Day 4 - Dynamic Frontend Components - General Ecommerce

Day 4, was about building dynamic frontend components that pull data from Sanity CMS or APIs to display marketplace information. The focus was on creating modular, reusable components and learning best practices for building scalable, responsive web applications that are ready for real-world scenarios.

Technical Report:

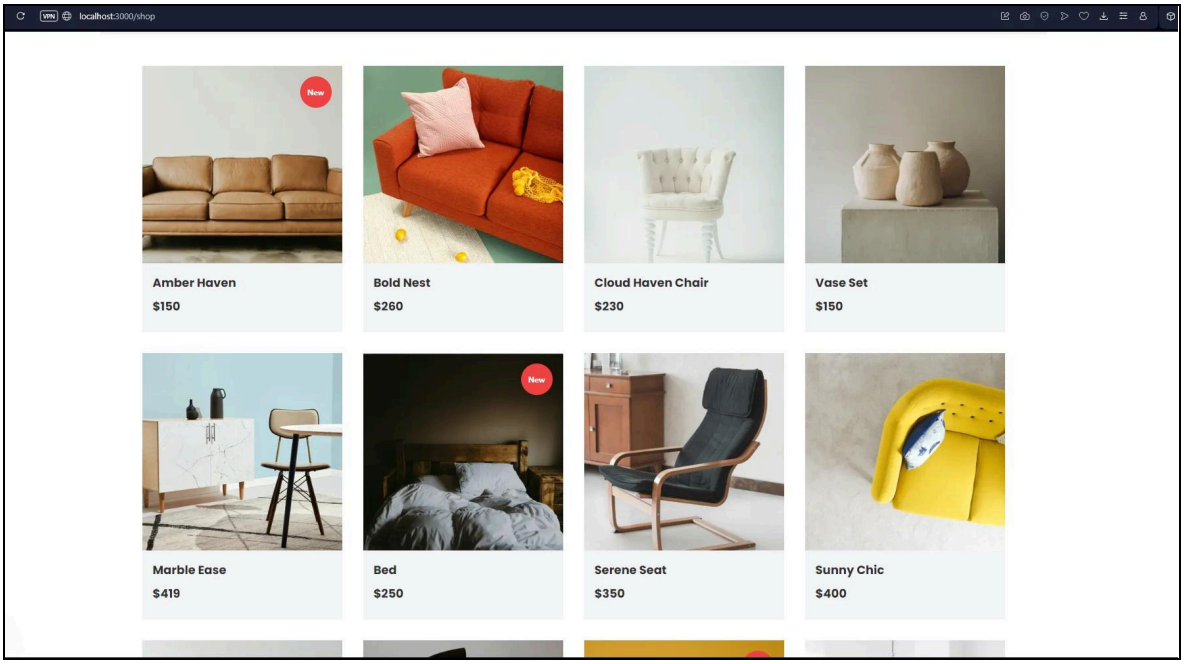
1. **Steps Taken to Build and Integrate Components:**
   The initial step in the development process was to create a component for the product card, which is displayed on the product's page. To achieve this, I used props to pass the fetched data from Sanity and display it dynamically within the card component. This approach allowed the product details to be rendered based on the data retrieved from the backend.

2. **Challenges Faced and Solutions Implemented:**
   One of the main challenges encountered in this specific day was the creation of a dynamic product detail page. To solve this, I utilized the product ID from the URL parameters, which allowed me to compare it against the data fetched from Sanity. If the ID matched, the corresponding product data would be displayed. This solution ensured that the correct product details were rendered based on the user’s selection.

