MODULE 2: BUILDING THE PRODUCT STORE

LESSON 6: Product Listing Page with Grid Layout

Learning Objective

By the end of this lesson, you will create a dedicated product listing page with filtering, sorting, and search functionality.

6.1 Creating the Products Page

Instead of showing all products on the homepage, let's create a dedicated `/products` page.

Step 1: Create `app/products/page.js`

javascript

```
"use client";
```

```
2.
3.  import { useState } from 'react';
4.  import Header from '@/app/components/Header';
5.  import Footer from '@/app/components/Footer';
6.  import ProductCard from '@/app/components/ProductCard';
7.  import products from '@/app/data/products';
8.
9.  export default function ProductsPage() {
10.   const [searchTerm, setSearchTerm] = useState('');
11.   const [selectedCategory, setSelectedCategory] = useState('All');
12.   const [sortBy, setSortBy] = useState('default');
13.
14.   // Get unique categories
15.   const categories = ['All', ...new Set(products.map(p => p.category))];
16.
17.   // Filter and sort products
18.   const filteredProducts = products
19.     .filter(product => {
20.       // Category filter
21.       if (selectedCategory !== 'All' && product.category !== selectedCategory) {
22.         return false;
23.       }
24.
25.       // Search filter
26.       if (searchTerm &&
    !product.name.toLowerCase().includes(searchTerm.toLowerCase()) &&
27.           !product.description.toLowerCase().includes(searchTerm.toLowerCase())) {
28.         return false;
29.       }
30.
31.       return true;
```

```
  })
  .sort((a, b) => {
    if (sortBy === 'price-low') {
      return a.price - b.price;
    } else if (sortBy === 'price-high') {
      return b.price - a.price;
    } else if (sortBy === 'name') {
      return a.name.localeCompare(b.name);
    }
    return 0;
  });

return (
  <div>
    <Header />

    {/* Page Header */}
    <section style={{
      backgroundColor: '#1e40af',
      color: 'white',
      padding: '3rem 2rem',
      textAlign: 'center'
    }}>
      <h1 style={{ fontSize: '2.5rem', marginBottom: '1rem' }}>
        Our Products
      </h1>
      <p style={{ fontSize: '1.2rem', opacity: 0.9, maxWidth: '600px', margin: '0 auto' }}>
        Discover authentic Sri Lankan products, crafted with care and delivered with love.
      </p>
    </section>

```

```
      <main style={{ padding: '2rem' }}>
        {/* Filters Bar */}
        <div style={{
          backgroundColor: 'white',
          padding: '1.5rem',
          borderRadius: '8px',
          boxShadow: '0 2px 4px rgba(0,0,0,0.1)',
          marginBottom: '2rem',
          display: 'flex',
          flexWrap: 'wrap',
          gap: '1rem',
          alignItems: 'center',
          justifyContent: 'space-between'
        }}>

          {/* Search */}
          <div style={{ flex: 1, minWidth: '250px' }}>
            <input
              type="text"
              placeholder="Search products..."
              value={searchTerm}
              onChange={(e) => setSearchTerm(e.target.value)}
              style={{
                width: '100%',
                padding: '0.75rem',
                border: '1px solid #e2e8f0',
                borderRadius: '4px',
                fontSize: '1rem'
              }}
            />
          </div>
```

```
94.
95.       {/* Category Filter */}
96.       <div style={{ minWidth: '200px' }}>
97.         <select
98.           value={selectedCategory}
99.           onChange={(e) => setSelectedCategory(e.target.value)}
100.              style={{
101.            width: '100%',
102.                padding: '0.75rem',
103.                border: '1px solid #e2e8f0',
104.                borderRadius: '4px',
105.                fontSize: '1rem',
106.                backgroundColor: 'white'
107.              }}
108.            >
109.              {categories.map(category => (
110.           <option key={category} value={category}>
111.             {category}
112.           </option>
113.         ))}
114.       </select>
115.     </div>
116.
117.     {/* Sort By */}
118.     <div style={{ minWidth: '200px' }}>
119.       <select
120.             value={sortBy}
121.           onChange={(e) => setSortBy(e.target.value)}
122.             style={{
123.               width: '100%',
124.               padding: '0.75rem',
```

```
            border: '1px solid #e2e8f0',
            borderRadius: '4px',
            fontSize: '1rem',
            backgroundColor: 'white'
          }}
        >
          <option value="default">Default</option>
          <option value="price-low">Price: Low to High</option>
          <option value="price-high">Price: High to Low</option>
          <option value="name">Name: A to Z</option>
        </select>
      </div>

      {/* Results Count */}
      <div style={{ color: '#64748b' }}>
        {filteredProducts.length} products found
      </div>
    </div>

    {/* Products Grid */}
    {filteredProducts.length > 0 ? (
      <div style={{
        display: 'grid',
        gridTemplateColumns: 'repeat(auto-fill, minmax(280px, 1fr))',
        gap: '2rem'
      }}>
        {filteredProducts.map(product => (
          <ProductCard key={product.id} product={product} />
        ))}
      </div>
    ) : (
```

```
            <div style={{
              textAlign: 'center',
              padding: '4rem',
              backgroundColor: '#f8fafc',
              borderRadius: '8px'
            }}>
              <h3 style={{ fontSize: '1.5rem', color: '#475569', marginBottom: '1rem' }}>
                No products found
              </h3>
              <p style={{ color: '#64748b' }}>
                Try adjusting your search or filter to find what you're looking for.
              </p>
            </div>
          )}
        </main>

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

6.2 Understanding the Filter Logic

Let's break down how the filtering works:

Search Filter:

