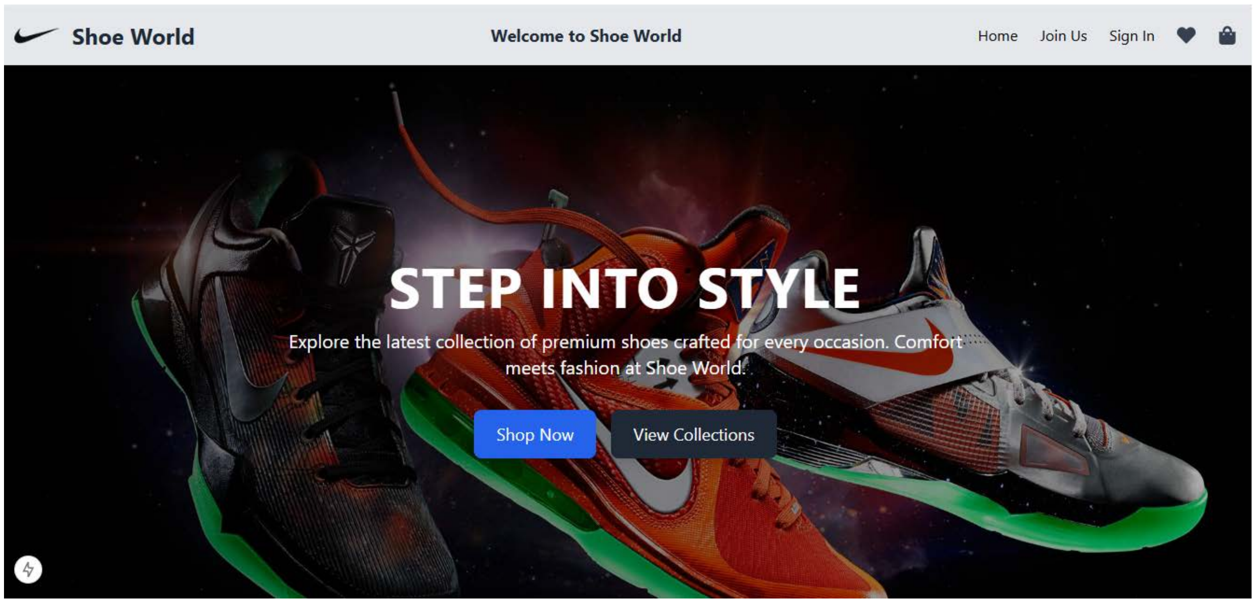
Prepared By : Shahzeena Samad

Date : 19th jan 2025


Shoe World
Welcome to Shoe World
Home
Join Us
Sign In
STEP INTO STYLE
Explore the latest collection of premium shoes crafted for every occasion. Comfort meets fashion at Shoe World.
Shop Now
View Collections

Day 4 - Dynamic Frontend Components - [SHOE WORLD]

Overview:

