Day 4 - Dynamic Frontend Components - General Ecommerce

Functional Deliverables

1. Screenshots or Screen Recordings

Product Listing Page

- **Description**: The product listing page dynamically renders data fetched from Sanity CMS or APIs. It includes features like grid layout, product name, price, image, and stock status.
- **Screenshot/Screen Recording**: [Attach screenshot/screen recording here.]

Individual Product Detail Pages

- **Description**: Each product has its own detailed page accessed via dynamic routing in Next.js. The pages display fields like product description, price, available sizes, or colors.
- **Screenshot/Screen Recording**: [Attach screenshot/screen recording here.]

Category Filters, Search Bar, and Pagination

- **Category Filters**: Dynamically filters products by category.
- **Search Bar**: Filters products by name or tags.
- **Pagination**: Breaks down large product lists into manageable pages.
- **Screenshot/Screen Recording**: [Attach screenshot/screen recording here.]

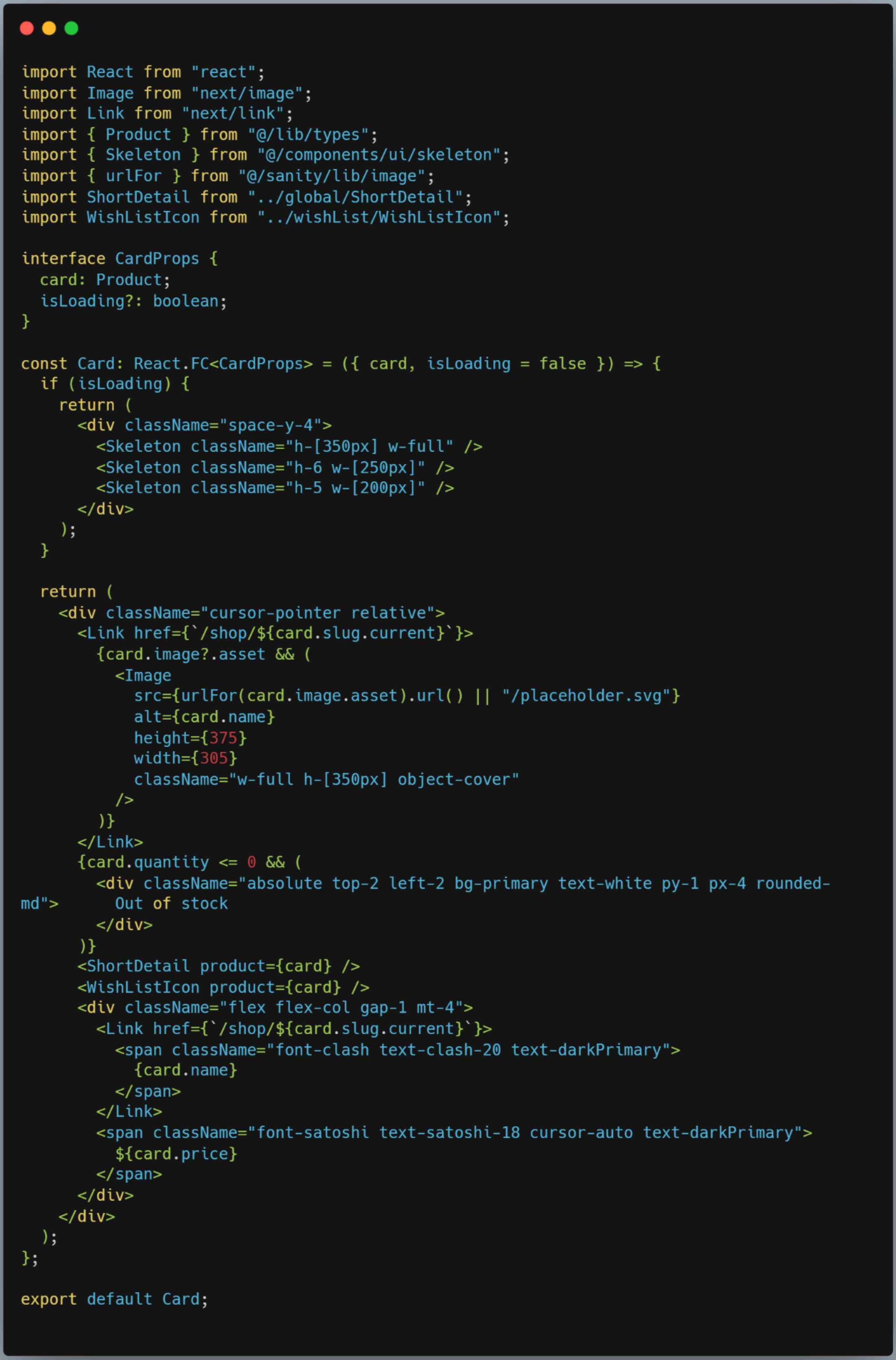
Additional Features

- **Related Products**: Displays similar or complementary products on the product detail page.
- **User Profile Components**: Shows user details, order history, and saved addresses.
- **Screenshot/Screen Recording**: [Attach screenshot/screen recording here.]

