Day 04

Building Frontend Components For Your Marketplace

By Kulsoom Imran

Components Built

1. Product Listing Grid

Purpose: Render all products data dynamically in a grid layout.

Details Displayed:

- Product Name
- Available Colors
- Price
- Category
- Status

2. Product Detail Component

Purpose: Display detailed information for a single product.

Details Displayed:

- Product Name
- Description
- Price
- Colors
- Status
- Add to Cart Option
- Related Products

3. Cart Component

Purpose: Display added items and manage the cart functionality.

Details Displayed:

- Product Name
- Product Image
- Price
- Quantity Control: Ability to increase or decrease the quantity of each product.
- Cart Summary:

- Total Number of Items: Displays the sum of all products in the cart.
- Total Price of Items: Shows the cumulative price of all items in the cart.

4. Wishlist Component

Purpose: Allow users to save products for future reference.

Details Displayed:

- Product Image
- Product Name
- Price

5. Filter Panel Component

Purpose: Provide users with multiple filtering options for precise results.

Filters Implemented:

- Price Range Slider (e.g., “Low to High,” “High to Low”)
- Category Selection
- Status Filters (e.g., "Just In," "Trending")

6. Related Products Component

Purpose: Suggest similar products based on the category.

Integration: Displayed on the product detail page.

7. Header and Footer Components

Purpose: Ensure consistent navigation and branding.

Details:

Header: Links to Home, LogIn, SignUp, Cart, Wishlist etc.

Footer: Links to legal pages, social media, and a brief brand overview.

8. Authentication:

Purpose: Enable users to create accounts and log in securely.

Implementation:

SignUp Form: Captures user details (Name, Email, Password) and stores them in Sanity using a preconfigured customer schema.

Login Form: Validates user credentials against data stored in Sanity and provides access upon successful login.

Features:

- Secure password handling (e.g., hashing).
- Error handling for invalid submissions.

Functionalities Implemented

1. Search Bar Functionality

Purpose: Allow users to search products by name or tags.

Features:

- Live search suggestions
- Integration with the Product Listing Grid

2. Filtering Functionalities

Filters Included:

- By Price Range: Dynamic slider to filter products by budget.
- By Categories: Filter products based on selected categories.
- By Status: Options like "Just In" or "Trending" to filter by relevance.

3. Dynamic Routing for Product Details

Purpose: Provide individual product pages.

Implementation: Used Next.js dynamic routing to fetch and display product details.

4. Cart Management

Features:

- Add, remove, or update product quantities.
- Calculate and display total price.
- Persist cart data using local storage.

5. Wishlist Functionality

Purpose: Save products for future reference.

Implementation:

- Used local storage to retain wishlist data.
- Allowed users to add/remove items with a single click.

6. Related Product Suggestions

Purpose: Improve user experience by suggesting relevant products.

Features:

Suggested based on categories.

7. Responsive Design

Purpose: Ensure seamless user experience across devices.

Implementation:

- Tailwind CSS media utilities for adaptive layouts.
- Mobile-first design for filter panels and grids.

Steps Taken to Build and Integrate Components and Functionalities

Project Setup:

- Initialized a Next.js project with TypeScript.
- Integrated Tailwind CSS for styling and configured global styles.
- Connected Sanity as the headless CMS for data storage and management.

Building Components:

- Developed modular components such as Products (filtered products, product card, product grid, related product), filter (price filter, category filter, status filter)
- Ensured reusability and adherence to design principles.

Integrating Functionalities:

- Built dynamic filtering using Tailwind CSS and Sanity data queries.
- Integrated a search bar with GROQ queries to fetch matching results in real-time.

Data Handling:

- Stored product and category data in Sanity.
- Utilized GROQ queries to dynamically fetch data as needed.
- Enforced type safety with TypeScript interfaces.

Routing:

- Used Next.js dynamic routing for individual product pages.
- Implemented deep links to specific categories and filtered results.

