MODULE 3: SHOPPING CART FUNCTIONALITY

LESSON 7: State Management with React Hooks

Learning Objective

By the end of this lesson, you will understand React Hooks and manage shopping cart state using `useState` and `useEffect`.

7.1 What are React Hooks?

Hooks are functions that let you "hook into" React features. Think of them as special tools that give your components superpowers.

Common Hooks:

- `useState` - Adds state (data) to components

- `useEffect` - Performs side effects (like saving to localStorage)
- `useContext` - Shares data between components
- `useReducer` - Complex state management

7.2 Creating the Cart Context

Instead of passing cart data through multiple components (prop drilling), we'll use Context to make cart available everywhere.

Step 1: Create `app/context/CartContext.js`

```
javascript
"use client";

import { createContext, useState, useContext, useEffect } from 'react';

// Create the context
const CartContext = createContext();

// Custom hook to use the cart context
export function useCart() {
 return useContext(CartContext);
}

// Provider component that wraps the app
export function CartProvider({ children }) {
 const [cart, setCart] = useState([]);
 const [isInitialized, setIsInitialized] = useState(false);

 // Load cart from localStorage on initial mount
 useEffect(() => {
  const savedCart = localStorage.getItem('ceylonmart-cart');
  if (savedCart) {
```

```
    setCart(JSON.parse(savedCart));
  }
  setIsInitialized(true);
}, []);

// Save cart to localStorage whenever it changes
useEffect(() => {
  if (isInitialized) {
    localStorage.setItem('ceylonmart-cart', JSON.stringify(cart));
  }
}, [cart, isInitialized]);

// Add item to cart
const addToCart = (product, quantity = 1) => {
  setCart(currentCart => {
    // Check if product already in cart
    const existingItem = currentCart.find(item => item.id === product.id);

    if (existingItem) {
      // Increase quantity
      return currentCart.map(item =>
        item.id === product.id
          ? { ...item, quantity: item.quantity + quantity }
          : item
      );
    } else {
      // Add new item
      return [...currentCart, { ...product, quantity }];
    }
  });
};

// Remove item from cart
const removeFromCart = (productId) => {
  setCart(currentCart => currentCart.filter(item => item.id !== productId));
};
```

```
// Update quantity
const updateQuantity = (productId, newQuantity) => {
 if (newQuantity < 1) {
  removeFromCart(productId);
  return;
 }

 setCart(currentCart =>
  currentCart.map(item =>
   item.id === productId ? { ...item, quantity: newQuantity } : item
  )
 );
};

// Clear cart
const clearCart = () => {
 setCart([]);
};

// Calculate total items
const cartItemsCount = cart.reduce((total, item) => total + item.quantity, 0);

// Calculate total price
const cartTotal = cart.reduce(
 (total, item) => total + item.price * item.quantity,
 0
);

// Value object to be provided
const value = {
 cart,
 addToCart,
 removeFromCart,
 updateQuantity,
 clearCart,
 cartItemsCount,
 cartTotal
```

```
  };

  return (
    <CartContext.Provider value={value}>
      {children}
    </CartContext.Provider>
  );
}
```

7.3 Wrapping the App with CartProvider

Step 1: Update `app/layout.js`

```javascript
import { CartProvider } from './context/CartContext';
import './globals.css';

export const metadata = {
  title: 'CeylonMart - Sri Lankan Products',
  description: 'Discover authentic Sri Lankan products',
};

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body>
        <CartProvider>
          {children}
        </CartProvider>
      </body>
    </html>
  );
```

```
}
```

7.4 Using Cart in ProductCard

Step 1: Update `app/components/ProductCard.js`

javascript

```
import Link from 'next/link';
import { useCart } from '@/app/context/CartContext';

function ProductCard({ product }) {
 const { addToCart } = useCart();

 const handleAddToCart = (e) => {
  e.preventDefault(); // Prevent Link navigation
  addToCart(product, 1);

  // Optional: Show feedback
  alert(`${product.name} added to cart!`);
 };

 return (
  <Link href={`/product/${product.id}`} style={{ textDecoration: 'none', color: 'inherit' }}>
   <div style={{
    border: '1px solid #e2e8f0',
    borderRadius: '8px',
    padding: '1rem',
    backgroundColor: 'white',
    boxShadow: '0 2px 4px rgba(0,0,0,0.1)',
    transition: 'transform 0.2s, box-shadow 0.2s',
    cursor: 'pointer',
    height: '100%',
    display: 'flex',
    flexDirection: 'column'
```

```
}}
onMouseEnter={e => {
 e.currentTarget.style.transform = 'scale(1.02)';
 e.currentTarget.style.boxShadow = '0 10px 15px -3px rgba(0,0,0,0.1)';
}}
onMouseLeave={e => {
 e.currentTarget.style.transform = 'scale(1)';
 e.currentTarget.style.boxShadow = '0 2px 4px rgba(0,0,0,0.1)';
}}
>
 {/* Product Image Placeholder */}
 <div style={{
  width: '100%',
  height: '200px',
  backgroundColor: '#f1f5f9',
  borderRadius: '4px',
  marginBottom: '1rem',
  display: 'flex',
  alignItems: 'center',
  justifyContent: 'center',
  color: '#64748b'
 }}>
  🖼️ {product.name}
 </div>

 {/* Product Details */}
 <h3 style={{ margin: '0 0 0.5rem 0', fontSize: '1.2rem' }}>
  {product.name}
 </h3>

 <p style={{ color: '#64748b', margin: '0 0 1rem 0', fontSize: '0.9rem', flex: 1 }}>
  {product.description.substring(0, 60)}...
 </p>

 <div style={{
  display: 'flex',
  justifyContent: 'space-between',
```

```
          alignItems: 'center',
          marginTop: 'auto'
        }}>
          <span style={{
            fontSize: '1.3rem',
            fontWeight: 'bold',
            color: '#1e40af'
          }}>
            Rs. {product.price.toLocaleString()}
          </span>

          <button
            onClick={handleAddToCart}
            style={{
              backgroundColor: '#1e40af',
              color: 'white',
              border: 'none',
              padding: '0.5rem 1rem',
              borderRadius: '4px',
              cursor: 'pointer',
              fontSize: '0.9rem'
            }}
          >
            Add to Cart
          </button>
        </div>
      </div>
    </Link>
  );
}

export default ProductCard;
```

