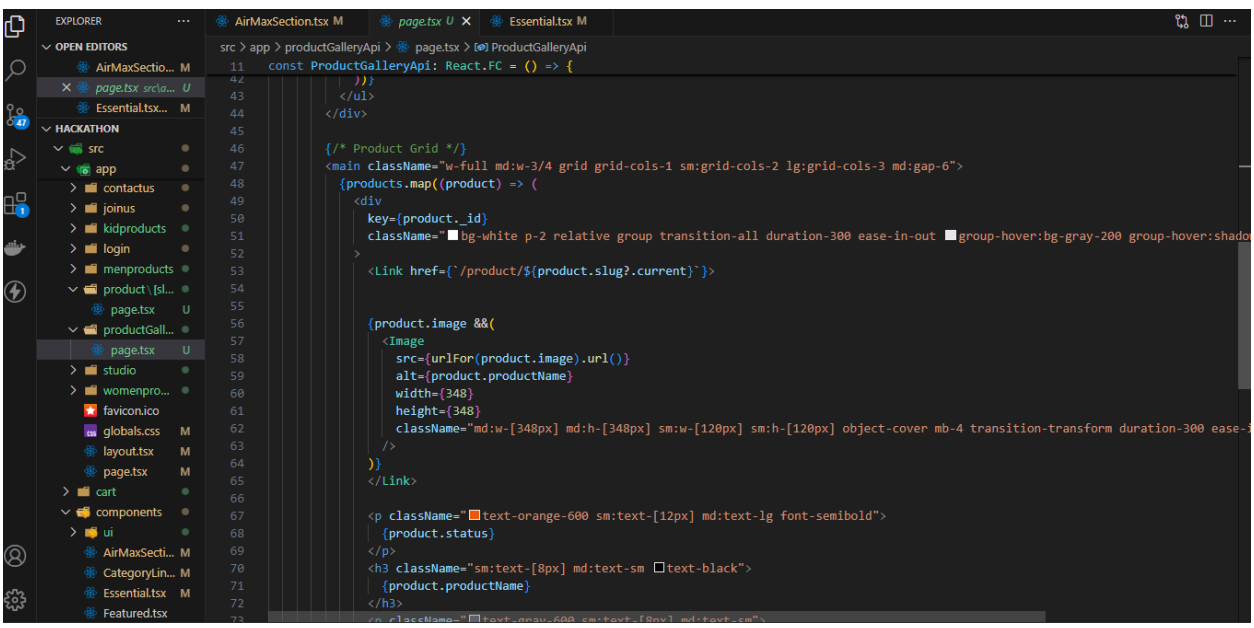
DAY 4 - BUILDING DYNAMIC FRONTEND COMPONENTS FOR YOUR MARKETPLACE

- Document Title: "Day 4 - Dynamic Frontend Components - [Nike-Template-3]"

Key Learning Outcomes:

