Low Level Design (LLD)

Shopping Cart

Document Version Control

Date Issued	Version	Description	Author
30-11-2024	1.0	Initial Draft of Low-Level Design Document	Ajanth, Arshekh John J D
2-12-2024	1.1	Added detailed component functions	Febin Anto K K
4-12-2024	1.2	Defined data structures for cart management	Castro R S Jeev
7-12-2024	1.3	Updated state management and API integration	Brijisha B S, Akshaya S S
9-12-2024	1.4	Added navigation flow and routing details	Jacob Daniel R
12-12-2024	1.5	Reviewed and finalized the LLD	Sham S

Contents

Abstract

With the rapid growth of eCommerce, the demand for user-friendly and efficient shopping platforms has significantly increased. To address this need, the development of a robust frontend for a shopping cart application is essential. This application is designed to enhance the online shopping experience by providing seamless product browsing, intuitive cart management, and responsive user interfaces.

The shopping cart frontend leverages modern technologies such as **React** for building dynamic user interfaces, **Redux** for state management, **Tailwind CSS** for responsive styling, and **AOS** for enhancing the user experience with smooth animations. By focusing on performance, scalability, and usability, this project aims to deliver a reliable solution for modern eCommerce platforms.

1 Introduction

1.1 Why this Low-Level Design Document?

The purpose of this document is to present a detailed description of the **Shopping Cart Frontend Application**. It will explain the features, functionalities, and behaviour of the application, as well as the constraints under which it must operate. This document is intended for both the developers and stakeholders and will serve as a blueprint for implementing the frontend system efficiently.

The main objective of the project is to create an interactive and responsive shopping cart that allows users to browse products, manage their cart, and proceed to checkout seamlessly. The project

leverages **React**, **Redux**, **Tailwind CSS**, and **AOS** to deliver a dynamic and visually appealing experience.

The shopping cart application includes:

- **Product Display**: A dynamic interface for users to view product details and select items.
- **Cart Management**: Real-time addition, removal, and modification of items in the cart.
- **Responsive Design**: Ensures compatibility across various devices, including desktops, tablets, and mobiles.
- **Scroll Animations**: Enhances user experience using smooth animations with **AOS**.
- **State Management**: Consistent and predictable application state handled by **Redux**.

This project shall be delivered in two phases:

- **Phase 1**: Implementation of core functionalities, including product display and cart management, using **React** and **Redux**.
- **Phase 2**: Enhancement of user experience with **Tailwind CSS**, **AOS**, and additional UI improvements.

1.2 Scope

This software system will be a **web application** designed to provide a seamless and interactive online shopping experience. The system will allow users to browse products, add or remove items from their cart, and proceed to checkout. Key features include:

- **Product Display**: A visually appealing interface for showcasing products with details like name price, and description.
- **Cart Management**: Real-time updates to cart items, enabling users to modify quantities or remove products.
- **Responsive Design**: Ensuring a consistent and user-friendly experience across all devices, including mobile, tablet, and desktop.
- **Scroll Animations**: Smooth transitions and animations to enhance user engagement using **AOS**.
- **State Management**: Efficient handling of application data using **Redux** for maintaining cart state and user interactions.

The focus of this system is to improve the user experience for shopping cart functionalities and provide a modern, responsive design for eCommerce platforms.

1.3 Constraints

The scope of this project is limited to **frontend functionalities**, including product display, cart management, and basic UI interactions. Backend integration (e.g., for user authentication and order processing) is beyond the current scope. The application will rely on API endpoints for product data and other dynamic content.

1.4 Risks

Several risks associated with the **shopping cart frontend project** have been identified:

- **Performance Issues**: High loading times or sluggish animations on low-end devices may affect the user experience.
- **Browser Compatibility**: Variations in browser rendering could lead to inconsistent designs or functionality.
- **API Dependency**: The frontend relies on APIs for product data, and any downtime or changes in the API structure could disrupt functionality.
- **State Management Complexity**: Errors in managing the application state (e.g., cart items or user sessions) could lead to a poor user experience.
- **Design Consistency**: Ensuring uniform responsiveness across various devices and screen sizes may pose a challenge.
- **Security Risks**: Although focused on the frontend, vulnerabilities such as XSS (Cross-Site Scripting) could pose risks if not addressed.

1.5 Out of Scope

The following activities and capabilities are **out of scope** for this project:

- **Backend Development**: The project focuses solely on the frontend. Backend functionalities such as user authentication, order processing, and database management are not included.
- **Payment Integration**: Features related to processing payments (e.g., through Stripe or PayPal) are not part of this project.
- **Advanced Analytics**: Dashboards or metrics to track user behavior and cart trends are excluded.
- **Inventory Management**: Managing product stock and availability is considered a backend responsibility.
- **Multi-Language Support**: The application will not include localization or translation features in this phase.

