EXPERIMENT 3

AIM:
Manage complex state with Redux or Context API

THEORY:

In modern web applications, state management is one of the most critical aspects of building scalable and maintainable systems. State refers to any data that a component or an application needs to remember, such as user information, product details, shopping cart items, or authentication status. As applications grow in size and complexity, managing state across multiple components becomes challenging.

Traditionally, state was managed locally within components using useState or setState. However, this approach becomes inefficient when multiple components need to access or update the same data, leading to a phenomenon called prop drilling. Prop drilling occurs when data has to be passed down through several levels of components, even when only deeply nested components require it. This makes the code harder to maintain, more error-prone, and less scalable.

To overcome these challenges, advanced state management solutions like Context API and Redux are widely used in React applications. Both provide mechanisms to share and update global state across the entire application without unnecessary prop drilling

1.1 The Problem of State Management in React

- **Local State**: Individual components often manage their own state using useState.
- **Prop Drilling Issue**: When a piece of state needs to be accessed by deeply nested components, props must be passed manually through intermediate layers, leading to poor scalability.
- **Complex Interactions**: In applications like E-Commerce, multiple features (cart, wishlist, authentication, filters, payment, etc.) depend on shared global state.

1.2 Context API

The Context API is a built-in feature of React that allows global state to be created and consumed across components. It works by creating a `Context` object, which provides a Provider (to supply data) and a Consumer (to access data). Any component wrapped in the Provider can consume the state without requiring intermediate components to pass it down manually.

In an E-commerce scenario:

- A global context can store the **user authentication status**, **cart items**, **wishlist items**, and **product filters**.

- This ensures that different pages (like product listings, cart page, or checkout page) can access the same data without redundant state management.

- For example, when a product is added to the cart, the cart badge in the navbar can update instantly, since both the navbar and product listing components consume the same context.

The Context API is simple, lightweight, and avoids external dependencies. However, it is most suitable for small to medium-scale applications. In very large applications with highly complex state logic, Context may become difficult to manage.

Advantages:

- No need for third-party libraries.
- Simpler for small to medium applications/
- Perfect for handling **authentication, theme, language, or cart items** in an e-commerce app.

Limitations:

- Less tooling support compared to Redux.
- May lead to performance issues if too many states are kept in a single context.

1.3 Redux

Redux is an external library for managing complex application state in a predictable manner. It is based on the principles of:

Single Source of Truth – The global application state is stored in a central store.

State is Read-Only – The only way to change the state is by dispatching actions.

Changes are Made with Pure Functions – Reducers are pure functions that specify how the state changes in response to actions.

In Redux, the flow is as follows:

- A component dispatches an action (e.g., “ADD_TO_CART”).
- The reducer function processes the action and updates the global state accordingly.
- The store holds the updated state, which can then be accessed by any component via useSelector.

In an E-commerce scenario:

- Redux can manage complex features like multi-step checkout, inventory synchronization, real-time cart updates, and order history management.
- For example, when a user checks out, multiple actions like “APPLY_DISCOUNT,” “UPDATE_STOCK,” and “GENERATE_ORDER” may occur simultaneously. Redux ensures predictable handling of these actions with a well-structured state flow.

Redux is ideal for large-scale applications with dynamic data and multiple interacting states, though it requires more boilerplate code compared to Context API.

Advantages:

- Works well with very large applications.
- Provides debugging tools like Redux DevTools.
- Scales easily as the application grows.
- Middleware (e.g., Redux Thunk, Redux Saga) allows handling asynchronous logic like API calls.

Limitations:

- More boilerplate code compared to Context.
- Steeper learning curve for beginners.

1.4 Choosing Between Redux & Context

- **Context API** → Best for small to medium projects, or when only a few global states are required (e.g., user login, cart, theme).
- **Redux** → Best for large applications with complex, frequently updated global states and asynchronous operations (e.g., fetching product lists, payment processing, multi-user orders).

COMPARISION

Feature	Context API	Redux
Setup Complexity	Easy	Medium to High
External Dependency	No	Yes
Performance for Large Apps	Moderate	High
Best Use Case	Small–Medium Apps	Large-Scale Apps
Debugging Tools	Basic	Excellent (Redux DevTools)

APPLICATION IN E-COMMERCE

In the E-commerce application experiment, the goal is to demonstrate how both Context API and Redux can be applied to manage complex state.

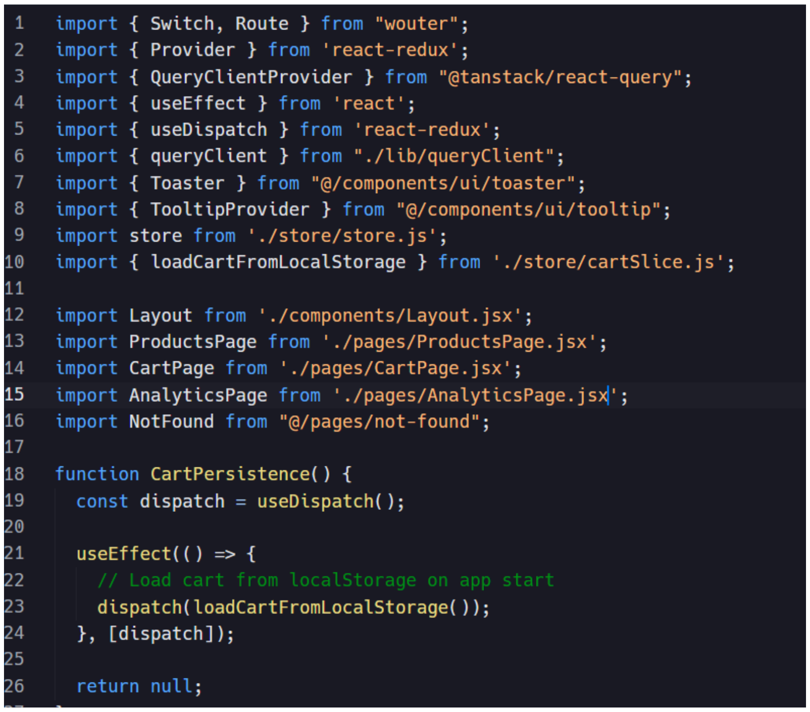
- With Context API, the experiment can showcase a simplified cart and wishlist management system.

- With Redux, the experiment can extend this into handling checkout processes, order management, and more advanced interactions.

Both approaches highlight how efficient state management prevents data inconsistency, improves user experience, and ensures scalability as the project grows.

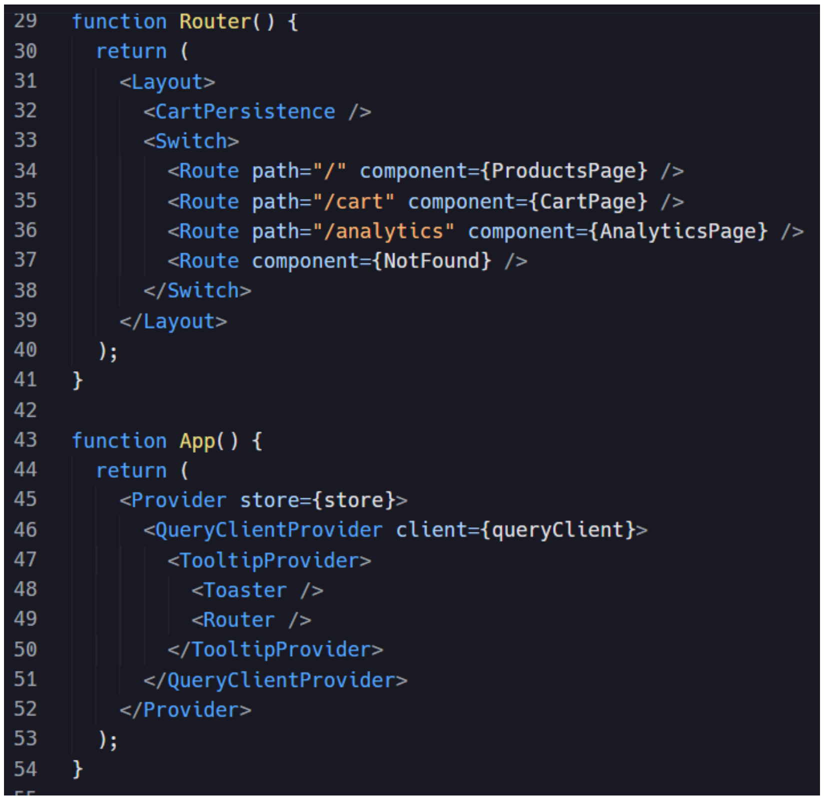
CODE

App.jsx



```jsx
import { Switch, Route } from "wouter";
import { Provider } from 'react-redux';
import { QueryClientProvider } from "@tanstack/react-query";
import { useEffect } from 'react';
import { useDispatch } from 'react-redux';
import { queryClient } from "./lib/queryClient";
import { Toaster } from "@/components/ui/toaster";
import { TooltipProvider } from "@/components/ui/tooltip";
import store from './store/store.js';
import { loadCartFromLocalStorage } from './store/cartSlice.js';

import Layout from './components/Layout.jsx';
import ProductsPage from './pages/ProductsPage.jsx';
import CartPage from './pages/CartPage.jsx';
import AnalyticsPage from './pages/AnalyticsPage.jsx';
import NotFound from "@/pages/not-found";

function CartPersistence() {
  const dispatch = useDispatch();

  useEffect(() => {
    // Load cart from localStorage on app start
    dispatch(loadCartFromLocalStorage());
  }, [dispatch]);

  return null;
```

Code 1.1

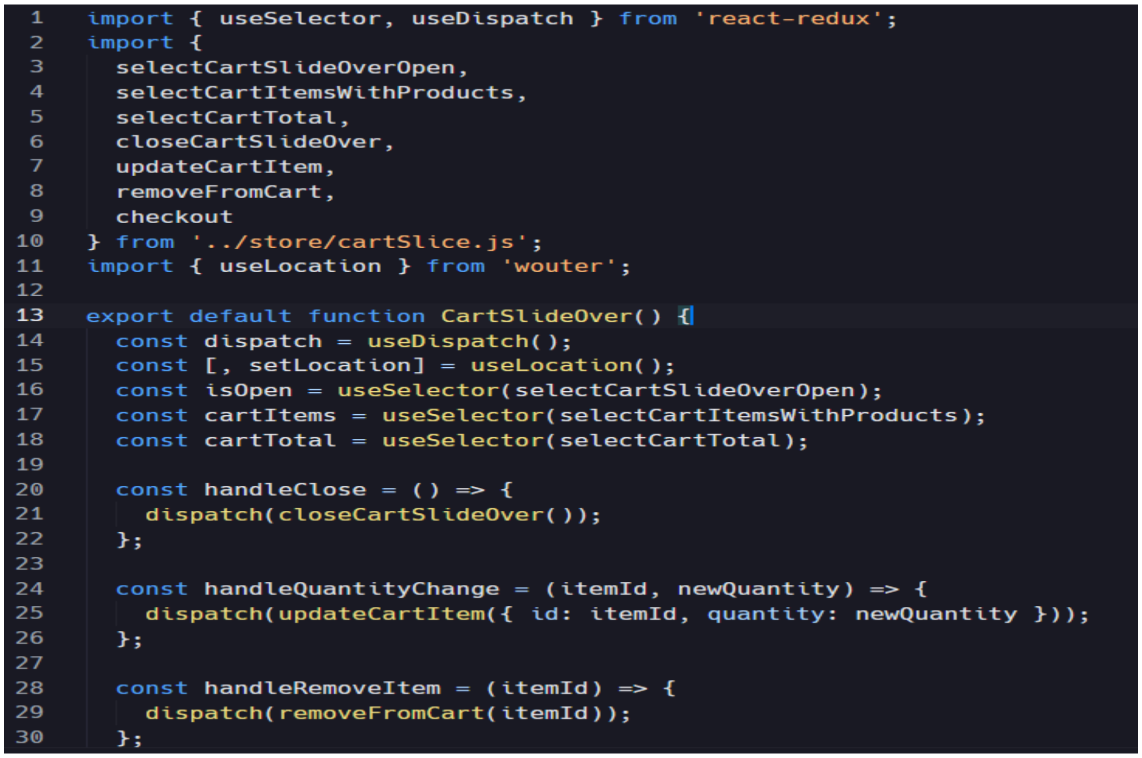


```jsx
function Router() {
  return (
    <Layout>
      <CartPersistence />
      <Switch>
        <Route path="/" component={ProductsPage} />
        <Route path="/cart" component={CartPage} />
        <Route path="/analytics" component={AnalyticsPage} />
        <Route component={NotFound} />
      </Switch>
    </Layout>
  );
}

function App() {
  return (
    <Provider store={store}>
      <QueryClientProvider client={queryClient}>
        <TooltipProvider>
          <Toaster />
          <Router />
        </TooltipProvider>
      </QueryClientProvider>
    </Provider>
  );
}
```

