Task 5 – Ideas for Shopping Cart Implementation

1. Goal of the Shopping Cart

The shopping cart allows users to:

- Collect products while browsing the shop
- View selected items in one place
- Modify quantities or remove products
- See the total price before checkout

The shopping cart acts as **temporary storage** between browsing and purchasing.

2. Where the Shopping Cart Should Be Stored

Best Practice: Server-side Session (PHP Sessions)

The shopping cart should be stored using **PHP sessions** on the server.

Reason:

- Sessions are secure
- Data cannot be manipulated easily by the user
- Each user gets an individual cart

Alternative approaches (briefly):

- Cookies / LocalStorage → easier but insecure
- Database → useful for logged-in users (advanced)

For our web shop (no login required yet), PHP sessions are the best choice.

3. Structure of the Shopping Cart Data

The shopping cart does **not** store full product information.
It stores **references** to products.

Each cart item contains:

- Product ID (pid)
- Product name
- Price (at the time of adding)
- Quantity

Why this is good practice:

- Lightweight
- Faster processing
- Product details can still be loaded from JSON files if needed

4. Adding Products to the Cart

Conceptual Flow:

1. User clicks **“Add to Cart”**
2. Product ID is sent to the server (via POST or GET)
3. Server checks:
    - Is the product valid?
    - Is it already in the cart?
4. If yes → increase quantity
5. If no → add as new cart item

The cart is then saved in the **PHP session**.

5. Shopping Cart Page (cart.php)

The shopping cart page should display:

- List of all selected products
- Product name
- Quantity
- Single price
- Subtotal per product
- Total cart price

User interactions on cart page:

- Increase or decrease quantity
- Remove product from cart
- Continue shopping
- Proceed to checkout

All calculations (totals, prices) should be handled **server-side**.

6. Updating and Removing Items

Quantity Update:

- User changes quantity
- Request sent to server
- Session data updated
- Page refreshed with new totals

Removing an Item:

- User clicks “Remove”

- Product ID sent to server
- Item deleted from session cart

7. Cart Lifetime

The shopping cart exists:

- As long as the session is active
- Until the browser is closed or session expires

This behavior is intuitive for users and easy to manage.

8. Error Handling

The system should handle:

- Missing product ID
- Invalid product ID
- Empty cart
- Direct access to cart without items

In such cases, a clear message should be shown (e.g. *“Your cart is empty”*).

9. Security Considerations

- Prices must **never** be trusted from the client
- Product IDs must be validated
- Total price must always be recalculated server-side
- Session data prevents manipulation

10. Optional Extensions (Future Work)

- Persist cart for logged-in users using database
- Merge guest cart after login
- AJAX updates without page reload
- Checkout and payment integration

Conclusion

The proposed shopping cart implementation:

- Uses **PHP sessions**
- Is secure and scalable
- Fits the current project scope
- Can easily be extended later