**Art Portfolio Website Documentation**

**Getting Started**

This project requires NodeJS and the express, fs, logger, mongodb, and morgan packages.

To start, run a terminal and change the directory to the project. Then, run the line: “npm install”. This should install the necessary packages to run the server and client.

Afterwards, to start the server, run the line “node memcrud.js”. You should get a message, “server started on port 3000” or something similar to that degree.

To view the webpage, open up a browser (preferably Chrome), and type in the url “localhost:3000/client/index.html”. This should connect to the webpage, with the showcases, images, likes, statistics, loading onto the page.

**How it Works**

The server runs via the code memcrud.js. This creates an application that has different routes for handling data. There are different CRUD operations and routes established within this file. The PUT method is utilized when a new showcase is created. The READ method is utilized when retrieving server data and images. The UPDATE method is utilized when new comments, artworks, likes, or showcase modifications are done. Finally, the DELETE method is utilized when a showcase is deleted, which removes the showcase from the JSON file and database.

The MongoDB database is used whenever data needs to be read or saved. A connection will get established to the database, and data is modified/read accordingly, then the connection is closed. The data is then saved locally in a variable and handled accordingly.

The client is used via an HTML file, which displays all the artworks, showcases, and functions for UI. The different buttons are connected to a fetch method/function, which calls specific routes and queries depending on the button clicked. This allows the server to save data and values from the user, as well as return any data that the client needs to display.

**Using the Webpage**

There are different features available to the Webpage. There are operations for viewing the website and sorting artworks, and operations for adding artwork/changing descriptions.

The top UI has buttons labeled “Showall Showcases”, “Your Favorites”, etc. Each button can change the way each artwork is displayed, such as all the showcases being shown, or only the chosen User’s liked artworks/commented artworks being displayed. On the row below, there are buttons to sort every artwork by like, comment, depending on the order. The User gets saved on reload, rather than being defaulted to a specific one, and is chosen based on the local persistence.

Displayed below are the different showcases already saved on the server. There are like buttons to add a like onto each artwork, arrows to navigate through the showcase, and a display of the art’s title, description, and different comments.

To the right of the artwork is the CRUD operations for manipulating the showcase and art descriptions. There is a comment box, which contains a textbox allowing users to post comments by typing in the box, and clicking the “Comment” button below. There is also local persistence to save drafts of comments when the website is reloaded.

There is also an editing menu with different functions and buttons accordingly. To add new artwork to the showcase, the image file name must be typed in (i.e. “photo.png”), and will be displayed if saved on the server. Title and description can be added by typing in the boxes below before adding the artwork, or by using typing in the title and description box, and clicking the adjacent button after the artwork has been added.

The other functions are for removing specific features of the showcase. The “remove comments” tool clears all the comments on the art, the “remove artwork” removes the artwork from the gallery, and the “remove showcase” deletes the showcase as a whole (including all the artwork in the gallery).

