For Jwt and cookie theory explanation visit:

<https://youtu.be/mGrVmEex6_g?si=DvlpBXCMMcxCkFw0>

<https://youtu.be/kMErso06vHo?si=W7nzv4DHxp_Rc_jW>

**Creating home page,login and signup page using ejs**

Keep it in views folder

**Home.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>

  </head>

  <body>

    <h1>

      Welcome to home <% if (locals.user) { %> <%=locals.user.name%> <% } %>

    </h1>

    <a href="/logout">logout</a>

  </body>

</html>

To understand locals.user ,locals.username – check ejs full tutorial to get stored user name using locals.username.

Every variable get stored in locals in ejs

**Signup.ejs**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

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

    <title>Signup</title>

  </head>

  <body>

    <h1>Signup</h1>

    <form action="/" method="post">

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

      <input type="text" name="email" />

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

      <input type="submit" />

    </form>

  </body>

</html>

**Login.ejs**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

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

    <title>Login</title>

  </head>

  <body>

    <h1>login</h1>

    <% if (locals.error) { %>

    <h2><%=locals.error%></h2>

    <% } %>

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

      <input type="text" name="email" />

      <input type="text" name="password" />

      <input type="submit" />

    </form>

  </body>

</html>

Creating endpoints using express and connecting to mongodb

const express = require("express");

const app=express()

app.set("view engine", 'ejs');

app.use(express.urlencoded({ extended: false }));

app.use(express.json());

const mongoose = require("mongoose");

//connecting to db

let connectDb = async () => {

    try {

      await mongoose.connect("mongodb://localhost:27017", { dbName: "jwttutorial" });

      console.log("conected mongodb");

    } catch (error) {

      console.log("error connecting to mongodb");

    }

  };

connectDb();

let user = mongoose.Schema({

  name: {

    type: String,

    required: true,

  },

  email: {

    type: String,

    required: true,

    unique: true,

  },

  password: {

    type: String,

    required: true,

  },

});

let User=mongoose.model('jwttut',user);

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

    res.render("signup");

});

On post request i.e, after submitting signup form

    <form action="/" method="post">

It goes to endpoint ‘/’ and method post => app.post(‘/’)

At this endpoint , we take input entered by user i.e, name,email and password . -req.body.name,req.body.email,req.body.password

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

Then before storing to database hash the password and then store

  let hash = await bcrypt.hash(password, 10);

  let user = new User({ name, email, password: hash });

  await user.save();

then redirecting to login page(/login endpoint)

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

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

  let hash = await bcrypt.hash(password, 10);

  let user = new User({ name, email, password: hash });

  await user.save();

  res.redirect("/login");

});

/login endpoint is serving login.ejs file

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

  res.render("login.ejs");

});

Login.ejs has a form which on submit – runs the endpoint /login (post method)

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

=> it execute app.post(‘/login’,(req,res)=>{}

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

in this post req, taking email and password entered by user

    let { email, password } = req.body;

then finding in db whether the user with this email exists or not

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

if user doesnot exists – send ‘user not registered’

    if (!user) {

      res.render("login", { error: "user is not registered" });

    }

If user with that email exists then compare entered password during login with already stored hashed password with respect to the entered email.

      let compare = await bcrypt.compare(password, user.password);

if comparition returns false- then send ‘password is incorrect’

      if (!compare) {

        res.render("login", { error: "password is  incorrect" });

      }

Else if passwords are matched –

Generate a token for that user using jwt.sign()

      let token = await jwt.sign({user }, "zeenath@7144");

      console.log(`token is ${token}`)

then store that token in a cookie using res.cookie

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

app.use(cookieParser());

const jwt = require("jsonwebtoken");

const bcrypt=require('bcrypt')

install and import cookie-parser npm package for cookie

      res.cookie("token", token);

then redirecting to home url endpoint

      res.redirect("/home");

in home url endpoint- instead of directing the user directly to home page, checking whether the user req has a cookie( in which token is stored)- for this using a middleware (auth.js)

import auth.js in main file app.js

const middleware=require('./auth')

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

  let user=req.user;

  res.render('home.ejs',{user})

})

Now see the code of auth.js

It checks whether req.cookies has some value or not- ( as along with req ,cookie also automatically gets send, if the same user access)

If(!req.cookie)- if req.cookie has no value then redirecting him to again the login page itself

    if(!req.cookie){

    res.redirect('/login')

    }

else – if some token exists in req.cookie – then do jwt.verify – to check if the token matches with the stored one

if yes then do next() so that the code inside app.get(‘/home’) gets execute

    else{

        let verified= await jwt.verify(token,'zeenath@7144');

        req.user=verified.\_doc;

        next();

    }

**auth.js**

const jwt=require('jsonwebtoken');

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

    let {token}= req.cookies;

    console.log(`req.cookies is ${req.cookies}`)

    console.log(`token of req.cooki is  ${token}`)

    if(!token){

    res.redirect('/login')

    }

    else{

        let verified= await jwt.verify(token,'zeenath@7144');

        req.user=verified.\_doc;

        next();

    }

};

module.exports=auth;

**app.js**

const express = require("express");

const app=express()

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

app.use(cookieParser());

const jwt = require("jsonwebtoken");

const bcrypt=require('bcrypt')

const middleware=require('./auth')

app.set("view engine", 'ejs');

app.use(express.urlencoded({ extended: false }));

app.use(express.json());

const mongoose = require("mongoose");

//connecting to db

let connectDb = async () => {

    try {

      await mongoose.connect("mongodb://localhost:27017", { dbName: "jwttutorial" });

      console.log("conected mongodb");

    } catch (error) {

      console.log("error connecting to mongodb");

    }

  };

connectDb();

let user = mongoose.Schema({

  name: {

    type: String,

    required: true,

  },

  email: {

    type: String,

    required: true,

    unique: true,

  },

  password: {

    type: String,

    required: true,

  },

});

let User=mongoose.model('jwttut',user);

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

    res.render("signup");

});

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

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

  let hash = await bcrypt.hash(password, 10);

  let user = new User({ name, email, password: hash });

  await user.save();

  res.redirect("/login");

});

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

  let user=req.user;

  res.render('home.ejs',{user})

})

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

  res.render("login.ejs");

});

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

  try {

    let { email, password } = req.body;

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

    if (!user) {

      res.render("login", { error: "user is not registered" });

    } else {

      let compare = await bcrypt.compare(password, user.password);

      if (!compare) {

        res.render("login", { error: "password is  incorrect" });

      }

      let token = await jwt.sign({user }, "zeenath@7144");

      console.log(`token is ${token}`)

      res.cookie("token", token);

      res.redirect("/home");

    }

  } catch (error) {

    console.log(error);

  }

});

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

  res.clearCookie('token')

  res.redirect("/");

});

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

  let { token } = req.cookies;

  if (!token) {

    res.redirect("/login");

  } else {

    let verified = await jwt.verify(token, "vinod@7144");

    req.user = verified.\_doc;

    next();

  }

};

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

  let user = req.user;

  res.render("home.ejs", { user });

});

app.listen(4000, () => {

  console.log("server runnig on 4000");

});

**Can check stored cookie in inspect->application->cookie in leftside panel**

**Mvc architecture:**

Writing the above code in mvc architecture

Create a controller folder – move all (req,res) functions into it

Make a routes folder add all endpoints – router.get(‘/’ ) , router.get(‘/login’) and posts reqs all into this.

In app.js – import router file into a variable routervar and do app.use(‘/’,routervar)

Create db folder inside it create conn.js – in this add mongodb related code and import it in main file app.js