Code 1.2

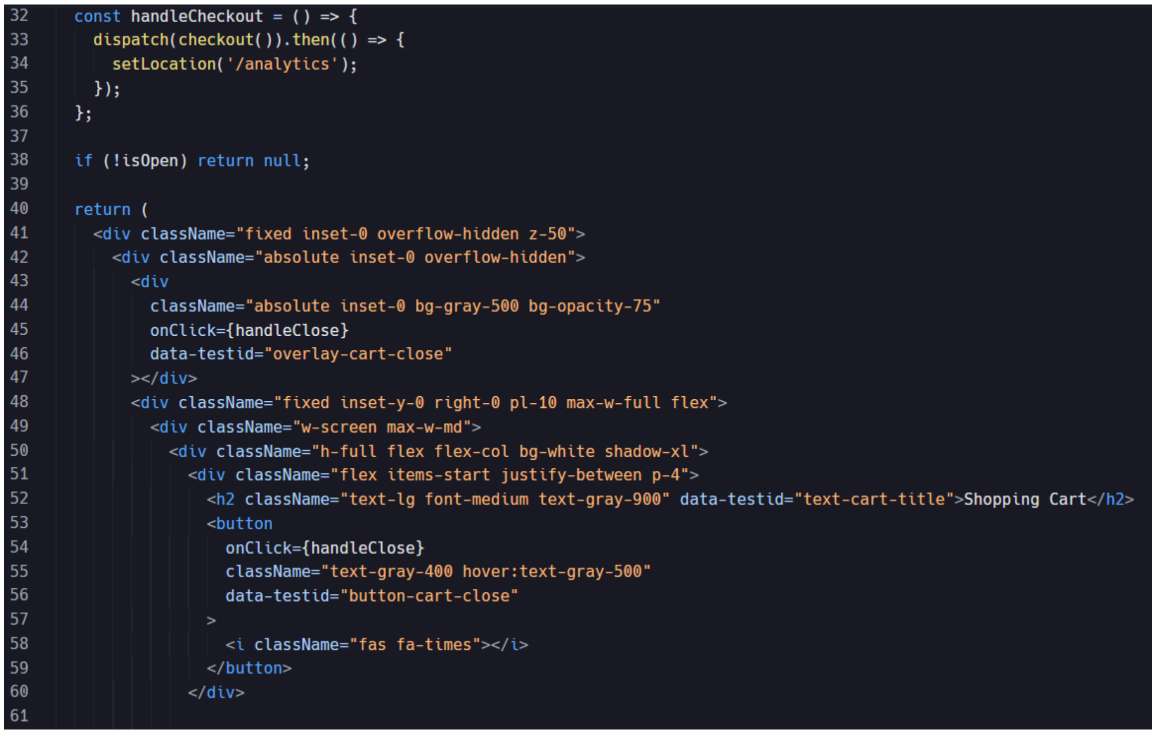
Components

Cart.jsx



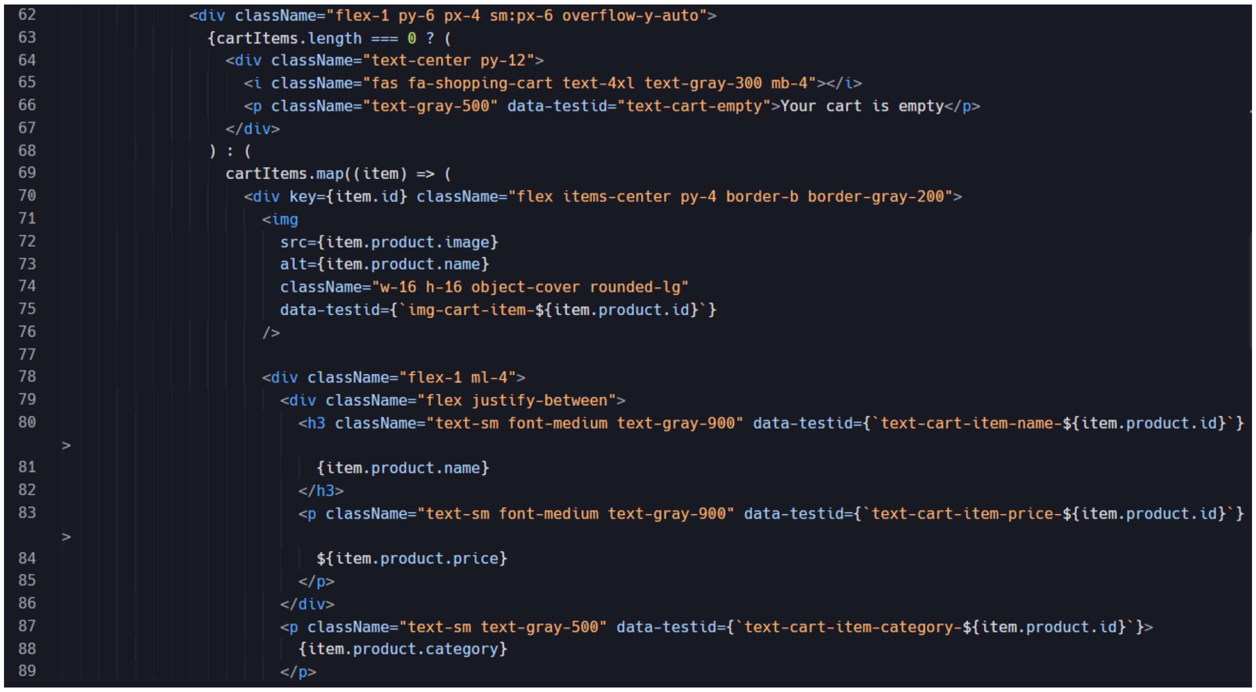
```jsx
import { useSelector, useDispatch } from 'react-redux';
import {
  selectCartSlideOverOpen,
  selectCartItemsWithProducts,
  selectCartTotal,
  closeCartSlideOver,
  updateCartItem,
  removeFromCart,
  checkout
} from '../store/cartSlice.js';
import { useLocation } from 'wouter';

export default function CartSlideOver() {
  const dispatch = useDispatch();
  const [, setLocation] = useLocation();
  const isOpen = useSelector(selectCartSlideOverOpen);
  const cartItems = useSelector(selectCartItemsWithProducts);
  const cartTotal = useSelector(selectCartTotal);

  const handleClose = () => {
    dispatch(closeCartSlideOver());
  };

  const handleQuantityChange = (itemId, newQuantity) => {
    dispatch(updateCartItem({ id: itemId, quantity: newQuantity }));
  };

  const handleRemoveItem = (itemId) => {
    dispatch(removeFromCart(itemId));
  };
```

Code 2.1



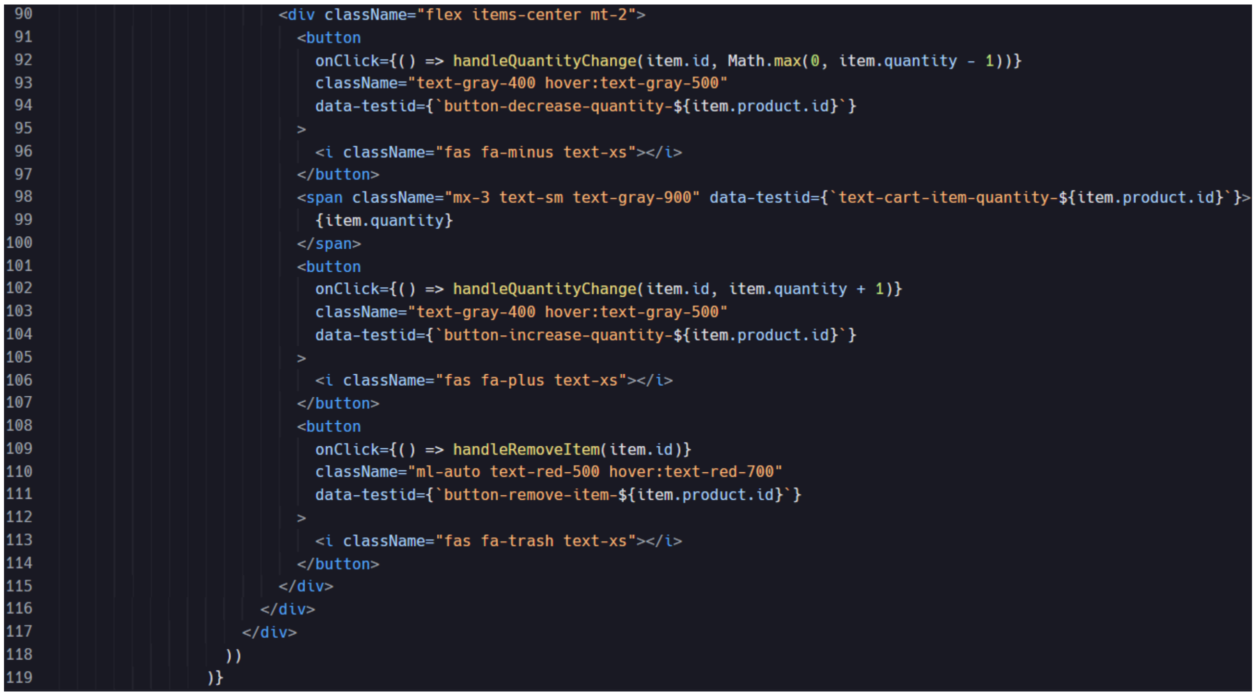
```jsx
  const handleCheckout = () => {
    dispatch(checkout()).then(() => {
      setLocation('/analytics');
    });
  };

  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 overflow-hidden z-50">
      <div className="absolute inset-0 overflow-hidden">
        <div
          className="absolute inset-0 bg-gray-500 bg-opacity-75"
          onClick={handleClose}
          data-testid="overlay-cart-close"
        ></div>
        <div className="fixed inset-y-0 right-0 pl-10 max-w-full flex">
          <div className="w-screen max-w-md">
            <div className="h-full flex flex-col bg-white shadow-xl">
              <div className="flex items-start justify-between p-4">
                <h2 className="text-lg font-medium text-gray-900" data-testid="text-cart-title">Shopping Cart</h2>
                <button
                  onClick={handleClose}
                  className="text-gray-400 hover:text-gray-500"
                  data-testid="button-cart-close"
                >
                  <i className="fas fa-times"></i>
                </button>
              </div>

```

Code 2.2



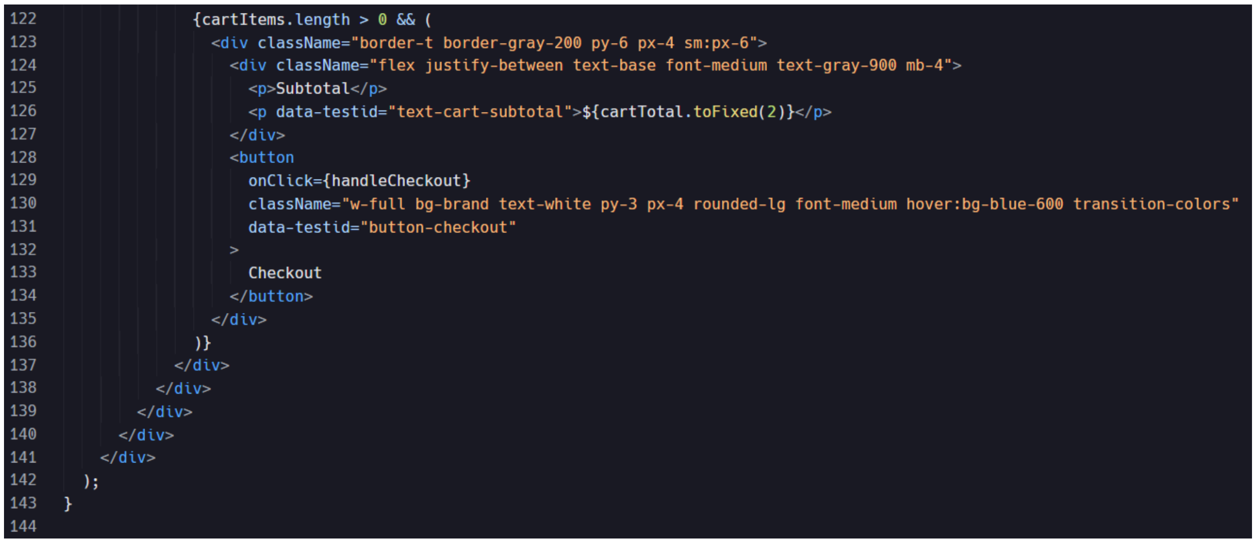
```jsx
<div className="flex-1 py-6 px-4 sm:px-6 overflow-y-auto">
  {cartItems.length === 0 ? (
    <div className="text-center py-12">
      <i className="fas fa-shopping-cart text-4xl text-gray-300 mb-4"></i>
      <p className="text-gray-500" data-testid="text-cart-empty">Your cart is empty</p>
    </div>
  ) : (
    cartItems.map((item) => (
      <div key={item.id} className="flex items-center py-4 border-b border-gray-200">
        <img
          src={item.product.image}
          alt={item.product.name}
          className="w-16 h-16 object-cover rounded-lg"
          data-testid={`img-cart-item-${item.product.id}`}
        />

        <div className="flex-1 ml-4">
          <div className="flex justify-between">
            <h3 className="text-sm font-medium text-gray-900" data-testid={`text-cart-item-name-${item.product.id}`}>
              {item.product.name}
            </h3>
            <p className="text-sm font-medium text-gray-900" data-testid={`text-cart-item-price-${item.product.id}`}>
              ${item.product.price}
            </p>
          </div>
          <p className="text-sm text-gray-500" data-testid={`text-cart-item-category-${item.product.id}`}>
            {item.product.category}
          </p>
```

Code 2.3



```jsx
          <div className="flex items-center mt-2">
            <button
              onClick={() => handleQuantityChange(item.id, Math.max(0, item.quantity - 1))}
              className="text-gray-400 hover:text-gray-500"
              data-testid={`button-decrease-quantity-${item.product.id}`}
            >
              <i className="fas fa-minus text-xs"></i>
            </button>
            <span className="mx-3 text-sm text-gray-900" data-testid={`text-cart-item-quantity-${item.product.id}`}>
              {item.quantity}
            </span>
            <button
              onClick={() => handleQuantityChange(item.id, item.quantity + 1)}
              className="text-gray-400 hover:text-gray-500"
              data-testid={`button-increase-quantity-${item.product.id}`}
            >
              <i className="fas fa-plus text-xs"></i>
            </button>
            <button
              onClick={() => handleRemoveItem(item.id)}
              className="ml-auto text-red-500 hover:text-red-700"
              data-testid={`button-remove-item-${item.product.id}`}
            >
              <i className="fas fa-trash text-xs"></i>
            </button>
          </div>
        </div>
      </div>
    ))
  )}
```

