# reverseengineering

### Overview

This application allows users to create, log in and sign out an account securely. All user's data is stored in MySQL workbench database.

### User Story

As a user wants to securely log into 'X', i want to know my personal details are safely secured and stored in the database.

### Programme Use

- Node.js

- Express

- MySQL

- Passport

- Sequelize

### App Setup

1. Create a GitHub repo called `Marketing` and clone it to your computer.

2. Make a package.json file by running `npm init` from the command line.

3. Install the Express npm package: `npm install express`.

4. Create a server.js file.

5. Install the Handlebars npm package: `npm install express-handlebars`.

6. Install MySQL npm package: `npm install mysql`.

7. Require the following npm packages inside of the server.js file:

\* express

\* set-up PORT

\* routes and sync database in terminal

### DB Setup

1. Inside your `Marketing` directory, create a folder named `db`.

2. In the `db` folder, create a file named `schema.sql`. Write SQL queries this file that do the following:

\* Create the `passport\_db`.

\* Switch to or use the `passport\_db`.

\* Create a `passport` table with these fields:

\* \*\*id\*\*: an auto incrementing int that serves as the primary key.

\* \*\*user\_email\*\*: a string.

\* \*\*password\*\*: a string.

3. Still in the `db` folder, create a `seeds.sql` file. In this file, write insert queries to populate the `Marketing` table with at least three entries.

4. Run the `schema.sql` and `seeds.sql` files into the mysql server from the command line

5. Now you're going to run these SQL files.

\* Make sure you're in the `db` folder of your app.

\* Start MySQL command line tool and login: `mysql -u root -p`.

\* With the `mysql>` command line tool running, enter the command `source schema.sql`. This will run your schema file and all of the queries in it -- in other words, you'll be creating your database.

\* Now insert the entries you defined in `seeds.sql` by running the file: `source seeds.sql`.

\* Close out of the MySQL command line tool: `exit`.

### Config Setup

1. Inside your `Marketing` directory, create a folder named `config`.

2. Create a `connection.js` file inside `config` directory.

\* Inside the `connection.js` file, setup the code to connect Node to MySQL.

\* Export the connection.

3. Create an `orm.js` file inside `config` directory.

\* Import (require) `connection.js` into `orm.js`

\* In the `orm.js` file, create the methods that will execute the necessary MySQL commands in the controllers. These are the methods you will need to use in order to retrieve and store data in your database.

\* `selectAll()`

\* `insertOne()`

\* `updateOne()`

\* Export the ORM object in `module.exports`.

### Model setup

1. Inside your `Marketing` directory, create a folder named `models`.

\* In `models`, make a `user.js` file.

\* Inside `user.js`, import `orm.js` into `user.js`

\* Also inside `user.js`, create the code that will call the ORM functions using 'bcrypt' specific input for the ORM for password hashing.

\* Export at the end of the `user.js` file.

### Controller setup

1. Inside your `Marketing` directory, create a folder named `controllers`.

2. In `controllers`, create the `passport\_controller.js` file for signing in, logging out and getting user specific data to be displayed on client side.

3. Inside the `passport\_controller.js` file, import the following:

\* Express

\* `user.js`

4. Create the `router` for the app, and export the `router` at the end of your file.

### View setup

1. Inside your `Marketing` directory, create a folder named `views`.

\* Create the `index.handlebars` file inside `views` directory.

\* Create the `layouts` directory inside `views` directory.

\* Create the `main.handlebars` file inside `layouts` directory.

\* Setup the `main.handlebars` file so it's able to be used by Handlebars.

\* Setup the `index.handlebars` to have the template that Handlebars can render onto.

\* Create a button in `index.handlebars` that will submit the user input into the database.