middleware.js

const res = require("express/lib/response")

const Campground = require('./models/campground');

const Review = require('./models/review')

const { campgroundSchema } = require('./schema');

const ExpressError = require('./utils/ExpressError');

const { reviewSchema } = require('./schema.js');

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

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

*// console.log(req.path, req.originalUrl);*

*// req.session.returnTo = req.originalUrl;*

        req.flash('error', 'You must login to access')

        res.redirect('/login');

    }

*else* {

        next();

    }

}

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

    const { id } = req.params;

    const campground = *await* Campground.findById(id);

*if* (!campground.author[0].equals(req.user.\_id)) {

        req.flash('error', 'You need to be author of this Campground to do that !!!')

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

    }

    next();

}

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

    const { id, rid } = req.params;

    const review = *await* Review.findById(rid);

*if* (!review.author[0].equals(req.user.\_id)) {

        req.flash('error', 'You need to be author of this review to do that !!!')

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

    }

    next();

}

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

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

*if* (error) {

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

*throw* new ExpressError(400, msg)

    }

*else* {

        next();

    }

}

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

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

*if* (error) {

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

*throw* new ExpressError(400, msg)

    }

*else* {

        next();

    }

}

app.js

const express = require('express');

const router = express.Router();

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

const catchAsync = require('../utils/catchAsync');

const passport = require('passport');

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

    res.render('users/register');

})

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

    const { user } = req.body;

*try* {

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

        const registerUser = *await* User.register(newUser, user.password)

        req.login(registerUser, 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) => {

    const urlRedirect = req.session.returnTo || '/campground'

    req.flash('success', 'Welcome back !!!');

    res.redirect(urlRedirect);

})

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

    req.logOut();

    req.flash('success', 'Goodbye');

    res.redirect('/');

})

module.exports = router;

routes/campgrounds.js

const express = require('express');

const router = express.Router();

const catchAsync = require('../utils/catchAsync');

const Campground = require('../models/campground');

const flash = require('connect-flash');

const { reqLogin, isAuthor, validateCampground } = require('../middleware');

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

    const campgrounds = *await* Campground.find();

    res.render('campgrounds/index', { campgrounds })

})

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

    res.render('campgrounds/new');

})

router.get('/:id/edit', reqLogin, isAuthor, catchAsync(async (req, res) => {

    const { id } = req.params;

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

    res.render('campgrounds/edit', { campground });

}))

router.put('/:id', validateCampground, reqLogin, isAuthor, catchAsync(async (req, res) => {

    const { id } = req.params;

*// console.log(id);*

    const cmpgrd = *await* Campground.findByIdAndUpdate(id, req.body.campground, { runValidators: true })

    res.redirect(`/campgrounds/${cmpgrd.\_id}`);

}))

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

    const { id } = req.params;

    const campground = *await* Campground.findById(id).populate({

        path: 'reviews',

        populate: {

            path: 'author'

        }

    }).populate('author');

*// console.log(campground);*

*if* (!campground) {

        req.flash('error', 'Campground not found');

        res.redirect('/campgrounds');

    }

    res.render('campgrounds/show', { campground });

}))

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

    const { campground } = req.body;

    const campgrounds = new Campground(campground);

    campgrounds.author[0] = req.user.\_id;

*await* campgrounds.save();

    req.flash('success', 'Successfully created new Campground');

    res.redirect('/campgrounds');

}))

router.delete('/:id', reqLogin, isAuthor, catchAsync(async (req, res) => {

    const { id } = req.params;

*await* Campground.findByIdAndDelete(id);

    req.flash('success', 'Successfully delete Campground');

    res.redirect('/campgrounds');

}))

module.exports = router;

models/user.js

const mongoose = require('mongoose');

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

const userSchema = new mongoose.Schema({

    email: {

        type: String,

        required: true,

        unique: true

    }

})

userSchema.plugin(passportLocalMongoose);

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

models/campground.js

const mongoose = require('mongoose');

const review = require('./review');

const Schema = mongoose.Schema;

campgroundschema = new Schema({

    title: String,

    price: Number,

    location: String,

    image: String,

    description: String,

    author: [{

        type: Schema.Types.ObjectId,

        ref: 'User'

    }],

    reviews: [{

        type: Schema.Types.ObjectId,

        ref: 'Review'

    }]

})

campgroundschema.post('findOneAndDelete', async function (doc) {

*await* review.deleteMany({

        \_id: {

            $in: doc.reviews

        }

    })

})

