Shalom Dawit

**14: Reverse Engineering Authentication**

Starting from the top to bottom I will try to explain the files in this project.

**Config:**-This folder has a child folder that contains *'isAuthenticated.js'.* This file is the middleware that allows access to the user verified site(s). Otherwise the file will redirect the user to the initial domain.

-The ‘*config.json*’ file holds the entrance credentials to our database when using either the development environment, testing environment or production environment.

-The ‘passport.js’ file holds the functions for how the developer wants passport to work in the context of this project.

**Models**: -This folder has two child files. The *user.js* file has the sequelize setup for the db holding user info. Bcrypt is also being used to hash the password so that users’ passwords are kept private even in the db.

-The ‘index.js’ file connects to the sql db using the info in the config.json file depending on which environment is in play. This file connects the contents of the db with the variables and functions in the rest the project

**Public:** This folder has two child directories and three child files. This folder is usually browser default on what is displayed on the page. Looking at the html files, the signup page displays the form to enter the new user information that will be saved(if done correctly) into the db. You also have a choice to click a *‘login*’ button which directs the user to the login html where the user enters their credentials that are already saved into the user db. The third file, welcome the user if they have correctly jumped through the hoops!

**JS**: This folder holds the front-end javascript files that interact with the user. The login.js file uses an event listener to save the input data from the user. The post request then takes that data (saved to a variable) to an api route on the domain. There are also other branches of actions if these things are successful or not successful. This is very similar to the signup.js file. The *members.js* displays dynamically the user name by using a get request.

**Stylesheets:** This folder contains the css file for styling the html files in the parent directory.

**Routes:** This folder deals with the back-end(server-side) routes. There are two files (law & order narrator’s voice). One that uses the data that the user inputs to tell the browser the path to go to and which html file to use on that path (*html-routes.js*). The other one has post requests and get requests that use the api paths to post data to them and get data from them as specified by their relative requests functions (*api-routes.js*).

The two files in the main directory of this project are server.js and package.json. Server.js the file that starts up the actions of the project and the order of said actions. Express is used to connect this project’s server to the user. Lastly, lines 25-30 is where the db is connected to the communication that will be going on between the server and the user. The rest of the file tells the browser what to look for on displaying the pages, connecting the routes that were defined in the routes folder and the connection credentials (defined in the config folder) and the passport module that sets up the authentication for the users (defined in the config folder).

The package.json file holds details of the project, instructions for node.js to install which modules, and the scripts for node command shortcuts along with file execution.

In conclusion, my future development would be to connect this project to an app that needs login authentication because none of that is included with this project. Also I would fix the promises issue and errors that is going on with this project.