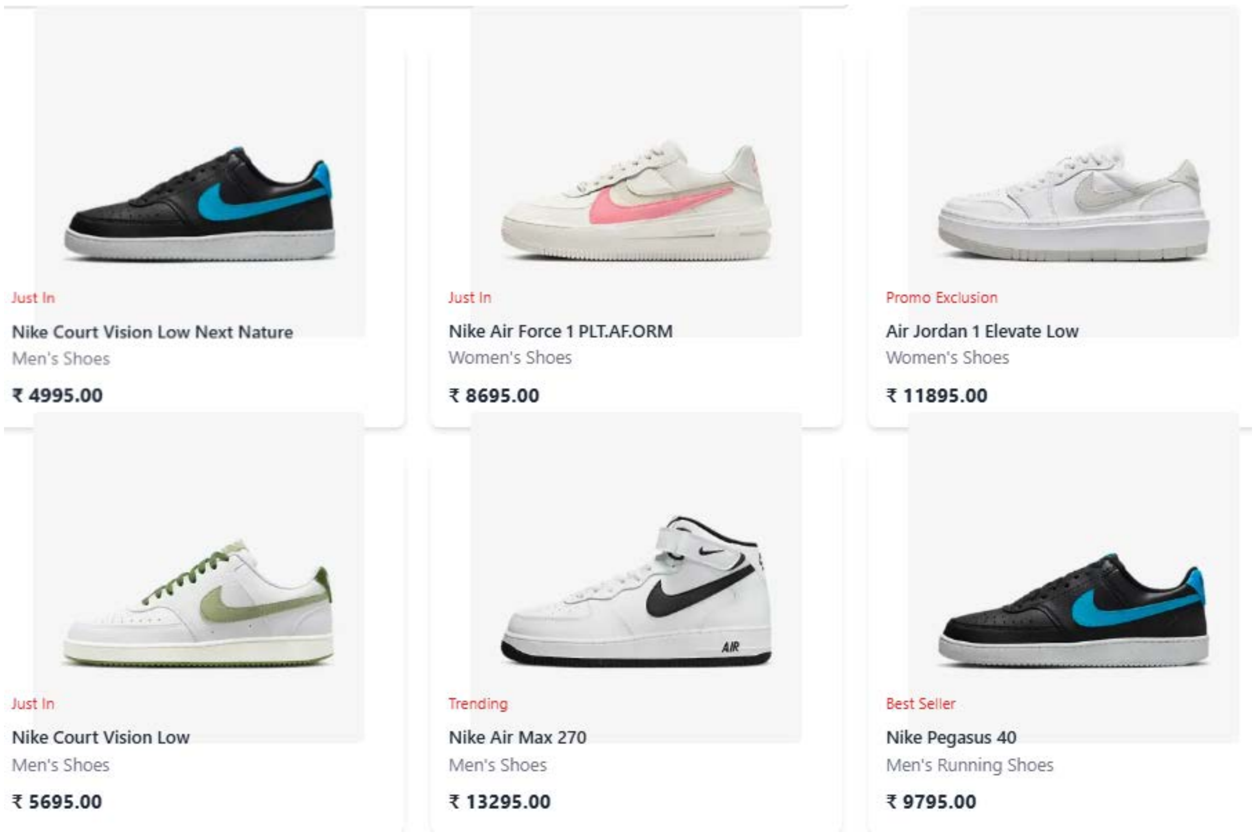
This report highlights the development of dynamic frontend components for the "Shoe World" eCommerce platform. The primary aim of the project was to create an engaging, seamless shopping experience with real-time data fetching, dynamic routing, and responsive design. Key features include product listing pages, detailed product pages, category filters, search functionality, and pagination, all built to optimize the user interface and enhance interaction.

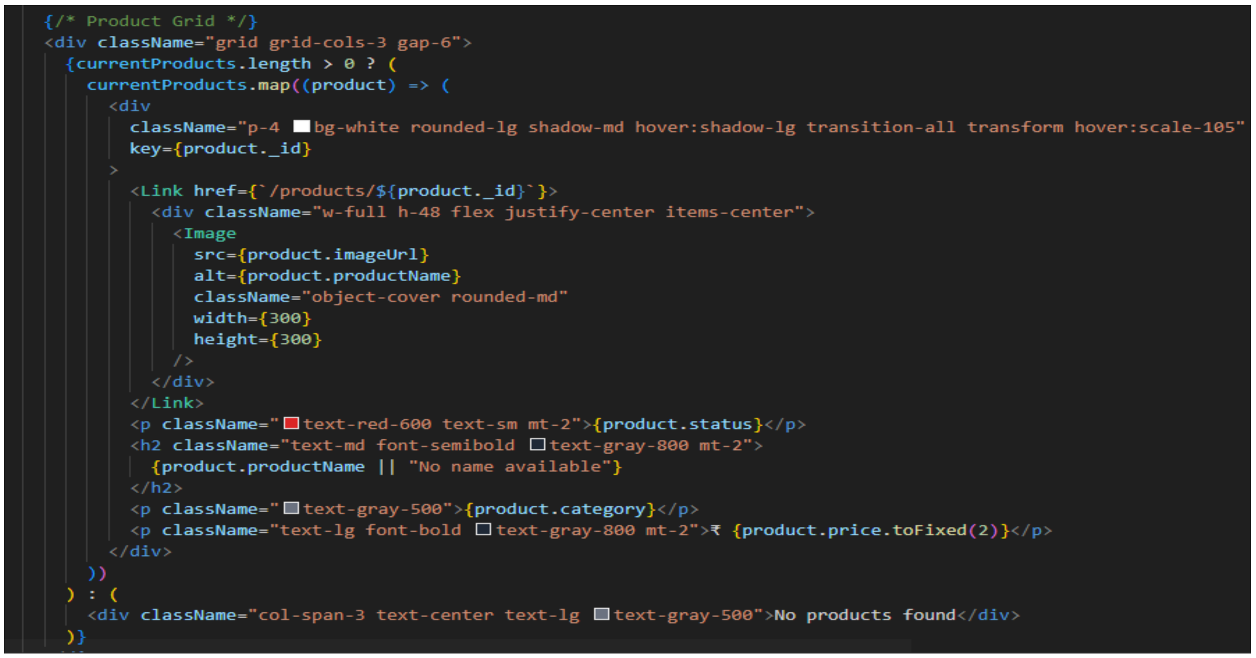
Project Deliverables:

1. **Dynamic Product Listing Page:**

    - Designed to display an updated list of products fetched dynamically from an API.