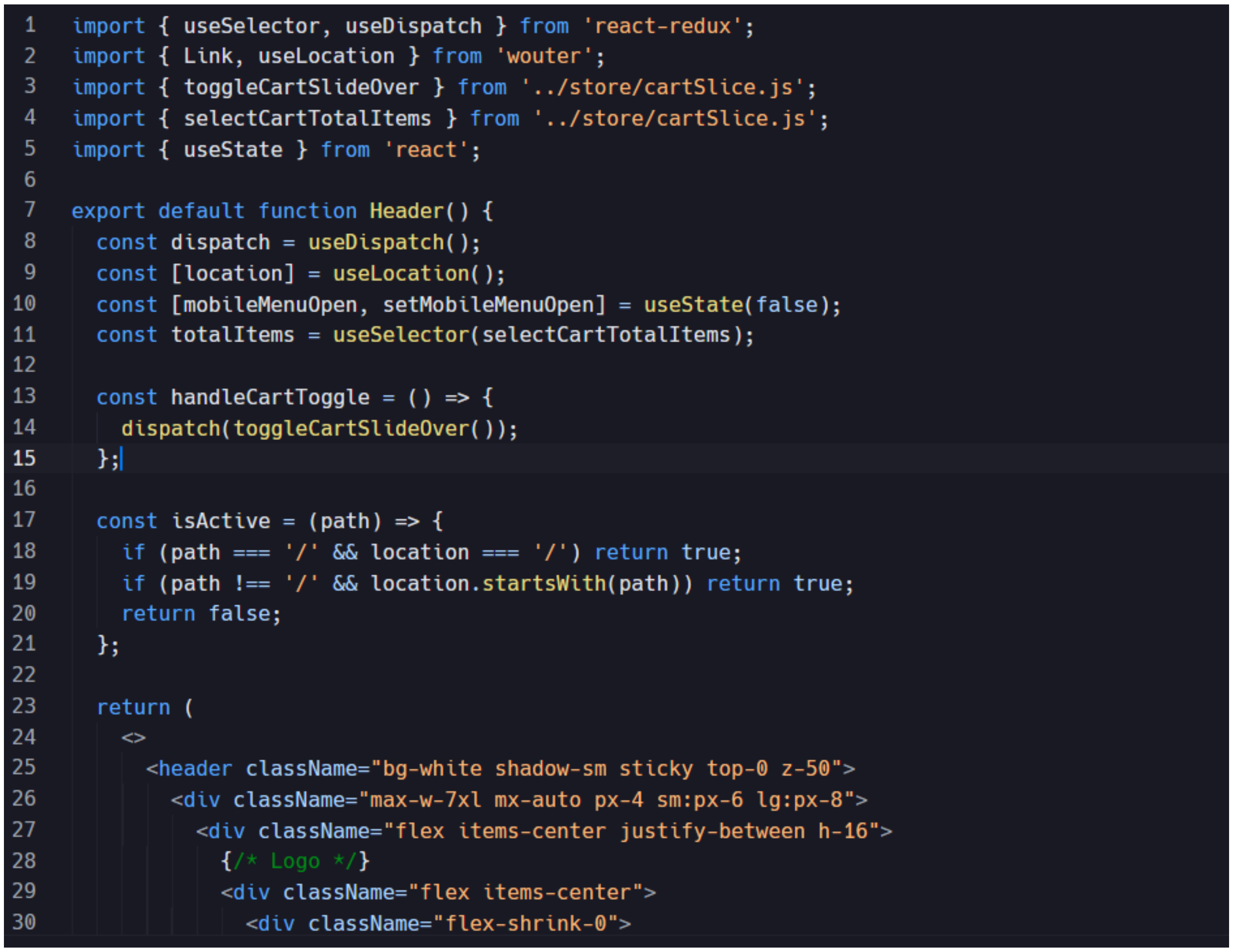
Code 2.4



```jsx
              {cartItems.length > 0 && (
                <div className="border-t border-gray-200 py-6 px-4 sm:px-6">
                  <div className="flex justify-between text-base font-medium text-gray-900 mb-4">
                    <p>Subtotal</p>
                    <p data-testid="text-cart-subtotal">${cartTotal.toFixed(2)}</p>
                  </div>
                  <button
                    onClick={handleCheckout}
                    className="w-full bg-brand text-white py-3 px-4 rounded-lg font-medium hover:bg-blue-600 transition-colors"
                    data-testid="button-checkout"
                  >
                    Checkout
                  </button>
                </div>
              )}
            </div>
          </div>
        </div>
      </div>
    </div>
  );
}

```

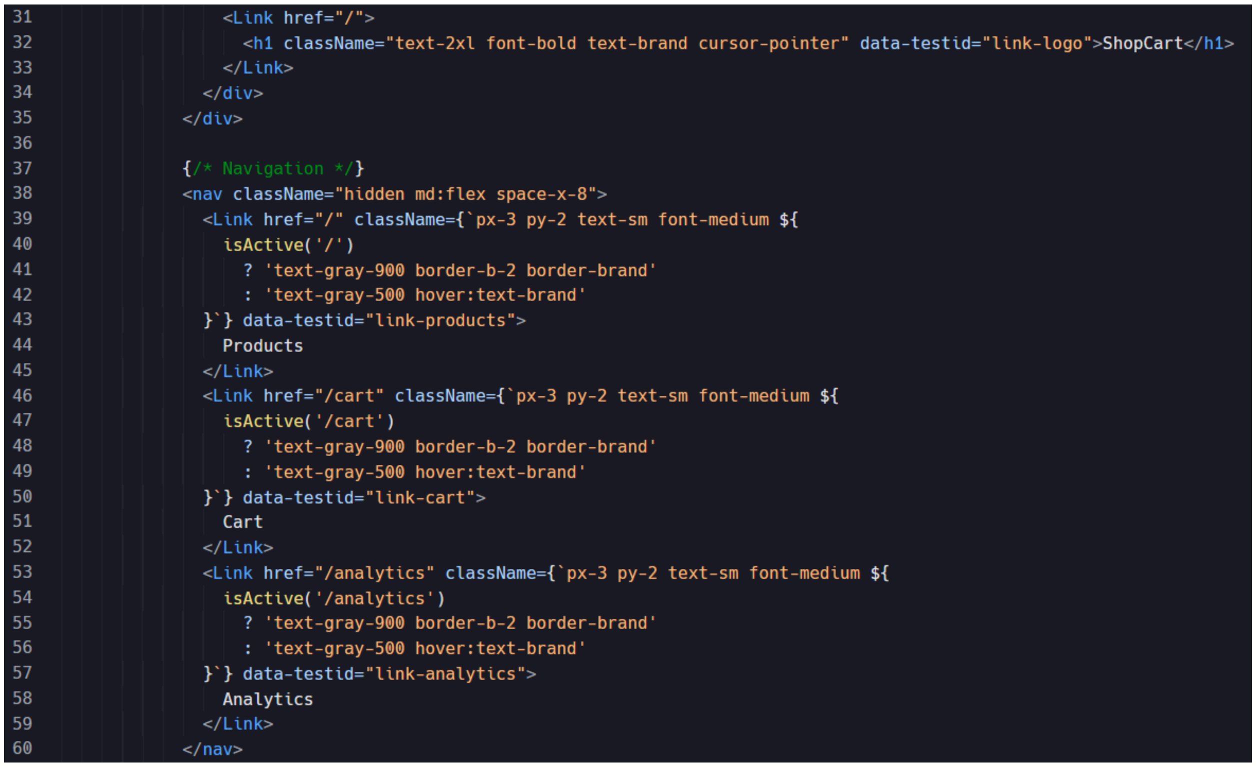
Code 2.5

Header.jsx



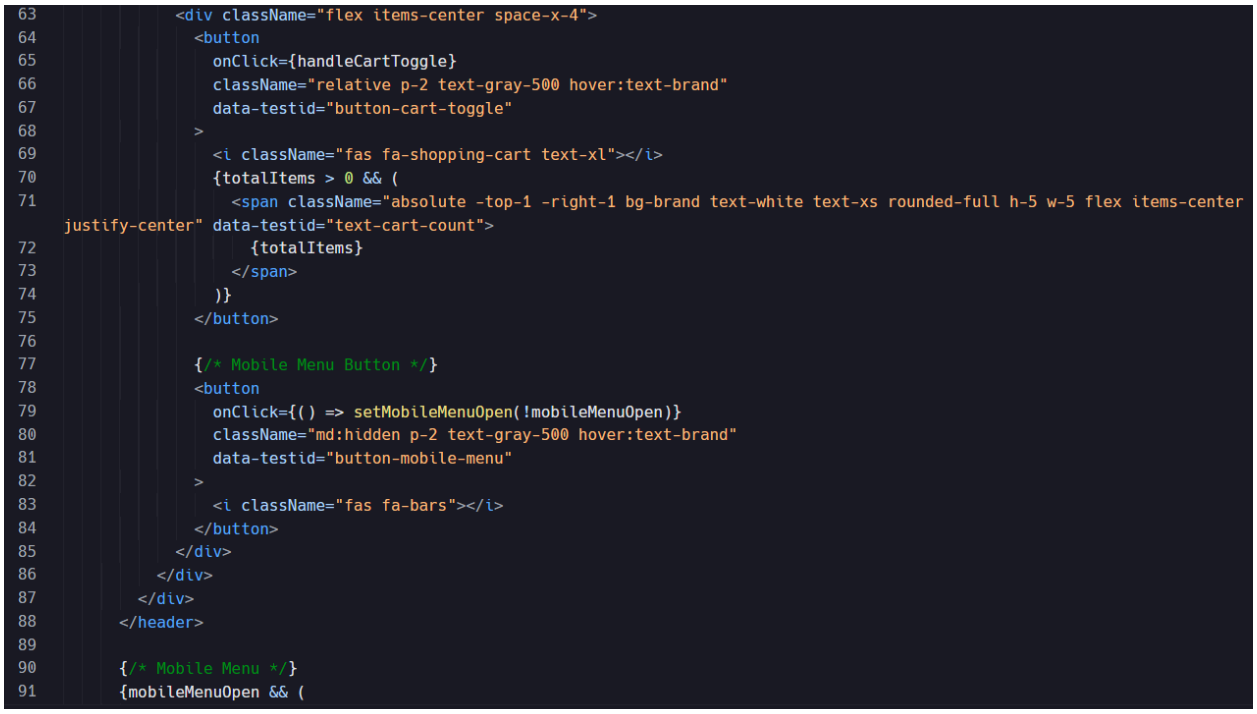
```jsx
import { useSelector, useDispatch } from 'react-redux';
import { Link, useLocation } from 'wouter';
import { toggleCartSlideOver } from '../store/cartSlice.js';
import { selectCartTotalItems } from '../store/cartSlice.js';
import { useState } from 'react';

export default function Header() {
  const dispatch = useDispatch();
  const [location] = useLocation();
  const [mobileMenuOpen, setMobileMenuOpen] = useState(false);
  const totalItems = useSelector(selectCartTotalItems);

  const handleCartToggle = () => {
    dispatch(toggleCartSlideOver());
  };

  const isActive = (path) => {
    if (path === '/' && location === '/') return true;
    if (path !== '/' && location.startsWith(path)) return true;
    return false;
  };

  return (
    <>
      <header className="bg-white shadow-sm sticky top-0 z-50">
        <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
          <div className="flex items-center justify-between h-16">
            {/* Logo */}
            <div className="flex items-center">
              <div className="flex-shrink-0">
```

Code 3.1



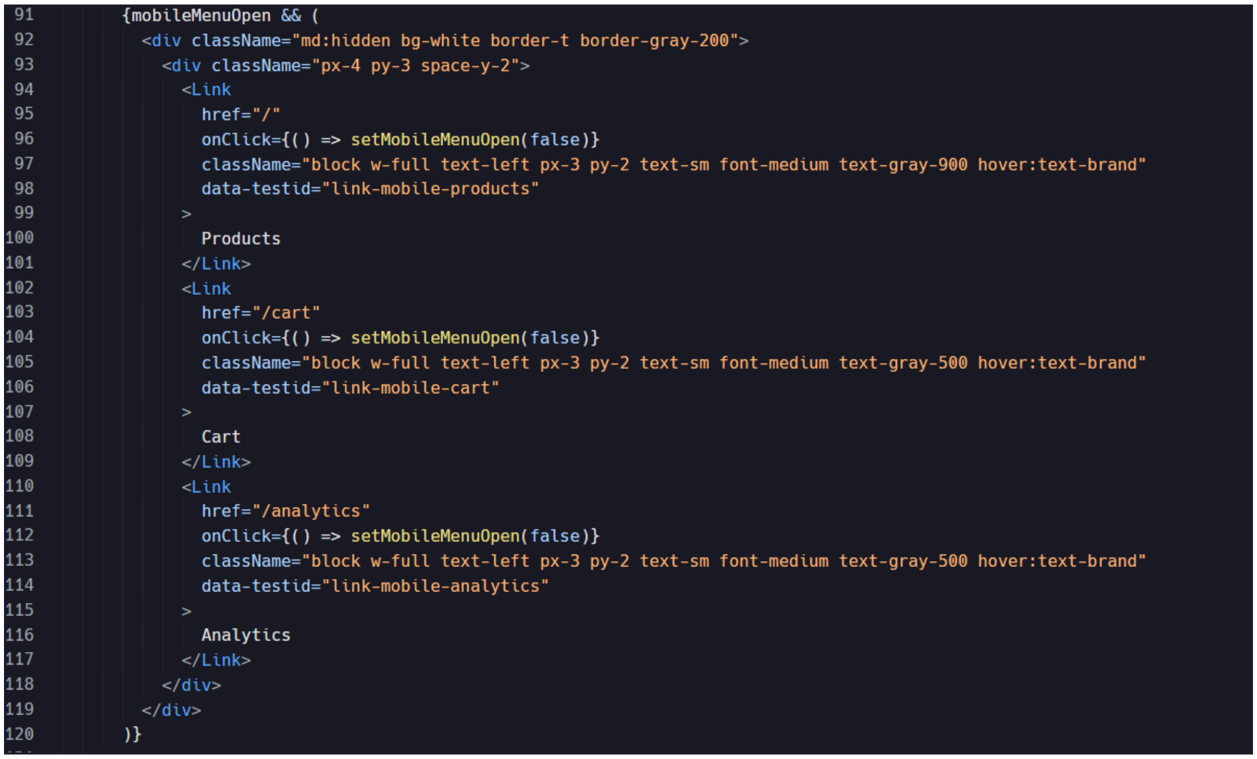
```jsx
            <Link href="/">
              <h1 className="text-2xl font-bold text-brand cursor-pointer" data-testid="link-logo">ShopCart</h1>
            </Link>
          </div>
        </div>

        {/* Navigation */}
        <nav className="hidden md:flex space-x-8">
          <Link href="/" className={`px-3 py-2 text-sm font-medium ${
            isActive('/')
              ? 'text-gray-900 border-b-2 border-brand'
              : 'text-gray-500 hover:text-brand'
          }`} data-testid="link-products">
            Products
          </Link>
          <Link href="/cart" className={`px-3 py-2 text-sm font-medium ${
            isActive('/cart')
              ? 'text-gray-900 border-b-2 border-brand'
              : 'text-gray-500 hover:text-brand'
          }`} data-testid="link-cart">
            Cart
          </Link>
          <Link href="/analytics" className={`px-3 py-2 text-sm font-medium ${
            isActive('/analytics')
              ? 'text-gray-900 border-b-2 border-brand'
              : 'text-gray-500 hover:text-brand'
          }`} data-testid="link-analytics">
            Analytics
          </Link>
        </nav>
```

Code 3.2



```jsx
            <div className="flex items-center space-x-4">
              <button
                onClick={handleCartToggle}
                className="relative p-2 text-gray-500 hover:text-brand"
                data-testid="button-cart-toggle"
              >
                <i className="fas fa-shopping-cart text-xl"></i>
                {totalItems > 0 && (
                  <span className="absolute -top-1 -right-1 bg-brand text-white text-xs rounded-full h-5 w-5 flex items-center justify-center" data-testid="text-cart-count">
                    {totalItems}
                  </span>
                )}
              </button>

              {/* Mobile Menu Button */}
              <button
                onClick={() => setMobileMenuOpen(!mobileMenuOpen)}
                className="md:hidden p-2 text-gray-500 hover:text-brand"
                data-testid="button-mobile-menu"
              >
                <i className="fas fa-bars"></i>
              </button>
            </div>
          </div>
        </div>
      </header>

      {/* Mobile Menu */}
      {mobileMenuOpen && (
```

