HW14 **Reverse Engineering Code**

This is an application and starter code for a sign up and login process.

As a NEW USER I can SIGN UP by inputting my email address and password, so that I can ACCESS THE WEBSITE.

As a NEW USER I can be REDIRECTED TO THE LOGIN PAGE, so that I can SIGN UP if I am a new user.

As an EXISTING USER I can LOG IN by inputting my email address and password that I inputted when I signed up, so that I can ACCESS THE WEBSITE.

The code file structure has been created using Sequelize Command Line Interface (CLI). The directories and files are explained below.

* Config
  + Middleware
    - isAuthenticated.js = This file establishes a middleware route that will route the user to the login page if they haven’t logged in yet.
  + Config.json = This file stores the connection details to our mysql database. This is how we’ll connect to our database since we’re using Sequelize.
  + passport.js = This file contains a function that validates the user’s email address and password.
* Models
  + index.js = This file is created by the Sequelize CLI. It collects all of the models in the models directory and associates them as needed. Also lets us reference the models directory when we setup api routes to communicate with our database and tables.
  + user.js = Sets up the model for the User table within the database. This file tells Sequelize things about the table it represents. It sets up the table, columns, and data to be stored in the table.
    - I’m not sure if storing the password validation method on the user.js model is an accepted practice.
* Node\_modules = Stores the external node modules that are required for this project.
* Public
  + Js
    - Login.js = This file is responsible for the frontend interaction and form setup on the login page. It also clears the data entered in the form. This file depends on the api-routes.js file to serve the login.html and login.js files. This file also depends on the login.html file.
      * There’s an API route included in this file for the signup api route. I don’t think this route should be included in this file. I think it should be moved to the api-routes.js file.
    - Member.js = This file does a GET request to validate the user that is on the page and display the correct HTML. This file depends on the members.html file.
    - Signup.js = This file is responsible for the frontend interaction and form setup on the signup page. This file depends on the signup.html file and the api-routes.js file.
  + Stylesheets
    - Style.css = This file is responsible for all of the HTML styling in all three HTML Files. This file depends on the HTML files.
  + Login.html = This file is responsible for the HTML content on the login page. This file depends on the html-routes.js file to serve the HTML content to the html route. This file also depends on the style.css file for styling and the login.js for the interactive functionality on the page.
  + Members.html = This file is responsible for the HTML content on the members page. This file depends on the html-routes.js file to serve the HTML content to the html route. This file also depends on the style.css file for styling and the members.js for the interactive functionality on the page.
  + Signup.html = This file is responsible for the HTML content on the signup page. This file depends on the html-routes.js file to serve the HTML content to the html route. This file also depends on the style.css file for styling and the signup.js for the interactive functionality on the page.
* Routes
  + Api-routes.js = This file is responsible for the data that can be displayed, created, updated or deleted from each site. This file depends on the models directory to determine how the data will be stored in the sql table and how the data will be displayed to the user. This file also depends on the passport.js file in the config folder to validate a user when they login.
  + Html-routes.js = This file is responsible for the HTML content (html file) that is displayed when a user hits a defined route. This file depends on the html files that will have the html content, the style.css file to server the styling of the html content, and the js files to handle the frontend interaction of the sites.
* Package-lock.json = Automatically generated for any directory where the Node packages are modified.
* Package.json = This file includes the information on the Node packages we’re using.
* Server.js = This file is responsible for setting up our connection to our Node server. This will allow us to display our files/application at the local host.

**Improvements**

Now that you understand the application and the starter code, you can make the following improvements:

1. Replace variables defined as ‘var’ with ‘const’.
2. Add alerts to let the user know if they need to sign up or log in.
3. Move the isAuthenticated.js file and Middleware directory to the routes directory.
4. Move the passport.js file to the public > js directory.