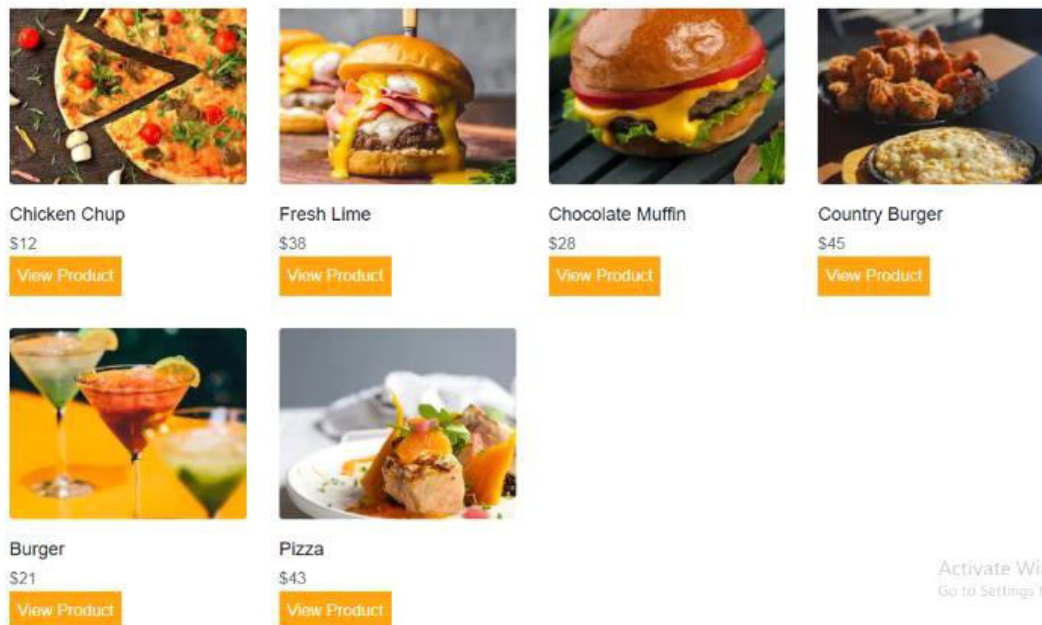# Day 4 - Dynamic Frontend Components

ScreenShot of:

***Product Listing Component:**

Search bar Functiolity:

🕐 A search bar is implemented using the SearchableProductList component.

🕐 The handleSearch function filters products based on the search query. It checks the

name field of each product and show them.

🕐 The search bar dynamically updates the filteredData as the user types.

Code:

Product Details (Dynamic Routes):

🕐 Shows a product image, name, description, price (original and discounted), and a

discount percentage for real-time.

🕐 Includes a "View Product" button linking to the product's detailed page using its _id.
API Integration:

1. Sanity API Fetching Logic: Since you're using Sanity CMS, you need to ensure data fetching

is properly managed.

🕐 Sanity Client Setup (if not already set up): In your lib folder, configure the Sanity

client:

🕐 Fetching Data in useEffect:

Searchbar.tsx U   Nav.tsx M   page.tsx ...\(6) M   page.tsx ...\shop M ✕

src > app > shop > ⬡ page.tsx > 🔅 page > 🔅 handleSearch

```tsx
"use client";
import React, { useState, useEffect } from 'react';
import { client } from '@/sanity/lib/client';
import { urlFor } from '@/sanity/lib/image';
import Header from '@/components/layout/Header';
import SearchableProductList from '@/components/layout/Searchbar';
import Link from 'next/link';
import { IoIosArrowDown } from "react-icons/io";
import Image from 'next/image';
const page = () => {
  const [datas, setDatas] = useState<any[]>([]);
  const [filteredData, setFilteredData] = useState<any[]>([]);
  const [searchQuery, setSearchQuery] = useState('');
  useEffect(() => {
    const fetchData = async () => {
      const result = await client.fetch(`*[_type == "food"]{
        _id,
        name,
        category,
        price,
        originalPrice,
        image,
        description,
        available,
        tags
      }`);
      setDatas(result);
      setFilteredData(result);
    };
    fetchData();
  }, []);
  const handleSearch = (event: React.ChangeEvent<HTMLInputElement>) => {
    const query = event.target.value.toLowerCase();
    setSearchQuery(query);
    const filtered = datas.filter(item =>
      item.name.toLowerCase().includes(query) ||
      item.description.toLowerCase().includes(query) ||
      item.category.toLowerCase().includes(query)
    );
    setFilteredData(filtered);
  };
  const uniqueCategories = datas.filter(
    (item, index, self) => index === self.findIndex((t) => t.category === item.category)
  );
  return (
    <div>
      <Header />
      <div className='mt-[100px] mb-[100px] px:6 lg:px-[120px]'>
        <div className='lg:flex gap-4'>
```