```
javascript
```

```
    if (searchTerm &&
        !product.name.toLowerCase().includes(searchTerm.toLowerCase()) &&
        !product.description.toLowerCase().includes(searchTerm.toLowerCase())) {
      return false;
    }
```

This checks if the search term appears in either the product name or description.

Category Filter:

```
javascript
    if (selectedCategory !== 'All' && product.category !== selectedCategory) {
      return false;
    }
```

Only show products matching the selected category (unless "All" is selected).

Sort Function:

```
javascript
  .sort((a, b) => {
    if (sortBy === 'price-low') {
      return a.price - b.price;  // Ascending
    } else if (sortBy === 'price-high') {
      return b.price - a.price;  // Descending
    }
  })
```

6.3 Updating the Header Navigation

Now let's update our Header to link to the new products page.

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

javascript

```
import Link from 'next/link';

function Header() {
  return (
    <header style={{
backgroundColor: '#1e40af',
      color: 'white',
      padding: '1rem',
      display: 'flex',
      justifyContent: 'space-between',
      alignItems: 'center'
    }}>
      <Link href="/" style={{ textDecoration: 'none', color: 'white' }}>
        <div style={{ fontSize: '1.5rem', fontWeight: 'bold' }}>
          🇱🇰 CeylonMart
        </div>
      </Link>

      <nav>
        <Link href="/" style={{ color: 'white', margin: '0 1rem', textDecoration: 'none' }}>
          Home
        </Link>
        <Link href="/products" style={{ color: 'white', margin: '0 1rem', textDecoration:
    'none' }}>
```

```
            Products
          </Link>
        <Link href="/cart" style={{ color: 'white', margin: '0 1rem', textDecoration: 'none' }}>
            Cart
          </Link>
        </nav>
      </header>
    );
  }

export default Header;
```

6.4 Adding Pagination

When you have many products, you need pagination. Let's add it.

Step 1: Add pagination state to `app/products/page.js`

Add these state variables:

```
javascript

  const [currentPage, setCurrentPage] = useState(1);

const productsPerPage = 9;
```

Step 2: Calculate paginated products

```
javascript

// After filteredProducts, calculate pagination
```

```
    const indexOfLastProduct = currentPage * productsPerPage;
    const indexOfFirstProduct = indexOfLastProduct - productsPerPage;
    const currentProducts = filteredProducts.slice(indexOfFirstProduct,
      indexOfLastProduct);

const totalPages = Math.ceil(filteredProducts.length / productsPerPage);
```

Step 3: Update the grid to use `currentProducts`

javascript

```
    {currentProducts.map(product => (
      <ProductCard key={product.id} product={product} />

