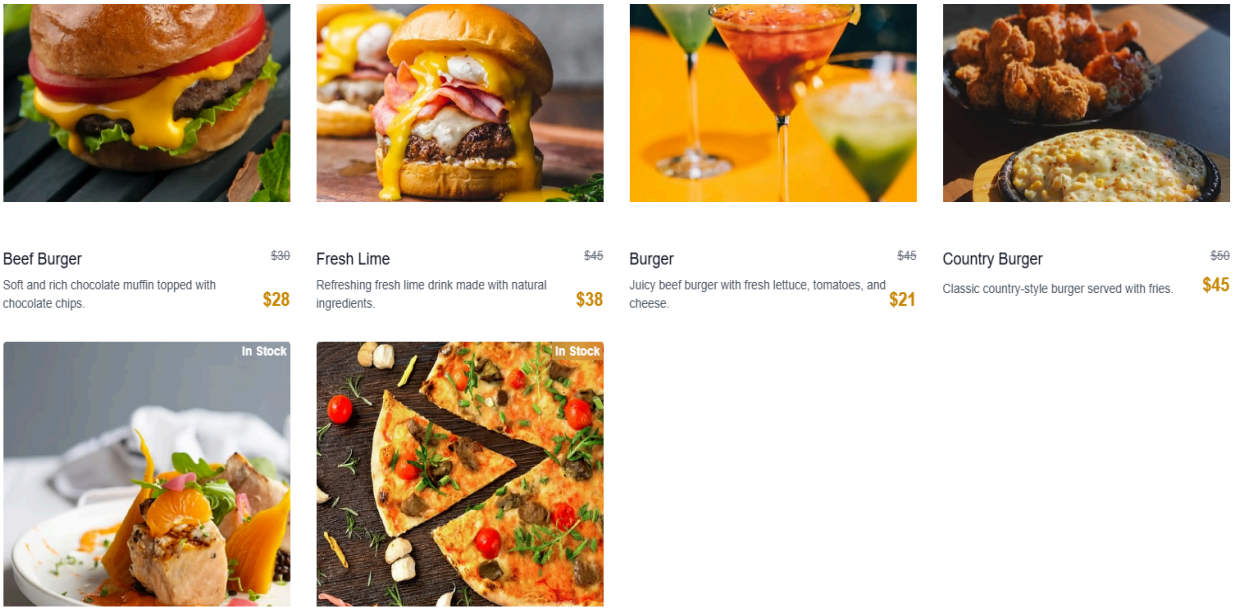
Create by Abdul Haseeb

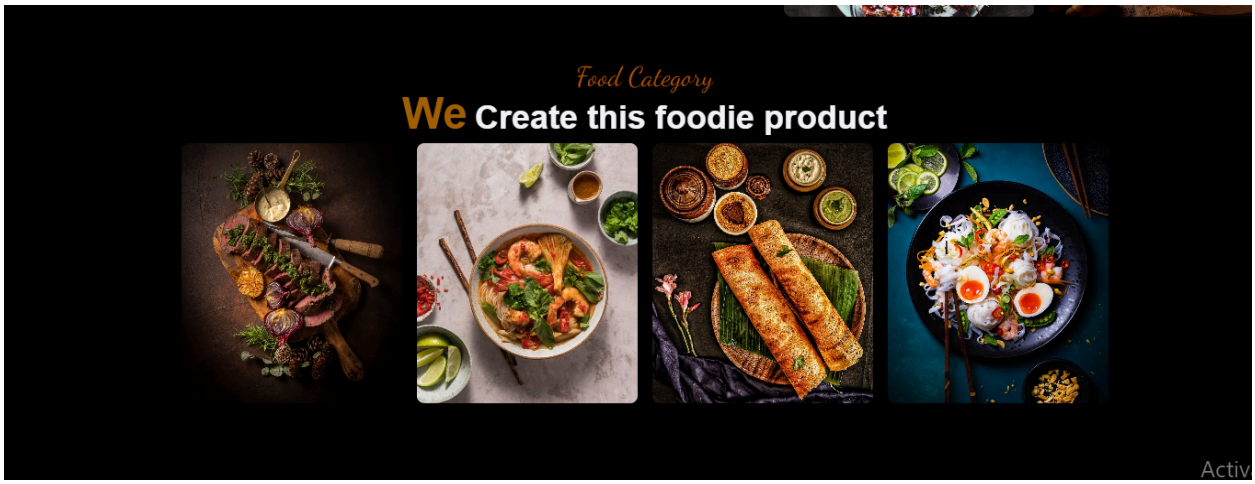
Day 4 - Dynamic Frontend Components-foods