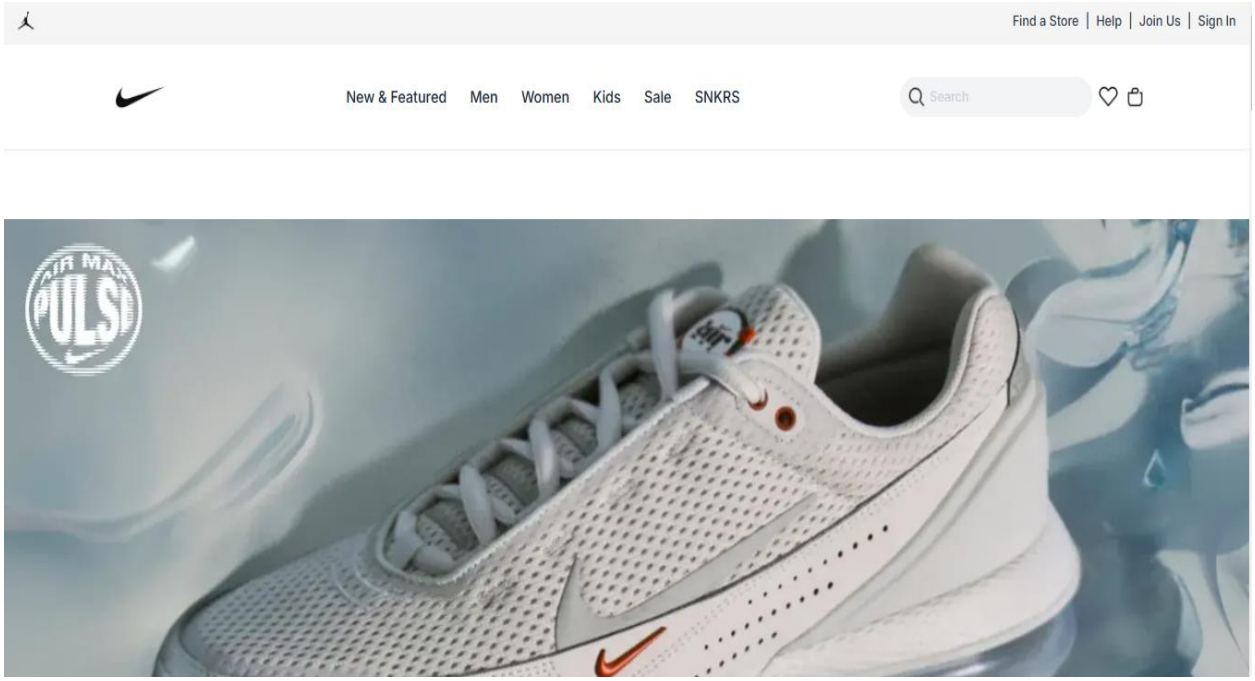
1. **Build dynamic frontend components to display data from Sanity CMS or APIs.**



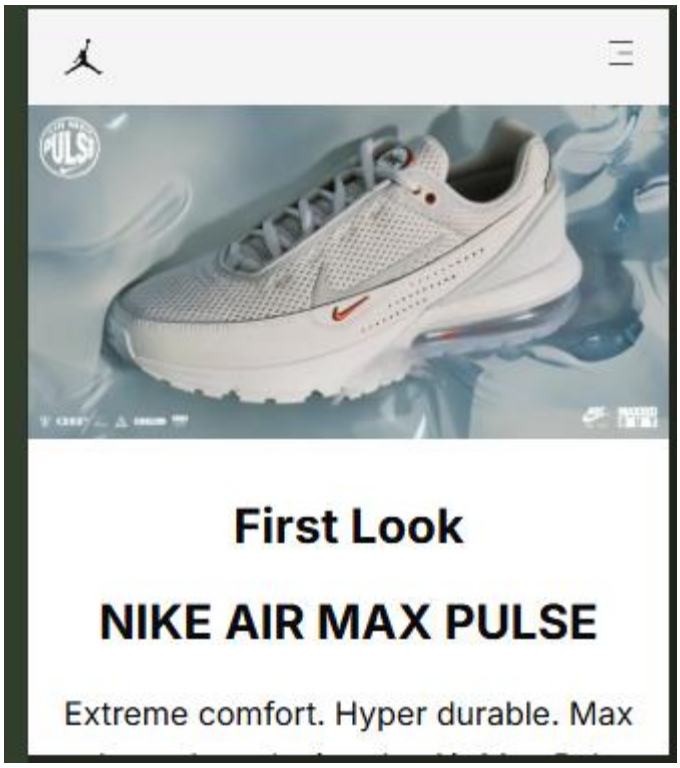
2. **Implement reusable and modular components.**
3. **Understand and apply state management techniques.**

4. **Learn the importance of responsive design and UX/UI best practices.**
5. **5. Prepare for real-world client projects by replicating professional workflows.**

<u>Laptop Screen RESPONSIVE</u>



Mobile Screen Responsive



Key Components to Build

1. **Product Listing Component:**

- Render product data dynamically in a grid layout.

```
{/* Product Grid */}
<main className="w-full md:w-3/4 grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 md:gap-6">
  {products.map((product) => (
    <div
      key={product._id}
      className="bg-white p-2 relative group transition-all duration-300 ease-in-out group-hover:bg-gray-200 group-hover:shado
    >
      <Link href={`/product/${product.slug?.current}`}>


      {product.image &&(
        <Image
          src={urlFor(product.image).url()}
          alt={product.productName}
          width={348}
          height={348}
          className="md:w-[348px] md:h-[348px] sm:w-[120px] sm:h-[120px] object-cover mb-4 transition-transform duration-300 ease-
        />
      )}
      </Link>

      <p className="text-orange-600 sm:text-[12px] md:text-lg font-semibold">
        {product.status}
      </p>
      <h3 className="sm:text-[8px] md:text-sm text-black">
        {product.productName}
      </h3>
```

