Trilogy Full Stack Flex Coding Bootcamp - University of Texas at Austin

14-Reverse Engineering Code

Tutorial for reverse engineering starter code

Purpose

As a developer, I want a walk-through of the codebase so that I can use it as a starting point for a new project.

App Summary

This app has a viewable database that a user can access when signed in.

File Summary:

Server.js

- Initializes the express server on port 8080
- Sets up the session variable for utilization
- Sets up the passport variable for utilization
- Creates the express app
- Creates the db variable to sync the models folder files
- Configures the middleware required for authentication
- Requires the html and api routes files and the app variable
- Syncs the database using sequelize middleware
- Logs a message to the user upon successful initialization

Config Folder:

Middleware - isAuthenticated.js

- Limits routes allowed when not signed in and redirects routes when not signed in

Config.json

- Creates database connection for dev, test, and production

Passport.js

- Sets up the passport variable
- Sets up the LocalStrategy variable
- Sets up the db variable to require models folder files
- Sets up sign in strategy:
    - On user sign in, code runs to determine:
        - If there is no user with the provided email, returns a message: “Incorrect Email”
        - If there is a user with the given email, but wrong password provided, returns a message: “Incorrect Password”
        - If neither of the above, it just returns the user
- Sequelize serializes and unserializes the user for the authentication state
- Exports the module = passport (which allows the LocalStrategy)

Public Folder:

JS folder

Login.js:

- References the login form, and reads the email and password inputs
- On submission, it validates if there’s an email and password entered
    - If an email and password have been entered, it runs the loginUser function and clears the form
    - loginUser does a post to the “api/login” route, which if successful, redirects the user to the members page.
        - If there is an error, it logs the error

Member.js:

- Makes a “get” request to figure out which user is logged in
    - Updates the HTML on the page

Signup.js:

- References the login form, and reads the email and password inputs

- On click of the signup button, it validates the email and password fields have text entries
    - If the required fields are populated, it runs the signUpUser function, which:
        - Does a post to the signup route.
            - If successful, it directs to the members page
            - If not successful, it throws a bootstrap alert

Stylesheets folder

Style.css:

- Provides top margin css for the signup and login form

Login.html:

- Provides HTML for the login page
- Dedicates this as the login page, tying it to the login.js file
- Houses the Login Form
- Has submit button to submit login info
- Has alternative route if user sign up is required

Members.html:

- Provides HTML for the members page
- Dedicates this as the members page, tying it to the members.js file
- Houses the Logout link (currently non-existing file)
- Provides a welcome message for the user with their name

Signup.html:

- Provides HTML for the signup page
- Dedicates this as the signup page, tying it to the signup.js file
- Houses the signup form
- Has submit button to submit sign up information
- Has alternative route if user login is required instead of signup.

Routes Folder:

Api-routes.js:

- Requires sequelize files (models) and passport file
- Sets up a module.exports function which takes in “app” as a parameter, to validate login credentials and send them to the members page
    - Otherwise, the user will be sent an error
- Provides a route for signing up a user
    - Securely stores the user’s password, and logs the user in
    - Sends an error if the user is not created successfully
- Provides a route for user log out
- Provides a route for using user data on the client side
    - If the user is not logged in, it sends an empty object back
    - If the user is logged in, it sends the user’s id and email (not password)

Html-routes.js:

- Requires path middleware to use relative routes to the HTML files
- Requires custom middleware for checking if a user is logged in
- When on the initial page, it sends the user to the members page if they already have an account
- When on the login page, it redirects the user to the members page if they already have an account
- Uses the “isAuthenticated” middleware to redirect a person to the signup page if they are not logged in and they try to access the “members” page.

Future Development

- A log out form and html page need to be developed to allow the user to log out using the currently existing link (for the currently non-existing form)
- Update CSS to make it more user friendly and visually appealing