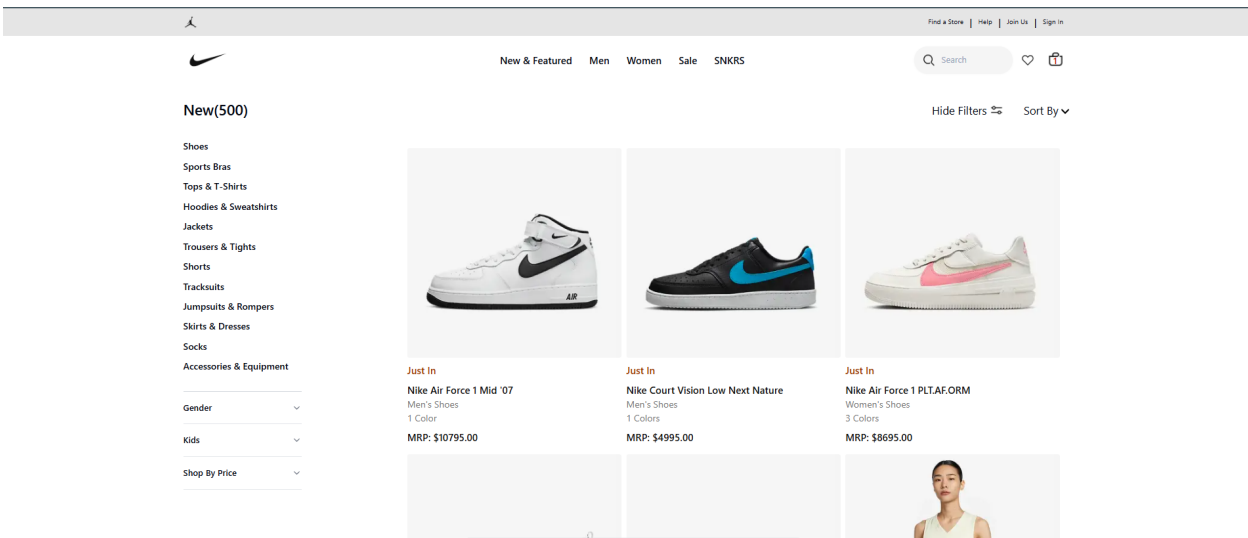


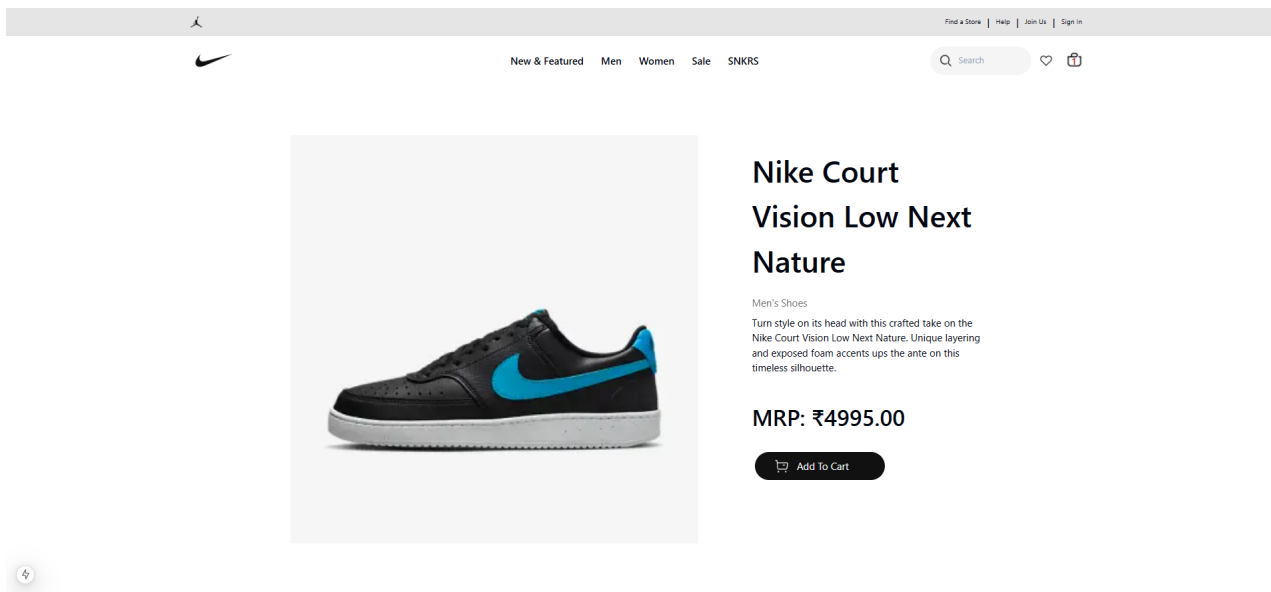
Day 4 - Dynamic Frontend Components – Nike