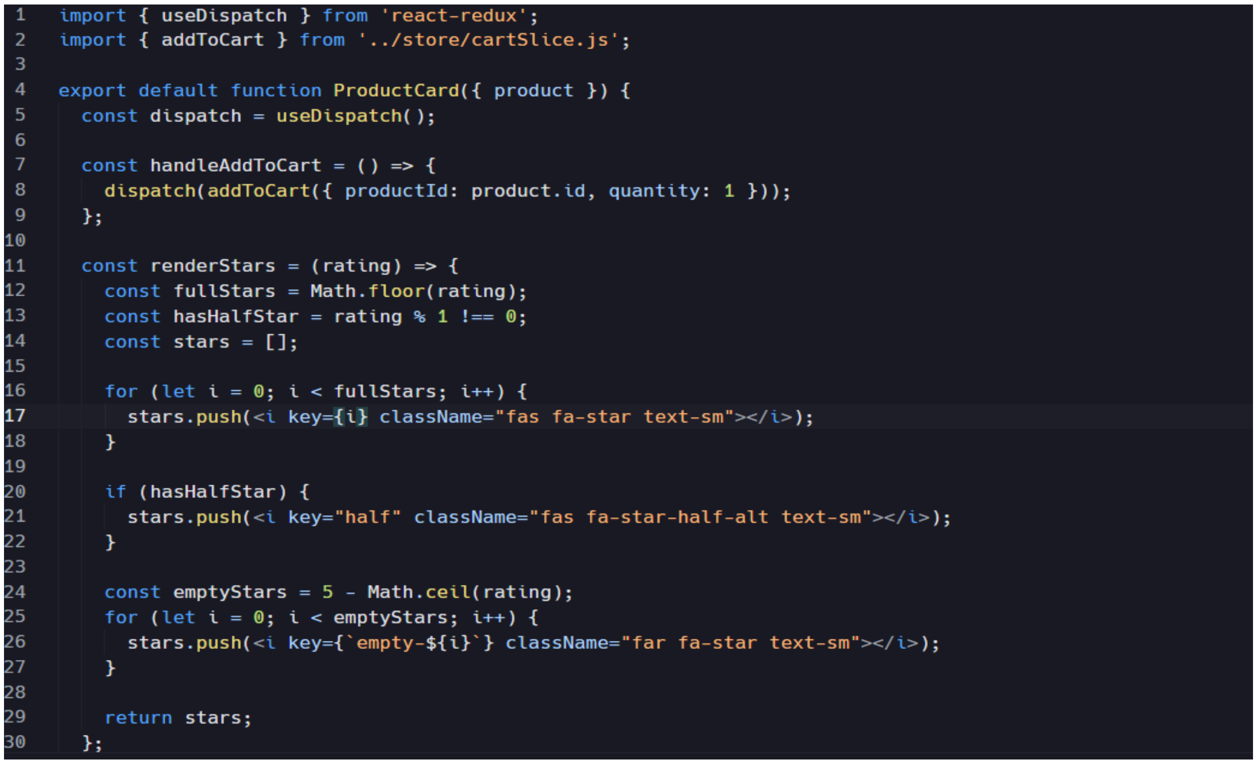
Code 3.3



```jsx
      {mobileMenuOpen && (
        <div className="md:hidden bg-white border-t border-gray-200">
          <div className="px-4 py-3 space-y-2">
            <Link
              href="/"
              onClick={() => setMobileMenuOpen(false)}
              className="block w-full text-left px-3 py-2 text-sm font-medium text-gray-900 hover:text-brand"
              data-testid="link-mobile-products"
            >
              Products
            </Link>
            <Link
              href="/cart"
              onClick={() => setMobileMenuOpen(false)}
              className="block w-full text-left px-3 py-2 text-sm font-medium text-gray-500 hover:text-brand"
              data-testid="link-mobile-cart"
            >
              Cart
            </Link>
            <Link
              href="/analytics"
              onClick={() => setMobileMenuOpen(false)}
              className="block w-full text-left px-3 py-2 text-sm font-medium text-gray-500 hover:text-brand"
              data-testid="link-mobile-analytics"
            >
              Analytics
            </Link>
          </div>
        </div>
      )}
```

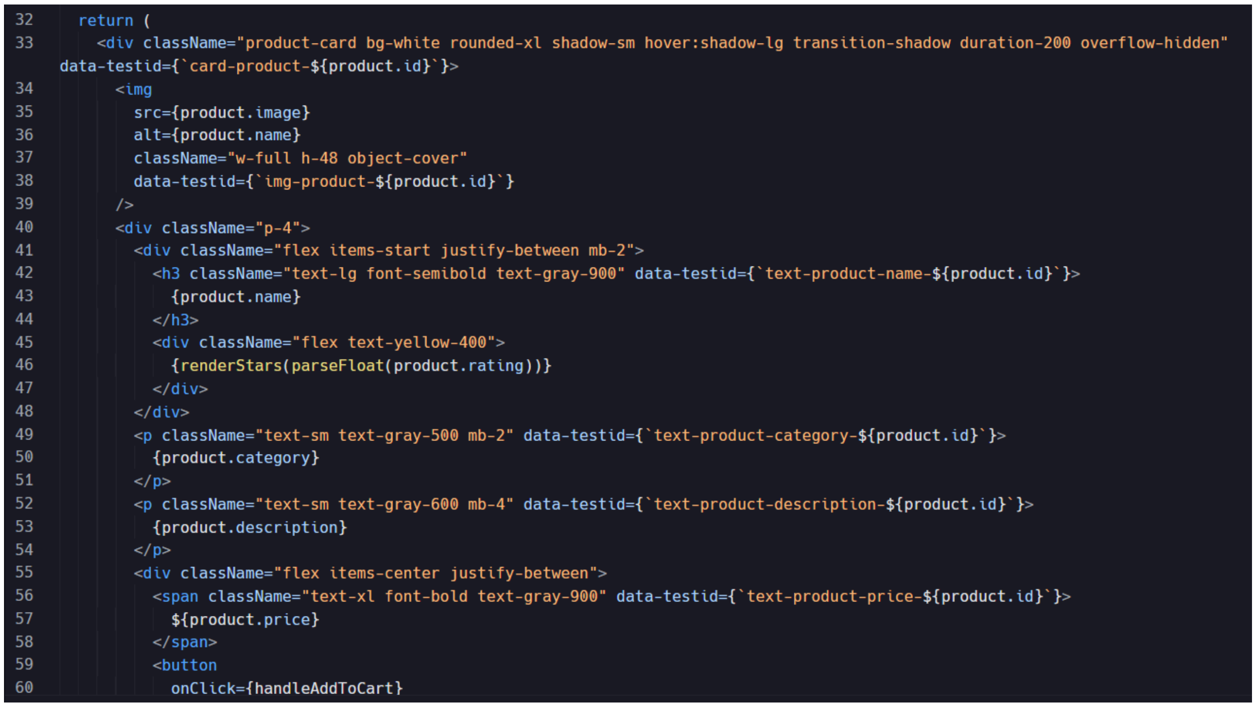
Code 3.4

ProductCard.jsx



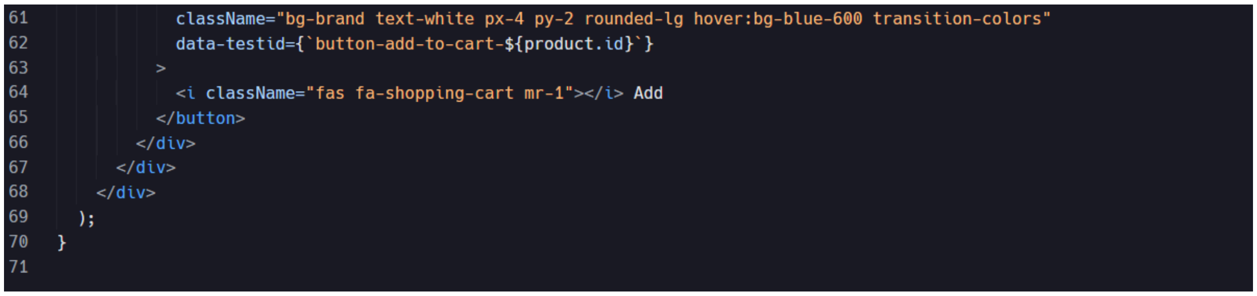
```jsx
import { useDispatch } from 'react-redux';
import { addToCart } from '../store/cartSlice.js';

export default function ProductCard({ product }) {
  const dispatch = useDispatch();

  const handleAddToCart = () => {
    dispatch(addToCart({ productId: product.id, quantity: 1 }));
  };

  const renderStars = (rating) => {
    const fullStars = Math.floor(rating);
    const hasHalfStar = rating % 1 !== 0;
    const stars = [];

    for (let i = 0; i < fullStars; i++) {
      stars.push(<i key={i} className="fas fa-star text-sm"></i>);
    }

    if (hasHalfStar) {
      stars.push(<i key="half" className="fas fa-star-half-alt text-sm"></i>);
    }

    const emptyStars = 5 - Math.ceil(rating);
    for (let i = 0; i < emptyStars; i++) {
      stars.push(<i key={`empty-${i}`} className="far fa-star text-sm"></i>);
    }

    return stars;
  };
```

Code 4.1



```jsx
  return (
    <div className="product-card bg-white rounded-xl shadow-sm hover:shadow-lg transition-shadow duration-200 overflow-hidden" data-testid={`card-product-${product.id}`}>
      <img
        src={product.image}
        alt={product.name}
        className="w-full h-48 object-cover"
        data-testid={`img-product-${product.id}`}
      />
      <div className="p-4">
        <div className="flex items-start justify-between mb-2">
          <h3 className="text-lg font-semibold text-gray-900" data-testid={`text-product-name-${product.id}`}>
            {product.name}
          </h3>
          <div className="flex text-yellow-400">
            {renderStars(parseFloat(product.rating))}
          </div>
        </div>
        <p className="text-sm text-gray-500 mb-2" data-testid={`text-product-category-${product.id}`}>
          {product.category}
        </p>
        <p className="text-sm text-gray-600 mb-4" data-testid={`text-product-description-${product.id}`}>
          {product.description}
        </p>
        <div className="flex items-center justify-between">
          <span className="text-xl font-bold text-gray-900" data-testid={`text-product-price-${product.id}`}>
            ${product.price}
          </span>
          <button
            onClick={handleAddToCart}
```

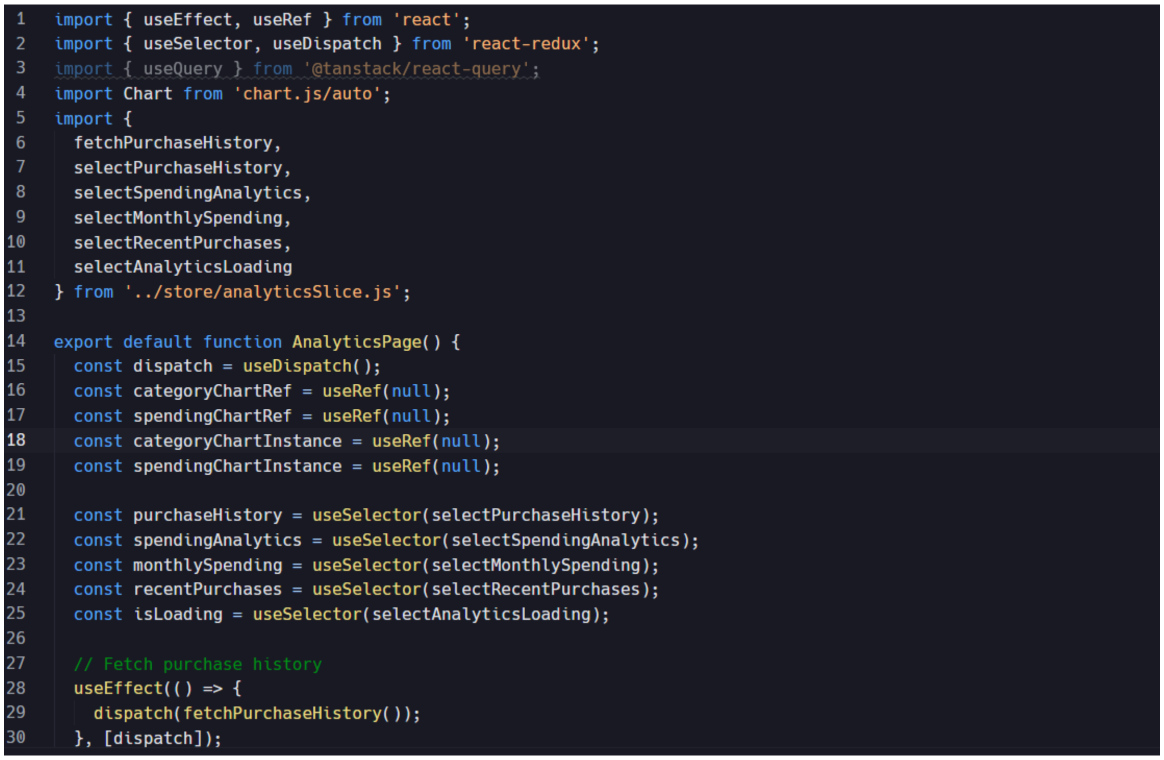
Code 4.2



```jsx
            className="bg-brand text-white px-4 py-2 rounded-lg hover:bg-blue-600 transition-colors"
            data-testid={`button-add-to-cart-${product.id}`}
          >
            <i className="fas fa-shopping-cart mr-1"></i> Add
          </button>
        </div>
      </div>
    </div>
  );
}
```