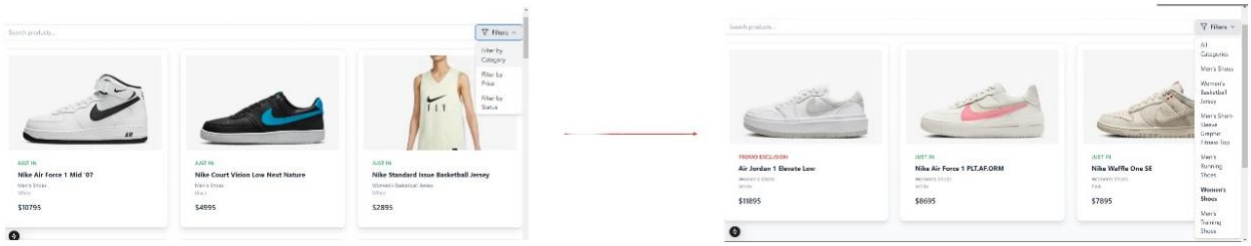
State Management:

- Managed cart and wishlist states with useState and useEffect
- Persisted user preferences and actions using local storage.

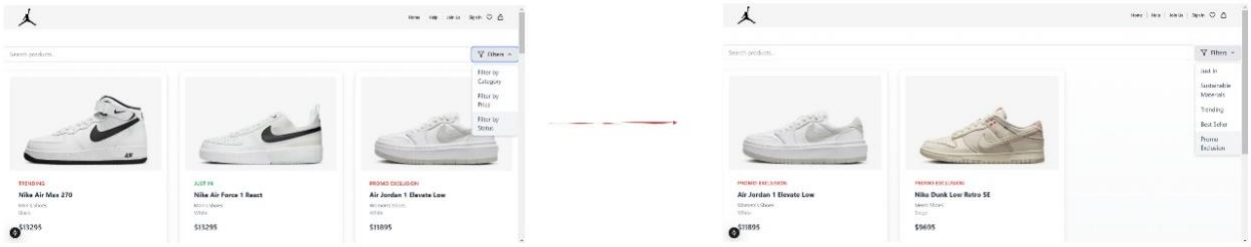
Best Practices Followed

- Modular design with reusable components for scalability.
- TypeScript for type safety and error prevention.
- Mobile-first, responsive design principles.
- Local storage to persist user data.
- Optimized API calls using caching techniques.

