Authentication

mongoose-encryption will automatically encrypts our password in binary long string which is simple authentication

syntax of mongoose-encryption:

import encrypt from …

const userSchema = new mongoose.Schema({

…

})

userSchema.plugin(encrypt,{secret:”Secrets”,encryptedFields:[“password”]})

md5 package will converts password into hash

syntax: md5(“password”) return hash including characters of “password”

It cannot be decrypted to return a password

It is more complex to hack

Simple Authentication index.js:

import bodyParser from "body-parser";

import dotenv from "dotenv";

import mongoose, { mongo } from "mongoose";

import express from "express";

import { fileURLToPath } from "url";

import path from "path";

import md5 from "md5";

dotenv.config();

mongoose.connect("mongodb://0.0.0.0/userDB")

*const* userSchema = {

  email:*String*,

  password:*String*

}

*const* user = new mongoose.model("users",userSchema)

*let* isUserExist;

*const* curPath = fileURLToPath(import.meta.url);

*const* \_\_dirname = path.dirname(curPath);

*const* pubPath =path.join(\_\_dirname,"public");

*const* PORT  = process.env.PORT;

*const* app = express();

app.use(bodyParser.urlencoded({extended: true}));

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

app.use(express.static(pubPath));

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

*res*.render("home")

})

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

*res*.render("register")

})

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

*res*.render("login")

})

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

  isUserExist = await user.findOne({email:*req*.body.email})

if(isUserExist)

{

*let* userExistError = {"error":"user already exists with this email"}

*res*.render("register",{userExistError})

}

else

{

*const* newUser = await user.create({email:*req*.body.email,password:md5(*req*.body.password)})

  newUser.save();

*res*.render("secrets");

}

})

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

*let* userLoginError;

  isUserExist = await user.findOne({email:*req*.body.email})

  if(isUserExist) {

  if(isUserExist.password == md5(*req*.body.password)) *res*.render("secrets")

  else {

    userLoginError = {"error":"Incorrect password"}

*res*.render("login",{userLoginError})

  }

} else {

  userLoginError = {"error":"Account not registered with this email"}

*res*.render("login",{userLoginError}) }

})

app.listen(PORT,()*=>*{

  console.log(`Server is running at port ${PORT}`)

})

bcrypt is used to convert password into hash with given no of times

Code:

import bodyParser from "body-parser";

import dotenv from "dotenv";

import mongoose, { mongo } from "mongoose";

import express from "express";

import { fileURLToPath } from "url";

import path from "path";

import bcrypt from "bcrypt";

dotenv.config();

mongoose.connect("mongodb://0.0.0.0/userDB")

*const* userSchema = new mongoose.Schema({

  email:String,

  password:String

})

*const* user = new mongoose.model("users",userSchema)

*let* isUserExist;

*const* curPath = fileURLToPath(import.meta.url);

*const* \_\_dirname = path.dirname(curPath);

*const* pubPath =path.join(\_\_dirname,"public");

*const* PORT  = process.env.PORT;

*const* app = express();

app.use(bodyParser.urlencoded({extended: true}));

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

app.use(express.static(pubPath));

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

*res*.render("home")

})

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

*res*.render("register")

})

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

*res*.render("login")

})

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

  isUserExist = await user.findOne({email:*req*.body.email})

if(isUserExist)

{

*let* userExistError = {"error":"user already exists with this email"}

*res*.render("register",{userExistError})

}

else

{

  bcrypt.hash(*req*.body.password,10,async (*err*,*hash*)*=>*{

*const* newUser = await user.create({email:*req*.body.email,password:*hash*})

  newUser.save();

*res*.render("secrets");

  })

}

})

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

*let* userLoginError;

  isUserExist = await user.findOne({email:*req*.body.email})

  if(isUserExist) {

    bcrypt.compare( *req*.body.password,isUserExist.password,(*err*,*result*)*=>*{

      if(*result*==true)

*res*.render("secrets")

    else

      {

        userLoginError = {"error":"Incorrect password"}

*res*.render("login",{userLoginError})

      }

    })

} else {

  userLoginError = {"error":"Account not registered with this email"}

*res*.render("login",{userLoginError}) }

})

app.listen(PORT,()*=>*{

  console.log(`Server is running at port ${PORT}`)

})

passport.js (automatically adds hash, login, register functionality and sessions and cookies)

required packages – “passport”, “passport-local”, “passport-local-mongoose” and “express-session”

write it after all app.use like bodyParser.url..etc. and before mongoose.connect

app.use(session({

  secret:process.env.SECRET,

  resave:false,

  saveUninitialized:false

}))

app.use(passport.initialize());

app.use(passport.session());

write it just after userSchema

userSchema.plugin(passportLocalMongoose) // for hash and salt and save into database

serialise means creates a cookie and deserialise means destroys the cookie