Functional Delivery

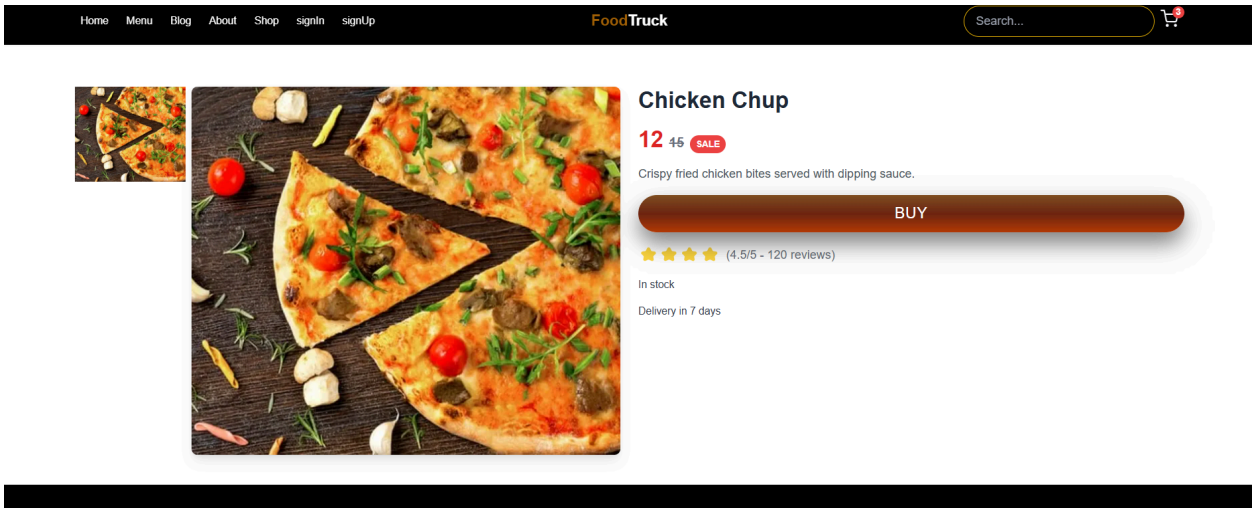
ScreenShot of Product Listing Component



* Category Component

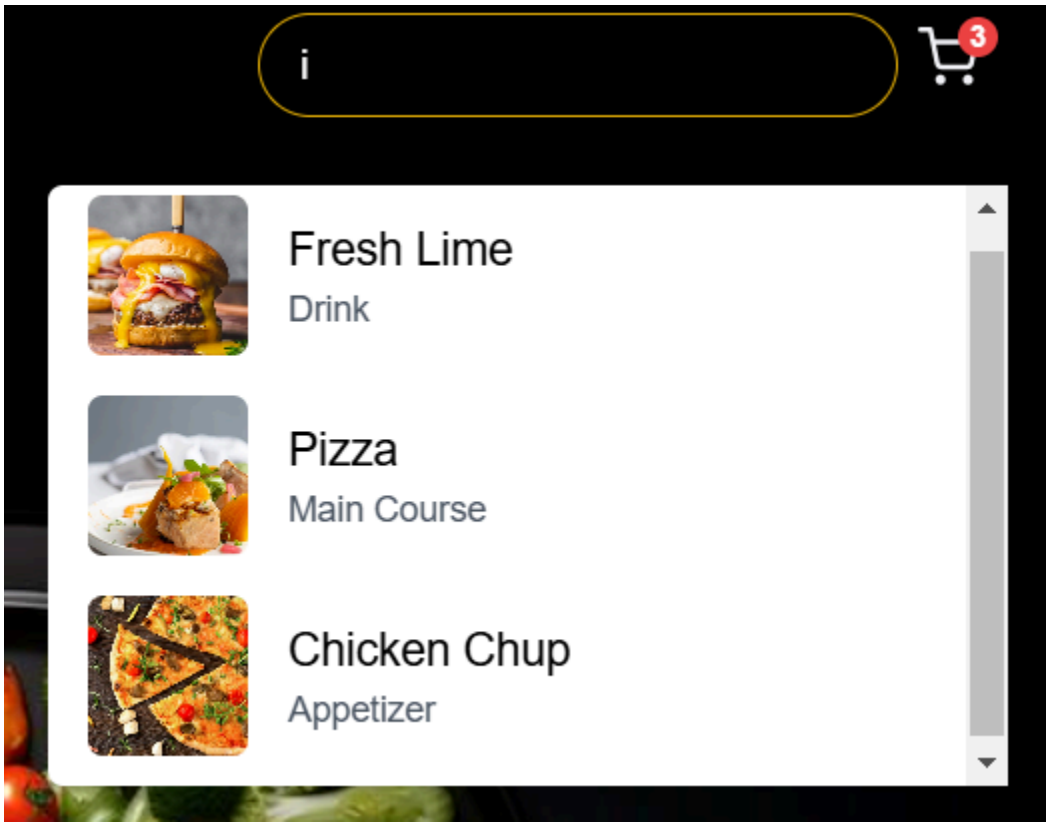


*Product Detail Component



Create by abdul haseeb

*Search Bar – This Screenshot is of the Navbar



Create by Abdul Haseeb

This is from Shop page where user can find products through Filter category

Sort by Category: burger

Beef Burger $30

Soft and rich chocolate muffin topped with chocolate chips. $28

burger $45

Refreshing fresh lime drink made with natural ingredients. $38

Create by Abdul Haseeb

Search bar Functionality 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.

Snippet code

```
const [products, setProducts] = useState([]);
 const [category, setCategory] = useState('');


 useEffect(() => {
   // Fetch data from Sanity
   const fetchProducts = async (category) => {
     const query = category
       ? `*[_type == "food" && category == "${category}"][0..6]{
           _id,
           name,
           category,
           price,
           originalPrice,
           tags,
           "image_url": image.asset->url,
           description,
           available
         }`
       : `*[_type == "food"][0..6]{
           _id,
           name,
           category,
           price,
           originalPrice,
           tags,
           "image_url": image.asset->url,
           description,
           available
         }`;

     try {
       const result = await client.fetch(query);

       setProducts(result);
     } catch (error) {
       console.error('Error fetching products:', error);
     }
   };

   fetchProducts(category);
 }, [category]);


 const handleCategoryChange = (e) => {
   setCategory(e.target.value);
 };
<div className="flex justify-around w-[332px]">
            <h1 className="text-lg text-black font-medium">Sort by Category:</h1>
            <select
              onChange={handleCategoryChange}
              value={category}
              className="text-black px-4 w-[236px] h-[40px] border ml-4 cursor-
pointer"
            >
              <option value="">All Categories</option>
              <option value="burger">burger</option>
              <option value="cheez broste">Cheez broste</option>
              <option value="drink">drink</option>
              <option value="Appetizer">Appetizer</option>
                 <option value="pizza">pizza</option>

            </select>
          </div>
          <div className="flex items-center w-[332px]">
            <h1 className="text-lg font-medium text-black">Show:</h1>
            <div className="flex items-center text-black justify-between px-4 w-
[236px] h-[40px] border ml-4 cursor-pointer">
              Default <IoIosArrowDown className="text-xl" />
            </div>
          </div>
        </div>
```

Product Details (Dynamic Routes):

