HTTP5222 Assignment 1 (Node.js Express app with JSON API)

Create a Node.js website using Express. The app will serve as an API (you will make requests to this API in a later assignment). Create at least two MongoDB collections to store data for your app. There should be admin pages in your Express app (using a template engine) to allow users to add data to all your collections. (DO NOT create a collection for menu links because this has already been done in class.) For other content you can hard-code some (when you later implement the public-facing pages in a later assignment) just to save some time.

You have two choices for your app content.

1. If you don't yet have a portfolio site, consider what data could be stored using a collection. For instance, what type of listable data would you have for a portfolio? It could be a collection of your projects and a collection of your skills.
2. If you do have a portfolio site and you don't want to create another one, the subject matter is up to you. For example, you can store data for a brochure site for a small business (e.g. you could have a collection for products, store locations, services, etc.).

Requirements:

1. Create MongoDB collections to store data for some of your app content (at least two collections).
2. Create an Express app for Node.js that has admin pages to **add data** to some MongoDB collections (at least two). The **admin pages** only should be rendered using a template engine (e.g. Pug or other). This means your admin pages are running server-side. (This is just to ensure you've got some practice with Express and rendering templates and also gives you an easy way to add data to your MongoDB collections.)
3. Add functionality to delete data as well. (Extra: add update for ease of use)
4. Your MongoDB collections should be online (e.g. on MongoDB Atlas) so that I can see the data in your app when I run the app code. (Make sure to make to add the entry under Network Access to **allow access from anywhere**.) Use realistic content in the collection (if you're working on your portfolio, the data should YOUR data).
5. Style the admin pages so it's presentable and it's easier to get to the admin pages (**this site is only an admin dashboard/site**, no public-facing pages).
    - You may use a CSS framework but change colours and fonts. (It shouldn't be obvious which framework was used at first glance.)
    - Your (admin) pages should be responsive. (You can keep it simple.)
6. Create API endpoints which return the JSON array for your collections (e.g. if two collections, you should have two API endpoints). (We'll cover how to create your own REST API and return JSON in week 7.)
7. Deploy the app online (include the URL to your deployment in a README).

You will be marked on:

- **Code quality (10)**
- **Design/Usability (5)** (design/usability for admin pages and usability of data structure)

This assignment is **individual** and is worth 17% of your final grade (scaled to be out of 17 on Bb). You can put your web app on GitHub and just submit your GitHub link (.gitignore the ***node_modules*** folder) along with the link to your deployed API. If you don't manage to deploy include any info needed to run the app (e.g. if DB credentials missing from GitHub—which is safer—just upload your .env file on Bb).

Due date: **June 30, 2025 @ 11:59pm**

Criteria	Excellent	Good	Satisfactory	Somewhat Unsatisfactory	Unsatisfactory
Code Quality (/10)	Code works, is efficient, well-commented and formatted. Code follows best practices. Data structure is good and makes sense for the content. Up to two very minor issues.	Code works, is commented and formatted well. Code mostly follows best practices but for a few minor issues.	Code is a little buggy, is not always commented or formatted well but a good attempt. Code not always efficient.	Code is buggy and poorly commented. Some issues following best practices.	Code doesn't work and is not commented. Code has many major issues.
Design (/5)	Design (for admin pages) is good. Admin website is easy to use and data structure from API is flexible/robust.	Design is mostly professional but with a few minor issues. Website is easy to use and follow but with some minor issues.	A good attempt but feels like student work. Website is mostly fine but with many minor issues or at most one or two major issues.	Design is not really complete and is obvious it's student work. Website has some major issues around usability/accessibility.	Design not really fleshed out at all. Design is not complete. Website is confusing/hard to use. Website not accessible.