Assignment 9: Implementing Edit and Delete Capabilities in the Shopping Cart

Background: In Assignment 8, you built functionality to add products to a catalog stored in a JSON file. Now, we will extend the shopping cart application by adding **Edit** and **Delete** capabilities. Users can adjust the quantity of items in their cart or remove items entirely. These changes will update the cart dynamically, using PHP to handle form submissions and update the cart data stored in the JSON file.

Objective: You will modify the existing shopping cart system by implementing:

1. Edit Capability: Users should be able to change the quantity of any product in their cart.
2. Delete Capability: Users can remove products from the cart entirely.

Modifying the cart.json file, which stores the cart data, will reflect these actions. After editing or deleting, the cart should update to display the current items and total cost.

Requirements:

1. Display Shopping Cart

- Create or extend a PHP page (cart.php) to display the shopping cart contents.
    - Show each item's product **name**, **price**, **quantity**, and **total cost** (price multiplied by quantity).
    - The cart should also display a summary of the **total number of items** and the **total cost** of the cart.

2. Edit Product Quantity

- Implement an "Edit" feature that allows the user to change the quantity of a product in the cart.
    - When the user updates the quantity, the cart data should be updated in the cart.json file.
    - Prevent negative or zero quantities (optional: if the quantity is set to zero, delete the item).
    - Recalculate the total cost of the cart after each edit.

3. Delete Product

- Add a "Delete" button for each product in the cart.
    - When the user clicks "Delete," the item should be removed from the cart (and from the cart.json file).
    - After deletion, the cart summary (total items and total cost) should be updated.

4. Form Handling

- Use PHP to handle form submissions for editing and deleting cart items.
    - Ensure that form submissions are processed securely, and the cart is updated correctly.
    - Redirect the user to the updated cart after each action (edit or delete).

Implementation Steps:

1. Read and Write Cart Data from cart.json

- Extend the existing JSON-based cart system. Store cart data in *cart.json* and display the cart contents by reading from this file.
- Example JSON structure for the cart:
  [
  {“name”:”product1”,”price”:3,”quantity”:3},
  {“name”:”product2”,”price”:1,”quantity”:2}
  ]

Display Shopping Cart

- On your cart.php page, read the cart.json file and dynamically display the cart contents.
- For each product, display:
    - Product name
    - Price
    - Quantity (with an editable input field)

- Total price for that product (price multiplied by quantity)
- A "Delete" button to remove the product from the cart

3. Edit Product Quantity

- Provide a form that allows the user to update the quantity of a product.
- After submitting the form, update the quantity in the cart.json file, and recalculate the cart's total cost.

4. Delete Product from Cart

- Implement a "Delete" button next to each product that allows the user to remove an item from the cart.
- When the delete button is clicked, remove the product from the cart.json file and update the cart accordingly.

Deliverables:

- **cart.php**: The main shopping cart page allows users to view, edit, and delete products from their cart.
- **product_catalog.php**: This is the product catalog page from Assignment 8, which allows users to continue adding products to the cart.
- **cart.json**: The JSON file that stores the cart data.
- **Any associated CSS**: To style and present the cart in a user-friendly manner.

Grading Criteria:

- **JSON File Handling (30%)**
  - Correct use of PHP to read and write to the cart.json file.
  - Data updates correctly based on user actions.
- **Edit and Delete Features (40%)**
  - Items can be edited, and their quantities updated successfully.
  - Items can be deleted from the cart, and the cart updates dynamically.
- **Form Handling and Validation (20%)**

  - Correct handling of forms for editing and deleting cart items.
  - Input validation for quantities (no negative numbers, etc.).

- **Code Quality and Presentation (10%)**
  - Clean and well-structured code.
  - User-friendly design and presentation of the shopping cart.