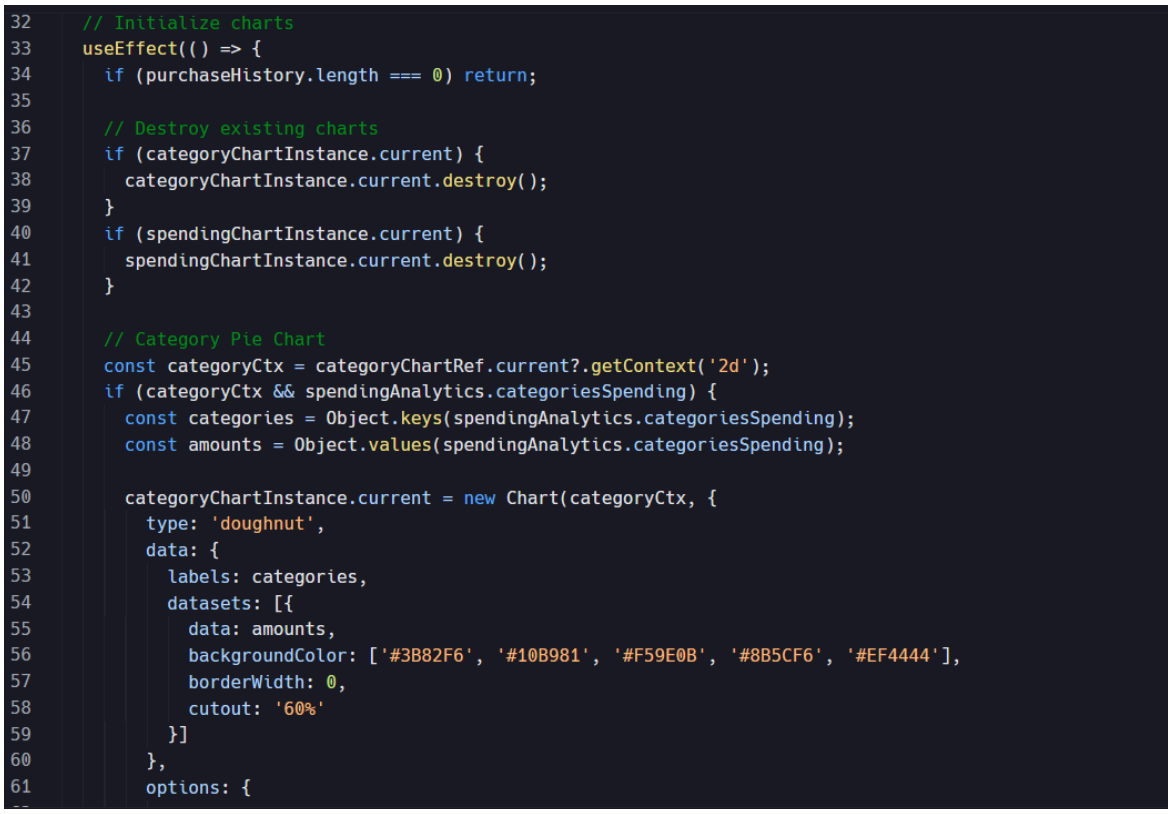
Code 4.3

AnalyticsPage.jsx



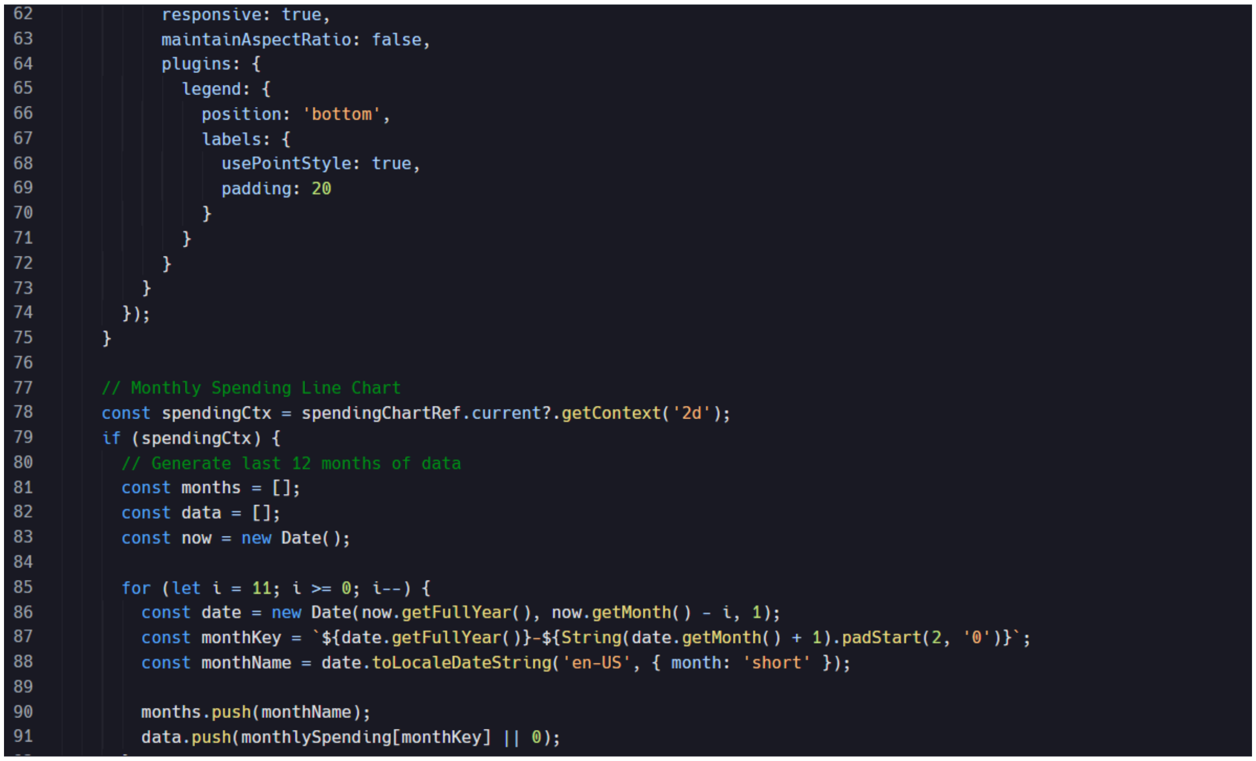
```jsx
import { useEffect, useRef } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { useQuery } from '@tanstack/react-query';
import Chart from 'chart.js/auto';
import {
  fetchPurchaseHistory,
  selectPurchaseHistory,
  selectSpendingAnalytics,
  selectMonthlySpending,
  selectRecentPurchases,
  selectAnalyticsLoading
} from '../store/analyticsSlice.js';

export default function AnalyticsPage() {
  const dispatch = useDispatch();
  const categoryChartRef = useRef(null);
  const spendingChartRef = useRef(null);
  const categoryChartInstance = useRef(null);
  const spendingChartInstance = useRef(null);

  const purchaseHistory = useSelector(selectPurchaseHistory);
  const spendingAnalytics = useSelector(selectSpendingAnalytics);
  const monthlySpending = useSelector(selectMonthlySpending);
  const recentPurchases = useSelector(selectRecentPurchases);
  const isLoading = useSelector(selectAnalyticsLoading);

  // Fetch purchase history
  useEffect(() => {
    dispatch(fetchPurchaseHistory());
  }, [dispatch]);
```

Code 5.1



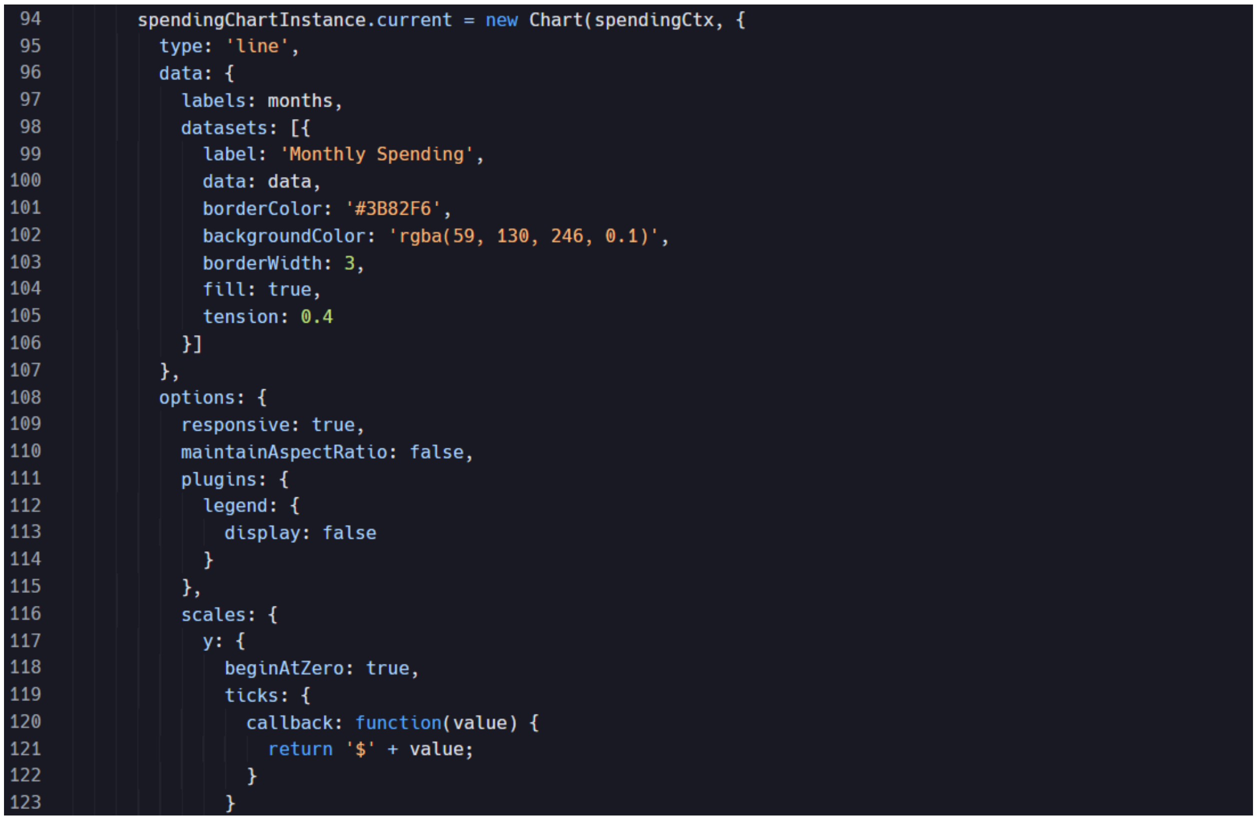
```jsx
  // Initialize charts
  useEffect(() => {
    if (purchaseHistory.length === 0) return;

    // Destroy existing charts
    if (categoryChartInstance.current) {
      categoryChartInstance.current.destroy();
    }
    if (spendingChartInstance.current) {
      spendingChartInstance.current.destroy();
    }

    // Category Pie Chart
    const categoryCtx = categoryChartRef.current?.getContext('2d');
    if (categoryCtx && spendingAnalytics.categoriesSpending) {
      const categories = Object.keys(spendingAnalytics.categoriesSpending);
      const amounts = Object.values(spendingAnalytics.categoriesSpending);

      categoryChartInstance.current = new Chart(categoryCtx, {
        type: 'doughnut',
        data: {
          labels: categories,
          datasets: [{
            data: amounts,
            backgroundColor: ['#3B82F6', '#10B981', '#F59E0B', '#8B5CF6', '#EF4444'],
            borderWidth: 0,
            cutout: '60%'
          }]
        },
        options: {
```

Code 5.2



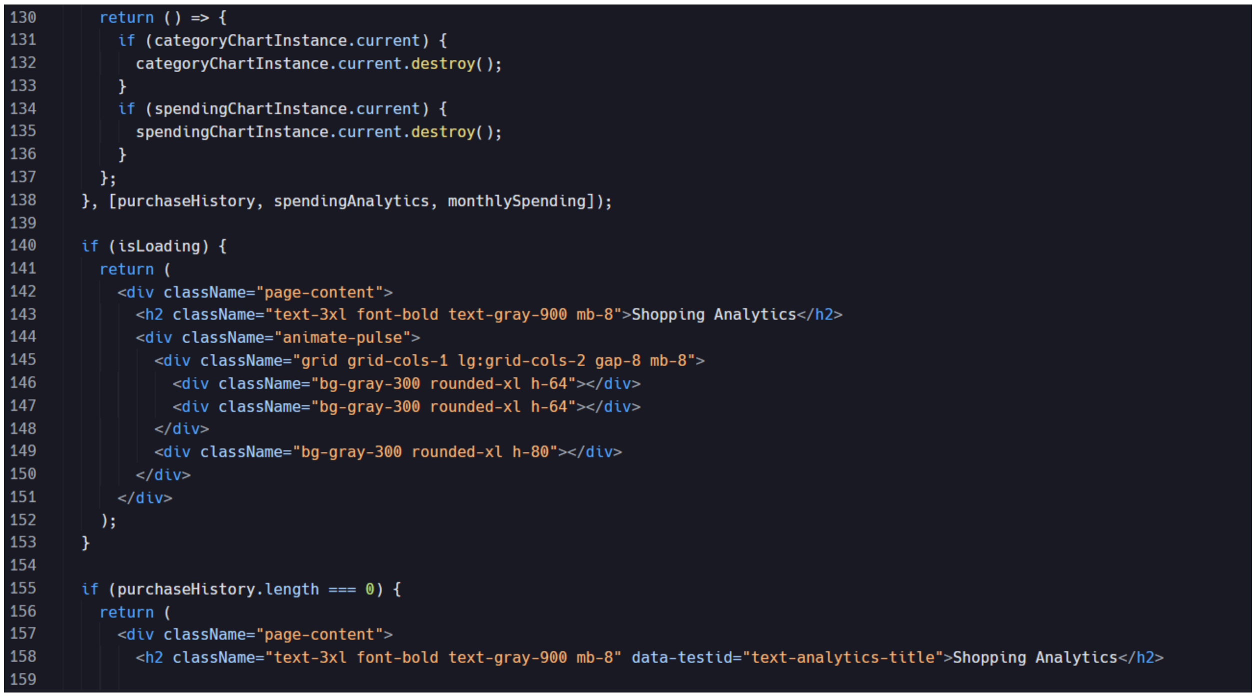
```
          responsive: true,
          maintainAspectRatio: false,
          plugins: {
            legend: {
              position: 'bottom',
              labels: {
                usePointStyle: true,
                padding: 20
              }
            }
          }
        }
      });
    }

    // Monthly Spending Line Chart
    const spendingCtx = spendingChartRef.current?.getContext('2d');
    if (spendingCtx) {
      // Generate last 12 months of data
      const months = [];
      const data = [];
      const now = new Date();

      for (let i = 11; i >= 0; i--) {
        const date = new Date(now.getFullYear(), now.getMonth() - i, 1);
        const monthKey = `${date.getFullYear()}-${String(date.getMonth() + 1).padStart(2, '0')}`;
        const monthName = date.toLocaleDateString('en-US', { month: 'short' });

        months.push(monthName);
        data.push(monthlySpending[monthKey] || 0);
```

