# Reverse-Engineering-Code

##Password Authentication

Using Passport.js an individual can register as a user. Once registered, the user can log into the page for access.

##User Story

As a user, I want to safely register with and log into the webpage.

##Dependencies

Ensure the following dependencies are installed: bcryptjs, express, express-session, mysql2, passport, passport-local, sequelize

##Usage

To begin using this program, you will need to clone this repository to your local storage, since it isn’t being hosted on Heroku in a way that will allow direct interaction. Then, do the following: 1. go into the config folder, open config.js and insert your personal log in information, 2. create a MYSQL database named "passport\_demo", 3. open the integrated terminal and type “npm i” to install your dependencies, and type "node server.js" to run the program and connect to the server, 5. open your browser and type "http://localhost:8080" in search bar to access the port your system is connected to on your local machine.

#File Explanations

<br /> ##CONFIG : MIDDLEWARE : isAuthenticated.js

This file controls what a user is able to see. If they user is authenticated, then they can move ahead. Otherwise, they will be redirected to the login/sign up page.

<br /> ##CONFIG : config.json

This configures the system’s connection to the server for the different environments.

<br /> ##CONFIG : passport.js

This tells passport.js that we want to log into the website with the listed credentials (email and password).

##MODELS : index.js

This connects to database and imports user login data. If you were working with other databases for your webpage, here is where you would call them together.

##MODELS : user.js

This model defines what a user is and what information it should contain and uses "bcrypt" for password hashing making the database more secure.

## ROUTES : api-routes.js

This file holds the routes for singing up, signing in and getting users specific data to be displayed client side

##ROUTES : html-routes.js

This file holds the routes that check whether a user is signed in or already has account and sends them to the correct html page. For example, if a user signs up, they will then be redirected to log in.

##package.json

This file has all package information, such as what dependencies should be installed (which allows you to just type “npm i” for installation), the application license, author, key words, test scripts, name of the application, what the main file is that should be accessed to run the application, etc.

##server.js

This file calls in what packages are required to run the application, establishes a PORT to listen to, creates express and middleware, sets up routes for the web pages and database interactions, and creates logs for successful and failed activities.