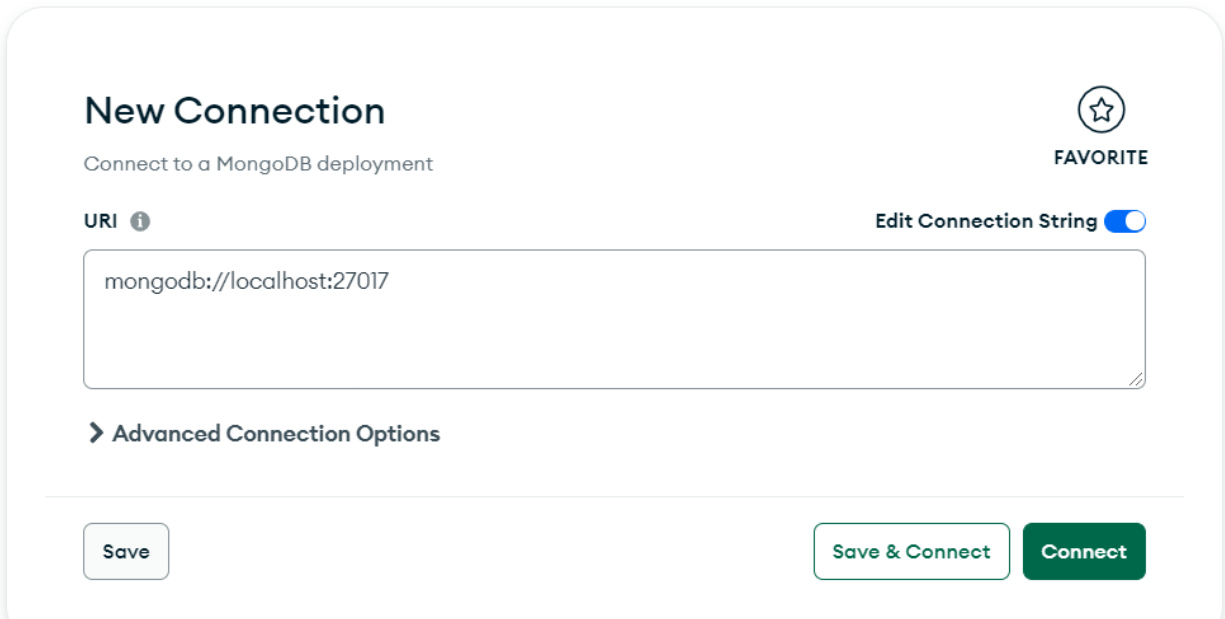


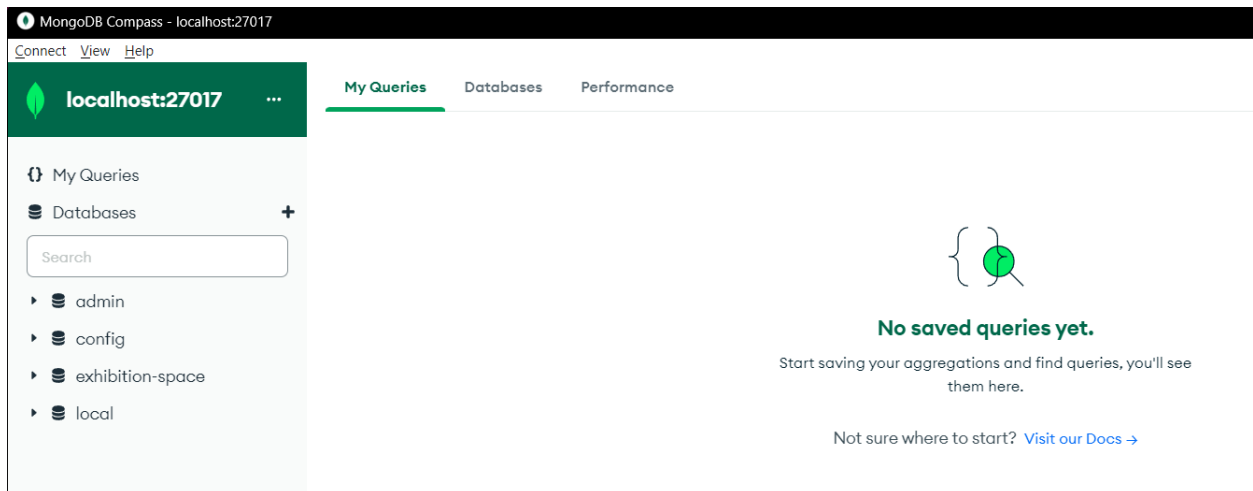
Youtube Link: [https://www.youtube.com/watch?v=E\\_MmjFIQugY](https://www.youtube.com/watch?v=E_MmjFIQugY)

To implement this program, please get the MongoDB Compass installed to run the database.

