

COMP 2537 Web Development 2

Assignment 1

COMP 2537 Web Development 2
Assignment 1

Introduction:

For this assignment you will be creating a simple website using Node.js. You will also be creating and connecting to a MongoDB database to store the users with their names, emails and passwords. You will create a publicly accessible MongoDB database using Atlas (at: <https://www.mongodb.com/atlas>).

If a user successfully authenticates (provides the valid email and password) a session will be created to store the user's information. A user is considered logged in if they have a valid session. You will store the **encrypted** session information in a **MongoDB session database**. A session must expire after 1 hour. If a user is not logged in (doesn't have a valid session) they must not be able to view the members page – it is a member's only page after all!

We will be taking input from the user to capture their account information and for them to log in. We'll need to properly deal with NoSQL Injection attacks in our MongoDB database so that users can't intentionally attack our database.

We will use a `.env` file to store encryption secrets and your MongoDB database credentials. Passwords must **not** be stored in your git repo. If they are, and your repo goes public or accidentally gets leaked, an attacker now has access to your database to do whatever they want with it! Let's keep our passwords out of the reach of hackers.

Your site needs to be hosted on a site like Render. Render is a fully managed web app hosting platform (<https://render.com/>). Render can provide many features including: 1) getting us off our localhost so we can share our sites with other people; 2) creates a domain which is publicly accessible; 3) installs and runs our node.js server; 4) automatically deploys our code when we make a commit and push our code to our main branch on github.

In the past I have used Heroku and Qoddi (similar hosting services), however, that was when they offered free development hosting. Now they charge money even for the simplest site. Render provides similar services for free.

COMP 2537 Web Development 2
Assignment 1

User Stories:

As you know, user stories are a key part of the agile development methodology as they help define valuable software features from the perspective of the end-user.

Here are some examples of user stories being implemented in this assignment:

- As a user, I want to be able to **sign up** for a new account so that I can access the members only area.
- As a user, I want to be able to **sign in** to my existing account so that I can access the members only area.
- As a user, I want to be able to **sign out** of my account so that nobody else can access it.
- As a user, I want to see a **home page** with the options to sign up or sign in if I am **not currently logged in**.
- As a user, I want to see a **home page** welcoming me and showing me the option to go to the members only area and sign out if I am currently **logged in**.
- As a user, I want to be able to see a **random image** when I access the members only area so that it is more visually interesting.
- As a user, I want to be informed if the **username or password** I entered **is incorrect** when signing in so that I can correct the mistake.

Recommended Software to Install:

Sourcetree: <https://www.sourcetreeapp.com/>



Sourcetree is a great GUI and visual tool for viewing and managing your Git repos. In my opinion it is better at seeing the current state of your repo than github.com, git on the command line, version control within VS Code and even GitHub Desktop.

Sourcetree is cross-platform, works on Windows and Mac and is free!

Studio 3T: <https://studio3t.com/>



Studio 3T is a great way to see and edit all of your MongoDB collections and data. Available for free on Windows, Mac and Linux.

VS Code: <https://code.visualstudio.com/download>



Visual Studio Code

A great, versatile IDE for all types of coding projects including websites written in Node.js.

Chrome Extensions:

There are also a few extensions for the Google Chrome Web Browser that might be helpful.

"Edit this cookie" by editthiscookie.com

Let's you see session cookies, edit them, or delete them.

<https://chrome.google.com/webstore/detail/editthiscookie/fngmhnnpilhplaeedifhccceomclgfbg?hl=en>

Note: Using this extension is not required. You can access cookies via the development tools, however, this provides a quick and easy way to see and delete cookies.

COMP 2537 Web Development 2
Assignment 1

Online Tools:

Some **GUID/UUID online generators**:

<https://guidgenerator.com/>

<https://www.uuidgenerator.net/guid>

<https://www.guidgen.com/>

These are useful sites when wanting to create unique and secret keys for session encryption.

Node Modules you will need:

In Node we will need some external libraries to make our website work:

- **express**
According to Express's official site (<https://expressjs.com/>): "Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications" and provides a myriad of utility methods and middleware for creating websites.
- **express-session** and **connect-mongo**
The express-session module is used to create a session that can ensure only those users that have logged in can see the members only page. The connect-mongo module is used to store your session data in an encrypted mongodb database. This way, if your server needs to be rebooted, you don't lose your server-side session information.
- **bcrypt**
BCrypt is a hashing library used to salt and hash passwords before storing them in a database. This is a requirement of this assignment and is good industry practice, since plain-text (unsalted, unhashed passwords) must not be stored in a database. If a database were to be breached through an attack, salted and hashed passwords created by BCrypt remain unknown and useless to the hacker.