*// campgroundschema.post.findOneAndDelete,async(function(x){*

*//     review.remove(*

*//         id: {$in: x.\_id}*

*//     )*

*// })*

const Campground = mongoose.model('Campground', campgroundschema);

module.exports = Campground;

models/review.js

const { string } = require('joi');

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

reviewschema = new Schema({

    body: String,

    rating: Number,

    author: {

        type: Schema.Types.ObjectId,

        ref: 'User'

    }

});

module.exports = mongoose.model('Review', reviewschema);

routes/reviews.js

const express = require('express');

const router = express.Router({ mergeParams: true });

const Campground = require('../models/campground');

const Review = require('../models/review')

const catchAsync = require('../utils/catchAsync');

const { reqLogin, validateReview } = require('../middleware');

router.delete('/:rid', reqLogin, catchAsync(async (req, res) => {

    const { id, rid } = req.params;

*await* Review.findByIdAndDelete(rid);

*await* Campground.findByIdAndUpdate(id, { $pull: { reviews: { $in: [rid] } } })

    req.flash('success', 'Successfully delete a review');

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

}))

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

    const { id } = req.params;

    const campground = *await* Campground.findById(id);

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

    review.author = req.user.\_id;

    campground.reviews.push(review);

*await* campground.save();

*await* review.save();

    req.flash('success', 'Successfully created a review');

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

}))

module.exports = router;

routes/users.js

const express = require('express');

const router = express.Router();

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

const catchAsync = require('../utils/catchAsync');

const passport = require('passport');

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

    res.render('users/register');

})

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

    const { user } = req.body;

*try* {

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

        const registerUser = *await* User.register(newUser, user.password)

        req.login(registerUser, 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) => {

    const urlRedirect = req.session.returnTo || '/campground'

    req.flash('success', 'Welcome back !!!');

    res.redirect(urlRedirect);

})

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

    req.logOut();

    req.flash('success', 'Goodbye');

    res.redirect('/');

})

module.exports = router;

views/campground/show.ejs

<% layout('layouts/boilerplate') %>

    <div *class*="card">

        <div *class*="row" *mb-3*>

            <div *class*="col-md-6">

                <img *src*="<%= campground.image %>" *class*="card-img-top" *alt*="...">

                <div *class*="card-body">

                    <div *class*="card-title">

                        <h4>

                            <%= campground.title %>

                        </h4>

                    </div>

                    <ul *class*="list-group list-group-flush">

                        <li *class*="list-group-item">

                            <%= campground.location %>

                        </li>

                        <li *class*="list-group-item"> Submitted by <%= campground.author[0].username %>

                        </li>

                        <li *class*="list-group-item">$<%= campground.price %> per day</li>

                        <li *class*="list-group-item">

                            <%= campground.description %>

                        </li>

                    </ul>

                    <% *if* (currentUser && campground.author[0].equals(currentUser.\_id)) { %>

                        <form *action*="/campgrounds/<%= campground.\_id %>?\_method=DELETE" *method*="POST">

                            <button *class*="btn btn-danger">Delete</button>

                        </form>

                        <% } %>

                </div>

            </div>

            <div *class*="col-md-6">

                <div *class*="text-center">

                    <h2>Leaves a Review</h2>

                </div>

                <% *if* (currentUser) {%>

                    <form *action*="/campgrounds/<%= campground.\_id %>/reviews" *method*="POST" *class*="validate-form"

*novalidate*>

                        <div *class*="card-text mb-3">

*<!-- <div class="mb-3"> -->*

                            <label *for*="rating" *class*="form-label">

                                <h5>Enter a Rating</h5>

                            </label>

                            <input *type*="range" *class*="form-range" *min*="1" *max*="5" *id*="rating" *name*="review[rating]">

                        </div>

                        <div *class*="card-text mb-3">

                            <label *for*="body" *class*="form-label">

                                <h5>Write a review</h5>

                            </label>

                            <textarea *name*="review[body]" *id*="body" *cols*="30" *rows*="5" *class*="form-control "

*required*></textarea>

                        </div>

                        <div *class*="card-text mb-3">

                            <button *class*="btn btn-primary">Submit</button>

                        </div>

                    </form>

                    <% } %>

                        <% *for*(let review of campground.reviews) {%>

                            <div *class*="card-body">

                                <h5 *class*="card-title mb-3">

                                    Rating : <%= review.rating %>

                                </h5>

                                <div *class*="card-subtitle mb-3">

                                    Contributted by : <%= review.author.username %>

                                </div>

                                <div *class*="card-text mb-3">

                                    Review : <%= review.body %>

                                </div>

                                <% *if* (currentUser && review.author.equals(currentUser.\_id)) {%>

                                    <form

*action*="/campgrounds/<%= campground.\_id %>/reviews/<%= review.\_id %>?\_method=DELETE"