7.5 Adding Cart to Product Detail Page

Step 1: Update `app/product/[id]/page.js`

Add the cart functionality:

```javascript
"use client";

import { useState } from 'react';
import { useParams } from 'next/navigation';
import Link from 'next/link';
import Header from '@/app/components/Header';
import Footer from '@/app/components/Footer';
import ProductCard from '@/app/components/ProductCard';
import products from '@/app/data/products';
import { useCart } from '@/app/context/CartContext';

export default function ProductPage() {
  const params = useParams();
  const productId = parseInt(params.id);
  const product = products.find(p => p.id === productId);
  const { addToCart } = useCart();
  const [quantity, setQuantity] = useState(1);
  const [addedToCart, setAddedToCart] = useState(false);

  if (!product) {
    // ... (not found handling)
  }

  const handleAddToCart = () => {
    addToCart(product, quantity);
    setAddedToCart(true);

    // Reset after 3 seconds
    setTimeout(() => setAddedToCart(false), 3000);
  };

  const handleBuyNow = () => {
```

```
    addToCart(product, quantity);
    // Navigate to cart page
    window.location.href = '/cart';
  };

  return (
    <div>
      <Header />

      <main style={{ padding: '2rem' }}>
        {/* Breadcrumb (same as before) */}

        {/* Product Details Grid */}
        <div style={{
          display: 'grid',
          gridTemplateColumns: '1fr 1fr',
          gap: '3rem',
          maxWidth: '1200px',
          margin: '0 auto'
        }}>
          {/* Left Column - Image (same) */}

          {/* Right Column - Details */}
          <div>
            <h1 style={{ fontSize: '2.5rem', margin: '0 0 1rem 0' }}>
              {product.name}
            </h1>

            <p style={{ fontSize: '1.1rem', color: '#475569', lineHeight: '1.6', marginBottom: '2rem' }}>
              {product.description}
            </p>

            <div style={{ marginBottom: '2rem' }}>
              <span style={{ fontSize: '2rem', fontWeight: 'bold', color: '#1e40af' }}>
                Rs. {product.price.toLocaleString()}
              </span>
```

```
          <span style={{ marginLeft: '1rem', color: '#059669', backgroundColor: '#d1fae5',
padding: '0.25rem 0.75rem', borderRadius: '9999px', fontSize: '0.875rem' }}>
            In Stock
          </span>
        </div>

        <div style={{ marginBottom: '2rem' }}>
          <label style={{ display: 'block', marginBottom: '0.5rem', fontWeight: 'bold' }}>
            Quantity:
          </label>
          <select
            value={quantity}
            onChange={(e) => setQuantity(parseInt(e.target.value))}
            style={{
              padding: '0.5rem',
              border: '1px solid #cbd5e1',
              borderRadius: '4px',
              width: '100px'
            }}
          >
            {[1,2,3,4,5].map(num => (
              <option key={num} value={num}>{num}</option>
            ))}
          </select>
        </div>

        {/* Success Message */}
        {addedToCart && (
          <div style={{
            backgroundColor: '#d1fae5',
            color: '#065f46',
            padding: '1rem',
            borderRadius: '4px',
            marginBottom: '1rem'
          }}>
            ✓ Added to cart successfully!
          </div>
```

```
)}

<div style={{ display: 'flex', gap: '1rem' }}>
 <button
  onClick={handleAddToCart}
  style={{
   flex: 2,
   backgroundColor: '#1e40af',
   color: 'white',
   border: 'none',
   padding: '1rem',
   borderRadius: '4px',
   fontSize: '1.1rem',
   fontWeight: 'bold',
   cursor: 'pointer'
  }}
 >
  Add to Cart
 </button>

 <button
  onClick={handleBuyNow}
  style={{
   flex: 1,
   backgroundColor: 'white',
   color: '#1e40af',
   border: '2px solid #1e40af',
   padding: '1rem',
   borderRadius: '4px',
   fontSize: '1.1rem',
   cursor: 'pointer'
  }}
 >
  Buy Now
 </button>
</div>
```

```
          {/* Rest of product details */}
        </div>
      </div>
    </main>

    <Footer />
  </div>
 );
}
```

Lesson 7 Summary

- Context API shares data across components without prop drilling
- `useState` manages component state
- `useEffect` handles side effects like localStorage
- Cart operations: add, remove, update, clear
- Always provide user feedback for actions

Practice Exercises

1. Add a mini cart dropdown in the header showing cart items
2. Implement "Add to Cart" animation (item flies to cart)
3. Show cart item count with a badge on the cart icon
4. Challenge: Add undo functionality when removing items