Code 5.3



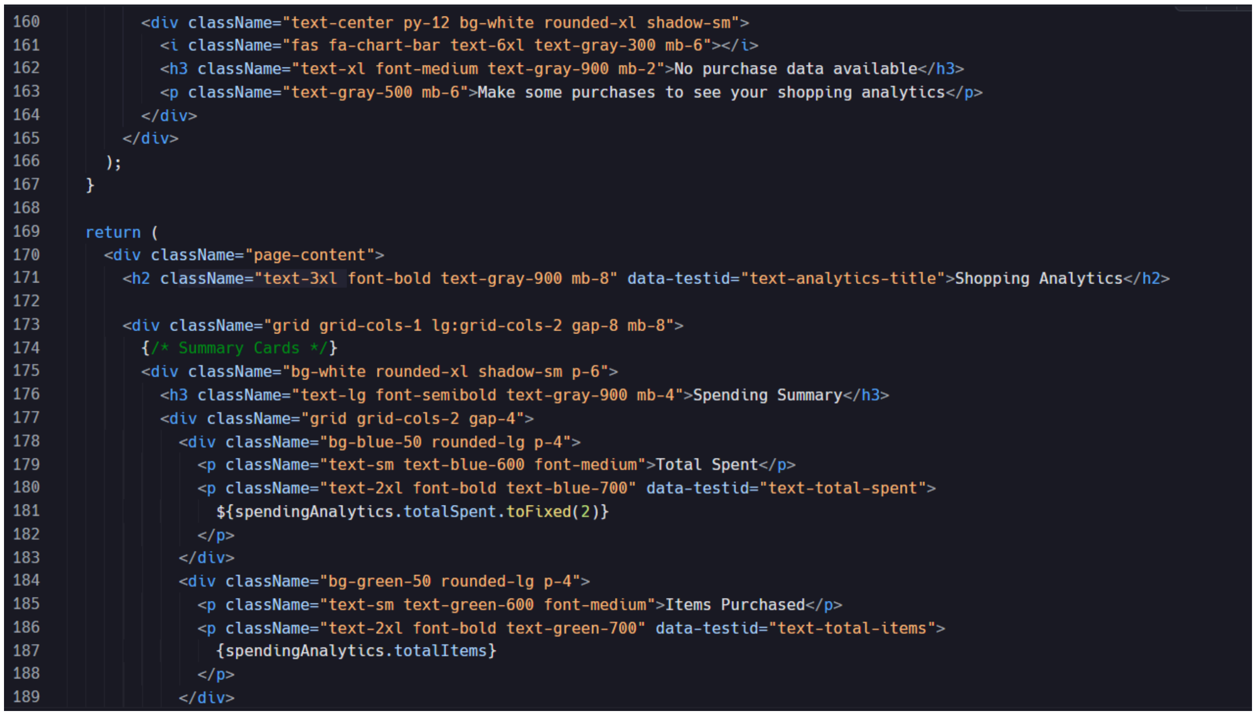
```
      spendingChartInstance.current = new Chart(spendingCtx, {
        type: 'line',
        data: {
          labels: months,
          datasets: [{
            label: 'Monthly Spending',
            data: data,
            borderColor: '#3B82F6',
            backgroundColor: 'rgba(59, 130, 246, 0.1)',
            borderWidth: 3,
            fill: true,
            tension: 0.4
          }]
        },
        options: {
          responsive: true,
          maintainAspectRatio: false,
          plugins: {
            legend: {
              display: false
            }
          },
          scales: {
            y: {
              beginAtZero: true,
              ticks: {
                callback: function(value) {
                  return '$' + value;
                }
              }
```

Code 5.4



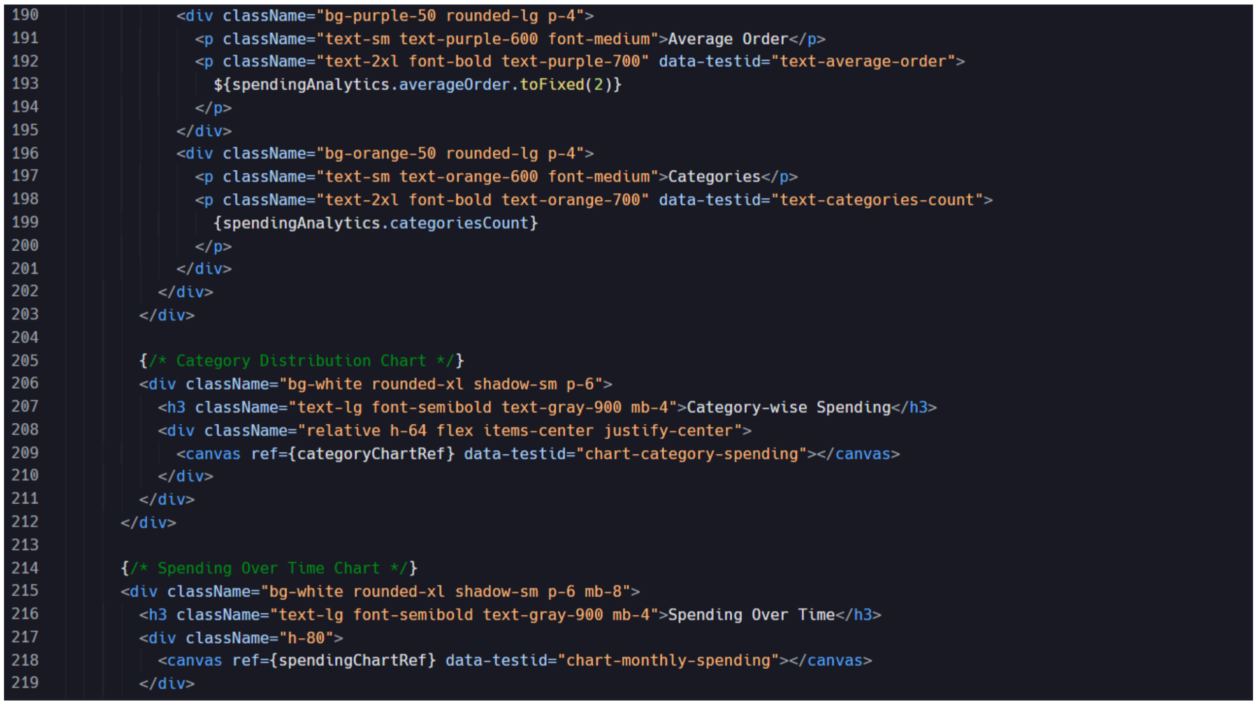
```
    return () => {
      if (categoryChartInstance.current) {
        categoryChartInstance.current.destroy();
      }
      if (spendingChartInstance.current) {
        spendingChartInstance.current.destroy();
      }
    };
  }, [purchaseHistory, spendingAnalytics, monthlySpending]);

  if (isLoading) {
    return (
      <div className="page-content">
        <h2 className="text-3xl font-bold text-gray-900 mb-8">Shopping Analytics</h2>
        <div className="animate-pulse">
          <div className="grid grid-cols-1 lg:grid-cols-2 gap-8 mb-8">
            <div className="bg-gray-300 rounded-xl h-64"></div>
            <div className="bg-gray-300 rounded-xl h-64"></div>
          </div>
          <div className="bg-gray-300 rounded-xl h-80"></div>
        </div>
      </div>
    );
  }

  if (purchaseHistory.length === 0) {
    return (
      <div className="page-content">
        <h2 className="text-3xl font-bold text-gray-900 mb-8" data-testid="text-analytics-title">Shopping Analytics</h2>

```

Code 5.5



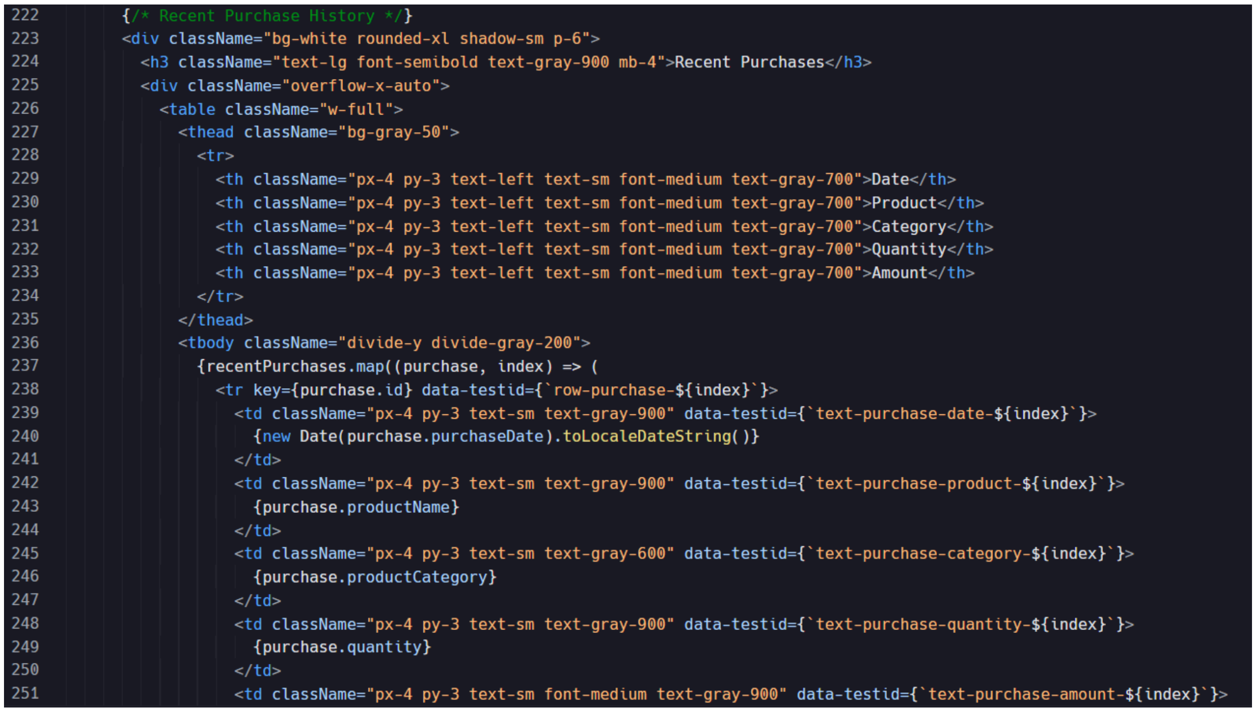
```
        <div className="text-center py-12 bg-white rounded-xl shadow-sm">
          <i className="fas fa-chart-bar text-6xl text-gray-300 mb-6"></i>
          <h3 className="text-xl font-medium text-gray-900 mb-2">No purchase data available</h3>
          <p className="text-gray-500 mb-6">Make some purchases to see your shopping analytics</p>
        </div>
      </div>
    );
  }

  return (
    <div className="page-content">
      <h2 className="text-3xl font-bold text-gray-900 mb-8" data-testid="text-analytics-title">Shopping Analytics</h2>

      <div className="grid grid-cols-1 lg:grid-cols-2 gap-8 mb-8">
        {/* Summary Cards */}
        <div className="bg-white rounded-xl shadow-sm p-6">
          <h3 className="text-lg font-semibold text-gray-900 mb-4">Spending Summary</h3>
          <div className="grid grid-cols-2 gap-4">
            <div className="bg-blue-50 rounded-lg p-4">
              <p className="text-sm text-blue-600 font-medium">Total Spent</p>
              <p className="text-2xl font-bold text-blue-700" data-testid="text-total-spent">
                ${spendingAnalytics.totalSpent.toFixed(2)}
              </p>
            </div>
            <div className="bg-green-50 rounded-lg p-4">
              <p className="text-sm text-green-600 font-medium">Items Purchased</p>
              <p className="text-2xl font-bold text-green-700" data-testid="text-total-items">
                {spendingAnalytics.totalItems}
              </p>
            </div>
```

Code 5.6



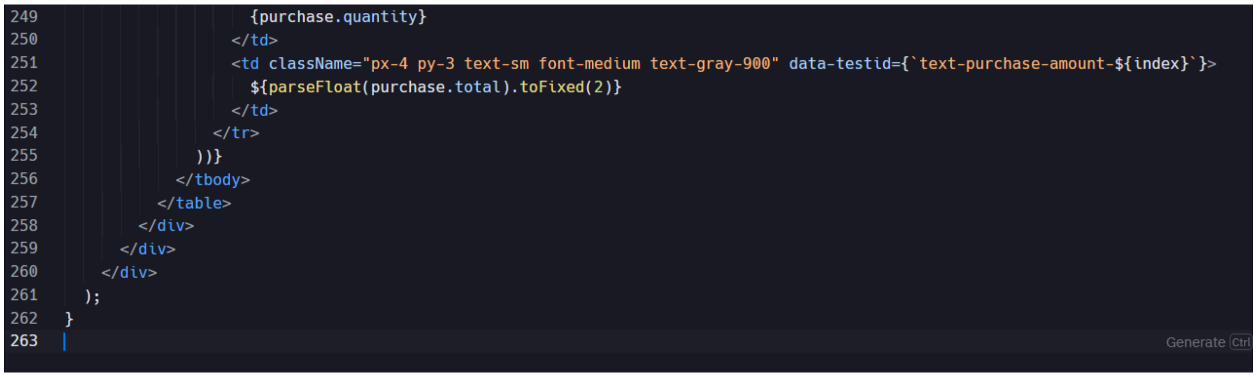
```
            <div className="bg-purple-50 rounded-lg p-4">
              <p className="text-sm text-purple-600 font-medium">Average Order</p>
              <p className="text-2xl font-bold text-purple-700" data-testid="text-average-order">
                ${spendingAnalytics.averageOrder.toFixed(2)}
              </p>
            </div>
            <div className="bg-orange-50 rounded-lg p-4">
              <p className="text-sm text-orange-600 font-medium">Categories</p>
              <p className="text-2xl font-bold text-orange-700" data-testid="text-categories-count">
                {spendingAnalytics.categoriesCount}
              </p>
            </div>
          </div>
        </div>

        {/* Category Distribution Chart */}
        <div className="bg-white rounded-xl shadow-sm p-6">
          <h3 className="text-lg font-semibold text-gray-900 mb-4">Category-wise Spending</h3>
          <div className="relative h-64 flex items-center justify-center">
            <canvas ref={categoryChartRef} data-testid="chart-category-spending"></canvas>
          </div>
        </div>
      </div>

      {/* Spending Over Time Chart */}
      <div className="bg-white rounded-xl shadow-sm p-6 mb-8">
        <h3 className="text-lg font-semibold text-gray-900 mb-4">Spending Over Time</h3>
        <div className="h-80">
          <canvas ref={spendingChartRef} data-testid="chart-monthly-spending"></canvas>
        </div>
```

