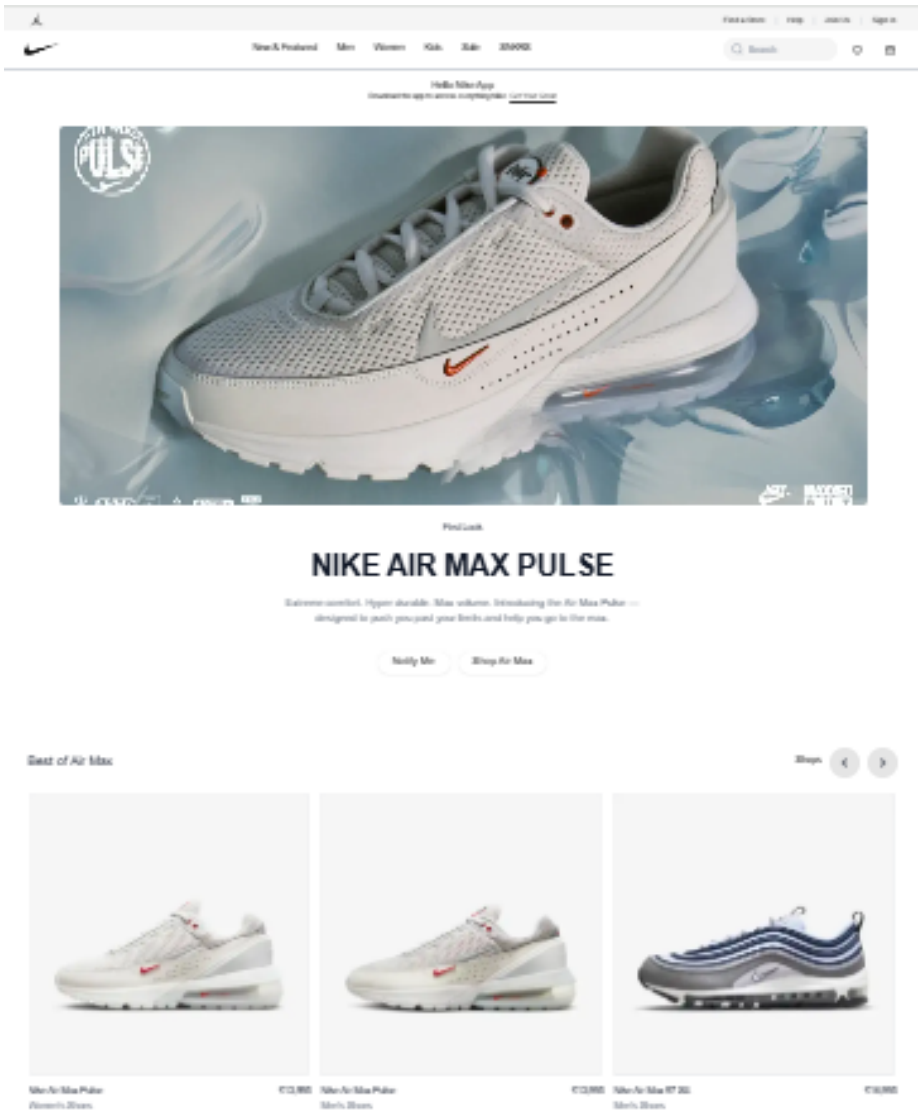
Day 4 - Dynamic Frontend Components - Flex Wear Marketplace

1. Introduction

Day 4 focused on building **dynamic frontend components** for the **Flex Wear Marketplace** using Next.js and Sanity CMS. The goal was to create reusable, scalable UI components that dynamically fetch and display data.

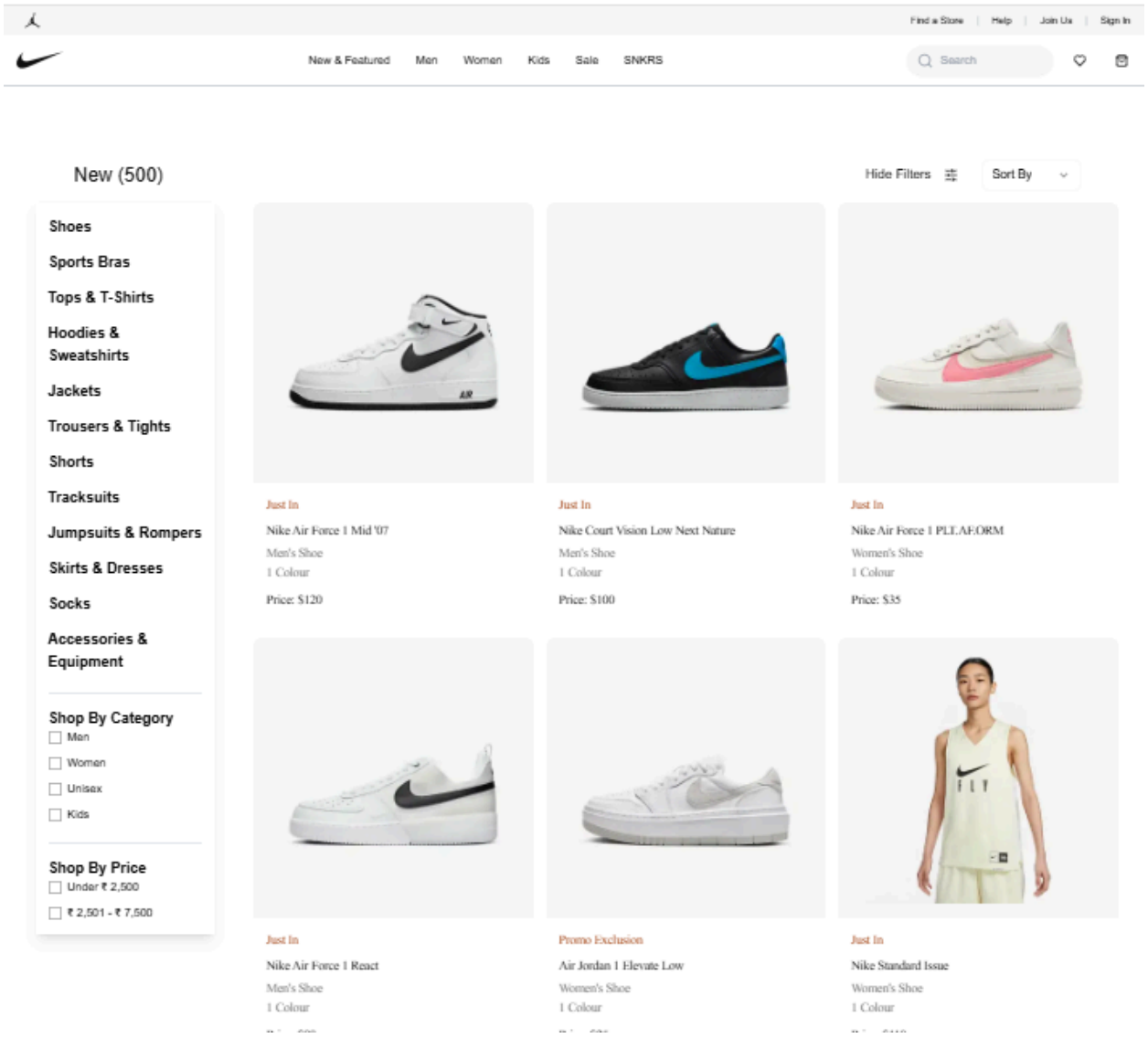


Prepared by:
Ayesha Nasir

2. Implementation Steps

✅ Product Listing Component

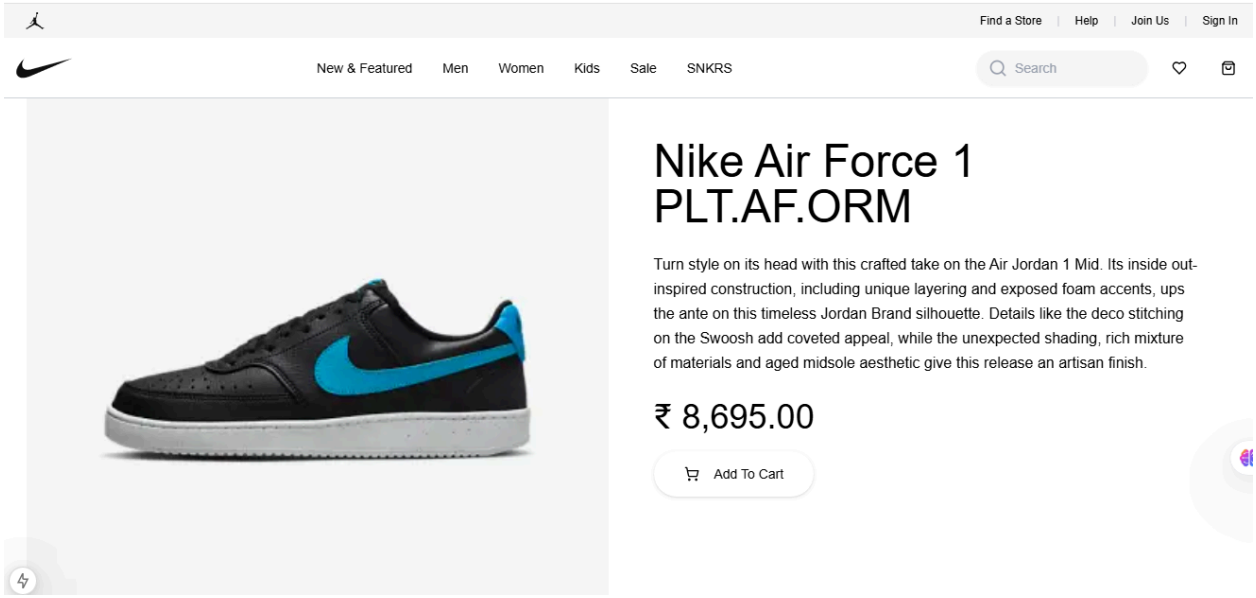
- Implemented a **grid layout** to display products dynamically.
- Integrated Sanity CMS API to fetch product details.
- Used Tailwind CSS for responsive design.



Prepared by:
Ayesha Nasir

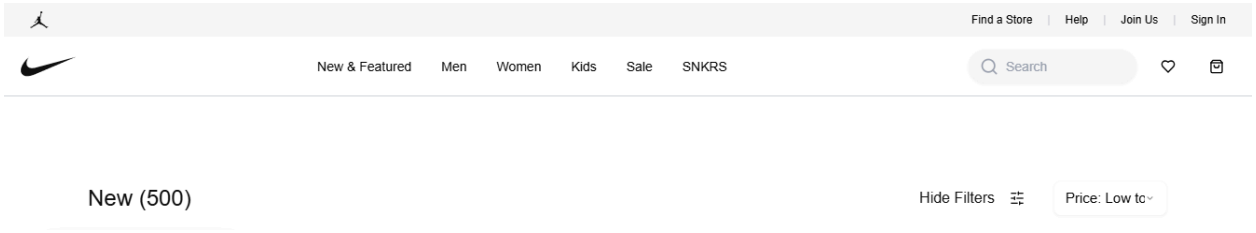
✅ Product Detail Component

- Created **dynamic routing (`/product/[id]`)** for individual product pages.
- Displayed detailed product information including price, description, and stock.



✅ Search Bar

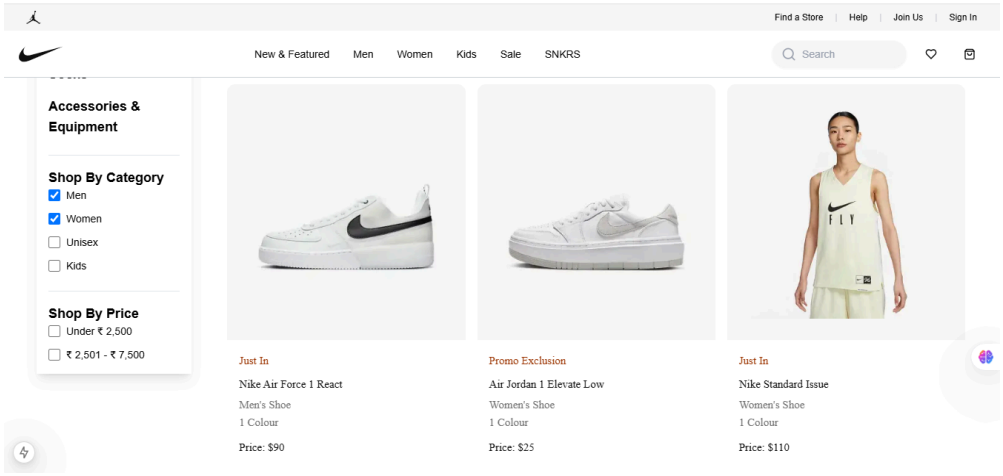
- Implemented search functionality to filter products by name.
- Used React state management for efficient filtering.



Prepared by:
Ayesha Nasir

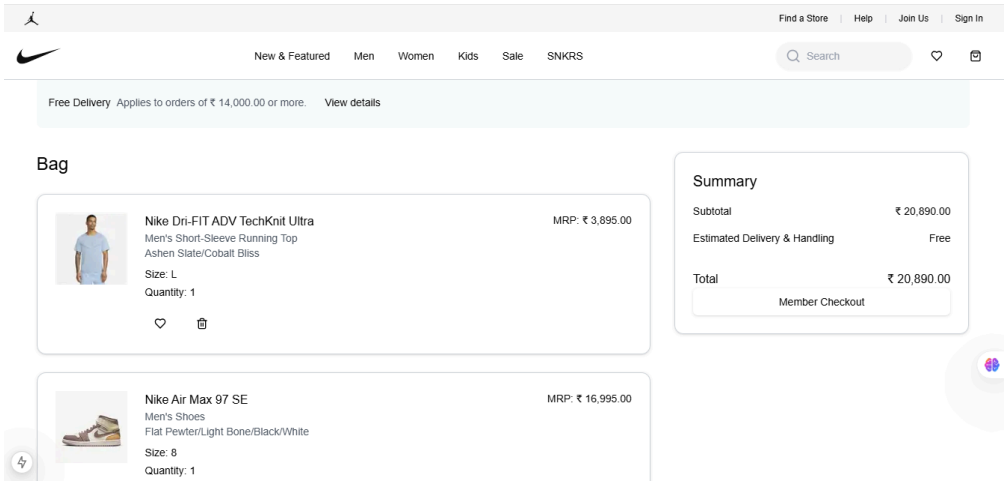
✅ Category Filtering

- Dynamically displayed product categories.
- Enabled category-based product filtering.



✅ Cart & Wishlist Components (If Implemented)

- Developed a **cart system** for adding and removing items.
- Integrated a **wishlist feature** using local storage.



Prepared by:
Ayesha Nasir

3. API Integration & Data Fetching

- Used **Sanity CMS** to store and retrieve marketplace products.
- Fetched data using **GROQ queries** inside Next.js components.
- Example API Query:

```
import { sanityFetch } from "@/sanity/lib/fetch";
import { allProducts } from "@/sanity/lib/queries";

type Product = {
  _id: string;
  productName: string;
  description: string;
  price: number;
  image: {
    asset: {
      url: string;
    };
  };
  category: string;
  inventory: number;
  status: string;
  colors: string[];
};
```

- Implemented **server-side rendering (`getServerSideProps`)** for fetching products efficiently.

```
export default async function Home() {
  const products: Product[] = await sanityFetch({
    query: allProducts,
  });

  return (
    <div>
      <h1>Products</h1>
      <div>
        {products.map((product) => (
          <div key={product._id} className="product-card">
            <img
              src={product.image?.asset?.url || "/placeholder.jpg"}
              alt={product.productName}
              className="product-image"
              width={"400px"}
            />
            <h3>{product.productName}</h3>
            <p>{product.description}</p>
            <p>Category: {product.category}</p>
            <p>Price: ${product.price}</p>
            <p>Inventory: {product.inventory}</p>
            <p>Status: {product.status}</p>
            <p>Colors: {product.colors.join(", ")}</p>
          </div>
        ))}
      </div>
    </div>
  );
}
```

4. Challenges & Solutions

🔴 Issue: Product Image Not Displaying

✅ **Fix:** Ensured `product.image?.asset?.url` exists before rendering.

🔴 Issue: Next.js Build Error with Tailwind CSS

✅ **Fix:** Adjusted Tailwind configurations and verified utility class availability.

🔴 Issue: API Data Not Fetching Correctly

✅ **Fix:** Verified Sanity project ID, dataset, and API query structure.

5. Best Practices Followed

- 📌 **Modular Components**: Created reusable UI components.
- 📌 **Responsive Design**: Used Tailwind CSS for mobile and desktop compatibility.
- 📌 **Optimized API Calls**: Implemented caching and efficient data fetching.
- 📌 **Error Handling**: Added fallback UI for missing product data.

Conclusion

Day 4 was a significant step in making the **Flex Wear Marketplace** dynamic and user-friendly. The integration of **Sanity CMS, Next.js, and Tailwind CSS** allowed for a scalable and professional UI. Looking forward to the next phase of development!

Prepared by:
Ayesha Nasir