Reverse Engineering Authentication Homework

Nuvi Johnson

**File Name: package.json**

High Level Purpose of File: This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies.

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | None |  |
| Dependencies ON this File  (Outputs): | None |  |
| Parts of the File: | * Bycryptjs for a secured way of storing passwords & creating hashes * Express is middleware that serves static files * Express-session will create a session id in a cookie and saves it server side that you can store data in mySQL or mongo. * Mysql12 allows to use mysql database * Passport is authentication middleware for node.js and provides different strategies for authenticating a user * Passport-local same as passport just locally * Sequelize lets you use manage the sql database | Project dependencies:      "bcryptjs": "2.4.3",      "express": "^4.17.0",      "express-session": "^1.16.1",      "mysql2": "^1.6.5",      "passport": "^0.4.0",      "passport-local": "^1.0.0",      "sequelize": "^5.8.6" |

**File Name: config.config.json**

High Level Purpose of File: credentials for development, test and production databases to connect to your mysql database.

**File Name: config.passport.js**

High Level Purpose of File: Allows & checks the user to sign in using email and password.

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | Requires passport module & passport-local module  Requires models folder | var passport = require("passport");  var LocalStrategy = require ("passport-local").Strategy;  var db = require ("../models"); |
| Outputs: | Module export is passport | module.exports = passport; |
| Parts of the File: | Passport will use local strategy for login email for username  User will sign in with email | passport.use(new LocalStrategy(    {      usernameField: "email"    }, |
|  | User will enter email and password,  If the email (or user) is not recognized, it returns incorrect email error to the user. | function(email, password, done) {      db.User.findOne({        where: {          email: email        }      }).then(function(dbUser) {        if (!dbUser) {          return done(null, false, {            message: "Incorrect email."          });        } |
|  | If the email is recognized, but the password is not recognized, then it returns incorrect password to the user.  Otherwise, it returns the user | else if (!dbUser.validPassword(password)) {          return done(null, false, {            message: "Incorrect password."          });        }        return done(null, dbUser);      });    }  )); |
|  | Sets the cookie id in the users browser | passport.serializeUser(function(user, cb) {    cb(null, user);  }); |
|  | Removes the cookie id from the users browser | passport.deserializeUser(function(obj, cb) {    cb(null, obj);  }); |

**File Name: config.middleware.isAuthenticated.js**

High Level Purpose of File: Middleware that restricts users to only navigate if signed in, restricts routes.

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Parts of the File: | Lets user continue | module.exports = function(req, res, next) {    if (req.user) {      return next();    } |
|  | Redirects if not logged in | return res.redirect("/");  }; |

**File Name: model.user.js**

High Level Purpose of File: Creates the user model for SQL database

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | Requires bcrypt for password hashing | var bcrypt = require("bcryptjs"); |
| Outputs: | Structure of the database for mySQL |  |
| Parts of the File: | Creates a new user model with sequelize  It has a user table that has email as a column  Email is required to be in the proper @ format and not null | module.exports = function(sequelize, DataTypes) {    var User = sequelize.define("User", {      email: {        type: DataTypes.STRING,        allowNull: false,        unique: true,        validate: {          isEmail: true        }      }, |
|  | Password is another column, the password can’t be null | password: {        type: DataTypes.STRING,        allowNull: false      }    }); |
|  | This is a method that looks at the hashed password in the database and compares to the password just input | User.prototype.validPassword = function(password) {      return bcrypt.compareSync(password, this.password);    }; |
|  | Then before the user can created it will automatically hash the password | User.addHook("beforeCreate", function(user) {      user.password = bcrypt.hashSync(user.password, bcrypt.genSaltSync(10), null);    });    return User;  }; |