Code 5.7



```
      {/* Recent Purchase History */}
      <div className="bg-white rounded-xl shadow-sm p-6">
        <h3 className="text-lg font-semibold text-gray-900 mb-4">Recent Purchases</h3>
        <div className="overflow-x-auto">
          <table className="w-full">
            <thead className="bg-gray-50">
              <tr>
                <th className="px-4 py-3 text-left text-sm font-medium text-gray-700">Date</th>
                <th className="px-4 py-3 text-left text-sm font-medium text-gray-700">Product</th>
                <th className="px-4 py-3 text-left text-sm font-medium text-gray-700">Category</th>
                <th className="px-4 py-3 text-left text-sm font-medium text-gray-700">Quantity</th>
                <th className="px-4 py-3 text-left text-sm font-medium text-gray-700">Amount</th>
              </tr>
            </thead>
            <tbody className="divide-y divide-gray-200">
              {recentPurchases.map((purchase, index) => (
                <tr key={purchase.id} data-testid={`row-purchase-${index}`}>
                  <td className="px-4 py-3 text-sm text-gray-900" data-testid={`text-purchase-date-${index}`}>
                    {new Date(purchase.purchaseDate).toLocaleDateString()}
                  </td>
                  <td className="px-4 py-3 text-sm text-gray-900" data-testid={`text-purchase-product-${index}`}>
                    {purchase.productName}
                  </td>
                  <td className="px-4 py-3 text-sm text-gray-600" data-testid={`text-purchase-category-${index}`}>
                    {purchase.productCategory}
                  </td>
                  <td className="px-4 py-3 text-sm text-gray-900" data-testid={`text-purchase-quantity-${index}`}>
                    {purchase.quantity}
                  </td>
                  <td className="px-4 py-3 text-sm font-medium text-gray-900" data-testid={`text-purchase-amount-${index}`}>
```

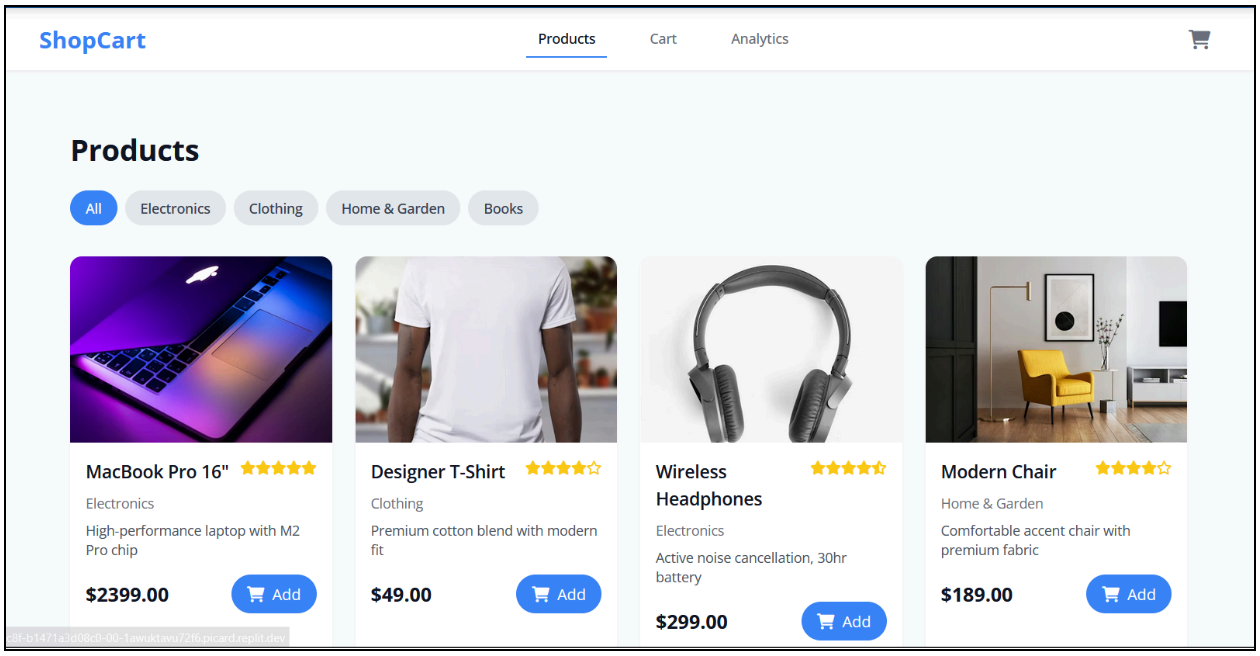
Code 5.8



```
                    {purchase.quantity}
                  </td>
                  <td className="px-4 py-3 text-sm font-medium text-gray-900" data-testid={`text-purchase-amount-${index}`}>
                    ${parseFloat(purchase.total).toFixed(2)}
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      </div>
    </div>
  );
}

```

Code 5.9

SCREENSHOTS



Products

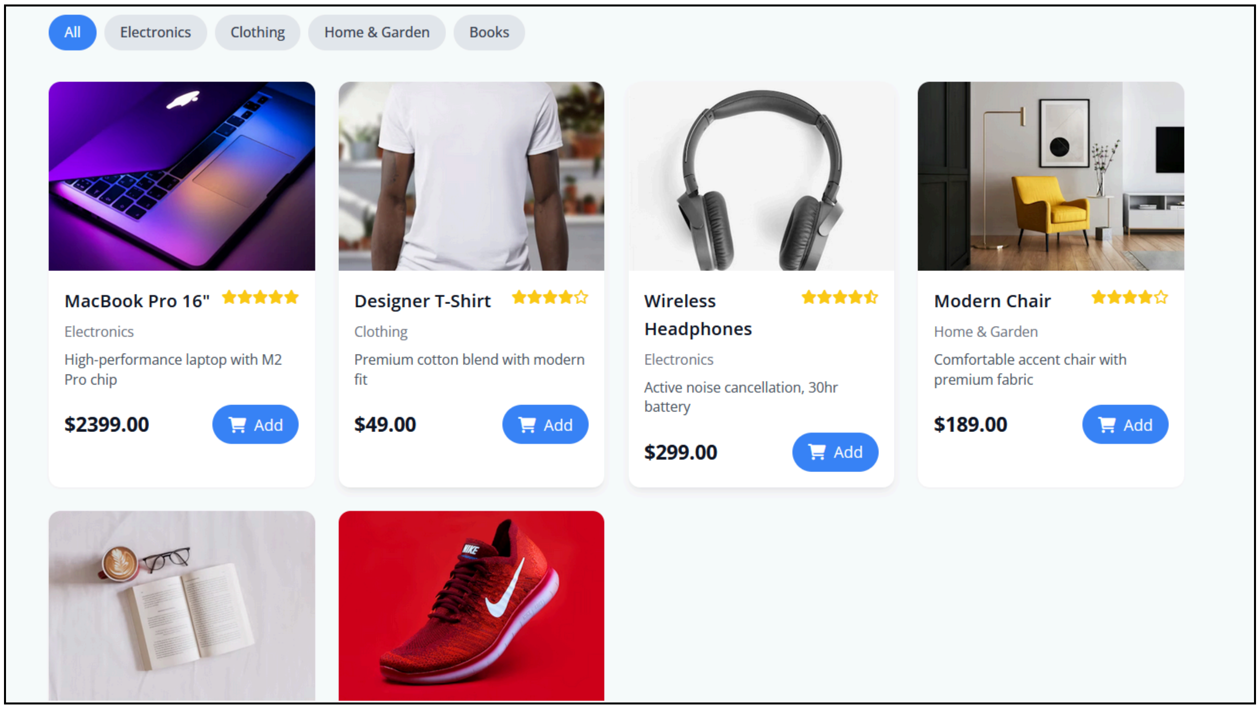
MacBook Pro 16"
Electronics
High-performance laptop with M2 Pro chip
$2399.00

Designer T-Shirt
Clothing
Premium cotton blend with modern fit
$49.00

Wireless Headphones
Electronics
Active noise cancellation, 30hr battery
$299.00

Modern Chair
Home & Garden
Comfortable accent chair with premium fabric
$189.00

Figure 1.1



MacBook Pro 16"
Electronics
High-performance laptop with M2 Pro chip
$2399.00

Designer T-Shirt
Clothing
Premium cotton blend with modern fit
$49.00

Wireless Headphones
Electronics
Active noise cancellation, 30hr battery
$299.00

Modern Chair
Home & Garden
Comfortable accent chair with premium fabric
$189.00

Figure 1.2

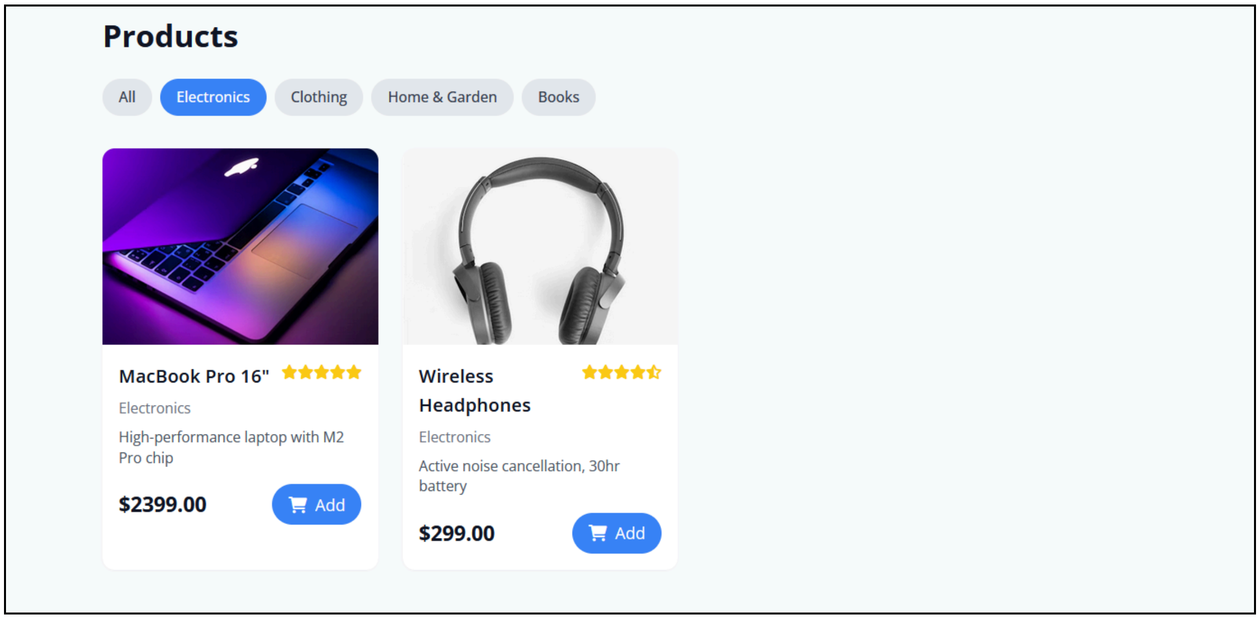


Figure 1.3

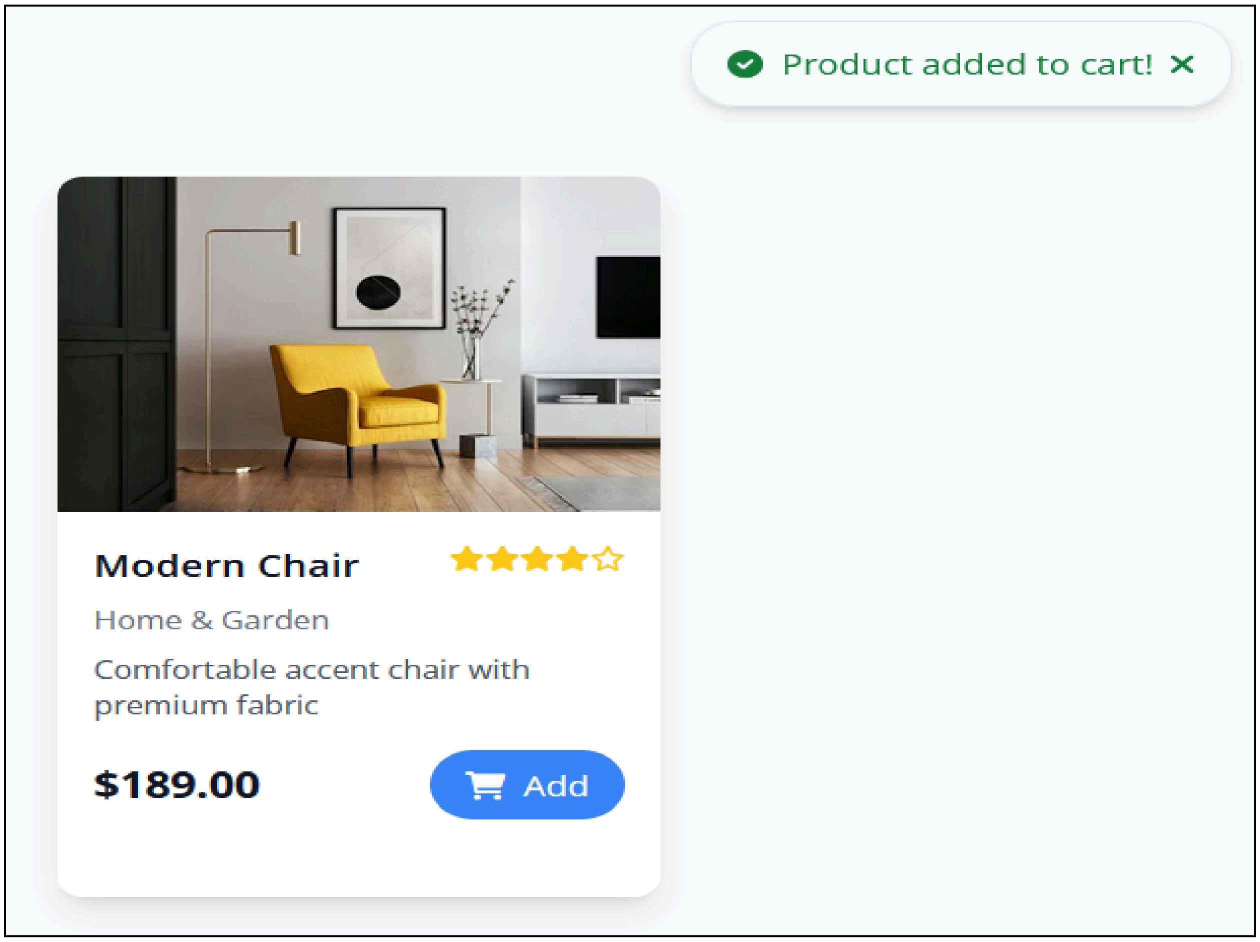
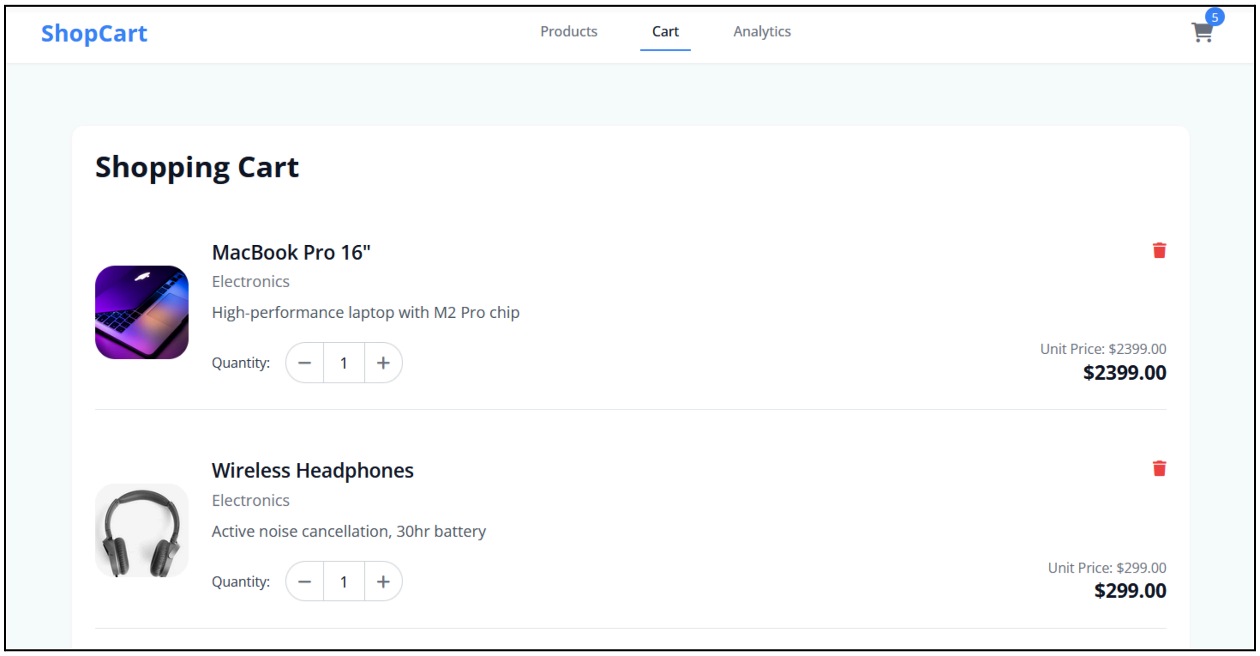


