MERN Stack Training
Company: Sensation Software Solutions
Student Name: Gurvinder Singh
Training Duration: 6 Months
Days: 76

Objective of the Day

The objective of Day 76 was to initiate and implement the Cart Management System on the GadgetShop frontend application. The primary goal was to ensure that authenticated users could add products to the cart, view the items dynamically, and manage the cart at a basic level. This phase focused on developing the Cart interface, data handling structure within React, interaction with product pages, and integrating a basic authentication check to secure actions related to the cart. The development aimed to build a foundation for the e-commerce flow by connecting product listing with the cart experience.

Overview of the Task

The cart system is a critical module in any e-commerce application, and on Day 76, the focus was to structure its core operational aspects. The task breakdown consisted of:

1. Designing the **Cart UI layout** and reusable components.
2. Developing an **add-to-cart action handler**.
3. Displaying items inside a **central cart layout**.
4. Implementing an **authentication verification layer** which only allows logged-in users to interact with the cart.
5. Structuring state management for cart operations.

The system was developed using **React (functional components), Hooks for state management, Tailwind CSS for styling, and shadcn/ui** for interface consistency.

Work Done on Day 76

1. Setting Up the Cart Component

A file named `Cart.jsx` was created within the `src/components` directory to manage the cart interface. This component is responsible for rendering all items added to the cart and displaying the subtotal and total values. The UI structure was divided into:

- Item listing container

- Product info section (image, name, price)
- Quantity modification area (future expansion planned)
- Remove item option placeholder
- Cart summary sidebar

This structure ensures scalability for upcoming features like quantity selection, promo codes, and checkout integration.

2. Configuring Cart State Management

For handling cart data, the `useState` hook was implemented initially, with a structure designed to transition into `Context API` or a state management library like Redux in later phases. The cart state includes:

- Product ID
- Name
- Price
- Image URL
- Category
- Quantity (default: 1)
- Subtotal calculation basis

The state design ensures that each operation on the cart (add, remove, list) can be executed without conflicts or redundancy.

3. Integrating Authentication Check

Basic authentication logic was integrated using a conditional check that verifies if the `authToken` or `userSession` is stored in localStorage. If the user is not logged in and attempts to add a product to the cart, the system prevents the action and navigates the user to the login page for security enforcement.

The authentication check was added to the Add to Cart button logic and Cart navigation logic, ensuring that only authenticated users can proceed.

4. Adding Add to Cart Button in Product Cards

Every product displayed on the product list page or details page now contains an Add to Cart button. On clicking the button, the respective product object is passed to the cart state handler and pushed inside the cart state array. The structure handles duplicate entries by checking if an item is already present before adding a new entry.

5. Styling and UI Development

Interface development was executed with Tailwind CSS utility classes to ensure responsive and scalable UI. Elements such as flex grids, spacing, borders, responsive breakpoints, and hover states were used. The layout supports tablet and mobile views seamlessly.

The UI includes:

- Product image left-aligned
- Details to the right
- Price aligned based on screen size
- Action buttons for user interactions
- Cart summary displayed with fixed width on the right side for desktop views

---

Hands-on Practice

After developing the initial cart structure, practical testing was initiated:

1. Multiple products were added to the cart to verify if the system correctly updated the state.
2. Duplicate entry prevention checks were tested for proper functionality.
3. Removal actions were tested conceptually by validating state manipulation (full delete functionality planned for further days).
4. Mobile responsiveness was verified using browser developer tools by testing system behavior on various breakpoints.
5. The authentication layer was checked by attempting cart modifications with no active session.

All tests verified that the baseline cart structure was working as intended.

Conclusion

Day 76 marked the beginning of the Cart System implementation in the GadgetShop project. The core functionality of adding items to the cart, saving their state, displaying them in a structured format, and implementing access control based on user login status was completed successfully. This provided a strong foundation and direction for upcoming work involving cart manipulation, checkout flow, data persistence, and API interaction. The necessary UI and functional groundwork is now in place to expand on advanced features in subsequent days.