**File Name: model.index.js**

High Level Purpose of File: Sign into mySQL and import table

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | Require npm packages fs, path, sequelize, config & it will be using the environment variables  Names basename = to that path for ease of the file  Names env to point to the environment, related to the config folder – using development here  Names db as an empty object | var fs        = require('fs');  var path      = require('path');  var Sequelize = require('sequelize');  var basename  = path.basename(module.filename);  var env       = process.env.NODE\_ENV || 'development';  var config    = require(\_\_dirname + '/../config/config.json')[env];  var db        = {}; |
| Outputs: | Exports as the database | module.exports = db; |
| Parts of the File: | Signs the user into mySQL using the config items | if (config.use\_env\_variable) {    var sequelize = new Sequelize(process.env[config.use\_env\_variable]);  } else {    var sequelize = new Sequelize(config.database, config.username, config.password, config);  } |
|  | The fs. readdirSyncmethod is used to synchronously read the contents of a given directory. The method returns an array with all the file names or objects in the directory | fs    .readdirSync(\_\_dirname)    .filter(function(file) {      return (file.indexOf('.') !== 0) && (file !== basename) && (file.slice(-3) === '.js');    }) |
|  | For each file take the new variable created called model and import the database model | .forEach(function(file) {      var model = sequelize['import'](path.join(\_\_dirname, file));      db[model.name] = model;    }); |
|  | If the database model has associated foreign keys across tables, allow them connected | Object.keys(db).forEach(function(modelName) {    if (db[modelName].associate) {      db[modelName].associate(db);    }  });  db.sequelize = sequelize;  db.Sequelize = Sequelize; |

**File Name: public.login.html**

High Level Purpose of File: Login page for the front end

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | Requires bootstrap and the style.cc files  Requires scripts for jquery and for the login.js file. | <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootswatch/3.3.7/lumen/bootstrap.min.css">    <link href="stylesheets/style.css" rel="stylesheet">    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>    <script type="text/javascript" src="js/login.js"></script> |
| Outputs: |  |  |
| Parts of the File: | Creates a nav bar at the top | <nav class="navbar navbar-default">      <div class="container-fluid">        <div class="navbar-header">        </div>      </div>    </nav> |
|  | Creates a container that has 1 row 2 columns  It has a form group that has email address  It has another form group that’s password  It has a button that has submit to log in  It has a sign up link that reroutes to signup.html | <div class="container">      <div class="row">        <div class="col-md-6 col-md-offset-3">          <h2>Login Form</h2>          <form class="login">            <div class="form-group">              <label for="exampleInputEmail1">Email address</label>              <input type="email" class="form-control" id="email-input" placeholder="Email">            </div>            <div class="form-group">              <label for="exampleInputPassword1">Password</label>              <input type="password" class="form-control" id="password-input" placeholder="Password">            </div>            <button type="submit" class="btn btn-default">Login</button>          </form>          <br />          <p>Or sign up <a href="/">here</a></p>        </div>      </div>    </div> |

**File Name: public.signup.html**

High Level Purpose of File: Signup page for the front end

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | Requires bootstrap and the style.cc files  Requires scripts for jquery and for the signup.js file. | <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootswatch/3.3.7/lumen/bootstrap.min.css">    <link href="stylesheets/style.css" rel="stylesheet">    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>    <script type="text/javascript" src="js/signup.js"></script> |
| Outputs: |  |  |
| Parts of the File: | Creates a nav bar at the top | <nav class="navbar navbar-default">      <div class="container-fluid">        <div class="navbar-header">        </div>      </div>    </nav> |
|  | Creates a container that has 1 row 2 columns  It has a form group that has email address  It has another form group that’s password  It has a button that has submit to sign up  It has a login link that reroutes to login.html | <div class="container">      <div class="row">        <div class="col-md-6 col-md-offset-3">          <h2>Sign Up Form</h2>          <form class="signup">            <div class="form-group">              <label for="exampleInputEmail1">Email address</label>              <input type="email" class="form-control" id="email-input" placeholder="Email">            </div>            <div class="form-group">              <label for="exampleInputPassword1">Password</label>              <input type="password" class="form-control" id="password-input" placeholder="Password">            </div>            <div style="display: none" id="alert" class="alert alert-danger" role="alert">              <span class="glyphicon glyphicon-exclamation-sign" aria-hidden="true"></span>              <span class="sr-only">Error:</span> <span class="msg"></span>            </div>            <button type="submit" class="btn btn-default">Sign Up</button>          </form>          <br />          <p>Or log in <a href="/login">here</a></p>        </div>      </div>    </div> |