Figure 1.4

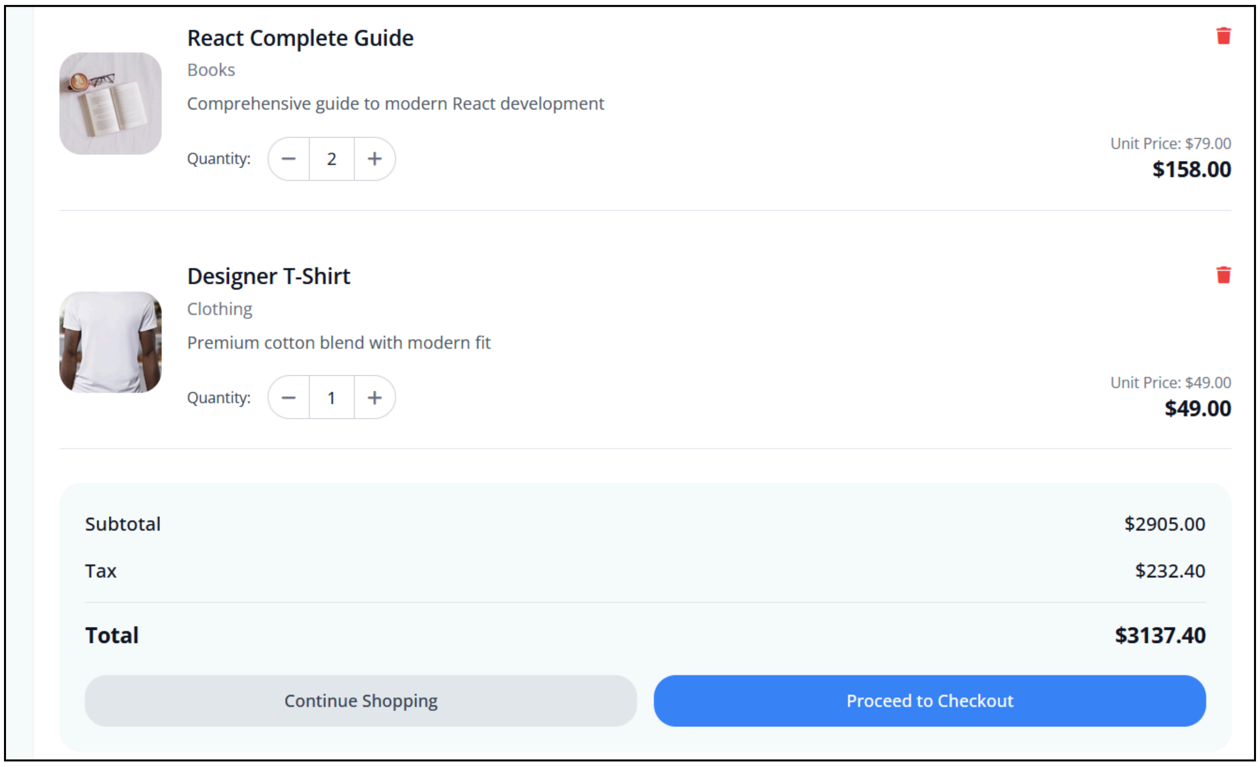


Shopping Cart

MacBook Pro 16"
Electronics
High-performance laptop with M2 Pro chip
Quantity: 1
Unit Price: $2399.00
$2399.00

Wireless Headphones
Electronics
Active noise cancellation, 30hr battery
Quantity: 1
Unit Price: $299.00
$299.00

Figure 1.5

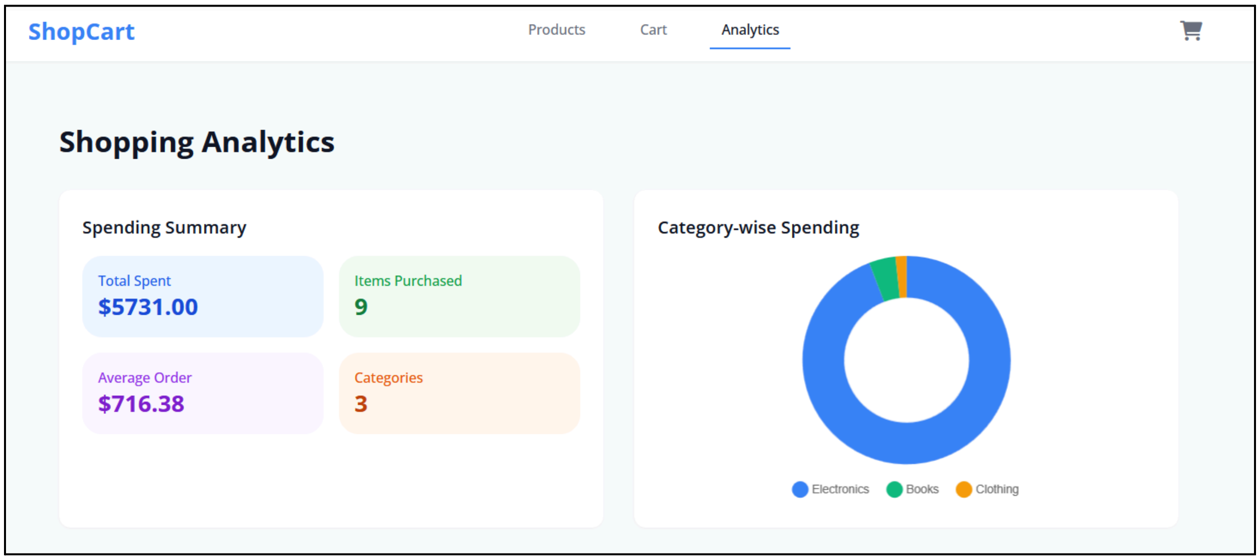


React Complete Guide
Books
Comprehensive guide to modern React development
Quantity: 2
Unit Price: $79.00
$158.00

Designer T-Shirt
Clothing
Premium cotton blend with modern fit
Quantity: 1
Unit Price: $49.00
$49.00

Subtotal $2905.00
Tax $232.40
Total $3137.40

Figure 1.6



Shopping Analytics

Spending Summary

Total Spent
$5731.00

Items Purchased
9

Average Order
$716.38

Categories
3

Category-wise Spending

Figure 1.7

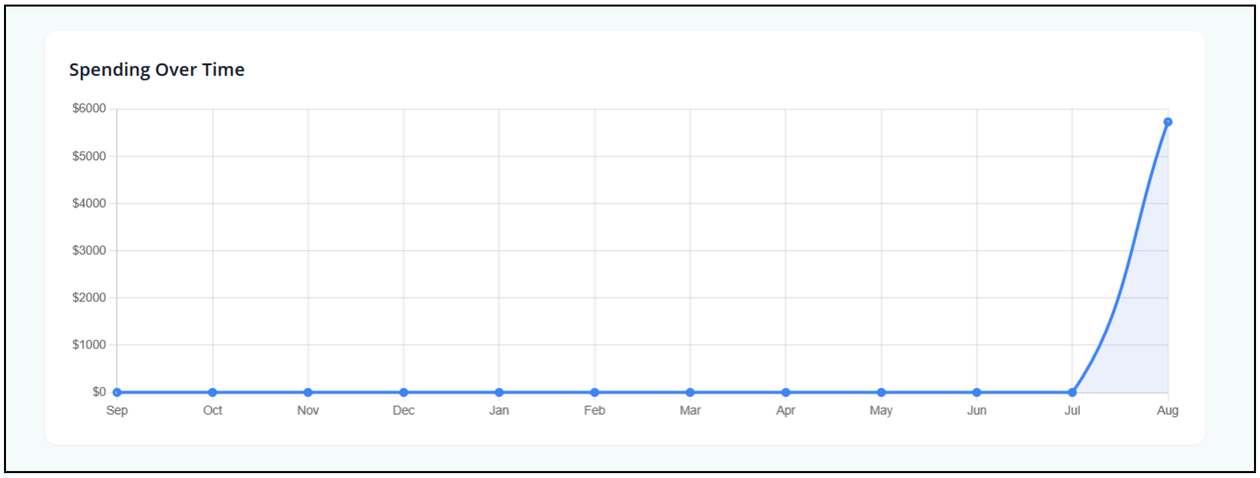
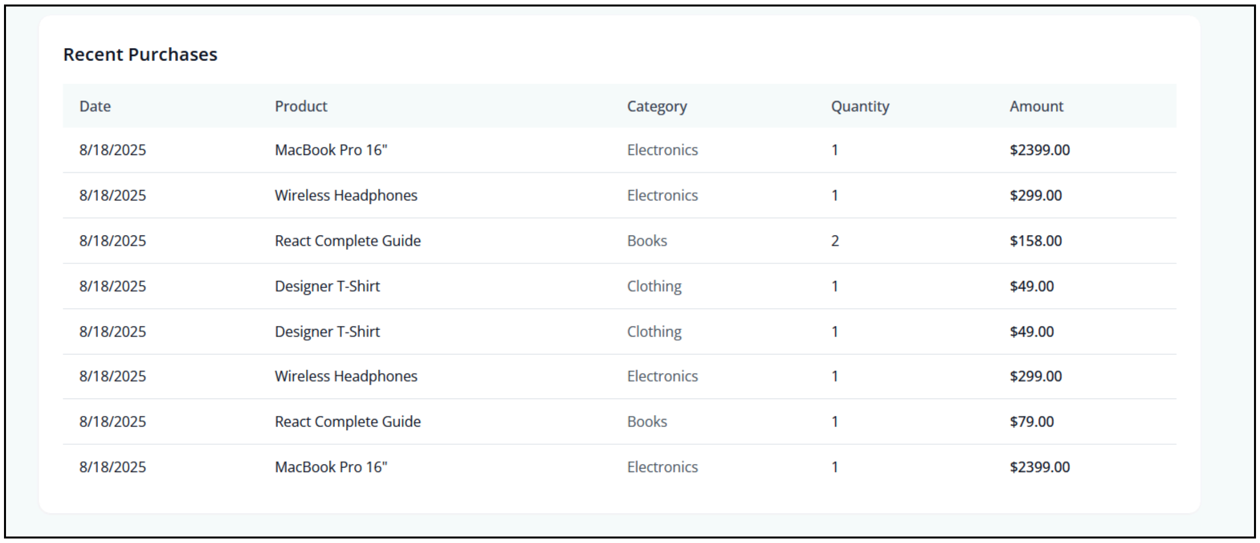


Figure 1.8



Recent Purchases

Date	Product	Category	Quantity	Amount
8/18/2025	MacBook Pro 16"	Electronics	1	$2399.00
8/18/2025	Wireless Headphones	Electronics	1	$299.00
8/18/2025	React Complete Guide	Books	2	$158.00
8/18/2025	Designer T-Shirt	Clothing	1	$49.00
8/18/2025	Designer T-Shirt	Clothing	1	$49.00
8/18/2025	Wireless Headphones	Electronics	1	$299.00
8/18/2025	React Complete Guide	Books	1	$79.00
8/18/2025	MacBook Pro 16"	Electronics	1	$2399.00

Figure 1.9

30% EXTRA CONTRIBUTION

In addition to implementing the core e-commerce functionality with Redux/Context API for managing complex states such as product listings, shopping cart updates, and user authentication, we made significant extra contributions that enhanced usability, scalability, and clarity of the system. These visible improvements account for approximately 30% extra work beyond the base requirements:

- **Cart State Reset After Checkout** – Implemented automatic clearing of cart state once the checkout process is completed. This ensured that stale cart data does not persist for the next session.
- **Error Handling in Actions** – Added validation and error handling for edge cases such as adding out-of-stock items, removing non-existent items, or exceeding quantity limits. This increased the reliability of the application.
- **State Persistence with Local Storage** – Integrated local storage synchronization so that cart and user state is preserved across page refreshes, improving the user experience.
- **Modular Reducer/Context Structure** – Organized reducers and contexts into separate modules (cart, user, products) for better scalability and easier debugging. This mirrors how larger production systems are structured.
- **Basic Middleware Logging (Redux only)** – For Redux implementation, included a simple logging middleware that tracks dispatched actions and resulting state changes, making the system more transparent and easier to debug.

CONCLUSION

In this project, we successfully demonstrated how complex state can be efficiently managed in an E-commerce application using Redux and the Context API. The implementation highlighted how global state management eliminates the issues of prop drilling and ensures that crucial data such as cart items, user authentication, product listings, and order details remain synchronized across the application.

By leveraging centralized state, we achieved better scalability, maintainability, and consistency within the application. Furthermore, integrating advanced techniques such as middleware for asynchronous operations (e.g., handling API calls for product fetching and checkout) and optimized use of selectors minimized unnecessary re-renders, improving the overall performance and responsiveness of the system.

Overall, the project not only achieved its primary goal of demonstrating complex state management but also showcased how thoughtful design decisions, performance-aware coding practices, and enhanced visualizations can transform a basic application into a more robust, efficient, and production-ready solution.