2 Technical specifications

2.1 Technologies Used:

The shopping cart application is built using the following modern web technologies:

- **React**: A JavaScript library for building dynamic user interfaces.
- **Redux**: A state management library to handle global state efficiently.
- **Tailwind CSS**: A utility-first CSS framework for creating responsive and visually appealing designs.
- **AOS (Animate On Scroll)**: A library to add smooth scroll animations, enhancing the overall user experience.
- **JavaScript (ES6+)**: The core programming language for implementing logic and interactivity.
- **HTML5 and CSS3**: The foundation of the frontend structure and design.

2.2 Development Tools:

- **Visual Studio Code**: The primary IDE used for development.
- **Git and GitHub**: Version control tools for tracking changes and collaboration.
- **Node.js and npm**: For managing dependencies and running the development environment.

2.3 Environment Requirements:

- **Operating System**: Windows, macOS, or Linux.
- **Browser**: Latest versions of Chrome, Firefox, Edge, or Safari for testing and usage.
- **Hardware**:
    - A system with at least 4 GB RAM and a modern processor for development.
    - Internet connectivity for dependency installation and API testing.

2.4 Functional Specifications:

- **Product Listing**:
    - Displays a list of products with images, names, prices, and descriptions.
- **Cart Management**:
    - Add, remove, and update product quantities in the cart.
    - Real-time calculation of total price.
- **Responsive Design**:
    - Adjusts layout and functionality for various screen sizes, including mobile, tablet, and desktop.
- **Animations**:
    - Smooth animations for scrolling and interacting with elements using AOS.

2.5 Deployment

The shopping cart frontend application will be deployed using a reliable hosting platform like **Netlify** or **Vercel**, ensuring seamless performance and accessibility.

Key Steps:

1. **Build the Application**: Use npm run build to generate optimized production files.
2. **Deploy to Hosting Platform**: Upload the /build folder to **Netlify**, **Vercel**, or **GitHub Pages**.
3. **Configure Routing**: Ensure single-page application (SPA) fallback is enabled for React.

Optimization:

- Minify JavaScript and CSS files for faster load times.
- Use a CDN for quick asset delivery.

Maintenance:

- Monitor performance using **Google Lighthouse**.
- Use CI/CD pipelines for automated updates.

3 Technology stack

Front End	ReactJs
Backend	---
Database	----
Deployment	Github or Vercel

4 Proposed Solution

The **shopping cart frontend application** is designed to provide users with an intuitive and seamless shopping experience. The solution leverages modern frontend technologies and best practices to achieve the desired functionality and user experience.

Key Components:

1. **Baseline Implementation**:
   - Use **React** and **Redux** for basic cart management, including adding, removing, and updating items.
   - Integrate a product list with dummy data to test core functionalities.
2. **Enhanced Frontend Solution**:
   - **Tailwind CSS**: Enhance the UI with responsive and modern design elements.
   - **AOS (Animate On Scroll)**: Add scroll animations to improve user engagement.
   - **Responsive Design**: Ensure the application works seamlessly across mobile, tablet and desktop devices.

Why Start with a Baseline?

Starting with a simple implementation allows the team to validate the application's core functionality (e.g., cart management and state handling). This baseline will serve as a foundation for enhancing the application with more advanced features and interactions.

Final Solution:

The final version of the application will combine:

- **React** for dynamic rendering.
- **Redux** for managing application state.
- **Tailwind CSS** for UI design.
- **AOS** for user experience improvements through animations.

This approach ensures a scalable, maintainable, and user-friendly shopping cart frontend.

5 Test cases

Test Case	Steps to Perform Test Case	Module	Pass/Fail
Add Item to Cart	1. Navigate to the product list. 2. Click the "Add to Cart" button on a product.	Cart Management	Pass
Remove Item from Cart	1. Navigate to the cart page. 2. Click the "Remove" button for an item in the cart.	Cart Management	Pass
Update Item Quantity	1. Navigate to the cart page. 2. Change the quantity of an item using the quantity selector.	Cart Management	Pass
Check Responsive Design	1. Open the application on different devices (mobile, tablet, desktop). 2. Verify layout consistency.	UI Design	Pass
Scroll Animations	1. Scroll through the product list. 2. Verify animations are triggered smoothly using AOS.	User Experience	Pass
Total Price Calculation	1. Add multiple items to the cart. 2. Verify the total price updates correctly.	Cart Management	Pass
API Data Loading	1. Mock API response for product list. 2. Verify products load correctly on the product list page.	Data Fetching	Pass

6 Key performance indicators (KPI)

· **Page Load Time**: Measure the time it takes for the shopping cart application to load and display content.
· **Cart Interaction Speed**: Evaluate how quickly the cart updates when items are added, removed, or modified.
· **User Engagement**: Track user interactions, such as the number of products viewed or added to the cart.
· **Bounce Rate**: Measure the percentage of users who leave the site without interacting with the cart.
· **Responsiveness**: Ensure the application functions seamlessly across different devices (mobile, tablet, desktop).
· **Animation Smoothness**: Assess the performance of AOS animations during scrolling for smooth transitions.
· **Error-Free Operation**: Monitor the number of errors encountered during cart operations or product data loading.