Image Upload

1. regular HTML does not allowed to upload image
2. Don’t put image on Mongo, Bson size limit of 16MB
3. GridFS can be utilized to save larger file.
4. Use Cloudinary to store image file (cloud services). paid tools which provide various services inclue scale & cropp the image. has a free tier. Other tools like AWS.
5. Mongo will only store the URL of image stored in Cloudinary.
6. HTML used application/ form-urlencoded enctype. to be able to upload image require “mutipart/form-data enctype
7. To use enctype “mutipart/form-data”, a middleware extension needed – “Multer”
8. Multer add a body object & a file object to the request object. Body object contains the text fileds of the form, the file object contain the file upload via the form.
9. Need Cloudinary credential to use their services.
10. install dotenv to conceal secret info (credential) for Cloudinary
11. Install Multer storage cloudinary

======= 532 The Multer Middleware ======

## views/campgrounds/new.ejs

<form action=”/campgrounds” method=”POST” novalidate c;ass=”validated-form” **enctype=”mutipart/form-data**”>

**<input type=”file” name=”image” id=”” multiple>**

**<div class=”mb-3”>**

**<label for=”’>Upload images</label>**

<div class=”form-file~~”~~ **custom-file**”>

<input type=”file” calss=”form-file-input” id=”image” **name=”image” multiple**>

**<label class=”form-file-label” for=”image”>**

<span class=”form-file-text **custom-file-label**”> Choose image(s)…</span>

<span class=”form-file-button”>Browse</span>

## Terminal

npm I multer

npm I dotenv

npm I cloudinary multer-storage-cloudinary

/routes/campgrounds.js

**const multer = require(‘multer’);**

**~~const upload = multer{dest: ‘uploads/’}~~**

const {storage} = require(‘../cloudinary’);

const upload = multer({storage})

router.route(‘/’)

**~~.post(upload.single(‘image’),(req.ews) =>{ //.post(upload.array(‘image’),(req.ews)-> for upload multiple files~~**

~~console.log(req.body, req.files)~~

~~res.send(req.body);~~

.post(isLoggedIn, validateCampground, upload.array(‘image’), catchAsync(campgrounds.createCampgroound)

=====534 environment variable with dotenv =========

Create an .env file to store secret & API key for Cloudinary

## app.js

if (process.env.NODE\_ENV !== “production”){

require(‘dotenv’).config(); *//for dotenv to use in development mode.*

console.log(process.env.SECRET)

console.log(process.env.API\_KEY)

## .env

CLOUDINARY\_CLOUD\_NAME=

CLOUDINARY\_KEY=

CLOUDINARY\_SECRET=

==== 535 Upload to cloudinary basics ======

Install Multer Storage Cloudinary for working with Multer & cloudinary in uploading & retreiving files to/from Cloudinary with Multer middleware.

## /cloudinary/index.js

const cloudinary = require(‘cloudinary).v2

const {CloudinaryStorage} = require(‘multer-storage-cloudinary’);

cloudinary.config({

cloud\_name: process.env.CLOUDINARY\_CLOUD\_NAME,

api\_key: process.env.CLOUDINARY\_KEY,

api\_secret: process. env.CLOUDINARY\_SECRET

]);

const storage = new CloudinatyStorage({

cloudinary,

params: {

folder: ‘yelpcamp’,

allowedFormats: [‘jpeg’, ‘png’, ‘jpg’]

}

});

module.exports = {

cloudinary,

storage

}

==== 536 Storing uploaded image links in Mongo=====

## /model/campground.js

const imageSchema = new Schema({

url: String,

filename: String

})

imageSchema.virtual(‘thumbnail’).get(function () {

return this.url.replace(‘/upload’, ‘/upload/w\_200’);

});

const CampgroundSchema = new Schema({

title: String,

images:[ imageSchema

~~{~~

[~~url: String~~](file:///C:\Users\Lenovo\Downloads\String)~~,~~

~~filename : String~~

~~}~~

],

## /controllers/campgrounds.js

const {cloudinary} = require(“../models/camptround’);

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

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

campground.images = req.files.map(f =>({url: f.path, filename: f.filename}))

campground.author = req.user.\_id;

await campground.save();

console.log(campground)

req.flash(‘success’, ‘Successfully made a new campground!’);

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

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

const { id} = req.params;

const campground = await Campground.findbyIdAndUpdate(id, {…req.body.campground})

const imgs = (req.files.map(f =>({url: f.path, filename: f.filename}))

campground.images.push(…imgs); //**Convert array into strings b4 push into array.**

await campground.save()

if (req.bpdy.deleteImages) {

for(let filename of req.body.deleteImages) {

await cloudinary.uploader.destroy(filename);}

await campground.updateOne({$pull: {images: {filename: {$in: req.body.delteImages}}