))}
```

Step 4: Add pagination controls after the grid

javascript

```
    {/* Pagination */}
    {totalPages > 1 && (
      <div style={{
        display: 'flex',
        justifyContent: 'center',
        gap: '0.5rem',
        marginTop: '3rem'
      }}>
        <button
          onClick={() => setCurrentPage(prev => Math.max(prev - 1, 1))}
          disabled={currentPage === 1}
          style={{
            padding: '0.5rem 1rem',
            border: '1px solid #e2e8f0',
```

```
        backgroundColor: currentPage === 1 ? '#f1f5f9' : 'white',
        color: currentPage === 1 ? '#94a3b8' : '#1e40af',
        borderRadius: '4px',
        cursor: currentPage === 1 ? 'not-allowed' : 'pointer'
      }}
    >
      Previous
    </button>

    {[...Array(totalPages)].map((_, i) => (
      <button
        key={i + 1}
        onClick={() => setCurrentPage(i + 1)}
        style={{
          padding: '0.5rem 1rem',
          border: '1px solid #e2e8f0',
          backgroundColor: currentPage === i + 1 ? '#1e40af' : 'white',
          color: currentPage === i + 1 ? 'white' : '#1e293b',
          borderRadius: '4px',
          cursor: 'pointer'
        }}
      >
        {i + 1}
      </button>
    ))}

    <button
      onClick={() => setCurrentPage(prev => Math.min(prev + 1, totalPages))}
      disabled={currentPage === totalPages}
      style={{
        padding: '0.5rem 1rem',
```

```
        border: '1px solid #e2e8f0',
        backgroundColor: currentPage === totalPages ? '#f1f5f9' : 'white',
        color: currentPage === totalPages ? '#94a3b8' : '#1e40af',
        borderRadius: '4px',
        cursor: currentPage === totalPages ? 'not-allowed' : 'pointer'
      }}
    >
      Next
    </button>
  </div>
)}
```

6.5 Adding Active Filters Display

Let users see what filters are active and clear them easily.

Add this after the filters bar:

javascript

```
{/* Active Filters */}
{(searchTerm || selectedCategory !== 'All') && (
  <div style={{
    marginBottom: '1rem',
    display: 'flex',
    gap: '0.5rem',
    alignItems: 'center'
  }}>
    <span style={{ color: '#64748b' }}>Active filters:</span>
```

```
290.
291.       {searchTerm && (
292.         <div style={{
293.           backgroundColor: '#e2e8f0',
294.           padding: '0.25rem 0.75rem',
295.           borderRadius: '9999px',
296.           display: 'flex',
297.           alignItems: 'center',
298.           gap: '0.5rem'
299.         }}>
300.           <span>Search: "{searchTerm}"</span>
301.           <button
302.             onClick={() => setSearchTerm('')}
303.             style={{
304.               border: 'none',
305.               background: 'none',
306.               cursor: 'pointer',
307.               fontSize: '1.2rem',
308.               color: '#64748b'
309.             }}
310.           >
311.             ×
312.           </button>
313.         </div>
314.       )}
315.
316.       {selectedCategory !== 'All' && (
317.         <div style={{
318.           backgroundColor: '#e2e8f0',
319.           padding: '0.25rem 0.75rem',
320.           borderRadius: '9999px',
```

```
        display: 'flex',
        alignItems: 'center',
        gap: '0.5rem'
      }}>
        <span>Category: {selectedCategory}</span>
        <button
          onClick={() => setSelectedCategory('All')}
          style={{
            border: 'none',
            background: 'none',
            cursor: 'pointer',
            fontSize: '1.2rem',
            color: '#64748b'
          }}
        >
          ×
        </button>
      </div>
    )}

    <button
      onClick={() => {
        setSearchTerm('');
        setSelectedCategory('All');
        setSortBy('default');
      }}
      style={{
        color: '#1e40af',
        textDecoration: 'underline',
        border: 'none',
        background: 'none',
```

```
            cursor: 'pointer'
          }}
        >
          Clear all
        </button>
      </div>
)}
```

Lesson 6 Summary

- Dedicated product listing pages improve navigation
- Combined filters (search, category, sort) provide great UX
- Pagination handles large product catalogs
- Active filters help users understand current view
- Always show "no results" state

Practice Exercises

1. Add price range filter (min and max price inputs)
2. Create a grid/list view toggle (show products as grid or list)
3. Add "Quick View" button that shows product details in a modal
4. Challenge: Save filter preferences in localStorage so they persist on page refresh