Code Deliverables

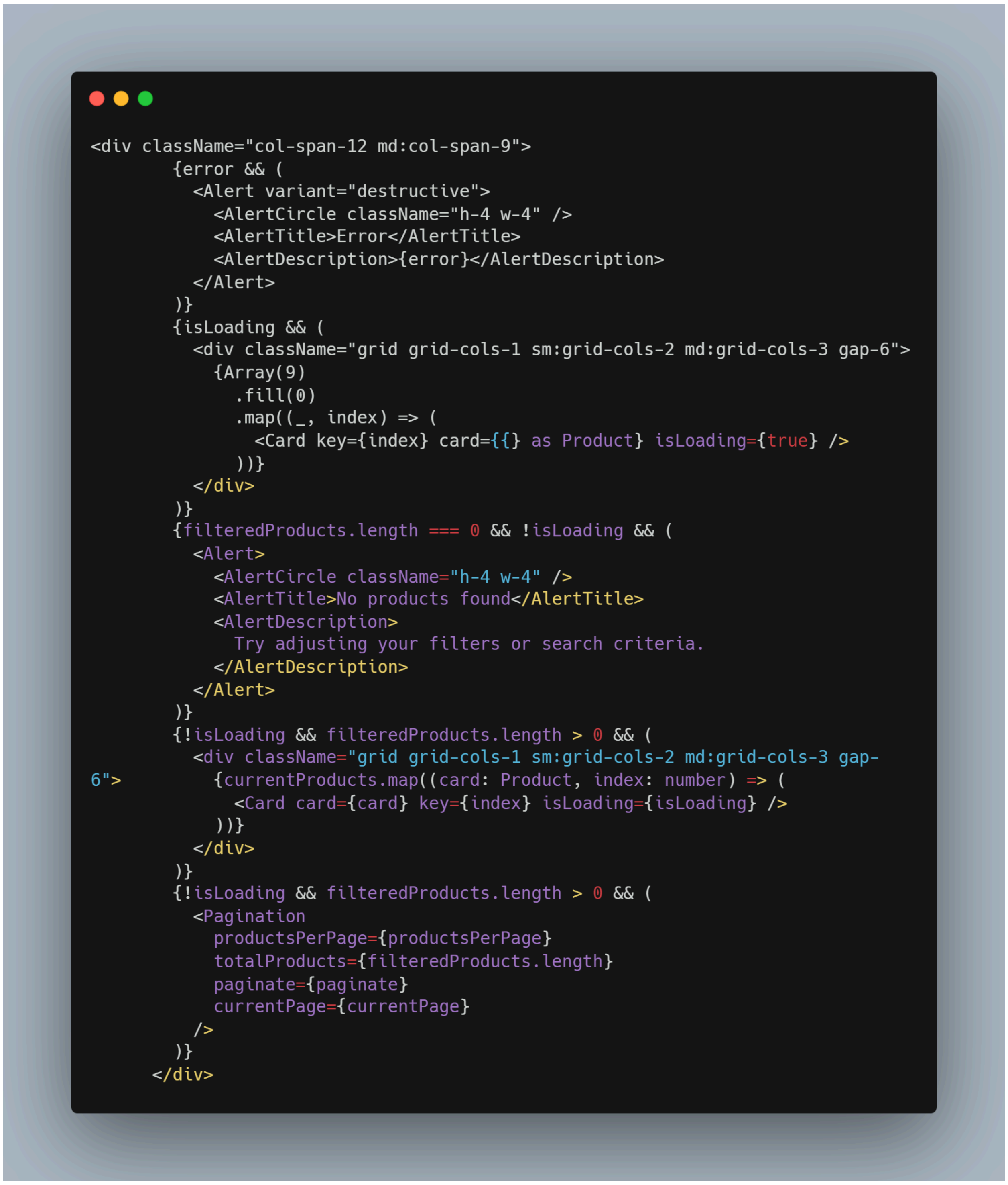
1. Key Component Code Snippets

ProductCard Component



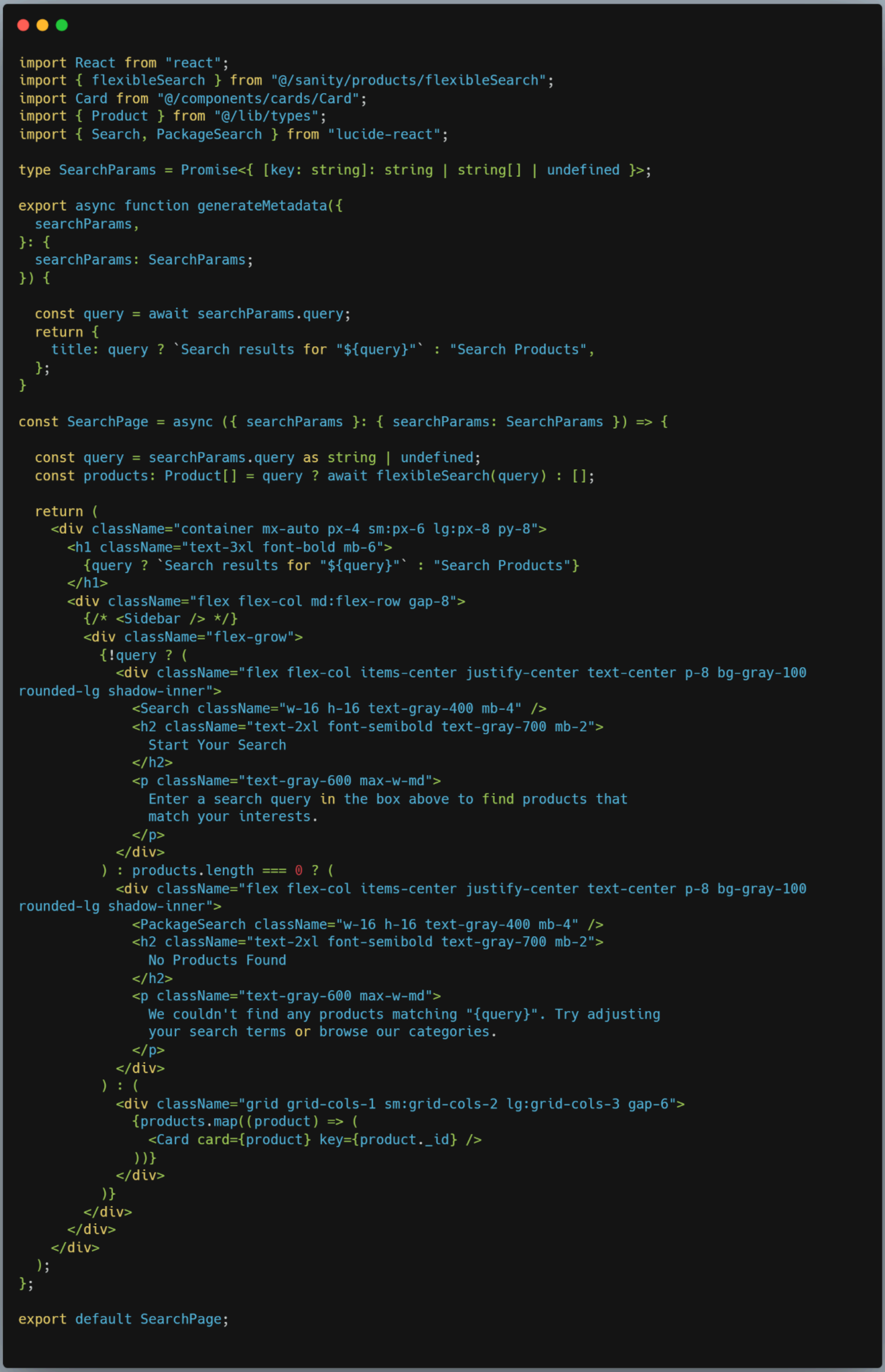
```tsx
import React from "react";
import Image from "next/image";
import Link from "next/link";
import { Product } from "@/lib/types";
import { Skeleton } from "@/components/ui/skeleton";
import { urlFor } from "@/sanity/lib/image";
import ShortDetail from "../global/ShortDetail";
import WishListIcon from "../wishList/WishListIcon";

interface CardProps {
  card: Product;
  isLoading?: boolean;
}

const Card: React.FC<CardProps> = ({ card, isLoading = false }) => {
  if (isLoading) {
    return (
      <div className="space-y-4">
        <Skeleton className="h-[350px] w-full" />
        <Skeleton className="h-6 w-[250px]" />
        <Skeleton className="h-5 w-[200px]" />
      </div>
    );
  }

  return (
    <div className="cursor-pointer relative">
      <Link href={`/shop/${card.slug.current}`}>
        {card.image?.asset && (
          <Image
            src={urlFor(card.image.asset).url() || "/placeholder.svg"}
            alt={card.name}
            height={375}
