DA219B - Lab 1

MongoDB Preparation

Sign up for a free MongoDB Atlas account if you haven't already: MongoDB Atlas.

- Make sure your IP is whitelisted in the MongoDB Atlas settings.
- Create a new MongoDB cluster and database.
- Obtain the connection string for your MongoDB database. This will be used in your Node.js application to connect to the database.

TASKS

1. Create an .env file and add the PORT and the CONNECTION_URL to it. (**Hard coding them will not be accepted.**)

2. Create a collection to store **dish** information. Each dish should include:

- id, name, ingredients, preparationSteps, cookingTime, and origin Add at least **five sample dishes**.
- At least one dish should be personal (e.g., cultural or favorite).
- Include **one custom field** such as spiceLevel, servings, or difficulty to show originality.

3. Create an HTML File (index.html)

- When you start your Node.js application, visiting http://localhost:5000 should display this page.
- This page will display the recipe data from your database.

4. Build CRUD Operations for Dishes using a REST API:

- **GET /api/dishes** – Return a JSON array with all dishes.
- **GET /api/dishes/:name** – Return a dish by name. Return 404 if it doesn't exist.
- **POST /api/dishes** – Add a new dish. Return 201 if successful, or 409 if it already exists.
- **PUT /api/dishes/:id** – Update an existing dish. Return 404 if it doesn't exist.
- **DELETE /api/dishes/:id** – Delete a dish. Return 404 if it doesn't exist.

5. Populate index.html with API Data

- Use **Fetch or Axios** to retrieve and display dish data (without any frontend frameworks).
- Create a table showing all recipes from the API. Each row should have:
    - an **"Update" button** in each row to allow editing and submitting updated data via PUT.
    - a **"Delete" button** with confirmation prompt.
- Provide a form for adding new recipes using POST. Update the table after the new recipe submission.

📌 Submission Requirements

✅ **Comment your code**: Add clear and meaningful comments in your code explaining key parts like database connection, route handling, and error management.

✅ **Set up .gitignore**: Make sure your .gitignore file excludes node_modules, .env, and any other personal or unnecessary files.

✅ **Use Git properly**: Create a Git repository and push your project. Show progress through multiple commits made over time — not just one big final commit.

✅ **Reflect briefly**: Add a short note describing any challenges you faced and how you solved them (1–2 sentences is enough).

✅ **Test all your routes**: Use Postman or your browser to test your GET, POST, PUT, and DELETE routes. Be ready to demonstrate these tests during the seminar session.

GOOD LUCK!

This is your chance to show your creativity and understanding. AI can help, but your real learning shows in how you test, reflect, and personalize your work.

— Deema Aloom
