SHOPPING CART

Document Version Control

Date Issued	Version	Description	Author
30 – 11 – ‘24	v1.0	Initial HLD	Ajanth
03 – 12 – ‘24	v1.1	Updated KPI — V1.1	Castro R S Jeev
06 – 12 – ‘24	v1.2	Updated LLD & Wireframe	Febin Anto K K
09 – 12 – ‘24	v1.3	Added Deployment Details	Jacob Daniel R
12 – 12 – ‘24	v1.4	Completed Final Submission	Sham S

Contents

Abstract:

This high-level design (HLD) document outlines the architecture and implementation details of a Shopping Cart website built using React, Redux, Tailwind CSS, and AOS. The platform aims to provide a seamless online shopping experience, enabling users to browse, select, and purchase products efficiently.

The frontend is developed using React to create dynamic and interactive user interfaces. Redux is employed for efficient state management, ensuring data consistency across components. Tailwind CSS is utilized to style the website, offering a modern and responsive design. AOS library is integrated to enhance user experience with smooth animations on scroll.

1.Introduction:

1.1 Why this High-Level Design Document?

A High-Level Design (HLD) document is a crucial artifact in the software development lifecycle. It provides a comprehensive overview of the system's architecture, key features, and technical implementation details.

The HLD will:

- **Clear Vision:** It establishes a shared understanding of the system's goals and scope among the development team, stakeholders, and clients.
- **Technical Blueprint:** It acts as a blueprint for the detailed design and implementation phases, guiding the development process.
- **Risk Mitigation:** It helps identify potential risks and challenges early in the development cycle, enabling proactive mitigation strategies.
- **Decision Making:** It supports informed decision-making regarding technology choices, resource allocation, and project timelines.
- **Effective Communication:** It facilitates effective communication between different teams and stakeholders involved in the project.

By providing a clear and concise overview of the system, this HLD document ensures that the Shopping Cart website is developed efficiently, meets the specified requirements, and delivers a high-quality user experience.

1.2 Scope:

This High-Level Design (HLD) document outlines the architecture and implementation details of a **shopping cart** application built using React, Redux, Tailwind CSS, and AOS. The application aims to provide a seamless online shopping experience, enabling users to browse, select, and purchase products efficiently.

1.3 Definitions:

Term	Description
Frontend	The user interface of the shopping cart application developed using React and styled with Tailwind CSS for responsiveness and interactivity.
API	APIs are used to fetch product details, cart updates, and manage user sessions in the shopping cart application.
Redux	Manages the state of the application, ensuring consistent and predictable updates to cart items, user sessions, and UI components.
Tailwind CSS	Provides a utility-first CSS framework, enabling rapid and consistent styling of the shopping cart application's interface.
AOS	Enhances user experience by animating elements like product cards and banners as they scroll into view.
IDE	Visual Studio Code is recommended for developing the React-based shopping cart frontend.

2.General Description:

2.1 Product Perspective:

The Shopping Cart application is built using **React**, **Redux**, **Tailwind CSS**, and **AOS** to provide a seamless eCommerce experience. It allows users to browse, select, and purchase products while managing the cart state efficiently. The frontend leverages **React** for dynamic UI components, **Redux** for centralized state management, **Tailwind CSS** for responsive and customizable styling, and **AOS** for adding smooth scroll animations to enhance the user experience

Problem statement:

The goal is to create a highly interactive and user-friendly **shopping cart** for an eCommerce platform, enabling users to manage products in their cart efficiently and proceed to checkout. The key features include:

- Viewing and managing cart items.
- Calculating prices, taxes, and discounts in real-time.
- Handling user authentication and session management.
- Implementing responsive design for seamless use across devices.

2.2 Proposed Solution:

The proposed solution is a **React-based shopping cart application** that integrates **Redux** for state management and **Tailwind CSS** for UI styling. Users can interact with the cart by adding or removing items, adjusting quantities, and viewing total prices. The **AOS** library will provide smooth animations for elements as they scroll into view, enhancing the overall experience. The cart also integrates with APIs for dynamic product data and user session management.

