Technical Report: Dynamic Routing and Modular Component Design in React and Next.js

Objective

This project demonstrates the implementation of a dynamic routing system for products in a Next.js application. The routing is based on product titles fetched from **Sanity CMS**, and the project emphasizes modularity and usability through reusable components such as ProductCard, SearchBar, and pagination handling in the main Page component.

Key Features and Design Decisions

1. Dynamic Routing for Product Details

Implementation

- The route for each product is dynamically generated using the product's title as a key. This ensures unique URLs for each product, improving SEO and user experience.

Code snippet for dynamic routing: (Appendix 1)

Key Functionality

- **Fetching Data**: Uses Sanity's GROQ query to fetch product details based on the title parameter passed in the URL.
- **Fallback Handling**: Displays a 404 page if the product is not found, ensuring a user-friendly error state.
- **Dynamic Rendering**: Product details, such as title, description, price, tags, and discount percentage, are rendered dynamically based on the fetched data.

---

2. Reusable Components

ProductCard

- A reusable component designed to display a preview of product information in a grid layout.
- Highlights:
    - Displays product image, title, description, price, and discount.
    - Provides a "New" badge for new products.
    - Includes a modal that opens on clicking the card to show detailed product information.

Code snippet: (Appendix 2)

SearchBar

- A component to filter products based on the user's input.
- Highlights:
    - Filters products in real-time.
    - Returns filtered results to the parent component for rendering.

Code snippet: (Appendix 3)

---

3. Main Page with Pagination

- Displays a grid of ProductCard components along with pagination and other components such as Navbar and Footer.
- Highlights:
    - Fetches all products from Sanity CMS on load.
    - Implements pagination logic to divide the products into pages, each containing 8 items.
    - Integrates the SearchBar for filtering.

Code snippet: (Appendix 4)

Conclusion

This project successfully creates a scalable and user-friendly e-commerce interface. Dynamic routing ensures flexibility, while modular components like ProductCard and SearchBar make the codebase maintainable and extensible. The integration with Sanity CMS enables easy content management, making this implementation ideal for modern web applications.

Appendix - 1:

```tsx
import { client } from "@/sanity/lib/client";
import { notFound } from "next/navigation";

interface ProductDetailsProps {
  params: {
    title: string;
  };
}

export default async function ProductDetails({ params }: ProductDetailsProps) {
  const { title } = params;

  const product = await client.fetch(
    `*[_type == "product" && lower(title) == $title][0]{
      title,
      "imageUrl": productImage.asset->url,
      description,
      price,
      tags,
      discountPercentage,
      isNew
    }`,
    { title: title.toLowerCase() }
  );

  if (!product) {
    return notFound();
  }

  return (
    <div className="container mx-auto p-4">
      <div className="flex flex-col md:flex-row items-center">
        <img
          src={product.imageUrl || "/default-image.jpg"}  // Use fallback image
          alt={product.title}
          className="w-full md:w-1/2 rounded-lg shadow-lg"
        />
        <div className="md:ml-8 mt-4 md:mt-0">
          <h1 className="text-3xl font-bold">{product.title}</h1>
          <p className="text-lg mt-2">{product.description}</p>
          <p className="text-xl font-semibold mt-4">${product.price}</p>
          {product.discountPercentage && (
            <p className="text-green-500 mt-2">
              Discount: {product.discountPercentage}%
            </p>
          )}
          {product.isNew && <p className="text-blue-500 mt-2">New Arrival!</p>}
          {product.tags?.length > 0 && (
            <div className="mt-4">
              <h4 className="font-semibold">Tags:</h4>
              <ul className="flex space-x-2">
                {product.tags.map((tag: string, index: number) => (
                  <li key={index} className="bg-gray-200 px-2 py-1 rounded-lg">
                    {tag}
                  </li>
                ))}
              </ul>
            </div>
          )}
        </div>
      </div>
    </div>
  );
}

