Launching:

Make sure data folder shares path with database initializer

Run database initializer to init mongodb database

Make sure server shares a path with all of the files sent in the submission

Run server with node

Bonus implementation:  
extra notification (reviews, follows, likes)

Queries for users and workshops

Deleting workshops, art and reviews

Design:

Design was built around not coupling data for effective queries, this broke down a bit near the end of the assignment when considering adding more complex queries but I think it still works well enough. minimal use of helper functions for stuff like repeated queries since abstracting in small functions fractures queries into several and distributes data across multiple functions making it hard to keep track of what exactly you have. Generally this abstraction would be considered good but for the smaller scale of the website and the fact it was built small feature by small feature makes this abstraction difficult to implement. Given a refactoring of the website I would probably try doing this for some very basic queries, since doing this for complex queries would likely end up with one off functions which I consider to be poor design.

Website is static html only really using js for sending data back and forth, keeps the logic centralized at one point without needing to track data across host and client and minimizes latency since only need one request for html instead of many for base html, js, and json to build the website client side.

The website is not the greatest in terms of scalability due to not wanting to couple data for simplicity. This means data is store once and additional linear or quadratic time queries are needed to search through all the data to get what is needed. this could be improved by adding copies and references to the data where needed but this additional complexity and logic was too much for my time constraints.

All server operations were made to be asynchronous to prevent the server form being held up at any one point, expression due to how common database calls are on routes and html rendering is on routes. Given a refactor many of the asynchronous parts could be abstracted out allowing for the core of the functions to remain synchronous but this wasn’t something in my time frame.

Data transfer I wouldn’t consider to be bad, through it could further be reduced through adding some for js logic client side and route to facilitate this, though given the small about of data being sent and rendered in the first place, the js files needed to facilitate this would make this benefit minimal, especially given image caching and the size of headers possible being around the same size as some of the small html files.

Youtube link: silent (mic stopped working) <https://youtu.be/O1db0ESX35A>

(audio) <https://youtu.be/nD1mIyWqUw8>