**File Name: public.members.html**

High Level Purpose of File: Signup page for the front end

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | Requires bootstrap and the style.cc files  Requires scripts for jquery and for the members.js file. | <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootswatch/3.3.7/lumen/bootstrap.min.css">    <link href="stylesheets/style.css" rel="stylesheet">    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>    <script type="text/javascript" src="js/members.js"></script> |
| Outputs: |  |  |
| Parts of the File: | Creates a nav bar at the top with logout button | <nav class="navbar navbar-default">    <div class="container-fluid">      <div class="navbar-header">        <a class="navbar-brand" href="/logout">          Logout        </a>      </div>    </div>  </nav> |
|  | Creates a container with 1 row, 2 columns where class = member-name will be placed in here from the members.js file and populate | <div class="container">      <div class="row">        <div class="col-md-6 col-md-offset-3">          <h2>Welcome <span class="member-name"></span></h2>        </div>      </div>    </div> |

**File Name: public.stylesheets.style.css**

High Level Purpose of File: Style for all front end pages

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | none |  |
| Outputs: | Uses bootstrap sign up and log in form styling. | form.signup,  form.login {    margin-top: 50px;  } |
| Parts of the File: | none |  |

**File Name: public.js.login.js**

High Level Purpose of File: Javascript file that supports the login page

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | login.html  Api-routes.js |  |
| Outputs: |  |  |
| Parts of the File: | Gets references for the forms and the inputs | $(document).ready(function() {    var loginForm = $("form.login");    var emailInput = $("input#email-input");    var passwordInput = $("input#password-input"); |
|  | On login, take the email and password and make sure that its not empty | loginForm.on("submit", function(event) {      event.preventDefault();      var userData = {        email: emailInput.val().trim(),        password: passwordInput.val().trim()      };      if (!userData.email || !userData.password) {        return;      } |
|  | If we do have the login info then we run the function loginUser and clear the form | loginUser(userData.email, userData.password);      emailInput.val("");      passwordInput.val("");    }); |
|  | The loginUser runs a post api that then redirects them to the member page if successful or logs an error if not successful. | function loginUser(email, password) {      $.post("/api/login", {        email: email,        password: password      })        .then(function() {          window.location.replace("/members");        })        .catch(function(err) {          console.log(err);        });    }  }); |

**File Name: public.js.signup.js**

High Level Purpose of File: Javascript file that supports the signup page

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | Signup.html  Api-routes.js |  |
| Outputs: |  |  |
| Parts of the File: | Gets references for the forms and the inputs | $(document).ready(function() {    var signUpForm = $("form.signup");    var emailInput = $("input#email-input");    var passwordInput = $("input#password-input"); |
|  | On sign up, take the email and password and make sure that its not empty | signUpForm.on("submit", function(event) {      event.preventDefault();      var userData = {        email: emailInput.val().trim(),        password: passwordInput.val().trim()      };      if (!userData.email || !userData.password) {        return;      } |
|  | If we do have the email and password, then run the signUpUser function | signUpUser(userData.email, userData.password);      emailInput.val("");      passwordInput.val("");    }); |
|  | The signUpUser runs a post api that then redirects them to the member page if successful or logs an error if not successful. | function signUpUser(email, password) {      $.post("/api/signup", {        email: email,        password: password      })        .then(function(data) {          window.location.replace("/members");        })        .catch(handleLoginErr);    } |
|  | If there is an error then create an alert that says 500 error. | function handleLoginErr(err) {      $("#alert .msg").text(err.responseJSON);      $("#alert").fadeIn(500);    }  }); |