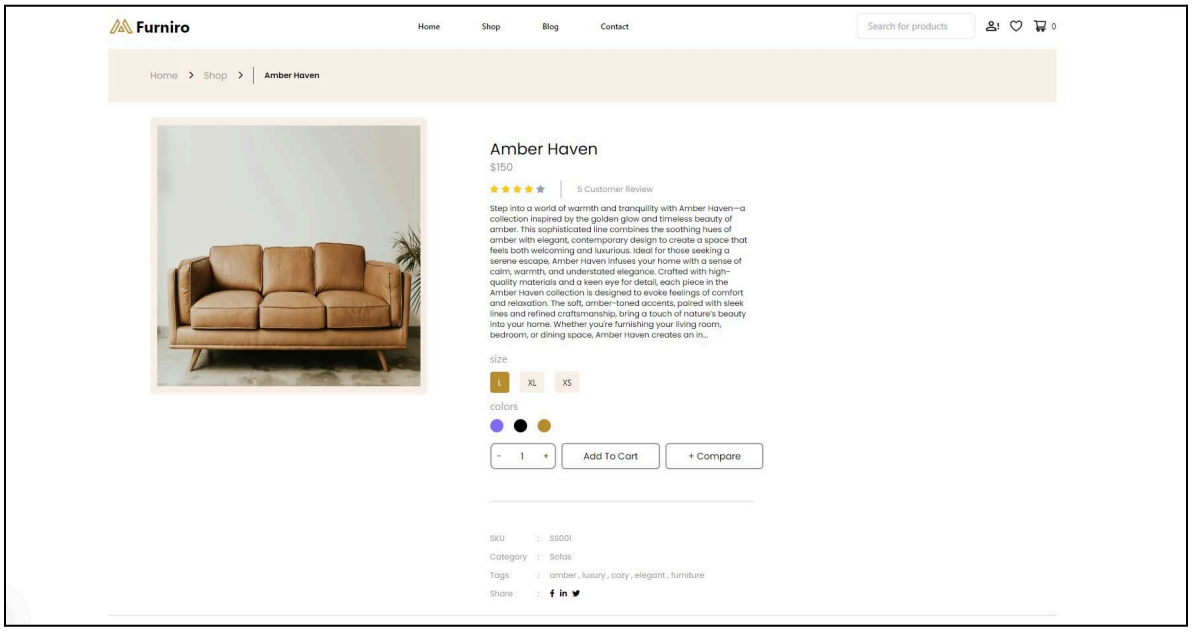
3. **Best Practices Followed:**
   Throughout the development process, I adhered to several best practices to ensure code maintainability and clarity. I included comments within the code to enhance its readability and make it easier to understand for future developers. Additionally, I created a separate component for the product card to maintain modularity and reusability. I also implemented error handling mechanisms to manage potential issues that could arise during data fetching or component rendering. These practices helped in making the application more robust and scalable.

Functional Deliverables (Screenshots):

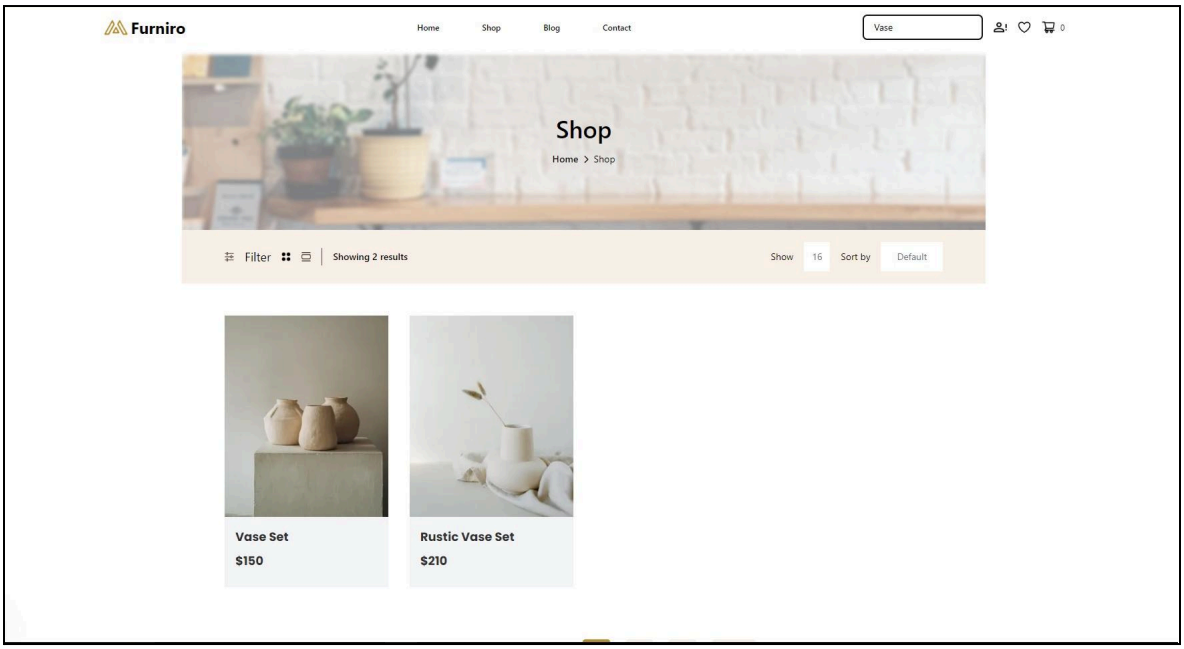
- The product listing page with dynamic data.