*method*="POST" *mb-3*>

                                        <button *class*="btn btn-danger btn-sm">Delete</button>

                                    </form>

                                    <% } %>

                            </div>

                            <% } %>

            </div>

        </div>

    </div>

    </div>

*<!-- <h1>*

*<%= campground.title %>*

*</h1>*

*<h2>*

*<%= campground.location %>*

*</h2>*

*<p>*

*<img src="<%= campground.image %>" alt="">*

*</p>*

*<p>*

*$<%= campground.price %> per day*

*</p>*

*<p>*

*<%= campground.description %>*

*</p>*

*<form action="/campgrounds/<%= campground.\_id %>?\_method=DELETE" method="POST">*

*<button>Delete</button>*

*</form>*

*<footer>*

*<p>*

*<a href="/campgrounds">Back to Full List</a>*

*</p>*

*<p>*

*<a href="/campgrounds/<%= campground.\_id %>/edit">Edit Campground</a>*

*</p>*

*</footer> -->*

views/campground/edit.ejs

<% layout('layouts/boilerplate') %>

    <div *class*="row">

        <div *class*="col-md-6 offset-3">

            <h1 *class*="text-center">

                Edit Campground

            </h1>

            <form *action*="/campgrounds/<%= campground.\_id %>?\_method=PUT" *method*="POST" *novalidate*

*class*="validate-form">

                <div *class*="mb-3">

                    <label *for*="title" *class*="form-label">

                        Campground title

                    </label>

                    <input *type*="text" *class*="form-control" *id*="title" *name*="campground[title]"

*value*="<%=campground.title %>" *required*>

                </div>

                <div *class*="mb-3">

                    <label *for*="location" *class*="form-label">

                        Location

                    </label>

                    <input *type*="text" *class*="form-control" *id*="location" *name*="campground[location]"

*value*="<%=campground.location %>" *required*>

                </div>

                <div *class*="mb-3">

                    <label *for*="image URL" *class*="form-label">

                        Image URL

                    </label>

                    <input *type*="text" *class*="form-control" *id*="image" *name*="campground[image]"

*value*="<%=campground.image %>" *required*>

                </div>

                <div *class*="mb-3">

                    <label *for*="price" *class*="form-label">

                        Campground Price

                    </label>

                    <div *class*="input-group">

                        <span *class*="input-group-text">$</span>

                        <input *type*="number" *class*="form-control" *name*="campground[price]" *aria-label*="Price per Night"

*id*="price" *value*="<%=campground.price %>" *required*>

                        <span *class*="input-group-text">.00</span>

                    </div>

                </div>

                <div *class*="mb-3">

                    <label *for*="description" *class*="form-label"> Campground Description</label>

                    <textarea *name*="campground[description]" *id*="description" *cols*="30" *rows*="5" *class*="form-control"

*required*>

                    <%=campground.description %>

                </textarea>

                </div>

                <% *if* (currentUser && campground.author[0].equals(currentUser.\_id)) {%>

                    <div *class*="mb-3">

                        <button *class*="btn btn-success">Submit</button>

                    </div>

                    <% } %>

            </form>

        </div>

    </div>

    //

*<!-- <form action="/campgrounds/<%= campground.\_id %>?\_method=PUT" method="POST">*

*<div>*

*<label for="title">Campground title</label>*

*<input type="text" id="title" name="campground[title]" value="<%=campground.title %>">*

*</div>*

*<div>*

*<label for="image">Image</label>*

*<input type="text" name="campground[image]" id="image" value="<%=campground.image %>">*

*</div>*

*<div>*

*<label for="location">Location</label>*

*<input type="text" name="campground[location]" id="location" value="<%=campground.location %>">*

*</div>*

*<div>*

*<label for="price">Price</label>*

*<input type="number" name="campground[price]" id="price" value="<%=campground.price %>">*

*</div>*

*<div>*

*<label for="description">Description</label>*

*<textarea name="campground[description]" id="description" cols="30" rows="10">*

*<%=campground.description %>*

*</textarea>>*

*</div>*

*<button>Submit</button>*

*</form>*

*<footer>*

*<p>*

*<a href="/campgrounds">Back to All Campgrounds</a>*

*</p>*

*</footer> -->*