Remember, in order to use these modules, you'll need to install them using:

```
npm i <module_name>
```

Example to install **express**:

```
npm i express
```

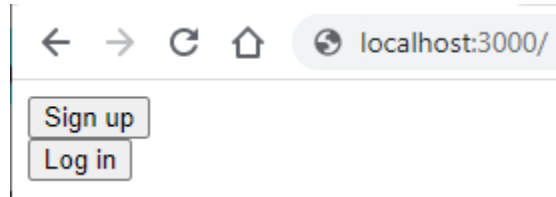
COMP 2537 Web Development 2
Assignment 1

Objectives:

Your Node.js website must have the following pages and functionality:

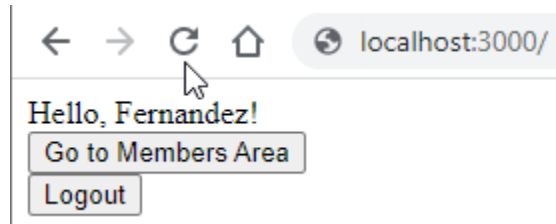
1. Home page – site: / method: GET

If the user is **not logged in**, this home page will have links to:



- a. Sign up (/signup)
- b. Log in (/login)

Else, if the user is **logged in**, this home page will:



- a. Say Hello **and** the name of the user – ex: Hello, Fernandez.
You must store the user's name in the session during login/signup and display it here.
- b. Provide a link to the members area (/members).
- c. Have a link that will log the user out (/logout).
This will end the session and redirect back to (/).

2. Sign up page – site: /signup method: GET

A screenshot of a web browser window. The address bar shows 'localhost:3000/signup/'. Below the address bar, the text 'create user' is displayed. Below the text, there are three input fields: 'name', 'email', and 'password'. Below the input fields, there is a 'Submit' button.

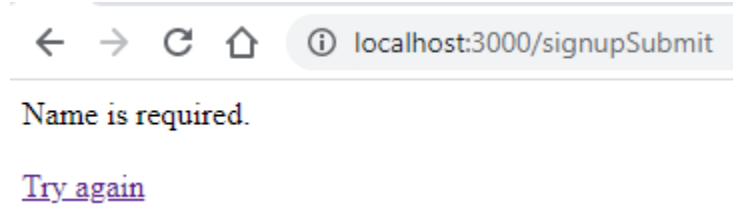
A form will gather the following information for sign up:

- a. Name
- b. Email
- c. Password

COMP 2537 Web Development 2
Assignment 1

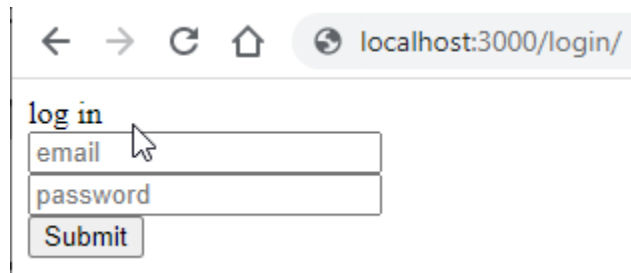
The signup form will `POST` the form fields.

You will validate all 3 inputs to make sure they aren't empty. If the user has forgotten to enter one or more of the 3 inputs, you will send back an appropriate message saying which field was missing. Ex: *Please provide an email address*. You will also provide a link back to the sign up page so the user may try again:



If the 3 fields are non-empty, add the user to your MongoDB database. Make sure that you are using Joi to validate the input (name, email and password) so that NoSQL Injection attacks are not possible. Add the **name**, **email** and a **BCrypted hashed password** as a user to the database. Then create a session and redirect the user to the `/members` page.

3. Log in page – site: `/login` method: `GET`



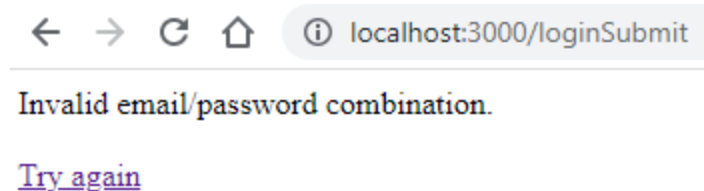
A form will gather the following information for sign up:

- d. Email
- e. Password

COMP 2537 Web Development 2
Assignment 1

The login form will `POST` the form fields. Check the user against the MongoDB database. Make sure that you are using Joi to validate the input (email and password) so that NoSQL Injection attacks are not possible. If a user with that email is found, check to see if the password matches the BCRYPTed password. If the email and passwords match store the user's name in a session and log the user in (redirect to `/members`).

