Day 4:

Marketplace Name: Women's Fashion E-Commerce

Documentation

Technical Report

This report summarizes the steps, challenges, and best practices followed during the development of dynamic frontend components for the marketplace project using **Next.js** and **Sanity CMS**.

Steps Taken to Build and Integrate Components

1. **Setting Up the Development Environment**
    - Initialized a Next.js project and configured the required dependencies, including `@sanity/client` for API integration.
    - Connected the project to the Sanity CMS
2. **Data Fetching and Integration**
    - Implemented GROQ queries to fetch data dynamically from Sanity CMS.
    - Tested API endpoints to ensure correct responses and data availability.
3. **Component Development**
    - **Product Listing Component**: Displayed products dynamically in a grid layout with details like name, price, image, and stock status.
    - **Product Detail Component**: Used dynamic routing (`/product/[id]`) to render detailed product information fetched based on unique IDs.
    - **Search Bar**: Built a search functionality that filters products by name or tags.
    - **Cart Component**: Implemented a stateful component to track and display added items, quantities, and total price.
4. **Styling and Responsiveness**
    - Used **Tailwind CSS** for styling components, ensuring responsiveness across devices.
    - Applied modern UX principles for an intuitive user interface.
5. **State Management**
    - Managed local component state using `useState`.
    - Used `useContext` to manage global state for the cart and user preferences.
6. **Testing and Debugging**

- Tested components manually by navigating through different routes and interacting with UI elements.

Challenges Faced and Solutions Implemented

1. **Data Not Displaying Properly**
   - **Challenge**: Products and categories were not rendering as expected in some components.
   - **Cause**: Incorrect GROQ queries or improperly mapped data fields.
   - **Solution**: Reviewed GROQ queries and compared them against the Sanity schema to ensure correct mappings. Logged fetched data to debug issues.
2. **Data Migration Issues**
   - **Challenge**: Data from external sources was not migrating properly into Sanity CMS.
   - **Cause**: Data fields did not align with the predefined schema in Sanity.
   - **Solution**: Adjusted the data structure of the external source to match the schema. Used Sanity's import tools and scripts to validate data before migration.
3. **TypeScript Errors**
   - **Challenge**: Type errors occurred frequently when working with data fetched from Sanity CMS.
   - **Cause**: Missing or incorrectly defined TypeScript types for the data.
   - **Solution**: Defined proper TypeScript interfaces and used them across components. Ensured all fields from Sanity were explicitly typed to avoid runtime errors.

Best Practices Followed During Development

1. **Reusable Component Design**
   - Developed modular components, such as `ProductCard` and `CategoryFilter`, to ensure flexibility and reusability.
   - Passed data via props, adhering to the principle of single-responsibility.
2. **Consistent State Management**
   - Used React's `useState` and `useContext` hooks to maintain a clear separation between local and global states.
3. **Responsive and Accessible Design**
   - Styled components using **Tailwind CSS**, ensuring responsiveness for mobile and desktop views.
   - Implemented accessibility features, such as keyboard navigation and ARIA roles.
4. **Error Handling**
   - Included error boundaries to catch and handle unexpected issues gracefully.
   - Logged errors when fetching data and displayed appropriate fallback UI.
5. **Documentation and Code Organization**

- Maintained clear file structures, separating components, hooks, and utility functions for better scalability.
- Added comments in code to explain complex logic and ensure easier maintenance.

