**MEN** Stack for Web App

Steps to making a web app using Node Express and Mongodb on MacOS 10.14.5

1. Installing Mongodb
   1. brew update
   2. brew install mongodb
   3. sudo mkdir -p /data/db makes directory for data
   4. sudo chown -R `id -un` /data/db gives permission to user
   5. mongod, waits for connection and ports
2. Setting up app.js and express using Node and NPM
   1. Npm init //name project different than any node packages
   2. Npm install <name of package> --save //express
   3. Optional: npm i -g (global) nodemon //restart server automatically when changes are made
   4. Starting server]
      1. This line creates server on localhost:3000
      2. *var* listener = app.listen(3000);

OR

*var* port = process.env.PORT || 3000;

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

*console*.log("Server Has Started!");

});

* + 1. This was a code for the cloud version
    2. app.listen(process.env.PORT, process.env.IP, ()*=>*{
    3. *console*.log("SERVER HAS STARTED");
    4. });
  1. Require packages
     1. *const* express = require("express"),
     2. app = express();
  2. Connecting to Mongoose
     1. mongoose.connect("mongodb://localhost:27017/house\_manager", {useNewUrlParser: true});

Description of certain features

1. Node.js
   1. app.set(“view engine”, “ejs”); so that the app knows to render .ejs files
   2. app.set(express.static(\_\_dirname + "/public")); so that style sheets are seen
   3. Things stored in the url route as a variable like /comment/:id id can be accessed by using req.params.id
   4. Modularize routes by putting them into their own js files
   5. In app.js use “app.use(name\_of\_routes);
   6. Forms need a post
      1. Form should send to route where items are being updated
   7. Rendered html pages need a get
   8. UPDATE/replace
   9. Method override npm install method-override --save
   10. ?\_method=PUT method = post
   11. Delete needs to be a from with action and method = post but with ?\_method=DELETE
   12. Middleware and req res next can be added to routes before the function callback
   13. If you have two id’s in the route, make them different.
       1. Example: campgrounds/:id/comments/:comment\_id/edit
       2. GET/POST/PUT/UPDATE
2. Checking stuff in database using mongo
   1. Show dbs
   2. Use <db name>
   3. Show collections
   4. db.<collection name>.find()
3. Be sure to include bootstrap and your own style sheet in the header.ejs in partials
4. Recommended Directories
   1. Images, middleware, models (mongoose), public (express specific i think), routes, views (also express specific)
5. Express.js
   1. Html template
      1. <%= %> shows javascript
      2. <% %> only runs js logic
   2. /:anything\_here\_is\_a\_variable that can be used in the route req.params.variable name
6. Mongoose
   1. Schema before models
   2. Models
   3. findbyidandupdate(req.params.id, req.body.name of array, callback func()=>
   4. Findbyidandremove
   5. Checking if user matches author example
      1. found Campground,author.id = mongoose obj
      2. Req.user.\_id = string
      3. Mongoose has a .equals method
   6. Create is equivalent to new and save
   7. ***.populate(“collectionsname”).exec***
      1. Puts the actual contents of collection into the list under the parent model, replacing the id
7. Node Modules
   1. Ejs
      1. Template for using javascript in html
   2. Express
      1. minimal and flexible Node.**js** web application framework
      2. light-weight web application framework to help organize your web application into an MVC architecture on the server side
   3. Express-sessions
      1. For creating user sessions like logins
   4. Dotenv
      1. Creates environment variables so when people see the code in the repo they can’t see passwords, also sets globals
      2. Process.env.VARIABLENAME
      3. Can be set in terminal using export VARIABLENAME=blahblahblah
   5. Body-parser
      1. app.use(bodyParser.urlencoded({}));
      2. Need for req.params.body (form returns)
   6. Passport for authentication (passport, passport-local, passport-local-mongoose)
      1. app.use(require(“express-session”)({

Secret: “idk what this does lol”, //used for hashing, the salt part

secret: "Once again Rusty wins cutest dog!",

resave: false,

saveUninitialized: false

}));

app.use(passport.initialize());

app.use(passport.session()); //need this so that you “stay” logged in

passport.use(new LocalStrategy(User.authenticate()));

passport.serializeUser(User.serializeUser());

passport.deserializeUser(User.deserializeUser());

1. Req.user to access user that is logged in
   1. methodOverride
      1. ?\_method=PUT method = post for update route
      2. ?\_method=DELETE method = post for delete route
   2. Async
      1. To use promises and async await functions to reduce callback
      2. Waterfall, better format of callbacks
   3. Cloudinary with Multer
      1. Image/pdf upload system from forms in html
   4. Nodemailer
      1. Allows for authentication through the app to send an email through your email.
      2. Password reset using tokens.
   5. Moment
      1. Used for timestamping models that get created like comment posts.
   6. Connect-flash
      1. Put the flash in the header.ejs so its on every page
      2. **Make flash before the redirect**, and do not render!
      3. you can either set a flash message on the req.flash object before returning a res.redirect() or you can pass the req.flash object into the res.render() function.
      4. if(err){

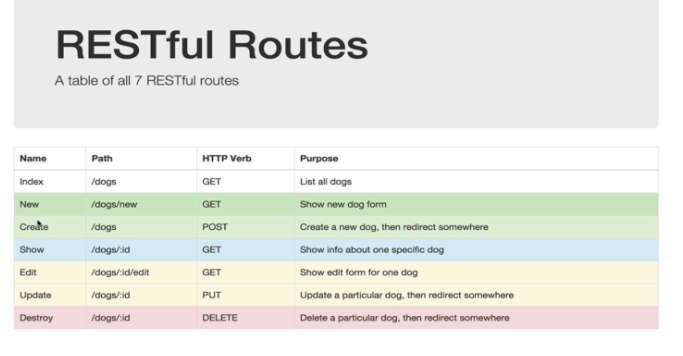
req.flash("error", err.message);

return res.redirect("/register");

}

1. HTML/CSS tips
   1. Display: inline;
   2. Give form an id and goto css and style it
   3. Forms: Value prefills the text area, where PLACEHOLDER only gives a shadow
   4. *Forms needs action which is route and method like post/get*
   5. *Need name in input in form in html*
2. JavaScript tips
   1. Condition && nextCondition
   2. Condition will terminate the entire if statement if condition is false so that the nextCondition won’t event run.

**Restful Routes:**



**House Manager Models needed:**

House:

String: Address, House Manager, Phone number, Rent, Image

Author: { Id:{Type: mongoose.Schema.Types.ObjectId,Ref: “User”},

Username: String}

Comments: [{Type: mongoose.Schema.Types.ObjectId,Ref: “Comment”}]

User:

First name, last name, username, password, email, avatar

Comments:

String: text,

Author: { Id:{Type: mongoose.Schema.Types.ObjectId,Ref: “User”},

Username: String