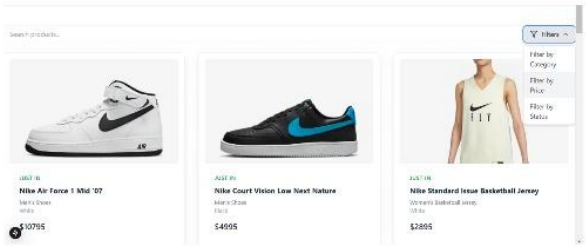
Filter by Category

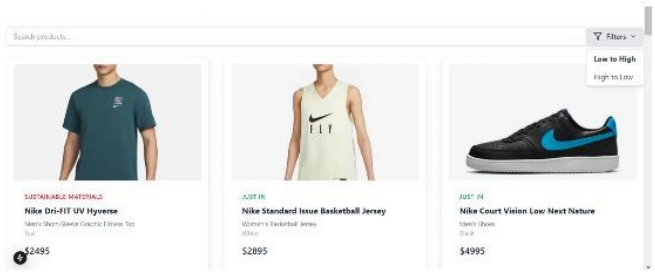


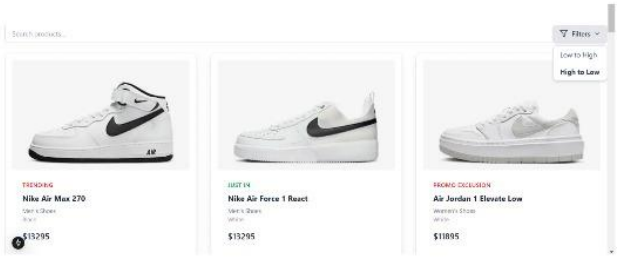
Filter by Status



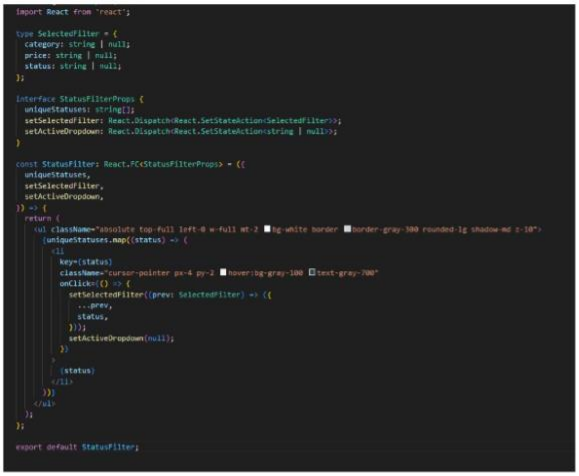
Filter by price





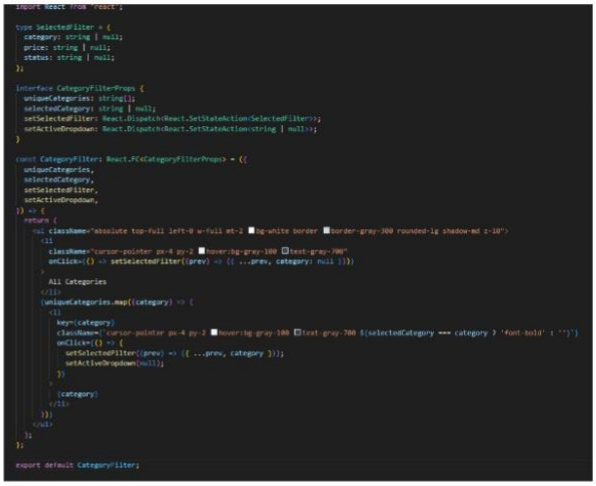


Filter by Status



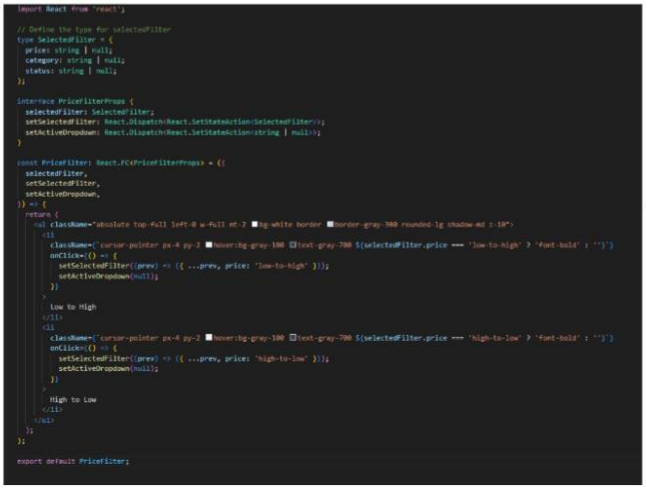
```tsx
import React from 'react';

type SelectedFilter = {
  category: string | null;
  price: string | null;
  status: string | null;
};

interface StatusFilterProps {
  uniqueStatuses: string[];
  setSelectedFilter: React.Dispatch<React.SetStateAction<SelectedFilter>>;
  setActiveDropdown: React.Dispatch<React.SetStateAction<string | null>>;
}

const StatusFilter: React.FC<StatusFilterProps> = ({
  uniqueStatuses,
  setSelectedFilter,
  setActiveDropdown,
}) => {
  return (
    <ul className="absolute top-full left-0 w-full mt-2 bg-white border border-gray-300 rounded-lg shadow-md z-10">
      {uniqueStatuses.map((status) => (
        <li
          key={status}
          className="cursor-pointer px-4 py-2 hover:bg-gray-100 text-gray-700"
          onClick={() => {
            setSelectedFilter((prev: SelectedFilter) => ({
              ...prev,
              status,
            }));
            setActiveDropdown(null);
          }}
        >
          {status}
        </li>
      ))}
    </ul>
  );
};

export default StatusFilter;
```

