COMP2406 Fall 2022
Assignment 5
Final Report

Joseph McNamara
101227263

Video: https://youtu.be/3TT-wV2nRU0

Note: My video is slightly over the time limit but everything to do with the actual website is done around minute 8, after that is just going through the code which I guess is an optional watch.

Running the project:

First, run the mongoDB database in the shell using the command “mongod –dirpath {path_to_project}/galaDB”

Once the database is running, initialize it using the database initializer. This can be done using the command “node database-initializer.js” in the directory that the project is stored in.

Finally, to run the server use the command “node server.js” in the directory that the project is stored in.

You will then be able to access the website by reaching the url “http://localhost:3000/”.

NOTE
In server.js on line 57, I had to use the address 0.0.0.0 instead of localhost, as it was the only way to connect to the database. If the database doesn’t connect, this might be the issue.
** SECOND NOTE **
I fixed the above issue but I’m gonna leave the note here instead. Hope you like my project :)

Design Decisions:

My design was split into five different routes:

- Login router
- Gallery router
- Search router
- User router
- Workshop router

These five different routes handled the 5 general sections that you could split my project into.

Gallery router:

A large part of my design decision was to decide where to store likes and reviews. It made sense to attribute these to the images, and have none of the information stored in the user. Therefore, all the likes and reviews are stored by storing the users information in the object of the image that they liked / reviewed. This way, it keeps it consistent when we are getting a users liked images etc. In doing this, we do not need to access the user in the database when we want to find their liked images and then also go get the image from the database, instead we can just search the images collection for images that have the users id (which we store in the session object for easy access) in their likes / reviews array.

In the gallery router, I handle uploading images, displaying the main gallery, toggling likes, adding and deleting reviews.

Instead of having separate requests for liking and unliking images, I have one request that checks if the user has liked an image or not, and then likes / unlikes it. This makes my front end very easy, as it just has to send one request, and there is no checking done on the front end side.

Login router:

Handled everything to do with logging in / registering, as well as setting up the session for the user. When logging in, the server checks to see if any of the fields are blank, and then checks to see if the user exists in the database. If all of these checks pass, the server creates the session variables which hold the username, user id, and a loggedIn flag.

The register functionality only registers the user and prompts them to login, which allows the session information to be handled purely by the login function.

User Router:

In the user router, I have lots of different middleman functions to get different data needed for each request. Something I could have done differently was make my middleman functions more modular, as currently I have separate functions like getLikedGallery and getReviewedGallery. If I could find a way to make one function get both of those galleries, it would be a lot easier.

The users store their followers as well as who they are following. This was a rushed idea, and it would make more sense to just store their followers or who they're following, but not both. The users also store their notifications list. Each notification send the link of the user who they are being notified about, the date that the notification was sent, and the actual notification message itself. Pug is used to build the actual notification.

Search Router:

The search function is relatively straightforward. There are three categories, and depending on which category is searched, we call a different find function on the gallery collection.

Other design decisions

Notifications are handled separately between new post notification and new workshop notification. This could probably be merged into one function that does both.

My view was set up like a pinterest type feed, where each post did not have the same height or width, but was scaled to fit in a 3 element width grid div. I feel like this could have been implemented cleaner, but I'm happy I was able to do it like this.

I decided on modals to display the images info, so when you click on the image it creates a popup rather than giving the image its own page. I feel like this compliments the pintrest type look I was going for. However, the modal is somewhat too large usually, and you can't see the entire image sometimes.