If the log in fails, send a message to the user with an appropriate message. Ex: *User and password not found*. Also provide a link back to the login page so that users can try again:

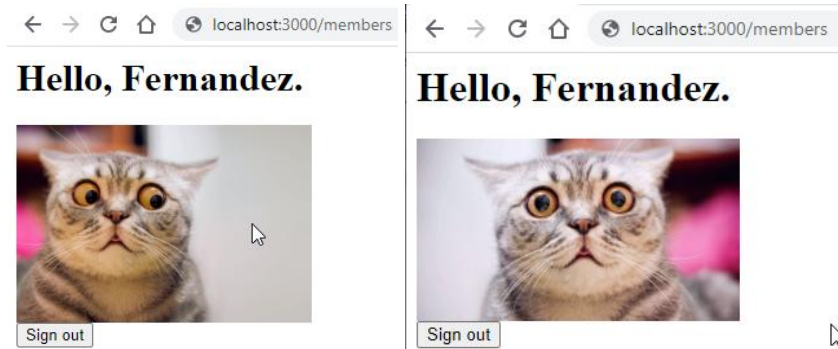


4. Members only page – site: `/members` method: `GET`

If the user has a **valid session** (is logged in):

- Say Hello **and** the name of the user – ex: Hello, Fernandez.
(use the name value stored in the session)
- Have a link that will log the user out (ends the session and redirects back to `/`).
- Display a random image from a selection of 3 images.

The images must be stored in a `/public` folder on the server.



If the user has **no session** (or an invalid one – i.e. not logged in):

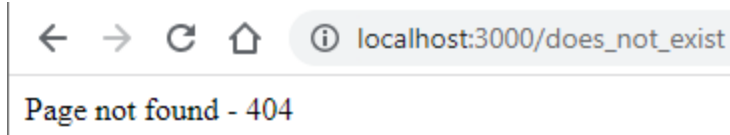
- Redirect to the home page (`/`).


5. Log out page – site: `/logout` method: `GET`

This page doesn't have any content, but must destroy the user's session (`req.session.destroy()`) and redirect back to the home page (`/`).

6. 404 page – site: any non-assigned URLs method: `GET`

COMP 2537 Web Development 2
Assignment 1



Any page that doesn't match the above sites will send an HTML status code of **404** and display page not found – Ex: /does_not_exist 

Your .env file:

Make sure that your username and password for connecting to your MongoDB database are stored in your .env file. Also, store your encryption keys for your sessions in the .env file. Make sure your .env is added to the .gitignore so that the .env file is NOT added to your git repo.
A .env file must **NEVER** be in your git repo!

Create a file called .env. This is where you will define all your users, passwords and secret keys.

```
MONGODB_HOST=YourMongoDBClusterHere
MONGODB_USER=YourUsernameHere
MONGODB_PASSWORD=YourPasswordHere
MONGODB_DATABASE=YourMongoDBDatabaseHere
MONGODB_SESSION_SECRET=Some_Session_Key_Here1
NODE_SESSION_SECRET=Some_Session_Key_Here2
```

Notes:

There is no space before or after the equals sign – ex:

```
MONGODB_USER=YourUsernameHere
```

NOT

```
MONGODB_USER = YourUsernameHere
```

While you *can* enclose the variable values within single or double quotes, it is common to omit them.

```
MONGODB_USER=YourUsernameHere
```

OR

```
MONGODB_USER="YourUsernameHere"
```

both work for defining variables.

More information here:

<https://www.npmjs.com/package/dotenv>

<https://hexdocs.pm/dotenvy/dotenv-file-format.html>

To use these variables in our code, we need to install the dotenv Node module using:

```
npm install dotenv
```

COMP 2537 Web Development 2

Assignment 1

And we'll need to the dotenv Node module like this:

```
require('dotenv').config();
```

And we can access those variables using `process.env` like this:

```
process.env.MONGODB_USER
```

and

```
process.env.MONGODB_PASSWORD
```

Pro Tip:

Many other Database providers (MongoDB, MySQL, and others) will provide us with a connection URL such as this (MongoDB):

```
mongodb+srv://my_user:xhov21ul@cluster0.fuu9a.mongodb.net/users
```

Or this (MySQL):

```
mysql://a0wj08yp3h60iax9:xhov21ul59ilv8e5@jtb9ia3h1pgevwb1.cbetxkdyhws  
b.us-east-1.rds.amazonaws.com:3306/kjm3rtal361lmlue
```

It is important to know how to parse these connection URLs into their individual parts.