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

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

## schema.js

module.exports.campgroundSchema = joi.object({

campground: joi.object({

title: joi.string().required(),

price: joi.number().required().min(0),

~~image: joi.string().required(),~~

location: joi.string(),

description: joi.string.required()

deleteImages: joi.array()

## views/campgrounds/show.ejs

<% for(let campground of campgrounds) {%>

<div class=”card” mb-3>

<div class=”row”>

<div class=”col-md-4”>

<img class=”img=fluid” alt=”” src=”<% =campground,images[0].url %”>

<div id="*campgroundCarousel*" class="carousel slide carousel-fade" data-bs-ride="carousel">

<div class="carousel-inner">

<% campground.images.forEach((img, i) => { %>

<div class="carousel-item <%= I === 0 ? ‘active’: ‘%>”>

<img src="<%= img.url%>" class="d-block w-100" alt="...">

</div>

<% } %>

<div class="carousel-item">

<img src="..." class="d-block w-100" alt="...">

</div>

<div class="carousel-item">

<img src="..." class="d-block w-100" alt="...">

</div>

</div>

<% if (campground.images.length > 1) {%> //**display arrow only if more than 1 image**

<button class="carousel-control-prev" type="button" data-bs-target="*campgroundCarousel* " data-bs-slide="prev">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

<span class="visually-hidden">Previous</span>

</button>

<button class="carousel-control-next" type="button" data-bs-target="*campgroundCarousel* " data-bs-slide="next">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

<span class="visually-hidden">Next</span>

</button>

<% } %>

</div>

<div class=”card mb-3”>

<% for(let img of campground.images) {%>

<img src=”<%= img.url%>” class=”card-img-top” alt=”…”>

<% } %>

====== 537 Display image in carousel ====

Copy carousel code from bootstrap5 to views/campgrounds/show.js

======538 fixing seeds =====

## seeds/index.js

const camp = new Campground({

…

images: [

{

~~\_id: 8790…….~~

Copy the images url & filename & pasted on every campground.

## views/campgrounds/index.ejs

<div class=”row”>

<div class=”col-md-4”>

<img class=”img-fluid” alt=”” src=”<%=campground.images[0].url %”>

=== 539 adding upload to edit page =====

Enable user to edit image file

## /views/campgrounds/edit.ejs

<form action=”/campgrounds/<%=campground.\_id%>?\_method=PUT” method=”POST” novalidate enctype=”multipart/form-data”>

~~<div class=”mb-3”>~~

~~<label class=”form-label” for=”image”>Add images</label>~~

~~<input type=”file” name=”image” id=”image” multiple>~~

<div class=”form-file~~”~~ **custom-file**”> (Copy from new.ejs)

<input type=”file” calss=”form-file-input” id=”image” **name=”image” multiple**>

**<label class=”form-file-label” for=”image”>**

<span class=”form-file-text **custom-file-label**”> **Add more image(s)…</**span>

<span class=”form-file-button”>Browse</span>

<div class=”mb-3”>

<% campground.images.forEach(function(img, i) { %>

<img src=”<%=~~img.url~~ img.thumbnail%>” class=”img-thumbnail”>

<div class=”form-check-inline”>

<input type =”checkbox” id=”image-<%=i%> name=”deleteImages[]” value=”<%=img.filename”>

</div>

<label for=”image-<%=i%”> Delete?</label>

<%})%>

<div class=”mb-3”>

<button class=”btn btn-info”>update Campground</button>

</div>

## /routes/campgrounds.js

router.route(‘/’)

.post(isloggedIn, upload.array(‘image’), validateCampground, catchAsync(Campgrounds.createCampground) //upload array must b4 validateCampground to produce body for validation

router.route(‘:/id’)

.put(isLoggedIn, isAuthor, upload.array(‘image’),validateCampground, catchAsync(campgrounds.updateCampground)

======540 customized file input =======

use “bs-custom-file-input” extension to help improve the function of input 🡺 Copy the link under boilerplate.(To display multiple filename on the file input dropdown)

* put the cdn link on the boilerplate
* put the statement “bsCustomFileInput.init()” under validatedForms.
* On the new.ejs change the input class into “custom-file” and “custom-file-label”



<div class="input-group">

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

<span class="input-group-text" id="inputGroupFileAddon01">Upload</span>

</div>

<div class="custom-file">

<input type="file" class="custom-file-input" id="inputGroupFile01"

aria-describedby="inputGroupFileAddon01">

<label class="custom-file-label" for="inputGroupFile01">Choose file</label>

</div>

</div>

## public/javascripts/validateForms.js

bsCustomFileInput.init()

==== 542 Delete image form =====

=====543 Delete image backend ====

====- 544 Adding thumbnail virtual property=====