To practice all the topics from this week and last week, I tried to write and run the to do list and the http request examples from the lectures. My goal was to purposefully break this or that portion of the code so I could understand the effect those specific lines of code have on the overall application.

The first error I ran into was “EADDRINUSE”. This meant that the port I was using was already in use. So, I went in and changed the port. Then, last week when learning Node.js, I did not have to type in my username and password for github.com. This week, I had to do it everytime I made a git pull request from Putty. This was a nuisance which I was able to solve by changing the repository from private to public.

Next to practice session, I ran the inSessions.js page with tweaks. First, I commented out “app.use(session())”. This broke the entire intSessions.js page. This means “app.use(sessions())” is vital to using sessions in the application. Not only that a secret, as in “app.use(sessions({secret: “some secret”});”, is required to run sessions. Furthermore, secret needs to have some value. Specifically, it’s value needs to be a string, a number will not work.

I ran into most trouble when destroying sessions. I could not figure out where to place the destroy session code without destroying the functionality of the entire web application. It is not logical to add “req.session.destroy()” within the route, because when a user is adding their name and then adding items to the to do list, their session has to stay intact. In this scenario, the only place where a session can end is if the user deliberates wants the session to end. Say the user is walking away from their computer and they do not want other to access their to do list on the same browser. So, I added an “End Session” button and in it I added the destroy session functionality. This worked, when the user clicks on the End Sessions button, the session is destroyed. But, it is not perfect. The user has to manually refresh the page to wipe all the text from their session.

From sessions I moved onto testing HTTP requests, POST data, and nested asynchronous calls. Following along the code for helloHttp.js and helloOrganization.js was intuitive. We have already completed assignments for each of these concepts on their own. We have made asynchronous get and post calls. The only new item this week is that now we are combining both techniques into one route. That is, we are sending GET request, POST request, and only after we have received response from our request, we will render the page, which will contain data received from our request. With a fair bit of back and forth I was able to write programs that were similar to helloHttp.js and HelloOrganization.js. I found that separating each callback into its own function alleviated a lot of confusion that comes with nesting multiple function into one.

Lastly, I wanted to practice destroying sessions, and here is where I ran into most trouble. I could not figure out where to place the destroy session code.

Then, I removed “secret:” from sessions(). I After running the intSessions.js file

The first problem had to do with setting up a .gitignore file. At first I did not add this file and did not exclude node modules and credentials.js file from uploading to github.