For MongoDB:

Below is the same connection URL color coded and labelled to distinguish the key parts of the URL.

```
mongodb+srv://my_user:xhov21ul@cluster0.fuu9a.mongodb.net/users
```

Protocol: **mongodb+srv**

User: **my_user**

Password: **xhov21ul**

Host: **cluster0.fuu9a.mongodb.net**

Port: (not specified)

Database: **users**

COMP 2537 Web Development 2
Assignment 1

And for MySQL:

```
mysql://a0wj08yp3h60iax9:xhov21ul59ilv8e5@  
jtb9ia3hlpgewb1.cbetxkdyhwsb.us-east-1.rds.amazonaws.com:3306/  
kjm3rtal3611mlue
```

Protocol: **mysql**
User: **a0wj08yp3h60iax9**
Password: **xhov21ul59ilv8e5**
Host: **jtb9ia3hlpgewb1.cbetxkdyhwsb.us-east-1.rds.amazonaws.com**
Port: **3306**
Database: **kjm3rtal3611mlue**

Create a Git Repo:

Create a git repo for your website. Make sure to add all files, commit and **push** them your origin on github.com. A single branch called `main` (or `master`) is fine.

As mentioned earlier, ensure you have excluded the `.env` file from your git repo.

Create a `.gitignore` file with the following contents:

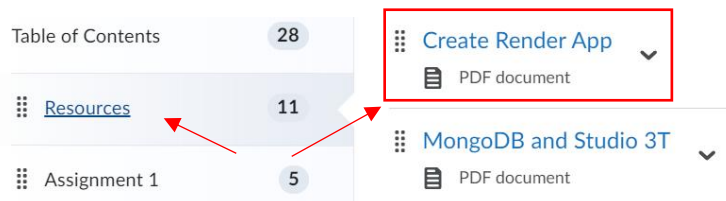
```
/node_modules  
package-lock.json  
.env
```

Hosting on Render (or other hosting service):

Create an account with Render (or another hosting service).

With your account, create an app that is linked to your git repo.

For more information on using Render see the Learning Hub in the Content -> Resources section



IMPORTANT! Remember to add your environment variables to the Render app. Otherwise, Render won't have access to all your secrets (passwords and log in information). Instructions for adding a `.env` file are in the "Create Render App" document on the Learning Hub.

COMP 2537 Web Development 2

Assignment 1

Protect against NoSQL Injection Attacks:

Any time we query a database and we include input from a user, we need to make sure the user hasn't intentionally (or unintentionally) added some NoSQL commands that would alter our query. We often include user input as part of our queries, such as: when we ask a user for their username, and lookup in the database to see if they exist; to add the username, email and password a user has entered in a form; or when a user requests to change their password and we update their password hash in the database.

User input can come in multiple ways such as form fields in POST requests and URL parameters.

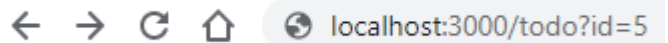
Form Fields

Sign Up:

name
email
password

Submit

URL parameters



If we were to include user input directly (without validating it first) from a form field like `req.body.username` or from a URL parameter like `req.params.id` in a MongoDB query like this:

```
await userCollection.find({username: req.params.id}).toArray();
```

The user might add some MongoDB code that could change the results of our query.

This is **BAD. This is VERY BAD!**

We need to validate that, what the user has provided us, doesn't contain any special characters and no MongoDB commands. This is why we need something a validator like Joi (another useful node module).

To install Joi:

```
npm i joi
```

With Joi we can ensure that the user input is just a string and not an array or a JSON object that MongoDB would interpret as a MongoDB command.

To use Joi in your code, you need to require it:

```
const Joi = require("joi");
```

COMP 2537 Web Development 2
Assignment 1

And in the following example, we make sure the `req.body.username` is a string (`.string()`), is not more than 20 characters (`.max(20)`) and is not empty (`.required()`). The schema defines a single variable and we validate `req.body.username` against the schema. If there is an error (a potential NoSQL attack), we will redirect to a different page, preventing the user's input from being executed as part of the query in the `userCollection.find()`.

```
const schema = Joi.string().max(20).required();

const validationResult = schema.validate(req.body.username);
if (validationResult.error !== null) {
  console.log(validationResult.error);
  res.redirect("/login");
  return;
}

const result = await userCollection.find({username: username}).toArray();
```