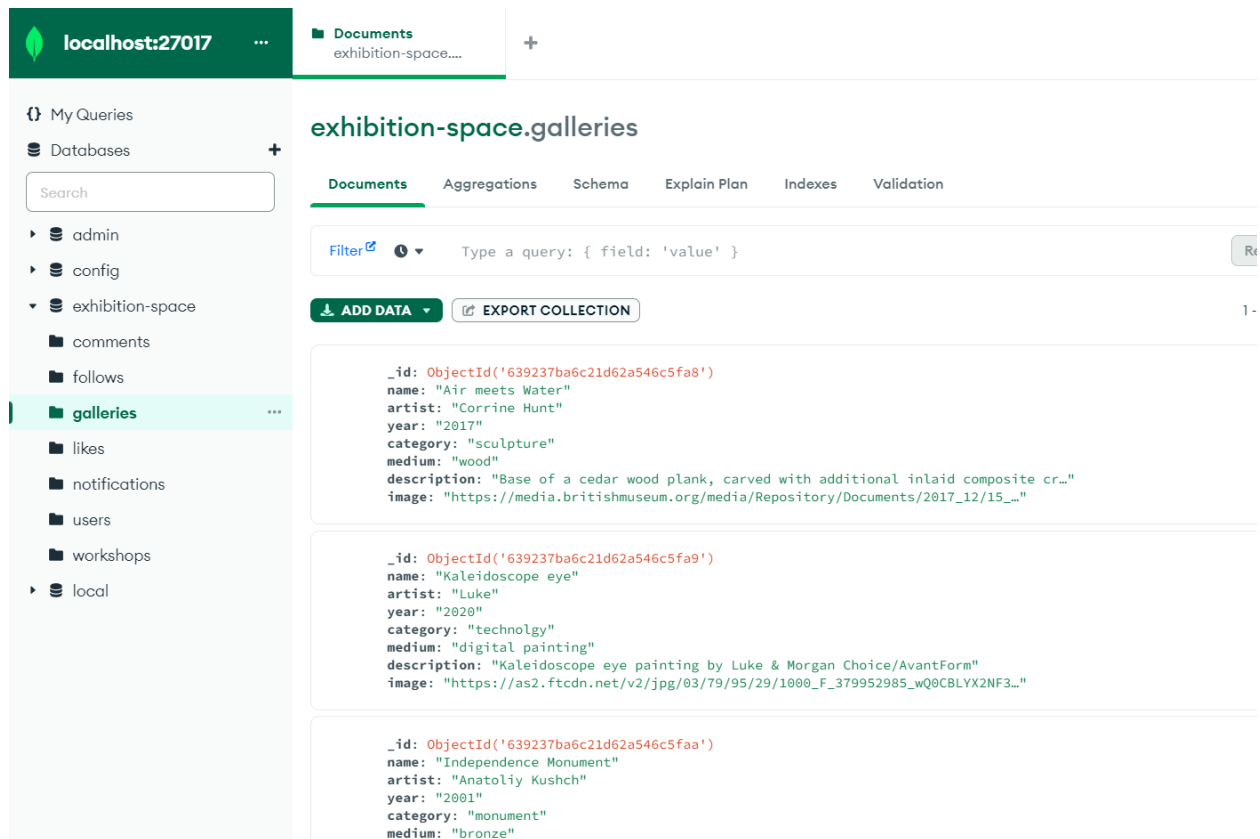
The steps to run it: 1. connect MongoDB to localhost:27017



The screenshot shows the 'New Connection' dialog in MongoDB Compass. At the top, it says 'New Connection' with a star icon and the word 'FAVORITE'. Below this, it says 'Connect to a MongoDB deployment'. There is a section for 'URI' with an information icon and a text input field containing 'mongodb://localhost:27017'. To the right of the URI field is a toggle for 'Edit Connection String' which is currently turned on. Below the URI field is a link for 'Advanced Connection Options'. At the bottom, there are three buttons: 'Save', 'Save & Connect', and 'Connect'.



2. add gallery.json file to the galleries



3. In the terminal, enter: npm install

4. In the terminal, enter: node server.js

5. Open: <http://localhost:3000>

In terms of designing this program, I created the RESTful CRUD APIs using the Express. Just simply require the express, then the server could easily handle the browser requests. It makes the whole code more concise and reduces the possibility of errors. Meanwhile, I used a lot of callback functions, to make all the input and output asynchronous. On the other hand, I saved the data in the database by MongoDB, which minimized of required data transfer. All the data was saved as an object, this includes imported artwork data (in 'galleries'), user liked works in 'likes', user data in 'users', etc. All of these features make scalability much higher and facilitate the maintenance of website data.

Other than this, I used another tool Mongoose, it makes the MongoDB much more easier. It provides a mapping from documents in the database to JavaScript objects in my program and provides many useful features to simplify our programming and interaction with Mongo. Additionally, it provides automatic reconnection attempts and automated keep-alive actions.

For the respective functions of these HTMLs I created:

index.html is the page to login

register.html is the page to register

addArtwork.html is the page to add the artwork

addWorkshop.html is the page to add the workshop

artist.html is the page to show the artist

home.html is the home page, display the first ten artworks and also the header

profile.html: you can change to artist or patron from here and check what you reviewed and liked

notification.html is the page to display the updates from the artists you followed.