/:id',async(req,res)=>{

    // const id=req.body.id;

    // console.log(req.body)

    const {id}=req.query

    console.log(req.params)

    const user=await User.findById(id);

    console.log(user)

    res.json({

        success:true,

        user:{}

    })

})

module.exports= router;