* My website is running on an express server. It has GET requests to render the webpages when the corresponding queries are entered (‘/’, ‘/home’, ‘/login’, etc). this means that since the website is currently being hosted on localhost:3000 if localhost:3000 is loaded then the base page is rendered (‘/’), when localhost:3000/home is loaded then the home page is rendered (‘/home’), etc
* The login system uses the node library called passport. One of the things that I did with this was create functions called checkAuthenticated and checkNotAuthenticated. When checkAuthenticated is called in the GET request for a page that requires a logged-in user to access (‘/inspire’, ‘/water’, ‘/home’, ‘/about’), it uses the isAuthenticated method which came from passport to determine the login status. If the user is logged in it allows the GET request to go through. Otherwise, it redirects to ‘/login’. And the same for checkNotAuthenticated for pages that can only be accessed if not logged in (‘/’, ‘/login’, ‘/register’), but there if isAuthenticated is true then the user is redirected to ‘/home’. Otherwise, the GET request is allowed to be continued.
* The view engine is set to EJS and all of the markup code is EJS instead of HTML to allow for if statements. This is used in the login code to create an error message popup in case of an error.
* The bcrypt node module is used for the login system to ensure security. When the POST request is sent from the register file, before saving the password, a variable is created which hashes the password directly from the request. This makes it so that at no point is their actual password stored in the server, creating a secure system.
* The method-override node module is used to allow for an HTTP DELETE request on the logout button on the ‘/home’ page, despite HTML (and EJS) only allowing GET and POST
* On the register page, there is a password generator. This has options to set the length and what kind of characters should be included. When the button is clicked to generate the password, it randomizes in the chosen character types for the chosen length, thus creating a random password
* On the Udemy project, the button with the clipboard on it was meant to be used to copy the generated password to the clipboard. But the DOM method which they used for it was no longer functional. Online, I was not able to find a replacement that worked to copy to the clipboard directly. I did find someone who wrote a function to copy the text in a given element. So in an event listener on the clipboard element, I made it so that when clicked, the text in the result element of the password generator will be selected, allowing the user to easily copy from there.
* On the list page (‘/inspire’), users are invited to inspire themselves by setting goals for themselves. There is an input form to enter their personal goals. When the form is submitted, an li element is added to the originally empty ul element below the form
* There are event listeners on each li, making it so that in the event of a left click, the completed class on the li will toggle, and in the event of a right click, the li element will be removed. The completed class, when on, has a text-decoration of line-through and a lighter color
* On the water tracking page, there is a larger cup that represents the total water drank and there are 8 smaller cups. Clicking on one of the empty cups will add the full class to that cup and all of the empty cups before it. Clicking on a full cup will remove the full class from all of the cups after it. If there are no full cups after it, it removes the full class from the cup which was clicked on