            width={305}
            className="w-full h-[350px] object-cover"
          />
        )}
      </Link>
      {card.quantity <= 0 && (
        <div className="absolute top-2 left-2 bg-primary text-white py-1 px-4 rounded-
md">      Out of stock
        </div>
      )}
      <ShortDetail product={card} />
      <WishListIcon product={card} />
      <div className="flex flex-col gap-1 mt-4">
        <Link href={`/shop/${card.slug.current}`}>
          <span className="font-clash text-clash-20 text-darkPrimary">
            {card.name}
          </span>
        </Link>
        <span className="font-satoshi text-satoshi-18 cursor-auto text-darkPrimary">
          ${card.price}
        </span>
      </div>
    </div>
  );
};

export default Card;
```

ProductList Component



```
<div className="col-span-12 md:col-span-9">
        {error && (
          <Alert variant="destructive">
            <AlertCircle className="h-4 w-4" />
            <AlertTitle>Error</AlertTitle>
            <AlertDescription>{error}</AlertDescription>
          </Alert>
        )}
        {isLoading && (
          <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-6">
            {Array(9)
              .fill(0)
              .map((_, index) => (
                <Card key={index} card={{} as Product} isLoading={true} />
              ))}
          </div>
        )}
        {filteredProducts.length === 0 && !isLoading && (
          <Alert>
            <AlertCircle className="h-4 w-4" />
            <AlertTitle>No products found</AlertTitle>
            <AlertDescription>
              Try adjusting your filters or search criteria.
            </AlertDescription>
          </Alert>
        )}
        {!isLoading && filteredProducts.length > 0 && (
          <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-
6">         {currentProducts.map((card: Product, index: number) => (
              <Card card={card} key={index} isLoading={isLoading} />
            ))}
          </div>
        )}
        {!isLoading && filteredProducts.length > 0 && (
          <Pagination
            productsPerPage={productsPerPage}
            totalProducts={filteredProducts.length}
            paginate={paginate}
            currentPage={currentPage}
          />
        )}
      </div>
```