- Include fields like:

```
OPEN EDITORS
  AirMaxSectio... M
  product.ts src\... U
  Essential.tsx... M
HACKATH...
  src
    components
      Hero2.tsx M
      Hero3.tsx
      Jordanspring.tsx
      Navbar.tsx M
    lib
    sanity
      lib
        client.ts U
        fetch.ts U
        image.ts U
        live.ts U
        queries.ts U
      schemaTypes
        index.ts U
        product.ts U
      env.ts U
      structure.ts U
    types
      products.ts U
  .env.local
  .eslintrc.json
  .gitignore
```

src > sanity > schemaTypes > product.ts > productSchema > fields

```
1  export const productSchema = {
2      name: 'product',
3      title: 'Product',
4      type: 'document',
5      fields: [
6        {
7          name: 'productName',
8          title: 'Product Name',
9          type: 'string',
10       },
11       {
12         name: 'category',
13         title: 'Category',
14         type: 'string',
15       },
16       {
17         name: 'slug',
18         title: 'Slug',
19         type: 'slug',
20         options: {
21           source: 'productName',
22
23
24         },
25
26       },
27       {
28         name: 'price',
29         title: 'Price',
30         type: 'number',
31       },
32       {
```

- Example layout: cards displaying product details.

```
const Carousel: React.FC = () => {
        <div className="flex space-x-4 overflow-x-auto sm:overflow-hidden">
          {visibleProducts.map((product) => (
            <div
              key={product._id}
              className="flex-shrink-0 w-full sm:w-1/2 md:w-1/3 bg-white p-4 text-center rounded-lg shadow ho
            >
              <Link href={`/product/${product.slug?.current}`}>

                <Image
                  src={urlFor(product.image).url()}
                  alt={product.productName}
                  width={400}
                  height={400}
                  className="mx-auto object-cover mb-4 rounded-lg"
                />
              </Link>
              <div className="md:px-10 sm:text-[8px] md:text-sm  flex sm:flex-col md:flex-row  items-center just
                <div className="text-start flex sm:items-center md:items-start flex-col" >
                  <h3 className="text-black  font-medium">
                    {product.productName}
                  </h3>
                  <p className="text-gray-600 sm:text-start md:text-start">{product.category}</p>
                </div>
                <div>
                  <p className="text-black ">MRP : ₹ {product.price}</p>
                </div>
              </div>
            </div>
          ))}
        </div>
      </div>
```

2. Product Detail Component:

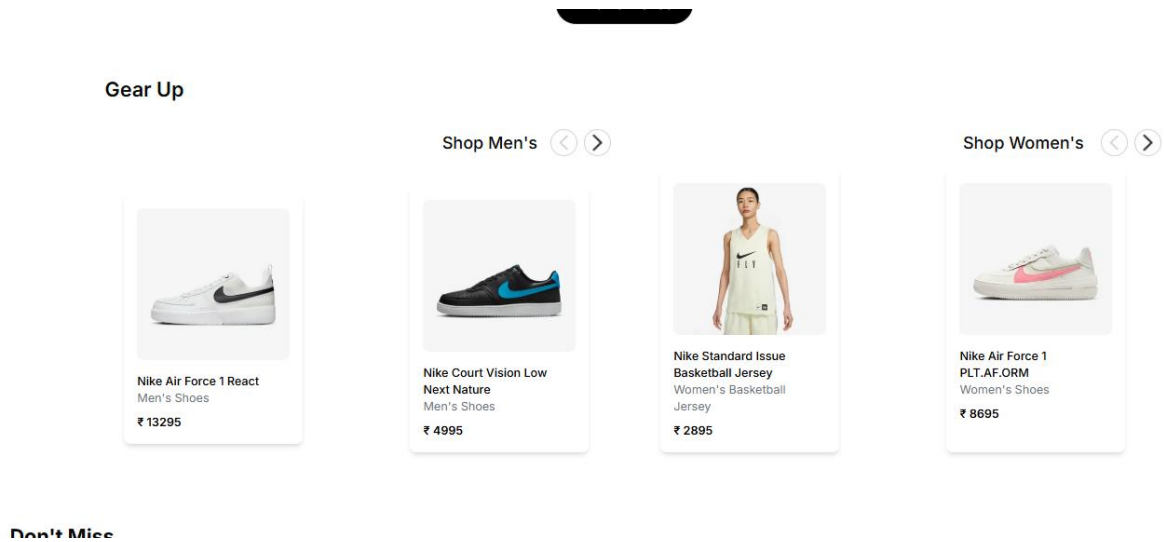
- **Create individual product detail pages using dynamic routing in Next.js.**