- Individual product detail pages with accurate routing and data rendering.



- Working Search Bar.



Code Deliverables:

- **Code snippet for product list.**

```
export default function Shop() {
const query = `*[_type == "product"][0...16]{
    title,
    _id,
    isNew,
    productImage {
      asset -> {
        url
      }
    },
    price
  }`;

  useEffect(() => {
    async function fetchProducts() {
      const data = await client.fetch(query);
      setProducts(data);
```

```
    }

    fetchProducts();
  }, []);
 {/* Products */}
      return (
 <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3
lg:grid-cols-4 gap-8 px-16 py-12">
        {filteredProducts.map((detail) => (
          <ProductCard key={detail._id} product={detail}
handleAdd={handleAdd} />
        ))}
      </div>);
}
```

- **Code snippet for Search bar.**

```
'use client';
// context/SearchContext.tsx
import React, { createContext, useState, useContext, ReactNode }
from 'react';

interface SearchContextType {
  searchTerm: string;
  setSearchTerm: (term: string) => void;
}

const SearchContext = createContext<SearchContextType |
undefined>(undefined);

export const SearchProvider: React.FC<{children: ReactNode}> = ({
children }) => {
  const [searchTerm, setSearchTerm] = useState('');

  return (
    <SearchContext.Provider value={{ searchTerm, setSearchTerm }}>
      {children}
    </SearchContext.Provider>
  );
};
```

```
export const useSearch = (): SearchContextType => {
  const context = useContext(SearchContext);
  if (!context) {
    throw new Error('useSearch must be used within a
SearchProvider');
  }
  return context;
};
```

- **Code snippet for ProductCard.**

```
const ProductCard = ( { product, handleAdd }: ProductCardProps ) =>
{
  return (
    <div className="sm:w-full w-[246px] h-[346px] sm:h-[406px]
bg-[#F4F5F7] flex flex-col relative">
      {/* Image Section with Hover Effect */}
      <div className="w-full h-[301px] relative group">
        <Image
          src={product.productImage.asset.url}
          alt="Product Image"
          width={285}
          height={301}
          className="object-cover w-full h-full"
        />
        {/* New Product Tag */}
        {product.isNew && (
          <p className="bg-red-500 text-white rounded-full w-12 h-12
flex items-center justify-center absolute top-4 right-4 text-xs
font-bold">
            New
          </p>
        )}
        {/* Overlay on hover only on image */}
        <div className="absolute inset-0 bg-black bg-opacity-50
opacity-0 group-hover:opacity-100 transition-opacity z-10"></div>
        {/* Hover Content (Button & Icons) */}
```

```
        <div className="absolute inset-0 flex justify-center
items-center opacity-0 group-hover:opacity-100 transition-opacity
z-20">
          <div className="text-center">
            <button onClick={() => handleAdd(product)}
className="bg-white sm:text-lg text-sm text-[#B88E2F] font-semibold
py-2 px-8 rounded-md mb-4">
              Add to Cart
            </button>
            <div className="flex sm:gap-x-8 gap-x-6 justify-center">
              <div className="flex items-center gap-x-1">
                <i className="fas fa-share-alt text-white"></i>
                <p className="sm:text-sm text-xs
text-white">Share</p>
              </div>
              <Link href={`/comparison`}>
                <div className="flex items-center gap-x-1">
                  <i className="fas fa-exchange-alt text-white"></i>
                  <p className="sm:text-sm text-xs
text-white">Compare</p>
                </div>
              </Link>
              <div className="flex items-center gap-x-1">
                <i className="fas fa-heart text-white"></i>
                <p className="sm:text-sm text-xs
text-white">Like</p>
              </div>
            </div>
          </div>
        </div>

      {/* Product Details */}
      <div className="p-4 sm:space-y-2 space-y-1 flex-grow">
        <Link href={`/shop/${product._id}`}>
          <h2 className="text-[#333333] font-bold sm:text-lg text-sm
font-poppins">
            {product.title}
          </h2>
        </Link>
```

```
        <p className="text-[#898989] font-poppins sm:text-sm
text-xs"></p>
        <div className="flex items-center gap-x-4">
          <h2 className="text-[#333333] font-bold sm:text-lg text-sm
font-poppins">
            ${product.price}
          </h2>
          <p className="text-[#898989] font-poppins text-sm
line-through"></p>
        </div>
      </div>
    </div>
  );
};

export default ProductCard;
```