SearchBar Component



```tsx
import React from "react";
import { flexibleSearch } from "@/sanity/products/flexibleSearch";
import Card from "@/components/cards/Card";
import { Product } from "@/lib/types";
import { Search, PackageSearch } from "lucide-react";

type SearchParams = Promise<{ [key: string]: string | string[] | undefined }>;

export async function generateMetadata({
  searchParams,
}: {
  searchParams: SearchParams;
}) {

  const query = await searchParams.query;
  return {
    title: query ? `Search results for "${query}"` : "Search Products",
  };
}

const SearchPage = async ({ searchParams }: { searchParams: SearchParams }) => {

  const query = searchParams.query as string | undefined;
  const products: Product[] = query ? await flexibleSearch(query) : [];

  return (
    <div className="container mx-auto px-4 sm:px-6 lg:px-8 py-8">
      <h1 className="text-3xl font-bold mb-6">
        {query ? `Search results for "${query}"` : "Search Products"}
      </h1>
      <div className="flex flex-col md:flex-row gap-8">
        {/* <Sidebar /> */}
        <div className="flex-grow">
          {!query ? (
            <div className="flex flex-col items-center justify-center text-center p-8 bg-gray-100
rounded-lg shadow-inner">
              <Search className="w-16 h-16 text-gray-400 mb-4" />
              <h2 className="text-2xl font-semibold text-gray-700 mb-2">
                Start Your Search
              </h2>
              <p className="text-gray-600 max-w-md">
                Enter a search query in the box above to find products that
                match your interests.
              </p>
            </div>
          ) : products.length === 0 ? (
            <div className="flex flex-col items-center justify-center text-center p-8 bg-gray-100
rounded-lg shadow-inner">
              <PackageSearch className="w-16 h-16 text-gray-400 mb-4" />
              <h2 className="text-2xl font-semibold text-gray-700 mb-2">
                No Products Found
              </h2>
              <p className="text-gray-600 max-w-md">
                We couldn't find any products matching "{query}". Try adjusting
                your search terms or browse our categories.
              </p>
            </div>
          ) : (
            <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6">
              {products.map((product) => (
                <Card card={product} key={product._id} />
              ))}
            </div>
          )}
        </div>
      </div>
    </div>
  );
};

export default SearchPage;
```

Documentation

Steps Taken

1. Fetched dynamic data from Sanity CMS and APIs.
2. Designed and implemented reusable components like `ProductCard`, `ProductList`, and `SearchBar`.
3. Integrated state management for the cart and wishlist features.
4. Implemented routing and pagination for seamless navigation.

Challenges Faced and Solutions

- **Challenge**: Managing state across multiple components.
    - **Solution**: Used React's Context API for global state management.
- **Challenge**: Implementing responsive design.
    - **Solution**: Utilized CSS Flexbox and Grid for adaptive layouts.
- **Challenge**: Fetching data dynamically.
    - **Solution**: Used Axios for API calls and optimized data fetching using server-side rendering.

Best Practices Followed

- Modular component design for reusability.
- State management with minimal re-renders.
- Mobile-first design approach for responsiveness.
- Proper folder structure and file naming conventions.

Repository Submission

- **GitHub Repository**: https://github.com/RehanTechForge/Marketplace-Builder-Hackathon-2025/tree/main
- **Folder Structure**:
    - `components/` - Contains all reusable components.
    - `app/` - Contains Next.js pages and routing logic.
    - `api/` - Contains scripts for API integration.