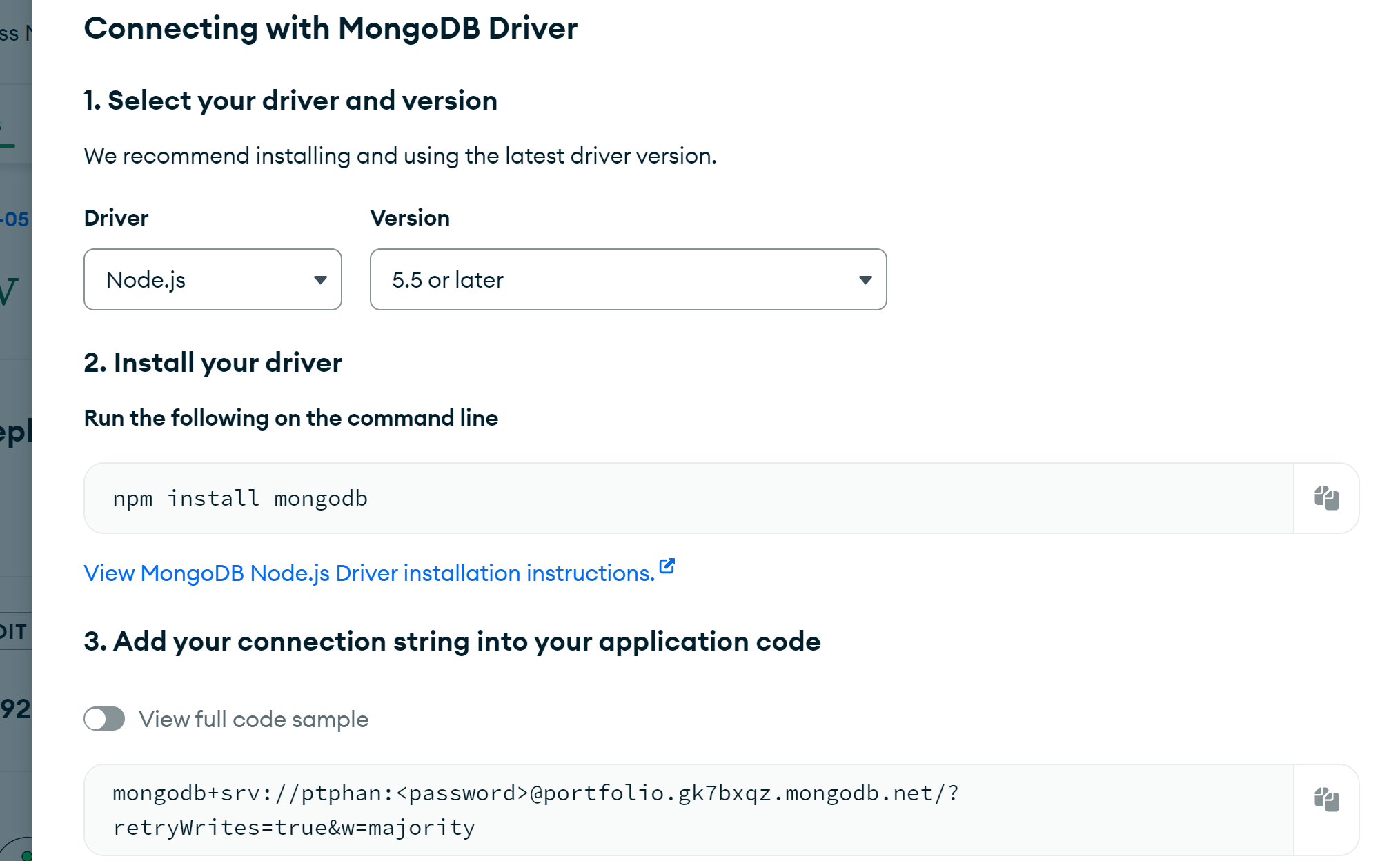
At the bottom of the page is an “Add a Showcase” textbox with a button next to the box. By typing in the textbox and clicking the adjacent button, a new showcase labeled by the textbox’s input will be created, and more art can be added to the showcase.

**Managing the Server/Database**

The images are stored on a directory, “/imgs/” on the server. To add images/art that can be accessed by the website, images can be saved into this directory.

The data is stored via 2 methods: local JSON file, and MongoDB database. Initially, the server will return the different showcases and images used by establishing a connection to the MongoDB database. However, if this fails, it uses the JSON file as a backup to load the website. The database stored can be accessed from the URL: <https://cloud.mongodb.com/v2/657b9b67abfe20528c8eadf6#/metrics/replicaSet/657b9b9ee064215917073950/explorer/data/101/find>

The page should be visible if access to the database is granted. On the website, you can authorize your IP and get a new user/password combo to connect to the database by visiting the overview, and clicking “connect” to the portfolio. State that you are using NodeJS drivers, and there should be a connection string to add to the server code as shown below:



Take the string and visit the file memcrud.js. At the top of the file, there should be a line, “const client = Mongo.MongoClient(…)”. Replace … with the connection string, and replace <password> with the password you’re using to connect to the database.

Now, you should be able to connect to the website and server accordingly. The webpage should be visible, and should load the different images, comments, descriptions, titles, and other features.

**Final Thoughts**

I would’ve liked to implement a way for users to upload the image to the server via the front-end, but there was no simple way to implement the feature without adding more complicated packages/technologies to the client side. However, the express package had image sending features available, which allowed the server to send images to the client for display. Many errors were involved with testing, which was solved through trial and error, and realizing that the test couldn’t be sped up without causing conflicts connecting to the server. Overall, I thought the project was a success, and reflects my journey on Web Development, and I believe I can utilize the skills learned in this class on my future endeavors.