

AMRAHA ANWAR

ROLL NO: 307318

SLOT: FRIDAY MORNING (9AM TO 12PM)

HACKATHON 3 –DAY 4

Dynamic Frontend Components- Comforty Ecommerce Marketplace

Day-4 documentation overview:

In day-4 we had to make some dynamic components, the components make our work easy, understandable, reusable and maintainable.

In this day we also had to add some basic functionalities, which are required in an Ecommerce marketplace.

The Components I developed:

I have developed the following components and will upgrade and add more later :

- Hero/ Header Component.
- Footer Component
- Add To Cart Component
- Checkout Component
- Search Bar Component
- Reviews and Ratings Component
- Wishlist Component
- Product Listing Component
- Dynamic Product Detail Component
- Inventory Component
- Shipment /Tracking Component
- Category Component
- Notifications Component
- Multi-Language Support Component
- FAQ component (only UI)
- Advanced Search Component

Functional Deliverable:

Screenshots of Functional deliverables :

Product Listing Page

All Products



Citrus Edge
\$20



Scandi Dip Set
\$40



Rose Luxe Armchair
\$20



Modern Cozy
\$20



Library Stool Chair
\$20



SleekSpin
\$20



Nordic Spin
\$30




Gray Elegance
\$8

Dynamic Product Detail page:

Sales


✓ Free Shipping On All Orders Over \$50


Eng ▾ Faqs ⓘ Need Help

 Comforty

Search products...

Search

 0

 0

Home Shop Product About Contact

Contact: (808) 555-0111

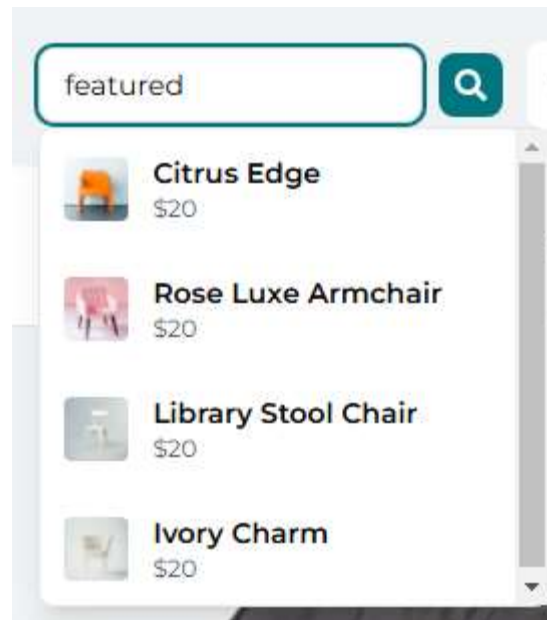


Rose Luxe Armchair

\$20.00 USD ~~\$30.00 USD~~

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

Advance Search Bar:



Code Deliverables:

