Fresh FASHION Finds E-commerce Shop

React Application

1. Introduction

The E-commerce Shop React Application is a frontend web application designed for managing an online store.

The project aims to provide users with a simple, efficient interface to browse products, add items to a shopping cart, and proceed to checkout. This project was developed as a part of exploring modern web development using React, Context API for state management, and React Router for navigation.

Objectives:

- To build an e-commerce frontend using React.
- To implement core e-commerce features like product listing, cart management, and checkout.
- To improve skills in state management and reusable components.

2. Architecture

The project is built using a component-based architecture, typical of React applications.

State management for the cart and sidebar is handled using Reacts Context API. The flow of data between components is centralized, making it easier to manage and scale.

Components:

- Product List Page: Displays a list of products with options to add items to the cart.
- Cart: Manages the state of items added to the cart, allows quantity adjustments, and displays the subtotal.
- Sidebar: Shows the items added to the cart with the ability to remove or clear the cart.
- Checkout: (Future enhancement) Handles the final checkout process.

Flow Diagram (Textual):

Product Page --> Sidebar --> Cart --> Checkout --> Order Confirmation.

The sidebar updates cart state and allows removing/updating items.

3. Features

- Product Listing: Displays products available for purchase.
- Add to Cart: Allows users to add products to their cart, update quantities, or remove items.
- Cart Sidebar: Users can view cart details from the sidebar at any point during shopping.
- Cart : Users can also see the whole cart by clicking view cart
- Subtotal Calculation: Automatically updates the subtotal as items are added or removed from the cart.
- Responsive Design: The UI is responsive and adapts to different screen sizes.

4. APIs

The project utilizes the FakeStore API to fetch product data. Below is a summary of the main API endpoints used:

- GET /products - Fetches a list of products

. - GET /products/:id - Fetches detailed information about a specific product.

Example response:

```
{
"id": 1,
 "title": "Product Name",
"price": 29.99,
 "description": "Product description...",
 "image": "https://imageurl.com",
...
}
```

5. Testing

Testing Strategy:

- Manual testing was conducted during development to ensure that the cart functionality works as expected.
- State management was checked by adding/removing items and validating the carts state across different components.

Future improvements could include adding unit and integration tests using Jest or React Testing Library.

6. Challenges and Learnings

Challenges:

- State Management: Handling the cart state globally while maintaining component-level isolation.- Sidebar Synchronization: Ensuring the sidebar and cart page reflect the same cart data required careful handling of state updates.

Learnings:

- Using React Context helped centralize cart-related logic, reducing redundancy.
- Gained deeper understanding of conditional rendering and dynamic state updates within React components.

7. Future Enhancements

- User Authentication: Add login/signup functionality to allow users to save their cart and manage orders.
- . Add a real payment gateway integration for checkout.
- . Implement user authentication and order history
- Unit Testing: Implement automated tests for key components using Jest or other testing frameworks.