2.3 Further Improvements:

Future improvements for the shopping cart application could include:

- Integration with backend technologies (e.g., **Node.js** or **Django**) for handling user authentication, product management, and order processing.
- Adding a wish list feature for users to save products for future purchases.
- Implementing payment gateway integration (e.g., **Stripe**, **PayPal**) for smooth checkout experiences.
- Adding real-time stock updates and product recommendations based on user behaviour.

2.4 Technical Requirements:

The shopping cart will be built using the following technologies:

- **React** for dynamic rendering of the UI.
- **Redux** for managing application state, including cart contents and user sessions.
- **Tailwind CSS** for creating a responsive and modern design that adapts to different screen sizes.
- **AOS** for adding scroll animations to product cards, cart details, and checkout pages.
- The application will be tested for cross-browser compatibility and mobile responsiveness.

2.5 Data Requirements:

The application will manage the following data:

- Product details, including name, price, description, and images.
- Cart data, including selected products, quantities, and total prices.
- User session data for managing logged-in states and personal information.
- The product data will be fetched from an API or database, and the cart state will be managed using **Redux** to ensure real-time updates across the application.

2.6 Tools used:

JavaScript and frameworks such as **React**, **Redux**, **Tailwind CSS**, and **AOS** are used to build the entire frontend of the shopping cart application, ensuring a dynamic and responsive user interface, efficient state management, and smooth scroll animations.

- **React**: A JavaScript library for building dynamic, interactive user interfaces, powering the main functionality of the shopping cart.
- **Redux**: A state management library to handle the global state of the application, ensuring consistent data flow, particularly for managing the cart and user sessions.
- **Tailwind CSS**: A utility-first CSS framework that enables rapid development of a responsive and customizable design for the shopping cart interface.

- **AOS (Animate On Scroll)**: A library for adding scroll animations to elements like product cards and cart details to enhance the user experience.
- **Visual Studio Code**: The preferred IDE for writing, testing, and debugging the React application.
- **GitHub**: Used for version control, collaborating with other developers, and keeping track of project changes.

Hardware Requirements

- **PC or Laptop**: Ensure that your system supports the necessary software for frontend development (e.g., Node.js, npm, React).
- **High-speed Internet Connection**: Required for fetching API data, interacting with online services, and deploying the application.
- **Responsive Device**: For testing the application's responsiveness across different screen sizes (mobile, tablet, and desktop).
- **Keyboard and Mouse**: Essential for development and testing.

2.7 Constraints

The **shopping cart frontend application** must be user-friendly, with an intuitive interface that allows users to easily browse, select, and manage products in their cart. It should be fully responsive, ensuring seamless usability across various devices (e.g., mobile, tablet, and desktop). The application should also be optimized for performance, minimizing loading times and ensuring smooth interactions. Users should not need to understand the underlying workings of the application, focusing only on the shopping experience.

2.8 Assumptions

The primary objective of this project is to create an efficient, visually appealing, and interactive shopping cart using React, Redux, Tailwind CSS, and AOS. It is assumed that:

- Users will have a modern web browser to access the shopping cart.
- The application will integrate smoothly with backend APIs for managing product data, user sessions, and order processing.
- The frontend will handle all state management through **Redux**, ensuring data consistency and real-time updates.
- The application will be designed to work seamlessly on various screen sizes and devices

3.Design Details:

Design Details for the Shopping Cart Project:

1. User Interface (UI) Design

1.1 Product List Page:

- Layout:
  - Grid layout with cards for each product.
  - Each card should include:
    - Product name.
    - Product price.
    - "*Add to Cart*" button.
- Styling:
  - Use a consistent color scheme (e.g., light background, accent colors for buttons).
  - Add padding and spacing between product cards.
  - Button:
    - Background color: White.
    - Hover effect: Darker shade.
    - Rounded corners for a modern look.