- **Include detailed fields such as: o Product Description o Price o Available Sizes or Colors**

```python
"use client";
import { addToCart } from "@/cart/cart";
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
import { groq } from "next-sanity";
import Image from "next/image";
import Link from "next/link";
import { Product } from "types/products";
import Swal from "sweetalert2";

interface productPageProps {
  params: Promise<{ slug: string }>;
}

async function getProduct(slug: string): Promise<Product> {
  return client.fetch(
    groq`*[_type == "product" && slug.current == $slug][0]{
        _id,
        productName,
        category,
        slug,
        price,
        inventory,
        colors,
        status,
        image,
        description,
        _type
        }`,
    { slug }
  );
}
export default async function ProductPage({ params }: productPageProps) {
  const { slug } = await params;
  const product = await getProduct(slug);

  // add to cart

  const handleAddToCart = (e: React.MouseEvent, product: Product) => {
    e.preventDefault();
    Swal.fire({
      position: "top-end",
      icon: "success",
      title: `${product.productName} added to cart`,
      showConfirmButton: false,
      timer: 1000,
    });
    addToCart(product);
  };

  return (
    <div className="md:pt-10 flex flex-col lg:flex-row items-center gap-8 p-6 max-w-6xl mx-auto">
      {/* Product Image */}
      <div
        data-aos="zoom-out-up"
        className="w-full lg:w-1/2 flex justify-center"
      >
        <Image
          src={urlFor(product.image).url()}
          alt={product.productName}
          width={400}
          height={400}
          className="object-contain"
        />
      </div>

      {/* Product Details */}
      <div className="w-full lg:w-1/2 flex flex-col gap-4">
        <h1 className="text-3xl font-bold">{product.productName}</h1>
        <p className="text-gray-600">{product.description}</p>
        <p className="text-2xl font-bold">₹ {product.price}</p>
        <Link href="/cart">
          <button
            className="bg-black text-white py-3 px-6 rounded-3xl w-48 flex items-center gap-2 hover:bg-gray-800"
            onClick={(e) => handleAddToCart(e, product)}
          >
            <span className="material-icons">
              {product.image && (
                <Image
                  src="/images/cart.png"
                  alt={product.productName}
                  width={22.36}
                  height={16.3}
                />
              )}
            </span>
            Add To Cart
          </button>
        </Link>

      </div>
    </div>
  );
}

