# Adding Authentication(51)

1) install :

npm i passport passport-local passport-local-mongoose

2)Create a new file -user.js under models:

const mongoose = require('mongoose')

const Schema = mongoose.Schema;

const passportLocalMongoose = require('passport-local-mongoose);

const UserSchema = new Schema({

email:{

type:string,

required: true,

unique: true

}

});

UserSchema.plugin(passportLocalMongoose);//username unique

module.exports = mongoose.model('User', UserSchema);

===configure passport =====

## app.js

const passport = require('passport');

const LocalStrategy = require('passport-local');

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

const userRoutes = require(‘./routes/users’)

const campgroundRoutes = require(‘./routes/campgrounds’);

const reviewsRoutes = require(‘./routes/reiews’);

app.use(passport.initialized());// must be after app.use(session)

app.use(passport.session())// for maintaining persistence log-in

app.use(‘/’, userRoutes);

app.use(‘/campgrounds’, campgroundRoutes)

app.use(‘/campgrounds/:id/reviews’, reviewsRoutes)

app.use((req, res, next) =>{

console.log(req.session)

res.locals.currentUser = req.user;

res.locals.success = req.flash(‘success’);

res.local.error = req.flash(‘error’);

passport.use(new LocalStrategy(User.authenticate()))//Use passport-Local strategy and user authenticate method.

passport.serializeUser(User.serializeUser());//get user data into session (storage)

passport.deserializeUser(User.deserializeUser());//get user data out of session (un-storage)

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

const user = new User({email: 'colt@gmail.com', username: 'colt'

const newUser = await user.register(user, 'chicken')// Pbkdf2 pw hash method. pw = 'chicken'

=====512. Register Form =======

=====513. Register Route Logic =====

## routes/users.js

const express = require('express')

const router = express.Router();

const catchAsync = require(‘../utils/catchAsync’)

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

const passport = require(‘passport’);

router.get('/register', (req, res) =>{ //create views/users/register.ejs

res.render('users/register)

})

router.post('/register', catchAsync(async(req, res, next) =>{

try{

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

const user = new User({email, username});

const registereduser = await User.register(user, password);

~~console.log(registeredUser);~~

req.login(registeredUser, err =>{

if (err) return next(err);

req.flash(‘success’,‘Welcome to Yelp-Camp”);

res.redirect(‘/campgrounds’);

})

} catch (e){

req.flash(‘error’, e.message);

res.redirect(‘/register’);

}))

router.get(‘/login’, (req, res)=>{

Res.render(‘users/login’);

})

router.post(‘/login’, passport.authenticate(‘local’, {failureFlash: true, failureRedirect:’/login’}), (req, res)=>{

req.flash(‘success’, ‘Welcome back!’);

const redirectUrl = req.session.returnTo || ‘/campgrounds’;

delete req.session.returnTo;

~~res.redirect(‘/campgrounds’);~~

res.redirect(redirectUrl);

})

router.get(‘/logout’, (req, res)=>{

req.logout();

req.flash(‘success’, ‘Goodbye!’);

res.redirect(‘campgrounds’);

})

module.exports = router;

-------------

## views/users/register.ejs

<% layout(‘layouts/boilerplate’)%>

<h1> Register </h1>

<form action='/register" method="POST" class="validated-form" novalidate>

username(form-control)

email(form-control)

password(form-control)

<button class=’btn btn-success”> Register</button>

</form>

-------------

==== 514. Login Routes ======

## views/users/login.ejs

<% layout(‘layouts/boilerplate’)%>

<h1> login</h1>

<form action=”/login" method="POST" class="validated-form" novalidate>

username(form-control)

password(form-control)

<button class=’btn btn-success”> Login</button>

</form>

====515. Loggin middleware =====

## routes/campgrounds.js

const {isLoggedin, isAuthor} = require(‘../middleware’)

router.get(‘/new’, isLoggedin, (req, res) =>{

~~if (!req.isAuthenticated()) {~~

~~req.flash(‘error’, ‘You must be signed in’);~~

~~return res.redirect(‘/login);~~

~~}~~

res.render(‘campgrounds/new’);

router.post(‘/’, isLoggedin, validateCampground, catchAsync(async(req, res, next)=>// create new campground.

Const campground = new Campground(req.body.campground);

Campground.author = req.user.\_id;

router.get(‘/:id/edit’, isLoggedIn, isAuthor, catchAsync(async (req, res) => // into edit campground

router.put(‘/:id’, isLoggedIn, isAuthor, validateCampground, catchAsync(async (req, res) => //Execute edit campground

const {id} = req.params;

~~const campground = await Campground.findById(id)~~

~~if (campground.author.equals(req.user.\_id)){~~

~~req.flash(‘error’, ‘No permission’)~~

~~return res.redirect(`/campgrounds/${id}`)}~~

campground = await Campground.findByIdAndUpdate(id,{…req.body.campground}

req.flash(‘success’, ‘Successfully updated campground’);

router.delete(‘/:id, isLoggedin, isAuthor, catachAsync(async (req, res)=> // delete campground

router.get(‘/:id’, catchAsync(async(req, res) =>{

const campground = await Campground.findById(req.params.id).populate(‘reviews’).populate(‘author’). Populate({

path: ‘reviews’,

populate: {

path: ‘author’})