In a slightly more complex example where we want to validate more than one variable at the same time, we can specify in our Joi schema which variables and which types are expected. This example validates both the username and the password from the form submission.

```
const schema = Joi.object(
{
  username: Joi.string().alphanum().max(20).required(),
  password: Joi.string().max(20).required()
});

const validationResult = schema.validate({req.body});
```

COMP 2537 Web Development 2
Assignment 1

Marking Guide:

Criteria	Marks
A home page links to signup and login, if not logged in; and links to members and signout, if logged in.	5 marks
A members only page that displays 1 of 3 random images stored on the server.	5 marks
The members only page will redirect to the home page if no valid session is found.	5 marks
The signout buttons end the session.	5 marks
A 404 page that "catches" all invalid page hits and that sets the status code to 404 .	5 marks
Session information is stored in an encrypted MongoDB session database. Sessions expire after 1 hour.	5 marks
Password is BCrypted in the MongoDB database.	5 marks
Your site is hosted in a hosting service like Render.	5 marks
All secrets, encryption keys, database passwords are stored in a <code>.env</code> file.	5 marks
The <code>.env</code> file is NOT in your git repo.	5 marks
Note: You will receive a mark of 0 if your code doesn't run (i.e. too many errors to run properly, cannot connect to the database, etc.)! I will be running the code on your Render app, please make sure it is available until after you receive your marks!	
Total:	50 marks

Submission Requirements:

Submission:
A link to your Youtube video demo. ex: https://youtu.be/c8nPuL5phG1
A link to your GitHub Repo with your Node.js application. (REMEMBER: Commit AND PUSH your code to GitHub!) ex: https://github.com/comp2537-s24/assignment1-studentname Include your self-graded checklist within the root of your git repo.
A link to your hosted site (on Render). ex: http://myawesome-2537-a1.onRender.com/

COMP 2537 Web Development 2
Assignment 1

Self-graded Checklist:

Find the self-graded assignment 1 checklist template and for each item, either **leave it blank** (if you did not complete the task fully) or put an **✖** next to it (indicating you completed the task).

Provide a totalled grade out of 50.

Fill in your name and set.

Example:

Student Name: **Nellie Jackman**

Student Set: **1A**

[] The signout buttons end the session. (incomplete)

[**✖**] The .env file is NOT in your git repo. (complete)

40/50

Include your self-graded checklist within your git repo in the root folder.

Deductions

The following deductions will apply if submission requirements are not met:

Requirement	Deduction	
	Out of 50	Percent
Missing Hosting Link	-10	20%
Missing or incomplete Checklist	-10	20%
Missing GitHub Link	-15	30%
Video is 10s to 60s too long	-5	10%
Video is >60s too long	-10	20%
Missing Video Link	-25	50%

COMP 2537 Web Development 2
Assignment 1

Video Demo Requirements:

Your video needs to include:	Example:
Your name	Patrick Guichon
Your set	1A
Which assignment you are demoing	Assignment 1

Notes about your video:
If we don't see all the functionality and features of your website demonstrated in the video, you will lose marks for not having completing this part of the website.
You do NOT need to show your face while recording the video, however, you must be narrating the video. Tell us what you are doing, as you are demoing your site.
Make sure we can hear your voice clearly, and at an adequate volume.
Max video length: 4m:00s.
Your video must NOT be longer than 4 minutes.

Demo the following functionality in this order:
1. Show the home page (/) running on Render (or other hosting service).
2. Show the users database in Studio 3T.
3. Show the session database in Studio 3T.
4. Click on the Sign up link and create a new user. After the user is created, the user is logged in and must be sent to the /members page.
5. Show the users database in Studio 3T – the new user must be added The password must be hashed (and not in plaintext).
6. Show the session database in Studio 3T – the session must be encrypted and the id must match your cookie in the browser.
7. Click on Sign out.
8. Show the session database in Studio 3T – the session must be deleted.
9. Delete your cookie and attempt to go directly to /members without a valid session.
10. From the login page enter a valid email address, but an invalid password. You must be provided an error message saying invalid password.
11. From the login page enter a valid email and password to show you are properly logged in.
12. Refresh the /members page to show your random pictures. Keep refreshing until all 3 images are shown.
13. Go to a page that doesn't exist example /doesnotexist to show the 404 page.
14. Show the code for your site in your editor: <ul style="list-style-type: none">- Show that you are using variables from your .env file.- Show that your sessions are set to expire after 1 hour.- Show that you are validating user inputs using Joi.
15. Show that your .env is added to your .gitignore and your .env is NOT in your git repo.