Working Search Bar Component:

```
// src/components/SearchBar.tsx
import { useState } from 'react';
import { useRouter } from 'next/navigation';
import { fetchAllProducts } from '@/sanity/lib/sanityClient'; // Adjust the
import based on your project structure
import { BsSearch } from 'react-icons/bs';

const SearchBar = () => {
  const [query, setQuery] = useState('');
  const router = useRouter();

  const handleSearch = async (e: React.FormEvent) => {
    e.preventDefault();
    if (!query) return;

    try {
      // Fetch all products from Sanity
      const products = await fetchAllProducts();

      // Find the product that matches the query
      const matchedProduct = products.find(product =>
        product.name.toLowerCase().includes(query.toLowerCase())
      );

      if (matchedProduct) {
        // Redirect to the product detail page using its ID
        router.push(`/product/${matchedProduct._id}`);
      } else {
        alert('No products found');
      }
    } catch (error) {
      console.error("Error fetching products:", error);
      alert('An error occurred while searching for products.');
    }
  };

  return (
    <form onSubmit={handleSearch} className='w-full sm:max-w-md relative'>
```

```
      <input
        type="text"
        placeholder="Search products..."
        value={query}
        onChange={(e) => setQuery(e.target.value)}
        className="w-full px-4 py-2 pl-10 border border-gray-200 rounded-full
        focus:outline-none focus:ring-2 focus:ring-pink-200
        text-sm transition-all duration-300
        hover:border-pink-100"
      />
      <BsSearch
        className='absolute left-3 top-1/2 transform -translate-y-1/2
        text -gray-400 hover:text-pink-500 transition-colors'
        size={22}
      />
    </form>
  );
};

export default SearchBar;
```

Product Details Component:

```
// src/components/ProductDetailClient.tsx
"use client"

import { ProductType } from '@/sanity/schemaTypes/productType';
import Image from 'next/image';
import { useCart } from '@/contexts/CartContext';

export default function ProductDetailClient({ product }: { product: ProductType
}) {
  const { addToCart } = useCart();

  const handleAddToCart = (product: ProductType) => {
    addToCart({
      id: product._id,
      name: product.name,
      price: product.price,
      image: product.image
    });
  };

  return (
    <div className="container mx-auto px-4 sm:px-6 lg:px-12 py-6 md:py-10">
```

```
      <div className="grid grid-cols-1 md:grid-cols-2 gap-6 md:gap-10">
        {/* Image Section */}
        <div className="w-full aspect-square md:aspect-[4/3] relative rounded-lg
shadow-lg overflow-hidden">
          <Image
            src={product.image}
            alt={product.name}
            fill
            sizes="(max-width: 768px) 100vw, (max-width: 1200px) 50vw, 33vw"
            className="object-cover rounded-lg"
            priority
          />
        </div>

        {/* Product Info Section */}
        <div className="flex flex-col justify-center space-y-4">
          <h1 className="text-2xl md:text-3xl lg:text-4xl font-bold text-gray-
900">
            {product.name}
          </h1>
          <p className="text-base md:text-lg text-gray-700">
            {product.fulldesc}
          </p>

          {/* Price and Rating Section */}
          <div className="flex flex-col space-y-2">
            <span className="text-2xl md:text-3xl font-semibold text-accent">
              <span className='text-blackish'>$</span>{product.price}
            </span>
            <div className="flex items-center text-lg text-gray-500">
              {[...Array(5)].map((_, i) => (
                <span
                  key={i}
                  className={`text-xl ${i < product.rating ? 'text-yellow-500' :
'text-gray-300'}`}
                >
                  ★
                </span>
              ))}
              <span className="ml-2 text-base text-gray-600">
                ({product.rating}/5)
              </span>
            </div>
          </div>
```

```
          {/* Add to Cart Button */}
          <button
            onClick={() => handleAddToCart(product)}
            className="w-full md:w-auto bg-accent text-white px-6 md:px-8 py-2
md:py-3 rounded-lg
              hover:bg-blackish focus:outline-none focus:ring-2 focus:ring-pink-
500 focus:ring-opacity-50
              transition duration-300 text-base sm:flex-col md:text-lg"
          >
            Add to Cart
          </button>
        </div>
      </div>
    </div>
  );
}
```