## middleware.js

const {campgroundSchema} = require(‘./schema.js’);

const ExpressError = require(‘,/utils/ExpressError’);

const Campground = require(‘./models/campground’);

const Review = require(‘./models/review’);

module.exports .isLoggedIn =(req, res, next) =>{

console.log(“REQ.User…”, req.user);

if (!req.isAuthenticated()) {

~~console.log(req.path, req.orignalUrl)~~

req.session.returnTo = req.originalUrl

req.flash(‘error’, ‘You must be signed in’);

return res.redirect(‘/login);

}

next();

}

module.exports.isAuthor =async(req, res, nest) =>{

const {id} = req.params;

const campground = await Campground.findById(id)

if (campground.author.equals(req.user.\_id)){

req.flash(‘error’, ‘No permission’)

return res.redirect(`/campgrounds/${id}`)}

next();}

module.exports.isReviewAuthor =async(req, res, nest) =>{

const {id, review.id} = req.params;

const review = await Review.findById(reviewid)

if (review.author.equals(req.user.\_id)){

req.flash(‘error’, ‘No permission’)

return res.redirect(`/campgrounds/${id}`)}

next();}

module.exports.validateReview = (req, rev, next) =>{

const {error} = reviewSchema.validate(req.body);

if (error) {

const msg = error.details.map(el => el.message).join(‘,’)

thow nex ExpressError(msg, 400)

} else {

Next();

}

}

=====516. Adding logout ====

## partial/navbar.ejs

<div class=”navbar-nav ml-auto”>

<% if(!currentUser) {%>

<a class=”nav-link” href=”/login”> Login </a>

<a class=”nav-link” href=”/register”> Register </a>

<% else {%>

<a class=”nav-link” href=”/logout”> Logout </a>

<%}%>

</div>

==== 517. Current User helper ====

Create current user status

====518. Fixed Register route ====

A register user is still not log-in, twig to make register user attained log-in status

=====519. Return to behaviour ====

Return user to last visited page.

# Adding Authorization(52)

====520. Adding Author to Campground =====

Create an author in campground DB which reference to user. Auto assign user who create new campground as author.

## models.campground.js

const CamgroundSchema = new Schema({

title: String,

image: String,

price: Number,

description: String,

location: String,

author:{

type: Schema.Types.ObjectId,

ref: ‘User’,

},

Reviews:{

{

Type:Schema,.type.Object,

Ref: ‘Review’

}

]

## Mongo

db.users.find({username: ‘lenny’} // => copy the objectId.

## Seeds/index.js

const seedDB = async() =>{

await Campground.deleteMany({});

for (let i=0; i<50; i++) {

const random1000 = Math.floor(Math.random()\* 1000);

const price = Math.floor(Math.random() \* 20) + 10;

const camp = new Campground({

author: ‘//pasted the objectId’

location: `${cities[random1000].city}, ${cities[random1000].state}`

title: `${sample(descriptors)} ${sample(places)}`,

image: ‘https://unsplash.com”

description: ‘Lorem’

## views/show.ejs

<li class =”list-group-item”>Submitted by <%= campground.author.username%></li>

<% if(currentUser && campground.author.equals(currentUser.id)) {%>

<div class=”card-body”>

<button class=”btn btn-danger”>Delete</button> //except author delete button will hide

<% }%>

<div class=”col-6”>

<% if(currentUser){%> // only logon user can see the review

<h2>Leave a Review</h2>

<form action=”/campgrounds/<%=campground.\_id%>/reviews”

<div class=”card-body”>

<h5 class=”card-title”>Rating: <%= review.rating %></h5>

<h6 class=”card-subtitle mb-2 text-muted”> by <%=review.author.username%></h6>

<p class=”card-text”>Review: <%= review.body %></p>

<% if(currentUser && review.author.equals(currentUser.id)) {%>

<form action=”/campgrounds/<%= campground.\_id%>/reviews/<%=review.\_id%>

<button class=”btn btn-sm btn-danger”>Delete</button>

</form>

<% } %>

=== 521. Showing & Hiding Edit/Delete ======

===522. Campground permission =========

====523. Authorization Middleware =======

Restructure middleware function into middleware file

====524. Review Permission =============

User without login not supposed to see the review form or send review

## terminal

node seeds/index.js

## models/review.js

Const reviewSchema = new Schema({

Body: String,

Rating: Number,

Author: {

Type: Schema.Types.ObjectId,

Ref: ‘User’

}

});

## Routes/reviews.js

Const { validateReview, isLoggedIn, isReviewAuthor} = require(‘../middleware’)

Router.post(‘/’, isLoggedIn, sReviewAuthor, validateReview, catchAsync(async (req, res) =>{

const campground = await Campground.findById(req.params.id);

const review = new Review(req.body.review);

review.author = req.user.\_id;

router.delte(‘/:reviewId’, isloggedIn, sReviewAuthor,catchAsync(async (req, res) =>{

====525. Review Authorization =============