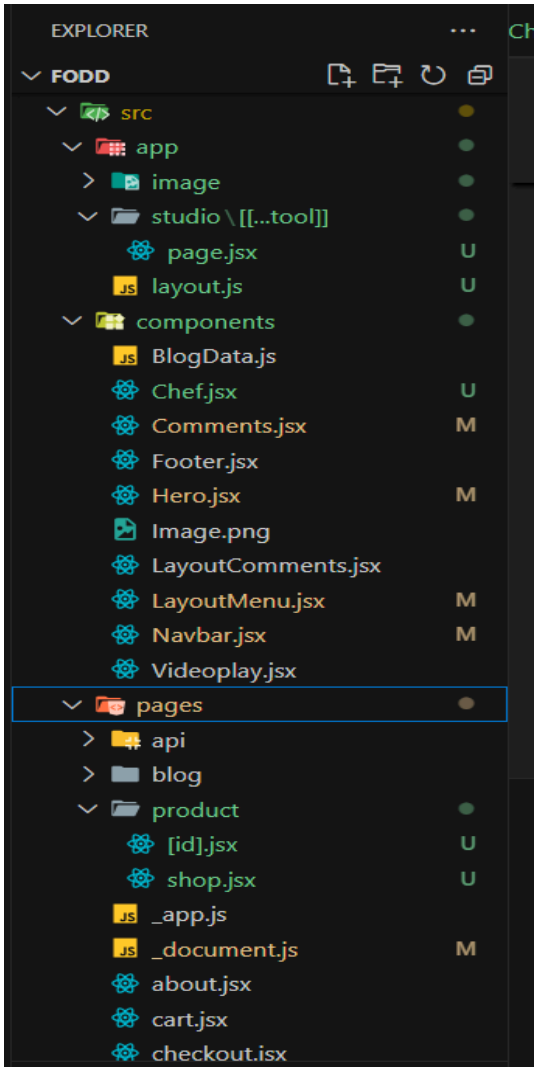
Shows a product image, name, description, price (original and discounted), and a discount percentage for real-time. Includes a "View Product" button linking to the product's detailed page using its _id code Review

```
// Static generation for the product page - fetching product IDs
export async function getStaticPaths() {
  const productsQuery = `*[_type == "food"]{ _id }`;
  const products = await client.fetch(productsQuery);

  const paths = products.map((product) => ({
    params: { id: product._id },
  }));

  return {
    paths,
    fallback: 'blocking',
  };
}

// Fetch product data statically for each product page
export async function getStaticProps({ params }) {
  const productId = params.id;
  const productQuery = `
    *[_type == "food" && _id == "${productId}"]{
      name, description, price, originalPrice, category, available,
      "image_url": image.asset->url, _id
    }
  `;
  const productData = await client.fetch(productQuery);

  if (!productData || productData.length === 0) {
    return { notFound: true };
  }

  return {
    props: {
      product: productData[0],
    },
    revalidate: 10,
  };
}

// Product Page Component
const ProductPage = ({ product }) => {
  if (!product) {
    return <div>Product not found.</div>;
  }

  const { name, description, price, originalPrice, available, image_url } = product;
```

Create by Abdul Haseeb

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

File Structure



Create by Abdul Haseeb

Product List Fetches data from a Sanity CMS backend for items of type "food". Displays each product in a card format, showing its name, description, price, discount, and an image

Preview code

```
 const [products, setProducts] = useState([]);
const [category, setCategory] = useState(''); // State for selected category

useEffect(() => {
  // Fetch data from Sanity
  const fetchProducts = async (category) => {
    const query = category
      ? `*[_type == "food" && category == "${category}"][0..6]{
          _id,
          name,
          category,
          price,
          originalPrice,
          tags,
          "image_url": image.asset->url,
          description,
          available
        }`
      : `*[_type == "food"][0..6]{
          _id,
          name,
          category,
          price,
          originalPrice,
          tags,
          "image_url": image.asset->url,
          description,
          available
        } `;

    try {
      const result = await client.fetch(query);

      setProducts(result);
    } catch (error) {
      console.error('Error fetching products:', error);
    }
  };

  fetchProducts(category);
}, [category]);
    const handleCategoryChange = (e) => {
  setCategory(e.target.value);
};
```

Create by Abdul Haseeb

Dynamic Routing Dynamic routing is used to handle routes like /shop/:id for individual product pages.

1.

Create Dynamic Route File: In the pages directory, create a [id].tsx file inside the shop folder: pages/shop/[id].tsx

2.

Get Product Data Dynamically:

Frontend Component	Styling and Responsiveness	Code Quality	Documentation and Submission	Final Review
✔	✔	✔	✔	✔

Create by Abdul Haseeb