Product List Page:

```
// src/components/ProductList.tsx
'use client';

import React, { useState, useEffect } from 'react';
import { ProductType } from '@/sanity/schemaTypes/productType';

import { useCart } from '../contexts/CartContext';
import { fetchProducts } from '../sanity/lib/sanityClient';
import Link from 'next/link';
import Image from 'next/image';

export default function ProductList() {
  const [products, setProducts] = useState<ProductType[]>([]);
  const [isLoading, setIsLoading] = useState(true);

  // Get the addToCart function from the cart context
  const { addToCart } = useCart();

  useEffect(() => {
    const getProducts = async () => {
      try {
        setIsLoading(true);
        const fetchedProducts = await fetchProducts();
        setProducts(fetchedProducts);
      } catch (error) {
        console.error('Error fetching products:', error);
```

```
      } finally {
        setIsLoading(false);
      }
    };

    getProducts();
  }, []);

  // Move handleAddToCart outside of useEffect
  const handleAddToCart = (product: ProductType) => {
    const cartProduct = {
      id: product._id,
      name: product.name,
      price: product.price,
      image: product.image,
    };
    addToCart(cartProduct);
  };

  if (isLoading) {
    return <div className="text-center py-10">Loading products...</div>;
  }

  return (
    <div className="container mx-auto px-4">
      <h2 className="
        text-center
        text-2xl
        sm:text-3xl
        mt-6
        md:text-4xl
        font-bold
        text-gray-800
        mb-8
        uppercase
        tracking-wider
      ">
        <span className='text-red-600'>━━</span>New Arrival's<span
className='text-red-600'> ━━</span>
      </h2>
    <div className="flex flex-wrap gap-4 p-4 w-full justify-center">
      {products.map((product) => (
        <div
          key={product._id}
          className="
```

```
            w-full
            sm:w-[calc(50%-1rem)]
            lg:w-[calc(33.333%-1rem)]
            xl:w-[calc(25%-1rem)]
            2xl:w-[calc(20%-1rem)]
            border
            border-gray-200
            rounded-xl
            overflow-hidden
            flex
            flex-col
            h-[450px]
            sm:h-[500px]
            md:h-[550px]
            transition-all
            duration-300
            ease-in-out
            hover:shadow-lg
          "
        >
          <Link href={`/product/${product._id}`} className="h-full flex flex-
col">
            <div className="relative w-full h-full">
              <Image
                src={product.image}
                alt={product.name}
                fill
                className="object-cover hover:scale-105 transition-transform
duration-300"
                sizes="(max-width: 640px) 100vw, (max-width: 768px) 50vw, (max-
width: 1024px) 33vw, 25vw"
                priority
              />
            </div>
            {/* Content Container */}
            <div className="p-4 flex-grow flex flex-col justify-between">
  <div>
    <h2 className="text-accent font-medium uppercase truncate text-sm sm:text-
base">
      {product.name}
    </h2>
    <p className="text-gray-500 line-clamp-2 text-xs sm:text-sm mb-2">
      {product.shortdesc}
    </p>
  </div>
```

```
  <div className="flex flex-col justify-between items-start">
    <span className="text-lg font-semibold">${product.price}</span>
    <div className="flex items-center">
      {[...Array(5)].map((_, i) => (
        <span
          key={i}
          className={`text-lg ${i < product.rating ? 'text-yellow-500' : 'text-
gray-300'}`}
        >
          ★
        </span>
      ))}
    </div>
    <button
  onClick={(e) => {
    e.preventDefault();
    handleAddToCart(product);
  }}
  className='
    min-w-[100px]  // Minimum width
    inline-block   // Prevents full width
    bg-accent
    hover:bg-accent-dark
    transition-all
    duration-300
    ease-in-out
    text-white
    py-2
    px-4
    rounded-lg
    text-sm
    font-medium
    active:scale-95
    hover:shadow-md
    transform
    hover:-translate-y-0.5
    focus:outline-none
    focus:ring-2
    focus:ring-accent-light
    text-center  // Ensure text is centered
  '
>
  Add to Cart
</button>
```

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