**File Name: public.js.member.js**

High Level Purpose of File: Javascript file that supports the member page

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | members.html  Api-routes.js |  |
| Outputs: |  |  |
| Parts of the File: | Runs a get route to see which user is signed in and updates the html on the member.html page | $(document).ready(function() {    $.get("/api/user\_data").then(function(data) {      $(".member-name").text(data.email);    });  }); |

**File Name: routes.api-routes.js**

High Level Purpose of File: Javascript file that has all the api routes within it

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | Requires model folder and the passport  Middleware passport.authenticate | var db = require("../models");  var passport = require("../config/passport"); |
| Outputs: |  |  |
| Parts of the File: | If the user has valid credentials per the passport.autheticate local strategy, then pass them to the members page, otherwise send error | module.exports = function(app) {    app.post("/api/login", passport.authenticate("local"), function(req, res) {      res.json(req.user);    }); |
|  | Route for signing up a user, if user is successful then it goes to the login, otherwise it sends error | app.post("/api/signup", function(req, res) {      db.User.create({        email: req.body.email,        password: req.body.password      })        .then(function() {          res.redirect(307, "/api/login");        })        .catch(function(err) {          res.status(401).json(err);        });    }); |
|  | Route for logging out the user | app.get("/logout", function(req, res) {      req.logout();      res.redirect("/");    }); |
|  | Route for getting user data, that if the user isn’t logged in, its blank otherwise send back the email and id | app.get("/api/user\_data", function(req, res) {      if (!req.user) {        res.json({});      } else {        res.json({          email: req.user.email,          id: req.user.id        });      }    });  }; |

**File Name: routes.html-routes.js**

High Level Purpose of File: Javascript file that has all the html routes within it

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | Requires path and middleware isAuthenticated  Each of the javascript files for each html (login, signup & members.js) all call this file | var path = require("path");  var isAuthenticated = require("../config/middleware/isAuthenticated"); |
| Outputs: |  |  |
| Parts of the File: | Send the user to sign up if they don’t have a login, if they do then send to member page | module.exports = function(app) {    app.get("/", function(req, res) {      if (req.user) {        res.redirect("/members");      }      res.sendFile(path.join(\_\_dirname, "../public/signup.html"));    }); |
|  | Send the user to sign up if they don’t have a login, if they do then send to member page | app.get("/login", function(req, res) {      if (req.user) {        res.redirect("/members");      }      res.sendFile(path.join(\_\_dirname, "../public/login.html"));    }); |
|  | If the member doesn’t have a log in and tries to access the member page, it sends them to signup | app.get("/members", isAuthenticated, function(req, res) {      res.sendFile(path.join(\_\_dirname, "../public/members.html"));    }); |

**File Name: server.js**

High Level Purpose of File: File that run post npm install node server.js – main file for all the components.

|  |  |  |
| --- | --- | --- |
|  | Functionality | Example Part (if applicable) |
| Dependencies FOR this File:  (Inputs): | Requires npm packages express & express-session  Requires passport that we configured before  Sets up the port and requires the models folder | var express = require("express");  var session = require("express-session");  var passport = require("./config/passport");  var PORT = process.env.PORT || 8080;  var db = require("./models"); |
| Outputs: |  |  |
| Parts of the File: | Create express app and configure the middleware | var app = express();  app.use(express.urlencoded({ extended: true }));  app.use(express.json());  app.use(express.static("public")); |
|  | Use sessions to keep track of users logged in status | app.use(session({ secret: "keyboard cat", resave: true, saveUninitialized: true }));  app.use(passport.initialize());  app.use(passport.session()); |
|  | Required routes needed | require("./routes/html-routes.js")(app);  require("./routes/api-routes.js")(app); |
|  | Syncing the database and console logging the success when it works | db.sequelize.sync().then(function() {    app.listen(PORT, function() {      console.log("==> 🌎  Listening on port %s. Visit http://localhost:%s/ in your browser.", PORT, PORT);    });  }); |