1. Go to Package-JSON and look at the dependencies to find out what the app may be doing
    a. **Look at Dependencies:**
        i. Bcrypt-nodejs:
            1. Using hash to store passwords in DB
        ii. Body-parser
            1. Allows for sending of form data as “req.body”
        iii. Express
            1. Node Framework
        iv. Mysql2
            1. Database
        v. Passport
            1. Authentication middleware for Express. Authenticates requests using plugins (strategies). Solely for authentication.
        vi. Passport-local(Passport strategy)
            1. Authenticating username and password
    b. **Look at the initial package information**
        i. Tells us to the app “starts” at server.js. Go to this file to begin reverse engineering.

    -After looking at the dependencies, we can tell that this will be some sort of an authentication app using passport and passport-local modules that will require an email and a password to authenticate.

2. Going to server.js
    a. A lot of familiar middleware, but some new stuff as well
        i. Familiar stuff: express, bodyParser
        ii. New stuff: session, passport
    b. See server.js file for detailed notes on the middlewares
        i. Mostly new middleware is authenticating the user and storing the session data in a secure way that also allows us to identify the true user.
    c. Requiring the models folder to be used for the app
    d. Setting public as the root directory for static files (Making note to look into it in future)
    e. Requiring the html and api routes (Checkout Routes in future)
3. Going to Model folder
    a. Index.js:
        i. Allows us to use the sequelize in our app
    b. User.js
        i. Sets the model of the database to be used in the app
4. Going to Public Folder
    a. Seperated with stylesheets and all the html pages
    b. JS
        i. Login.js:

        1. Taking login from user with email input and passport input and posting it to the server (both username and password is required)
    ii. Member.js
        1. Displays the member name and email on the page
    iii. Signup.js
        1. Posting the user information to the server
5. Going to Routes
    a. Html-routes.js
        i. Requiring path, and "isAuthenticated" from config folder (Making note to visit afterwards)
        ii. Setting up the html routes based on whether the member is logging in, signing up and/or if the login was verified.
    b. APi-routes.js
        i. Requiring Models and **Passport (in config folder)**
        ii. Accepting and the post requests from public JS files, then sending back the appropriate information based on the requests
        iii. Post for signup is creating a new user in the database.
        iv. Login is using "passport.authenticate" to authenticate the users
        v. Get for logout is logging out the user and redirecting to the Root Directory
6. Going to Config Folder
    a. Passport.js
        i. Requiring Passport, passport-local, and models
        ii. Authenticating, serializing and deserializing information of the user.
        iii. See comments for detailed info on each new

    b. Config.json
        i. Development files should match local information when downloading it to local drive to test
    c. isAuthenticated.js