```

3. Category Component:

- **Display categories dynamically fetched from the data source.**

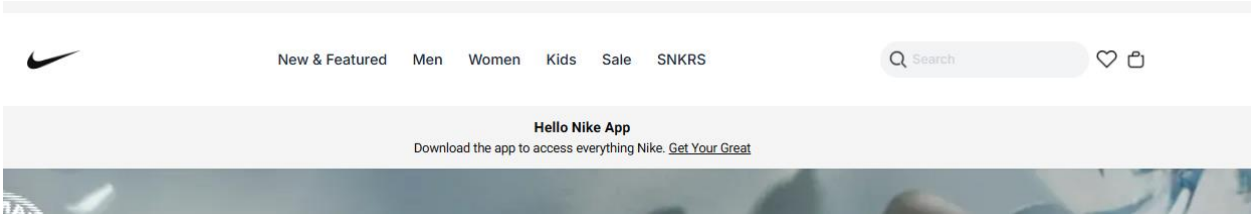


- **Enable filtering of products by selected categories.**

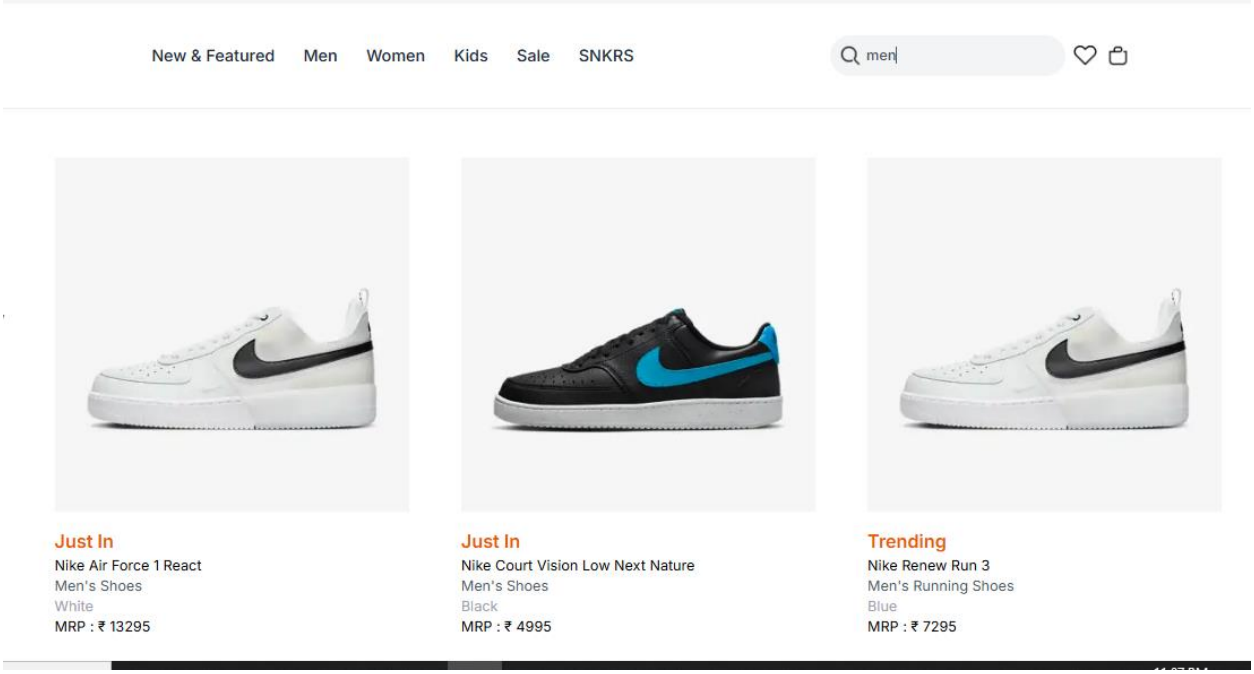
```
const GearUp: React.FC = () => {
  const [menProducts, setMenProducts] = useState<Product[]>([]);
  const [womenProducts, setWomenProducts] = useState<Product[]>([]);
  const [currentMenIndex, setCurrentMenIndex] = useState(0);
  const [currentWomenIndex, setCurrentWomenIndex] = useState(0);

  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const fetchedProducts: Product[] = await client.fetch(allProducts);

        // Filter products by category
        const men = fetchedProducts.filter(
          (product) =>
            product.category?.toLowerCase().includes("men")
        );
        const women = fetchedProducts.filter(
          (product) =>
            product.category?.toLowerCase().includes("women")
        );

        setMenProducts(men);
        setWomenProducts(women);
      } catch (error) {
        console.error("Error fetching products:", error);
      }
    };
```

4. Search Bar:

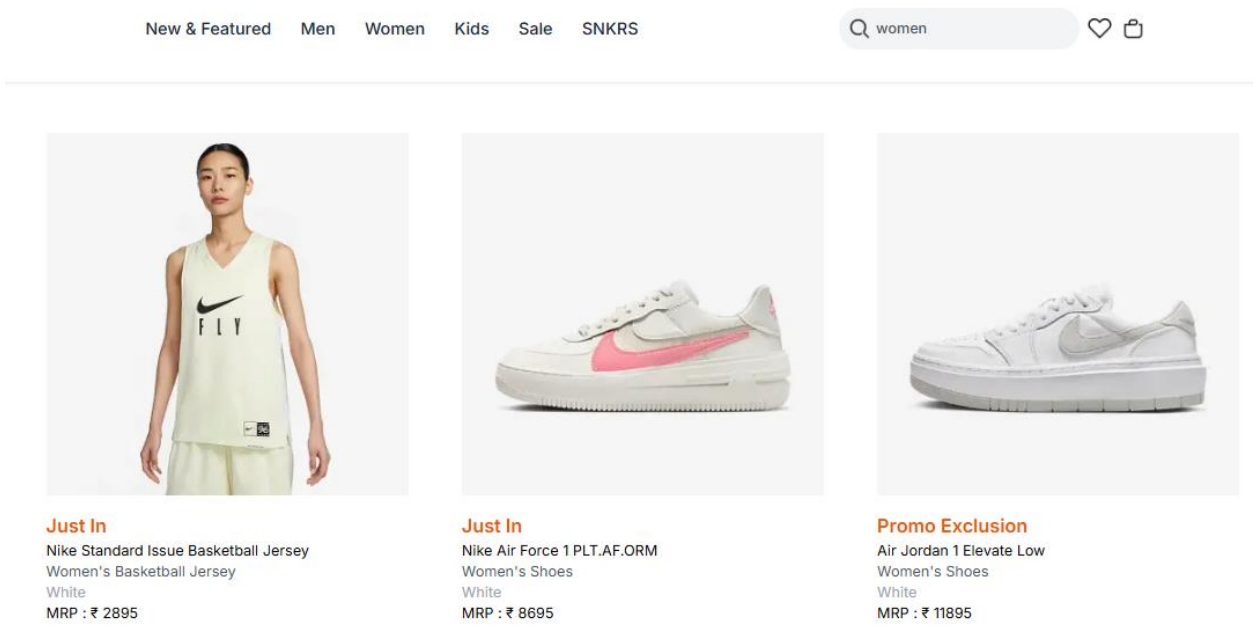
- **Implement search functionality to filter products by name or tags.**



Searching men product:

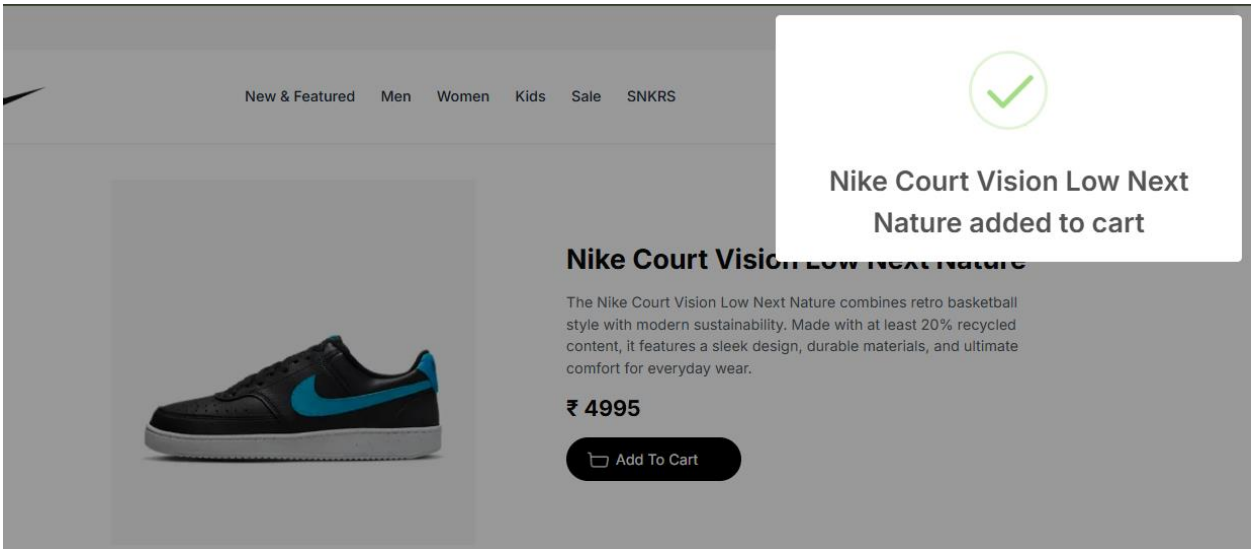


Searching women product:

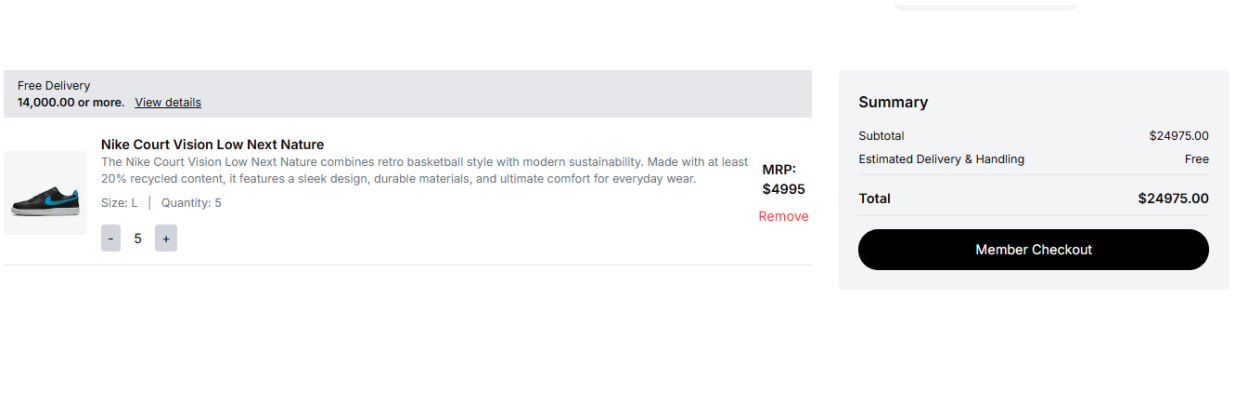


5. Cart Component:

- **Display added items, quantity, and total price.**
- **Use state management for tracking cart items.**

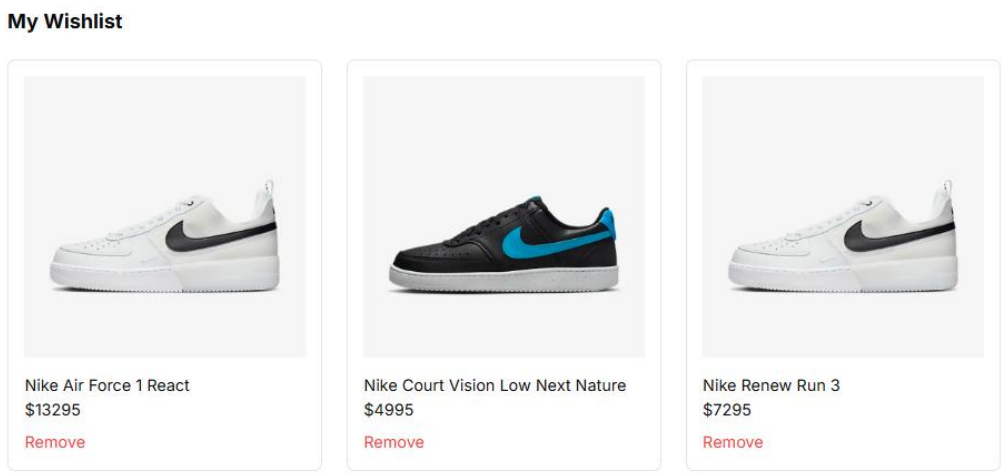


Add and remove :

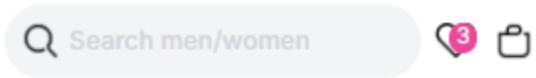


6. Wishlist Component:

- **Allow users to save products for future reference.**
- **Use local storage or a global state management tool to persist data.**



Icons number add:



- ***Checklist for Day 4: Self-Validation Checklist:***
- ***Frontend Component Development: ✔***
- ***Styling and Responsiveness: ✔***
- ***Code Quality: ✔***
- ***Documentation and Submission: ✔***
- ***Final Review: ✔ ✘***