Filter by Category



```tsx
import React from 'react';

type SelectedFilter = {
  category: string | null;
  price: string | null;
  status: string | null;
};

interface CategoryFilterProps {
  uniqueCategories: string[];
  selectedCategory: string | null;
  setSelectedFilter: React.Dispatch<React.SetStateAction<SelectedFilter>>;
  setActiveDropdown: React.Dispatch<React.SetStateAction<string | null>>;
}

const CategoryFilter: React.FC<CategoryFilterProps> = ({
  uniqueCategories,
  selectedCategory,
  setSelectedFilter,
  setActiveDropdown,
}) => {
  return (
    <ul className="absolute top-full left-0 w-full mt-2 bg-white border border-gray-300 rounded-lg shadow-md z-10">
      <li
        className="cursor-pointer px-4 py-2 hover:bg-gray-100 text-gray-700"
        onClick={() => setSelectedFilter((prev) => ({ ...prev, category: null }))}
      >
        All Categories
      </li>
      {uniqueCategories.map((category) => (
        <li
          key={category}
          className={`cursor-pointer px-4 py-2 hover:bg-gray-100 text-gray-700 ${selectedCategory === category ? 'font-bold' : ''}`}
          onClick={() => {
            setSelectedFilter((prev) => ({ ...prev, category }));
            setActiveDropdown(null);
          }}
        >
          {category}
        </li>
      ))}
    </ul>
  );
};

export default CategoryFilter;
```

Filter by Price

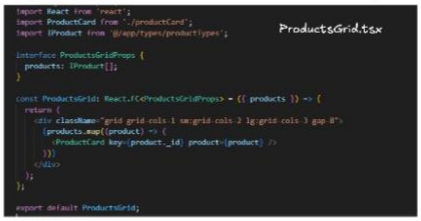


```tsx
import React from 'react';

// Define the type for selectedFilter
type SelectedFilter = {
  price: string | null;
  category: string | null;
  status: string | null;
};

interface PriceFilterProps {
  selectedFilter: SelectedFilter;
  setSelectedFilter: React.Dispatch<React.SetStateAction<SelectedFilter>>;
  setActiveDropdown: React.Dispatch<React.SetStateAction<string | null>>;
}

const PriceFilter: React.FC<PriceFilterProps> = ({
  selectedFilter,
  setSelectedFilter,
  setActiveDropdown,
}) => {
  return (
    <ul className="absolute top-full left-0 w-full mt-2 bg-white border border-gray-300 rounded-lg shadow-md z-10">
      <li
        className={`cursor-pointer px-4 py-2 hover:bg-gray-100 text-gray-700 ${selectedFilter.price === 'low-to-high' ? 'font-bold' : ''}`}
        onClick={() => {
          setSelectedFilter((prev) => ({ ...prev, price: 'low-to-high' }));
          setActiveDropdown(null);
        }}
      >
        Low to High
      </li>
      <li
        className={`cursor-pointer px-4 py-2 hover:bg-gray-100 text-gray-700 ${selectedFilter.price === 'high-to-low' ? 'font-bold' : ''}`}
        onClick={() => {
          setSelectedFilter((prev) => ({ ...prev, price: 'high-to-low' }));
          setActiveDropdown(null);
        }}
      >
        High to Low
      </li>
    </ul>
  );
};

export default PriceFilter;
```

products/page.tsx

FilteredProduct.tsx

ProductsGrid.tsx



ProductCard.tsx

ProductDetails.tsx