1.2 Cart Page:

- Layout:
  - List items vertically with the following details:
    - Product name.
    - Product price.
    - Quantity selector (dropdown or +/- buttons).
    - Remove button.
  - Total price at the bottom.
  - "*Clear Cart*" button next to the total price.
- Styling:
  - Highlight each cart item with a subtle border or background color.
  - Keep total price and buttons sticky at the bottom for easy access.

1.3 Navigation Bar:

- Items:
  - Home (Product List).

- Special Product
- About us
- Cart (with item count badge).

- Styling:
  - Use a top bar with a consistent color.
  - Badge:
    - Circle with the number of items in the cart.
    - Position: Top-right of the cart icon.
    - Color: Red for the badge, white for the text.

2. Responsive Design:

- Mobile View:
  - Switch to a single-column layout for the product and cart pages.
  - Use a collapsible menu for navigation.
- Desktop View:
  - Use a grid layout for product cards (e.g., 3-4 items per row).
- Media Queries:

```
@media (max-width: 768px) {
 .product-card {
  width: 100%; /* Full width on mobile */
 }
 .nav-bar {
  flex-direction: column;
 }
}
```

3. State Management:

- Cart State:
  - Use a global state or context to manage the cart.
  - Store cart data in the following format:

```
[
 { id: 1, name: "Product 1", price: 100, quantity: 1 },
 { id: 2, name: "Product 2", price: 200, quantity: 2 },
];
```

- UI Updates:
    - Dynamically update the cart count in the navigation bar.
    - Re-render the cart page when items are added, removed, or updated.

4. Fonts:

- Use modern fonts like Roboto, Open Sans, or Lato.

5. Icons:

- Use Font Awesome or React Icons for:
    - Cart icon in the navigation bar.
    - Remove button in the cart.
    - Quantity adjustment (+/-).

6. Interactivity:

- Add hover and click effects:
    - Buttons: Change background color and scale slightly on hover.
    - Links: Underline or color change on hover.

7. Accessibility:

- Add ARIA labels for buttons and navigation:

  <button aria-label="Add Product 1 to cart">Add to Cart</button>
- Use sufficient color contrast between text and backgrounds.

4. Performance:

The shopping cart application should ensure seamless user interactions, such as quickly adding items to the cart, updating the cart state, and navigating between pages. Efficient state management is critical to avoid delays or lags, ensuring a smooth user experience. Additionally, performance optimization techniques like lazy loading for assets and minimizing re-renders should be considered.

4.1 Reusability:

The code and components written for this project should follow modular design principles. Components like Product Card, Cart Item, and state management logic should be reusable in other e-commerce or similar projects without modification. Proper abstraction and documentation will enhance reusability.

4.2 Application Compatibility:

The project is built using ReactJS, ensuring compatibility across modern web browsers (e.g., Chrome, Firefox, Edge). Each component (e.g., product list, cart page) performs a specific task, with React ensuring the proper flow of data through props and state management.

4.3 Resource Utilization:

The application is lightweight and ensures efficient utilization of browser resources. Features such as efficient rendering, caching, and minimal use of external libraries help reduce processing overhead. The application prioritizes responsiveness even on devices with limited resources.

4.4 Deployment:

The application is deployed on GitHub Pages for easy access and demonstration. The deployment process uses build tools like React Scripts to bundle and optimize the project. The project can also be hosted on other platforms like Netlify or Vercel for scalability and enhanced performance.

Conclusion:

The Shopping Cart Project successfully demonstrates the core functionalities of a modern e-commerce application. It provides a user-friendly interface for browsing products, adding them to a cart, and viewing cart details. By leveraging technologies like ReactJS, the project ensures efficient state management and seamless user interactions.

This project also emphasizes modular and reusable code, adhering to best practices in frontend development. The absence of a database in this version simplifies the architecture, making it ideal for showcasing frontend skills while leaving room for future scalability. Potential enhancements include integrating a backend for data persistence, implementing authentication, and adding dashboards for administrative functionalities.

Overall, this project serves as a strong foundation for developing more advanced e-commerce solutions, showcasing practical application development skills in a structured and scalable manner.