Screenshots of:

1. Product Listing Component:

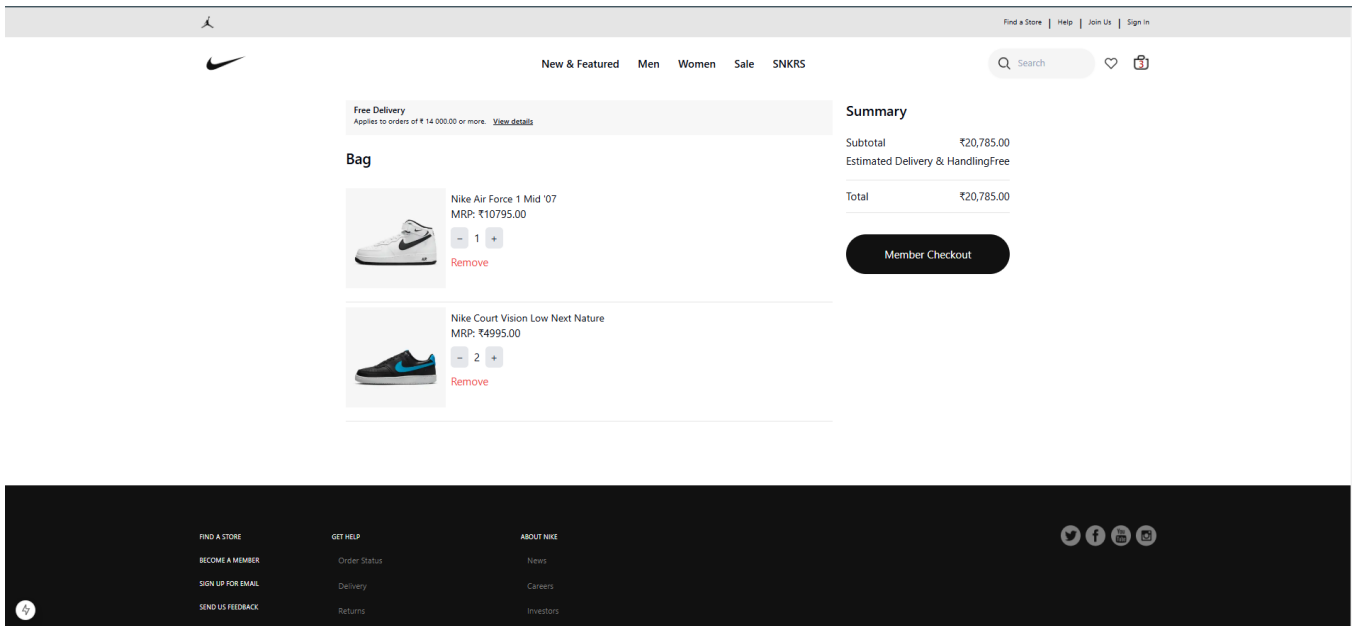


2. Dynamic Functionality:



Umer Adnan (00053633)

3.Cart Functionality:



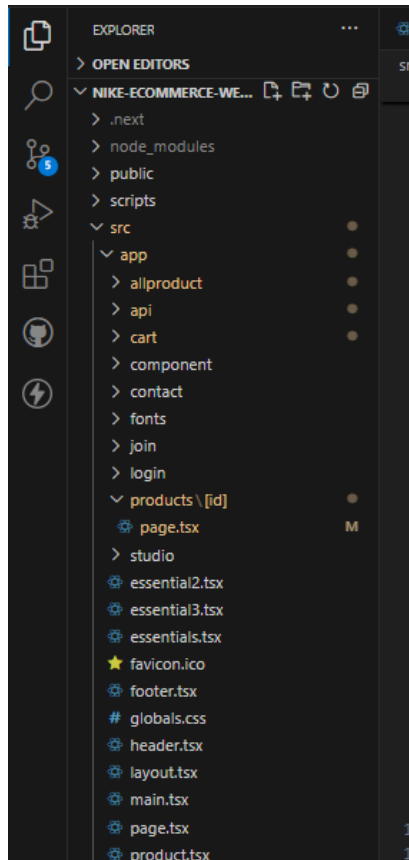
Code:

```
page.tsx M product.tsx X
src > app > product.tsx > Product
8 export default function Product() {
53 > <div className="flex justify-between py-8">...
71 </div>
72
73 /* Slider */
74 <div className="overflow-hidden relative">
75 <div
76   className="flex transition-transform duration-500"
77   style={{
78     transform: `translateX(-${currentIndex * (370 + 15)}px)`,
79     gap: "20px",
80     width: `${cards.length * (370 + 15)}px`,
81   }}
82 >
83   {cards.map((card, index) => (
84     <div
85       key={index}
86       className="flex-shrink-0"
87       style={{ width: "370px" }}
88     >
89       <div className="bestSellingBox">
90         <div className="addToCartOverlay"><Link href={`../allproduct`}>View Product Page</Link></div>
91         <Image src={card.image} alt={card.title} className="mainImage" />
92       </div>
93       <div className="flex justify-between text-[15px] font-medium py-2">
94         <div>{card.title}</div>
95         <div>{card.price}</div>
96       </div>
97       <div className="text-[#757575] text-[15px]">{card.category}</div>
98     </div>
99   ))}
100 </div>
101 </div>
102 </div>
103 );
104 }
105
```

Umer Adnan (00053633)

Product Details (Dynamic Routes):

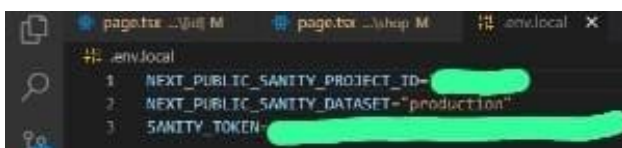
- Shows a product image, name, description, price (original and discounted), and a discount percentage for real-time.
- Includes a "View Product" button linking to the product's detailed page using its `_id`.



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:**

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:

```
page.tsx M X
src > app > shop > [id] > page.tsx > Page
220
221 const Page = async ({ params }: { params: { id: string } }) => {
222   const datas = await client.fetch(
223     `[_type == "food" && _id == "${params.id}"][0]{
224       name,
225       category,
226       price,
227       originalPrice,
228       image,
229       description,
230       available,
231       tags
232     }`
233   );
234   const relatedImages = await client.fetch(
235     `[_type == "food" && "${datas.tags[0]}" in tags][0...3]{
236       url: image.asset->url
237     }`
238   );
239   return (
240     <div>
241       <Header/>
242       <div className="mt-[100px] mb-[20px] container px-1 mx-auto">...
243     </div>
244     {shopdetail.map((datashop)=>{
245       return(
246         <div className="mt-[0px] mb-[100px] lg:w-4/5 px-12 mx-auto">
247           <div className="flex gap-10">
248             <button className="bg-bordercoloryellow text-whitetext p-2">Description</button><button>Rev
249           </div>
250           <div className="flex flex-col gap-4 text-[14px]">
251             <p>{datashop.description}</p>
252             <p>{datashop.senddespara}</p>
253           </div>
254         </div>
255       )
256     })}
257   )
258 }
```

Final Checklist

Frontend Component Development	Styling and Responsiveness	Code Quality	Documentation and Submission	Final Review
✓	✓	✓	✓	✓