```

Appendix - 2:

```tsx
import React, { useState } from "react";
import Image from "next/image";

interface ProductCardProps {
  product: {
    title: string;
    description: string;
    price: number;
    productImage?: {
      asset?: {
        _id?: string;
        url?: string;
      };
    };
    discountPercentage?: number;
    isNew?: boolean;
  };
}

const ProductCard: React.FC<ProductCardProps> = ({ product }) => {
  const [isModalOpen, setIsModalOpen] = useState(false);

  const handleProductClick = () => {
    setIsModalOpen(true);
  };


  const imageUrl =
    product.productImage?.asset?.url || "/image-4.png";

  return (
    <>

      <div
        className="card border shadow-lg rounded-lg overflow-hidden cursor-pointer"
        onClick={handleProductClick}
      >

        <Image
          src={imageUrl}
          width={300}
          height={200}
          alt={product.title}
          className="object-cover w-full h-[200px]"
        />
        <div className="card-body p-4">

          <h5 className="card-title font-bold text-lg">{product.title}</h5>

          <p className="card-text text-gray-600">
            {product.description.substring(0, 80)}...
          </p>

          <h5 className="card-title font-bold text-lg mt-2 text-blue-500">
            {product.discountPercentage ? (
              <>
                Discounted: $
                {(
                  product.price *
                  (1 - product.discountPercentage / 100)
                ).toFixed(2)}
                <br />
                <span className="line-through text-gray-500">
                  Original: ${product.price}
                </span>
              </>
            ) : (
              `Price: $${product.price}`
            )}
          </h5>
          {product.isNew && (
            <span className="bg-green-500 text-white px-2 py-1 rounded-full inline-block mt-2">
              New
            </span>
          )}
        </div>
      </div>


      {isModalOpen && (
        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50">
          <div className="bg-white p-6 rounded shadow-lg overflow-y-auto max-h-screen relative w-[90%] md:w-[70%] lg:w-[50%]">

            <button
              className="absolute top-4 right-4 text-gray-500 hover:text-gray-700 bg-gray-200 rounded-full p-2"
              onClick={() => setIsModalOpen(false)}
            >
              &times;
            </button>

            <h2 className="text-2xl font-bold mb-4">{product.title}</h2>
            <Image
              src={imageUrl}
              width={500}
              height={400}
              alt={product.title}
              className="rounded mb-4 w-full h-[300px] object-cover"
            />
            <div className="mb-4">
              <h3 className="text-lg font-semibold mb-1">Description:</h3>
              <p>{product.description}</p>
            </div>
            <div className="mb-4">
              <h3 className="text-lg font-semibold mb-1">Price:</h3>
              <p>
                {product.discountPercentage ? (
                  <>
                    Discounted Price: $
                    {(
                      product.price *
                      (1 - product.discountPercentage / 100)
                    ).toFixed(2)}
                    <br />
                    Original Price: ${product.price}
                  </>
                ) : (
                  `$${product.price}`
                )}
              </p>
            </div>
            {product.isNew && (
              <div className="bg-green-500 text-white px-2 py-1 rounded-full inline-block">
                New
              </div>
            )}
          </div>
        </div>
      )}
    </>
  );
};

export default ProductCard;
```

Appendix - 3:

```tsx
import React from "react";

interface SearchBarProps {
  products: any[];
  onSearchResults: (results: any[]) => void;
}

const SearchBar: React.FC<SearchBarProps> = ({ products, onSearchResults }) => {
  const handleSearch = (query: string) => {
    const results = products.filter((product) =>
      product.title.toLowerCase().includes(query.toLowerCase())
    );
    onSearchResults(results);
  };

  return (
    <div className="flex items-center justify-center w-full">
      <div className="relative w-full max-w-xs">
        <div className="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none">
          <svg className="w-4 h-4 text-gray-500" fill="none" stroke="currentColor" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M21 21l-6-6m2-5a7 7 0 11-14 0"></path></svg>
        </div>
        <input
          type="text"
          placeholder="Search..."
          className="py-2 px-3 pl-10 rounded-md border border-gray-300 focus:ring-blue-500 focus:border-blue-500 w-full"
          onChange={(e) => handleSearch(e.target.value)}
        />
      </div>
    </div>
  );
};

export default SearchBar;
```

Appendix - 4:

```tsx
"use client"

import React, { useState, useEffect } from "react";
import { client } from "@/sanity/lib/client";
import Hero from "./Components/Hero";
import Sec_Hero from "./Components/Sec_Hero";
import ProductCard from "./Components/ProductCard";
import SearchBar from "./Components/SearchBar";
import Slide from "./Components/Slide";
import ShareSec from "./Components/ShareSec";

const Home = () => {
  const [products, setProducts] = useState<any[]>([]);
  const [filteredProducts, setFilteredProducts] = useState<any[]>([]);
  const [currentPage, setCurrentPage] = useState(1);
  const productsPerPage = 8;

  useEffect(() => {
    const fetchProducts = async () => {
      const data = await client.fetch(`*[_type == "product"]`);
      setProducts(data);
      setFilteredProducts(data);
    };

    fetchProducts();
  }, []);

  const handlePageChange = (page: number) => {
    setCurrentPage(page);
  };

  const indexOfLastProduct = currentPage * productsPerPage;
  const indexOfFirstProduct = indexOfLastProduct - productsPerPage;
  const currentProducts = filteredProducts.slice(indexOfFirstProduct, indexOfLastProduct);

  const totalPages = Math.ceil(filteredProducts.length / productsPerPage);

  return (
    <div>
      <Hero />
      <Sec_Hero />
      <SearchBar products={products} onSearchResults={setFilteredProducts} />
      <div className="container mx-auto mt-4">
        <div className="grid grid-cols-1 sm:grid-cols-3 lg:grid-cols-4 gap-4">
          {currentProducts.map((product) => (
            <ProductCard key={product.title} product={product} />
          ))}
        </div>
      </div>
      <div className="container mx-auto mt-8">
        <div className="flex justify-center mt-6 mb-6">
          <button
            onClick={() => handlePageChange(currentPage - 1)}
            disabled={currentPage === 1}
            className="px-4 py-2 bg-blue-500 text-white rounded disabled:bg-gray-400"
          >
            Previous
          </button>
          <span className="mx-4">
            Page {currentPage} of {totalPages}
          </span>
          <button
            onClick={() => handlePageChange(currentPage + 1)}
            disabled={currentPage === totalPages}
            className="px-4 py-2 bg-blue-500 text-white rounded disabled:bg-gray-400"
          >
            Next
          </button>
        </div>
      </div>
      <Slide />
      <ShareSec />
    </div>
  );
};

export default Home;
```