Day 4 - Code Snippets

1. ProductCard Component

This component displays an individual product with its image, title, and price.

```
// components/ProductCard.tsx
import Image from "next/image";
import Link from "next/link";

export default function ProductCard({ product }: { product: any }) {
  return (
    <Link href={`/product/${product.slug.current}`} className="block
bg-white rounded-lg shadow-md overflow-hidden">
      <div className="relative h-64 w-full">
        <Image
          src={product.thumbnail || "/placeholder.svg"}
          alt={product.title}
          fill
          className="object-cover"
        />
      </div>
      <div className="p-4">
        <h3 className="text-lg font-semibold">{product.title}</h3>
        <p className="text-primary
font-bold">${product.price.toFixed(2)}</p>
      </div>
    </Link>
  );
}
```

2. ProductList Component

This component maps through an array of products and renders a grid of `ProductCard` components.

```
// components/ProductList.tsx
import ProductCard from "./ProductCard";

export default function ProductList({ products }: { products: any[] })
{
  return (
    <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3
lg:grid-cols-4 gap-6">
      {products.map((product) => (
        <ProductCard key={product._id} product={product} />
      ))}
    </div>
  );
}
```

3. SearchBar Component

A simple search bar component that triggers a callback with the search query.

tsx

CopyEdit

```
// components/SearchBar.tsx
import { useState } from "react";

export default function SearchBar({ onSearch }: { onSearch: (query:
string) => void }) {
  const [query, setQuery] = useState("");

  const handleSearch = (e: React.FormEvent) => {
    e.preventDefault();
    onSearch(query);
  };

  return (
    <form onSubmit={handleSearch} className="flex items-center">
      <input
        type="text"
        placeholder="Search products..."
        value={query}
        onChange={(e) => setQuery(e.target.value)}
        className="border border-gray-300 p-2 rounded-l"
      />
      <button type="submit" className="bg-primary text-white p-2
rounded-r">
        Search
      </button>
    </form>
  );
}
```

---

4. Dynamic Routing & API Integration

a. Category Page with Dynamic Routing

This page uses dynamic routing to display products for a given category.

```
// app/category/[categorySlug]/page.tsx
import { client } from "@/sanity/lib/client";
import ProductList from "@/components/ProductList";
import { notFound } from "next/navigation";

export default async function CategoryPage({ params: { categorySlug }
}: { params: { categorySlug: string } }) {
  // Fetch category details
  const categoryQuery = `
    *[_type == "category" && slug.current == $slug][0]{
      _id,
      title
    }
  `;
  const category = await client.fetch(categoryQuery, { slug:
categorySlug });
  if (!category) return notFound();

  // Fetch products in the category
  const productsQuery = `
    *[_type == "product" && category->slug.current == $slug]{
      _id,
      title,
      slug,
      price,
      "thumbnail": thumbnail.asset->url
    }
  `;
  const products = await client.fetch(productsQuery, { slug:
categorySlug });

  return (
    <main className="container mx-auto px-4 py-8">
      <h1 className="text-3xl font-bold mb-6">{category.title}</h1>
      <ProductList products={products} />
    </main>
  );
}
```

This document provides a snapshot of key components and scripts used for API integration and dynamic routing within the project. Each snippet is designed to be modular, making it easy to maintain and extend as needed.