

## ANGULAR LAB 4: SHOPPING CART

**Task:** Build an Angular app to call your Express cart items API and display results. Then use a form to add items and button clicks to delete items.

### What does the application do?

- Fetches all the cart items from your cart API (Express Lab 1 or Express Lab 2). Displays the items with all of their details: product, price, and quantity.
- Allow the user to delete individual items from the cart.
- Allow the user to fill out a form to add additional items to the cart.

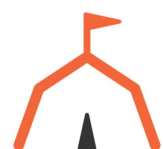
### Build Specifications:

1. Build an Angular app
2. Create an interface called **Item**, having these properties
  - a. **product**: string, **price**: number, **quantity**: number, **id**?:number;
3. Create a service named **CartApi** in Angular.
  - a. Add a **getAllItems()** method that uses http to make a GET request to your `/cart-items` API endpoint.
  - b. Add a **deleteItem(id:number)** method that uses http to make a DELETE request to your `/cart-items/:id` API endpoint.
  - c. Add an **addItem(item:Item)** method that uses http to make a POST request to your `/cart-items` API endpoint.
4. Create a component named **Products**
  - a. Call the CartApi service **getAllItems** method. Display the cart items from the service in the **Products** view.
  - b. Each item displayed in **Products** should include a delete button. When that button is clicked, call the CartApi service **deleteItem** method with the appropriate item's ID. When the call completes, call **getAllItems** again to refresh the display.
  - c. Add a form where the user can fill out a new product, price, and quantity. When the form is submitted, call the CartApi service **addItem** method with the appropriate Item object based on the form inputs. When the call completes, call **getAllItems** again to refresh the display.

### Extended Challenges:

- Enable items to be edited (or perhaps just the quantity changed). Use the PUT endpoint.

continued on next page...



## Sample designs:

