Name: Divine Jojolola

Student Number: 101274703

Project Name: Open Gallery

Course Number: COMP 2406B

Youtube Link: https://www.youtube.com/watch?v=jStgepJkEfc

List of files and purpose:

1) Models.js: This file is essential for structuring and managing the database of an art-related platform using Mongoose with MongoDB. Its main purposes are:

   Defining Schemas: It outlines schemas for entities like Artworks, Users, Reviews, and Workshops, ensuring structured and detailed data representation.

Establishing Relationships: It links these entities through references, enabling interconnected data management.**Data Integrity and Searchability**: The file includes validation rules for data integrity and indexes for efficient searching.

Model Creation: It converts schemas into models, facilitating database interactions like creation, querying, and updating of records.

Module Exporting: By exporting these models, it allows their use across the application, promoting a modular and organized codebase.

2) Database-initializer.js: This file automates the process of populating a MongoDB database for an art platform. It connects to the database, reads art-related data from a JSON file, and uses this data to create and store new artist and artwork records in the database. The script ensures that each artist and their associated artworks are correctly linked and stored, streamlining the setup of the database with relevant content. Once the process is complete, it closes the database connection.
3) databaseHelpers.js: This file defines a collection of asynchronous functions for managing and interacting with data in an art platform's database. These functions handle various tasks such as retrieving, creating, updating, and

deleting records for artworks, users, reviews, and workshops. It includes functionalities like user authentication, liking and unliking artworks, following and unfollowing artists, managing reviews, and enrolling in workshops. The module is designed to provide a comprehensive set of operations required for the smooth functioning of the platform, ensuring efficient and organized data handling. The functions are then exported for use across the application.

5) 404.ejs: This HTML file is a template for a 404 error page, used in a web application when a requested resource is not found. The structure includes standard HTML tags with embedded JavaScript and CSS links for styling and functionality. The **<%- include("../partials/header", {isLoggedIn: isLoggedIn, notifications: notifications}); %>** line suggests the use of a templating engine (like EJS) to include a header partial, which adapts based on the user's login status and notifications. The content of the page features a humorous message acknowledging the missing resource, aimed at informing users in a light-hearted way that the page or content they are looking for cannot be found.

6) Artists.ejs: This HTML file is designed as a web page to display a list of artists, likely as part of an art-related website. It sets up the basic structure with meta tags, a title, and links to a CSS stylesheet and a JavaScript file for styling and interactive functionality.
   The page uses a templating engine, indicated by tags like **<%- include(...); %>** and **<% artists.forEach(function(artist) { %>** for dynamic content generation. The **<%- include("../partials/header", ...); %>** line includes a header section that adapts based on user login status and notifications.
   The main content of the page is within a div with the ID **artists-container**. It features a title and a container (** **) where artist details are dynamically inserted. Each artist's information is displayed using the **artistCard** partial, which is populated for each artist in the **artists** array (passed to the template engine).
7) Artwork.ejs: this renders the page for a single artwork
8) Artworks.ejs: this renders the page for all artworks
9) Home.ejs: this renders the home page

10)Login.ejs: this renders the login page

11)newArtwork.ejs: this renders the page for creating an artwork

12)newWorkshop.ejs: this renders the page for creating a workshop

13)register.ejs: this renders the page for creating a new patron

14) user.ejs: this renders the profile of a user

15) workshop.ejs: this renders the page of a single workshop

16) artistCard.ejs: this renders the card of an artist with its picture and text

17) artworkCard.ejs: this renders the card of an artwork with its picture and title

18)header.ejs: this renders the header of the site

19)server.js: it is responsible for handling all server actions of the site

20) serverHelpers.js: it has some helper functions for getting information from the database

How To Run Program

1) Navigate to root folder
2) Run `npm install` in terminal
3) Navigate to database folder
4) Run `node database-initializer.js` in terminal
5) Navigate to root folder
6) Run `node server.js` in terminal

Discussion and critique of my overall design

In developing the software, a key focus was placed on modularity and scalability. The project's architecture was meticulously organized, with files strategically divided into folders based on their functional relationships and similarities. This organization not only simplifies navigation through the codebase but also enhances its coherence and maintainability.

A significant emphasis was also placed on the reusability of functions. By designing functions to be as versatile as possible, the codebase becomes more robust and adaptable to various contexts, significantly aiding in scalability and future-proofing the application.

Furthermore, considerable attention was given to schema design. A well-thought-out schema is critical in maintaining data integrity and ensuring consistent relationships across various database schemas. This approach guarantees a stable and efficient database structure, essential for the application's long-term performance and scalability.

Lastly, the project adheres to best practices in REST API design, with a focus on intuitive and consistent naming conventions. This practice not only streamlines development and integration processes but also ensures that the API is user-friendly and easily navigable by other developers, fostering better collaboration and efficiency.

In summary, the development approach adopted for this project was rooted in principles of clarity, efficiency, and future adaptability, ensuring a robust and scalable software solution.

Extra functionalities:

1) I display enrolled users in a workshop
2) Artist gets notification when they create an artwork

Stylic Decisions:

I went for modern designs and garnered design inspiration from popular sites such as Instagram and Dribbble. I relied heavily on flexbox and grid to ensure that my design remained intact across multiple screens

Known Errors:

The pagination does not go pass page 2.

Sometimes, my node js server crashes on its own. All you need to do is just restart it and everything will be fixed