ANGULAR LAB 4: SHOPPING CART

Task: Create a folder that houses two subfolders. One subfolder for the Express server and the other for your Angular application. Both subfolders will be unique repos on Github. Build a REST API with an Express server. Create a module that contains routes for your front-end to communicate with. Test the endpoints with Postman. Add a front-end to display the shopping cart items from your back-end API.

What does the application do?

6. The back-end REST API provides access to an array of shopping cart items.
7. The back-end will have routes for GET, POST, PUT, and DELETE which allows our front-end to communicate with our server. Each route will be handling the following functionality:
    a. `GET /cart-items` returns an array of all items
    b. `POST /cart-items` for now, log the **body** to the console. (later, this will add a new item to the list)
    c. `PUT /cart-items/:id` for now, log the **id** param and the **body** to the console. (later, this will replace an item in the list)
    d. `DELETE /cart-items/:id` for now, log the **id** param to the console. (later, this will delete an item from the list.)
8. When you serve this Angular application, it fetches the shopping cart items from the API and displays them **beautifully**.

Build Specifications:

Server Side

6. Use Express to create your server.
7. Require the module that will contain the routes you have created.
8. Start your server out with a hard-coded array of cart items, each including `id`, `product`, `price`, and `quantity`.
9. Test your endpoints using Postman.

Client (Angular) Side

1. Build an Angular app
2. Create an interface called `Item`:
    a. product: string, price: number, quantity: number, edit?: boolean;
3. Create a component named `Products`
4. Create a service named `Cart` in Angular. Give it a `getAllItems()` method that uses http to make a GET request to your /cart-items API.
5. Display the cart items from the service in the `Products` view.
6. For this part of the lab, we do NOT yet need to handle POST, PUT, and DELETE on the Angular side.

Bonus:

1. Modify your POST endpoint to add an item to the array.
2. Modify your DELETE endpoint to remove an item from the array, based on the ID.
3. Modify your PUT endpoint to replace an item in the array, based on the ID.

Sample designs: