

## **Day 4 - Dynamic Frontend Components - [NIKE STORE]**

### **Introduction:**

This report summarizes the process of building dynamic frontend components for a marketplace. The primary objective of the project was to create an engaging and efficient user interface with real-time data fetching and dynamic routing. This involved implementing essential features such as product listing pages, product detail pages, category filters, search functionality, and pagination. The goal was to provide users with an interactive and responsive shopping experience through dynamic components.

### **Functional Deliverables:**

#### **Product Listing Page with Dynamic Data:**

The product listing page was designed to display a dynamic list of products fetched from an API. This page serves as the main interface for browsing products.

#### **Product Detail Page:**

Each product in the product listing page links to its own dedicated product detail page. This page displays more detailed information about the product, including the product name, description, price, and images.

#### **Category Filter:**

The category filter allows users to narrow down the displayed products by category (e.g., "Women's Basketball," "Men's Running Shoes," etc.). It enhances the browsing experience by allowing users to view only the items relevant to them.

#### **Search Bar:**

The search bar allows users to search for products by name or description. It helps users find specific products quickly and easily.

## Pagination:

Pagination is used to break down large product lists into smaller pages, improving load times and overall usability.

## Summary Of Each Component:

### Product Listing Page with Dynamic Data:

- Dynamic product cards displaying product information such as name, price, and image.
- Real-time data fetching from an API to display updated product information.



Just In  
**Nike Air Force 1 Mid '07**  
Men's Shoes  
MRP: ₹ 10795.00



Just In  
**Nike Court Vision Low Next Nature**  
Men's Shoes  
MRP: ₹ 4995.00



Promo Exclusion  
**Air Jordan 1 Elevate Low**  
Women's Shoes  
MRP: ₹ 11895.00



**Just In**  
**Nike Court Vision Low**  
Men's Shoes  
MRP: ₹ 5695.00



**Just In**  
**Nike Blazer Mid '77 Vintage**  
Men's Shoes  
MRP: ₹ 8495.00



**Just In**  
**Nike Air Force 1 PLT.AF.ORM**  
Women's Shoes  
MRP: ₹ 8695.00



**Just In**  
**Nike Air Force 1 React**  
Men's Shoes  
MRP: ₹ 13295.00



**Promo Exclusion**  
**Air Jordan 1 Elevate Low**  
Women's Shoes  
MRP: ₹ 11895.00



**Just In**  
**Nike Standard Issue Basketball Jersey**  
Women's Basketball Jersey  
MRP: ₹ 2895.00



**Promo Exclusion**  
**Nike Dunk Low Retro SE**  
Men's Shoes  
MRP: ₹ 9695.00



**Sustainable Materials**  
**Nike Dri-FIT UV Hyverse**  
Men's Short-Sleeve Graphic Fitness Top  
MRP: ₹ 2495.00



**Just In**  
**Nike Zoom Fly 5**  
Men's Running Shoes  
MRP: ₹ 11295.00

**Product Detail Page:**

- Dynamic routing using React Router to show product-specific data based on the product ID.
- Real-time rendering of product details fetched from the API.



## Nike Air Force 1 Mid '07


The Nike Air Force 1 Mid '07 delivers timeless style with premium leather and mid-cut design. Perfect for everyday wear, it provides exceptional comfort and durability. The iconic Air-Sole cushioning adds responsive support for long-lasting performance.

Category: Men's Shoes

Color: White

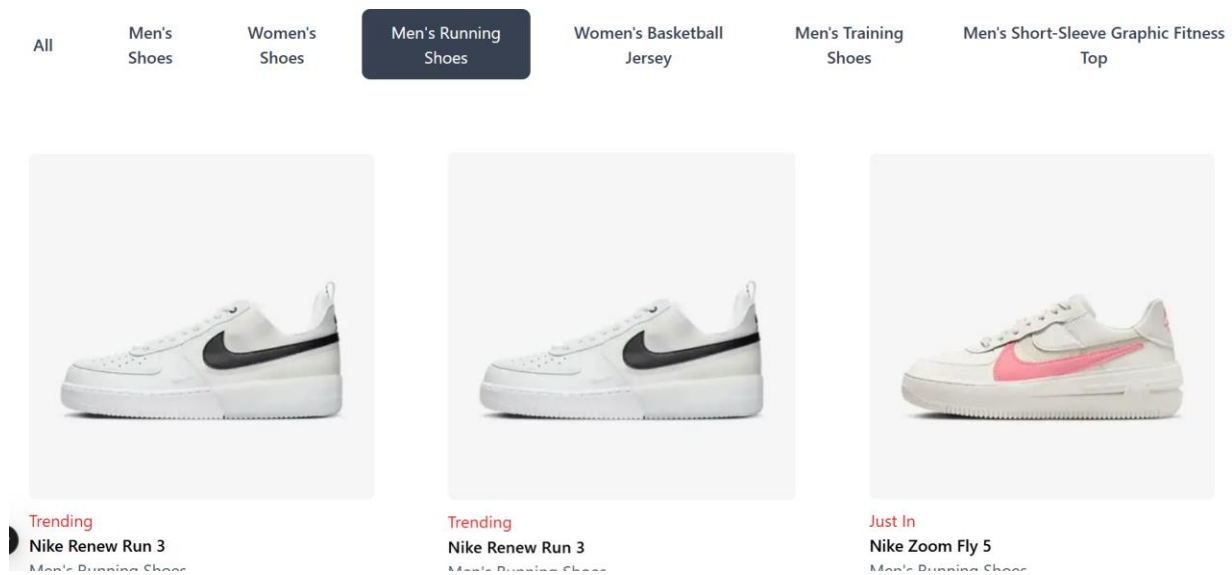
Inventory: 20

MRP: ₹ 10795.00

 Add To Cart

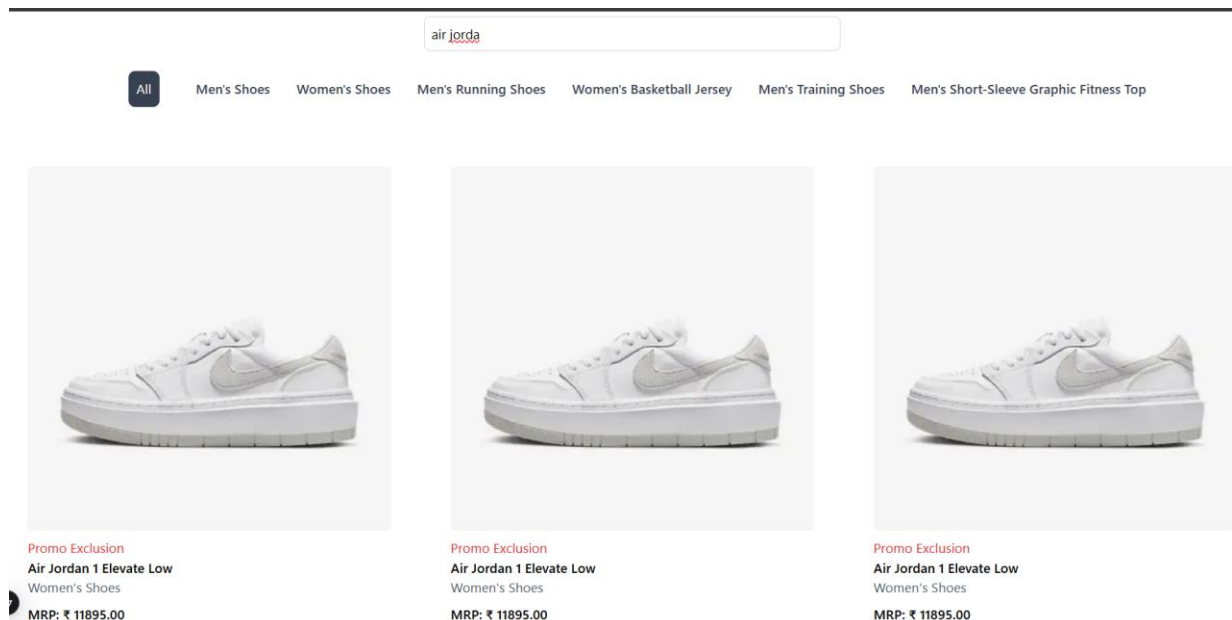
### Category Filter:

- **Category Buttons:** Clicking on a category filter button will fetch and display products belonging to that category.
- **Dynamic Filtering:** The displayed products are filtered dynamically based on the selected category.



## Search Bar:

- **Real-Time Search:** The search filters the displayed products as the user types.
- **State Management:** The query input is managed using React hooks (useState).



## Pagination:

- **Page Navigation:** Users can navigate between pages of products using "Previous" and "Next" buttons.

- **Dynamic Rendering:** Only a subset of products is rendered per page to improve performance.



Just In

Nike Air Force 1 PLT.AF.ORM

Women's Shoes

MRP: ₹ 8695.00

Prev

3 - 5

Next

## Code Deliverables:

### Product Listing Page with Dynamic Data:

```
<div className="p-6 flex flex-col transform transition-all hover:scale-105" key={product._id}>
  <Link href={` /products/${product._id}`}>
    <div className="flex justify-center items-center w-full h-full">
      <Image
        src={product.imageUrl}
        alt={product.productName}
        className="object-cover rounded-md"
        height={500}
        width={500}
      />
    </div>
  </Link>
  <p className="text-red-600 mt-2">{product.status}</p>
  <h2 className="text-md font-semibold text-black">{product.productName || "No name available"}</h2>
  <p className="text-gray-600">{product.category}</p>
  <p className="text-md font-semibold mt-2 text-black">MRP: ₹ {product.price.toFixed(2)}</p>
</div>
))
```

## Product Detail Page:

```
if (id) {
  const fetchProductDetails = async () => {
    try {
      const response = await fetch(`/api/products/${id}`, { method: 'GET' });
      if (!response.ok) throw new Error(`Error fetching product: ${response.statusText}`);
      const data = await response.json();
      setProduct(data);
    } catch (error) {
      console.error('Error fetching product details:', error);
      setError('Failed to load product details');
    } finally {
      setLoading(false);
    }
  };
  fetchProductDetails();
}, [id]);
```

## Category Filter:

```
const CategoryFilter = ({ categories, selectedCategory, onCategoryChange }: CategoryFilterProps) => {
  return (
    <div className="flex justify-center mb-5 pb-5 space-x-4">
      {[ "All", ...categories ].map((category) => (
        <button
          key={category}
          className={`btn btn-outline-dark ${selectedCategory === category ? "bg-gray-700 text-white" : "text-black"} py-2 px-4 rounded-lg transition`}
          onClick={() => onCategoryChange(category)}
        >
          {category}
        </button>
      ))}
    </div>
  );
};

export default CategoryFilter;
```

## Search Bar:

```
return (  
  <div>  
    { /* Search Bar */}  
    <div className="mb-4 flex justify-center">  
      <input  
        type="text"  
        value={searchQuery}  
        onChange={handleSearchChange}  
        placeholder="Search for products..."  
        className="border ■ border-gray-300 rounded-lg p-2 w-1/3"  
      />  
    </div>  
  </div>  
)
```

## Pagination:

```
{ /* Pagination Controls */}  
<div className="flex justify-center mt-6">  
  <button  
    className="px-4 py-2 ■ bg-gray-500 ■ hover:bg-neutral-100 font-medium rounded mr-2"  
    onClick={() => handlePageChange(currentPage - 1)}  
    disabled={currentPage === 0}  
  >  
    Prev  
  </button>  
  <span className="px-4 py-2">  
    {currentPage + 1} - {totalPages}  
  </span>  
  <button  
    className="px-4 py-2 ■ bg-gray-500 ■ hover:bg-neutral-100 font-medium rounded ml-2"  
    onClick={() => handlePageChange(currentPage + 1)}  
    disabled={currentPage === totalPages - 1}  
  >  
    Next  
  </button>  
</div>  
</div>
```

## Challenges and Solutions:

1. **Challenge:** Integrating dynamic data fetching with React components.  
**Solution:** Used `useEffect` to trigger data fetching on component mount and updated the state with the response using `useState`.
2. **Challenge:** Handling large product lists and optimizing performance.  
**Solution:** Introduced pagination to reduce the load time by displaying smaller subsets of data at a time.
3. **Challenge:** Dynamic routing for individual product pages.



**Solution:** Leveraged React Router's `useParams` hook to retrieve the correct product data based on the dynamic route.

### **Best Practices:**

1. **Component Reusability:** Used reusable components like `ProductCard` and `ProductList` to keep the code modular.
2. **Efficient State Management:** Utilized React hooks (`useState` and `useEffect`) for clean state and side-effect management.
3. **Separation of Concerns:** Ensured each component focused on a single responsibility for easier debugging and maintenance.