In the folder config folder middleware in the isAuthenticate.js you are restricting routes a user is not allowed to be in. Then when the user is logged in sends them to the restricted route.
If they are not logged in send them back to the home page. The config.json is just the set ups to the database with mysql, passport, and database testing. Passport.js is setting up the user login with username/email and password. Then uses LocalStrategy to have the user use an email rather than a username. Function of email, password and done just finds the email and password that the user has entered. Then the function then sends the user an error message saying the incorrect email does the same thing with the validPassword function if none of the above returns the user. In order to help keep authentication state across HTTP requests, sequelize needs to serialize and deserialize the user then exporting our configured passport.

Models folder file index.js sets up the path, fs, database, and linking up the config files and development. Then use the variable files in the database and create a new user. Set up the link to the database. User.js is setting up a bcrypt to make it where no one can know the users passwords then setting up the requirements for making an account then checks if the user has put in the right requirements to the email and password. Then bcrypt hashes out the users password to make it not known to the public but only to the user making the account.

Public folder js folder file login.js getting references to form and inputs. When a form is submitted make sure the user has put in an email and password. When we have an email, run the loginUser to clear the form. Then loginUser posts to the api/login and if it is successful sends them the the members.html if error sends them a log error. Member.js then does a get request to figure out who logged in and updates the HTML page to that user. When the signup button is clicked make sure the user puts in the right requirements for the making an email and password to make an account. Then does a post to the signup route if correct redirects to members page otherwise sends the user an error if it throws a bootstrap alert.

Stylesheets set up how the webpage looks like. Then login.html is the webpage for the user to login. If logged in goes to the members.html page, then if the user doesn't have an account send them to the signup.html to make an account. Folder routes then have api-routes to send information to the database for the signup, login, logout, and user_data. Html-routes sends users to the pages for members and login to link it all together.

The server.js file is requiring necessary npm packages and passports as we have configured it. Setting up the port and requiring models for syncing, creating express apps and configuring middleware needed for authentication. Uses sessions to keep track of our user’s login status then requiring our routes. Syncing our database and logging in a message to the user upon success.