Search Bar Component

```

/Search.tsx

"use client";

import { useRouter } from "next/navigation";
import { useState } from "react";
import { FaSearch } from "react-icons/fa"; // Search icon
import { ImSpinner8 } from "react-icons/im"; // Loading spinner

export default function Search() {
  const [searchTerm, setSearchTerm] = useState("");
  const [searchResults, setSearchResults] = useState<any[]>([]);
  const [isLoading, setIsLoading] = useState(false);
  const [noResults, setNoResults] = useState(false); // State to track no results
  const router = useRouter();

  // Handle search
  const handleSearch = async () => {
    if (!searchTerm.trim()) return; // Ignore empty search terms

    setIsLoading(true);
    setNoResults(false); // Reset no results state
    try {
      const response = await fetch(`/api/search?q=${encodeURIComponent(searchTerm)}`);
      const data = await response.json();

      if (data.length === 0) {
        setNoResults(true); // No matching products found
      } else {
        setNoResults(false); // Reset no results state
      }

      setSearchResults(data);
    } catch (error) {
      console.error("Error searching products:", error);
      setNoResults(true); // Show error message if search fails
    } finally {
      setIsLoading(false);
    }
  };

  // Handle "Enter" key press
  const handleKeyPress = (e: React.KeyboardEvent) => {
    if (e.key === "Enter") {
      handleSearch();
    }
  };

  // Redirect to product detail page when a product is selected
  const handleProductClick = (slug: string) => {
    router.push(`/product/${slug}`);
    setSearchTerm(""); // Clear the search term
    setSearchResults([]); // Clear the search results
  };

  return (
    <div className="relative">
      <div className="flex items-center">
        { /* Compact Search Input */ }
        <input
          type="text"
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          onKeyDown={handleKeyPress} // Trigger search on "Enter"
          placeholder="Search..."
          className="w-32 sm:w-48 px-3 py-2 rounded-lg border border-gray-300 focus:outline-none
focus:ring-2 focus:ring-customDarkBlue text-sm"
        />
        { /* Search Button */ }
        <button
          onClick={handleSearch}
          className="ml-2 p-2 bg-customDarkBlue text-white rounded-lg flex items-center justify-center"
        >
          {isLoading ? (
            <ImSpinner8 className="animate-spin w-4 h-4" /> // Loading spinner (smaller size)
          ) : (
            <FaSearch className="w-4 h-4" /> // Search icon (smaller size)
          )}
        </button>
      </div>

      { /* Display search results or no results message */ }
      {searchResults.length > 0 ? (
        <div className="absolute top-10 left-0 w-64 bg-white border border-gray-200 rounded-lg shadow-
lg z-50 max-h-60 overflow-y-auto">
          {searchResults.map((product) => (
            <div
              key={product._id}
              onClick={() => handleProductClick(product.slug)} // Redirect to product detail page
              className="p-3 hover:bg-gray-100 cursor-pointer"
            >
              <div className="flex items-center gap-3">
                <img
                  src={product.imageUrl}
                  alt={product.title}
                  className="w-8 h-8 object-cover rounded"
                />
                <div>
                  <h3 className="font-semibold text-sm">{product.title}</h3>
                  <p className="text-xs text-gray-600">{product.price}</p>
                </div>
              </div>
            </div>
          )
        )}
        </div>
      ) : noResults ? (
        <div className="absolute top-10 left-0 w-64 bg-white border border-gray-200 rounded-lg shadow-
lg z-50 p-3">
          <p className="text-gray-600 text-sm">This product doesn't exist.</p>
        </div>
      ) : null}
    </div>
  );
}

```



Product Detail component

Dynamic Product Detail Component:

```
import { client } from "@sanity/lib/client";
import ProductDetails from "@components/ProductDetails";

async function getData(slug: string) {
  const query = `*[ _type == "products" && slug.current == "${slug}" ][0]{
    _id,
    title,
    price,
    priceWithoutDiscount,
    description,
    "image": image.asset->,
    "slug": slug.current,
    price_id,
    inventory,
    badge,
    reviews[]->{
      name,
      rating,
      comment,
      date
    }
  }`;
  const data = await client.fetch(query);
  return data;
}

export const dynamic = "force-dynamic";

export default async function ProductDetailPage({
  params,
}: {
  params: { slug: string };
}) {
  const data = await getData(params.slug);

  if (!data) {
    return <div>Product not found</div>;
  }

  return (
    <div className="max-w-screen-2xl mx-auto overflow-x-hidden px-5 sm:px-10 md:px-10 lg:px-28 py-20">
      <ProductDetails product={data} />
    </div>
  );
}
```

Best Practices Followed:

- **Meaningful Variables**
- **Meaningful Function Names**
- **Everything is separated in components to maintain reusability and understandable**
- **Added comments to make code understandable**
- **Refactored the code to give it a clean look**

Conclusion:

In this day I read the whole document, understand it completely then I separated each code into components, made them reusable, readable and SEO friendly. Then I added some required functionalities and ended here by explaining everything in my document with screenshot proofs.

