Reverse Engineering Code

— PASSWORD AUTHENTICATION —

Allows users to create an account, log into the account and sign back out securely. All user data is stored in a mysql database.

— USER STORY —

Users who want to safely log in to "X", I want to know my personal details are safely stored so that I don't have to worry about using "X".

— TECH USED —

- BCRYPTJS
- EXPRESS
- EXPRESS-SESSION
- MYSQL2
- PASSPORT
- LOCAL
- SEQUELIZE

— GETTING STARTED —

When using this app, clone this repository into your local storage. Once this is complete, please follow these steps;

1. Create a mysql db called "passport_demo"
2. Go into the config file, open config.js and insert your personal data ie username, password etc.
3. Open terminal in current repo and run "npm i" to install all node packages
4. While in terminal, run "node server.js" and you will successfully connect to server
5. Open browser and put "http://localhost:8080" in search bar

6. Enjoy using the app!

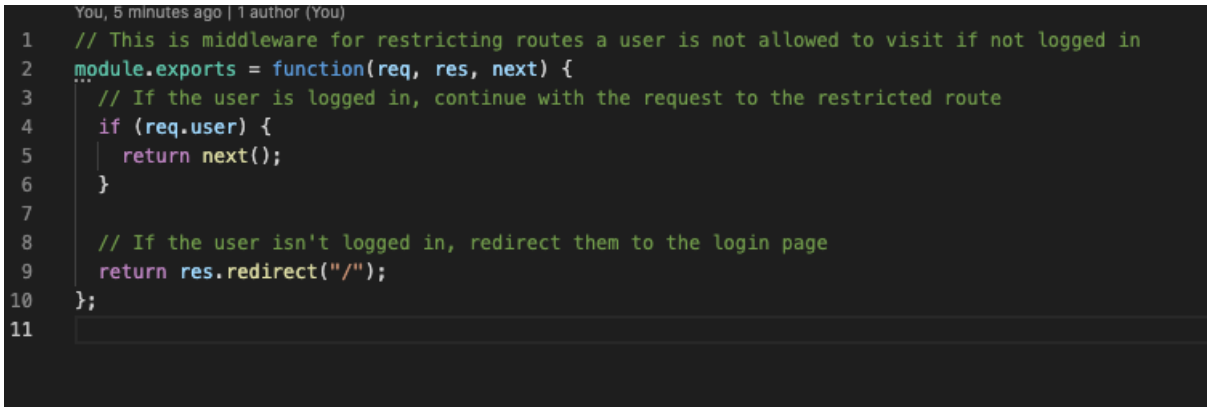
Reverse Engineering Code

— Codebase —

Config/middleware/isAuthenticated.js

The **isAuthenticated.js** is a function declaration be used as middleware for the application. The function checks whether or not the user credentials are correct and then logs in or redirects them.

The function is only declared in this file it is called in HTML routes.



Config/config.json

The **config.json** file serves as a connection between the code/server and the database.

The “development” and “test” keys are for the developers as they are creating, debugging and testing the app this will connect directly to the local mysql (or one of developers preference) database.

In the above the username, password, database and host should all be derived from the developers local mysql.

When the application is read and the developer would like to launch their application onto a hosting website such as HEROKU this is when the application will use the “production” key. The below info will need to be populated by the developer with information given by JawsDB.

Config/passport.js

The **passport.js** file is used to authenticate the username and password to ensure safe login and registration. The file contains a new instance of LocalStrategy class being created, the parameters are input in this case an object and a call back. The object tells us that the username will be an email. In the call back the function looks for a the email using sequelize in the database. This will return a promise with another call back in the **then()** statement of the promise. The logic in this checks to see if the email and password is valid if they are correct it returns the function done with the details of the user.

Models/index.js

The **index.js** file creates a new instance of sequelize class connected with the MySQL database. Then it runs a loop that reads through the current folder (models/) and adds the sequelize models defined in all the **.js** files as key value pairs in the “db” object. One per file for example “User” will become

User: {Model info goes here}

These models are connected to the database with sequelize (Note they are not the tables in the mysql db it is a representation of the tables done using object orientated mapping). Now for example when you require “models/” it will have a models.User which you can use sequelize commands on.

Models/user.js

In the **user.js** file the code uses models to do a definition of the class User. As this is a code driven database sequelize is used to define the users prior to updating the database. The user.js does not create any instances of the user class. The information to populated this will be from the post request sent by the browser. A module bcrypt is required by the file to ensure password validation when users are created. **User.js** will be exported as a module and required in the index.js file.

Public/js/login.js

This is all front-end code will ensure a form is sent to the server using fetch when a user logs into the application. The code uses jquery to add an event listener on the submit form it then sends a post request using jquery/ajax.

Public/js/members.js

This is front end code that has a get request for the /members route.

Public/js/signup.js

This is all front-end code will ensure a form is sent to the server using fetch when a user signs up for the application.

Routes/api-routes.js

The api-routes.js is a file that defines the routes for the server.js file, the file has 2 post requested for when the member signs up and when the member logs-in, the signup post request uses sequelize to create a new User and the login request uses the passport module to authenticate the users details before allowing log-in. The get requests are for the log-out and to send the user data in json format.

Routes/html-routes.js

The html-routes.js is get requests to render the html webpages.

/server.js

The server.js file is where everything is brought together. It requires the necessary npm packages and modules. It uses express to set up the server/app and all the necessary middleware for needed for authentication. It then requires the routes and ensures that the server is listening on the specified server depending whether it is being hosted or on the local host.