- Each product is represented by a card showing key information such as product name, price, and image.
- The listing page allows users to browse products efficiently and view a variety of footwear options.

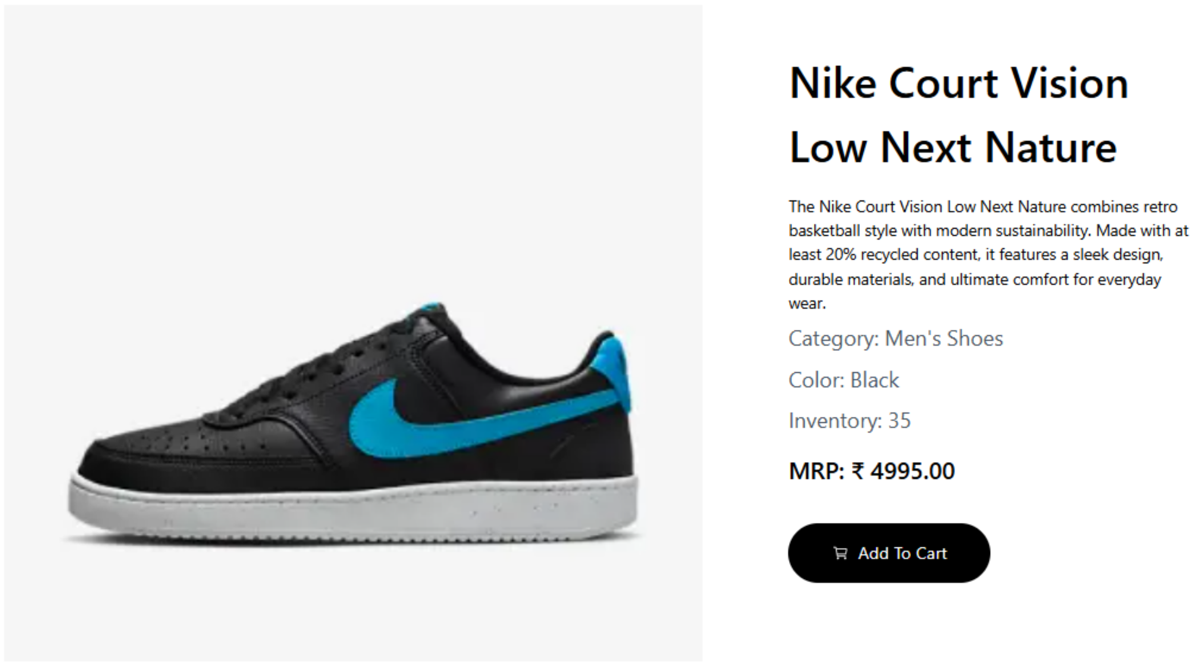




```
{/* Product Grid */}
<div className="grid grid-cols-3 gap-6">
  {currentProducts.length > 0 ? (
    currentProducts.map((product) => (
      <div
        className="p-4 bg-white rounded-lg shadow-md hover:shadow-lg transition-all transform hover:scale-105"
        key={product._id}
      >
        <Link href={`/products/${product._id}`}>
          <div className="w-full h-48 flex justify-center items-center">
            <Image
              src={product.imageUrl}
              alt={product.productName}
              className="object-cover rounded-md"
              width={300}
              height={300}
            />
          </div>
        </Link>
        <p className="text-red-600 text-sm mt-2">{product.status}</p>
        <h2 className="text-md font-semibold text-gray-800 mt-2">
          {product.productName || "No name available"}
        </h2>
        <p className="text-gray-500">{product.category}</p>
        <p className="text-lg font-bold text-gray-800 mt-2">₹ {product.price.toFixed(2)}</p>
      </div>
    ))
  ) : (
    <div className="col-span-3 text-center text-lg text-gray-500">No products found</div>
  )}
```

2. Product Detail Pages:

- Each product links to a dedicated product detail page with specific information like description, pricing, specifications, and product images.
- The page is dynamically routed using React Router, fetching detailed product data based on the product's unique ID.



Nike Court Vision Low Next Nature

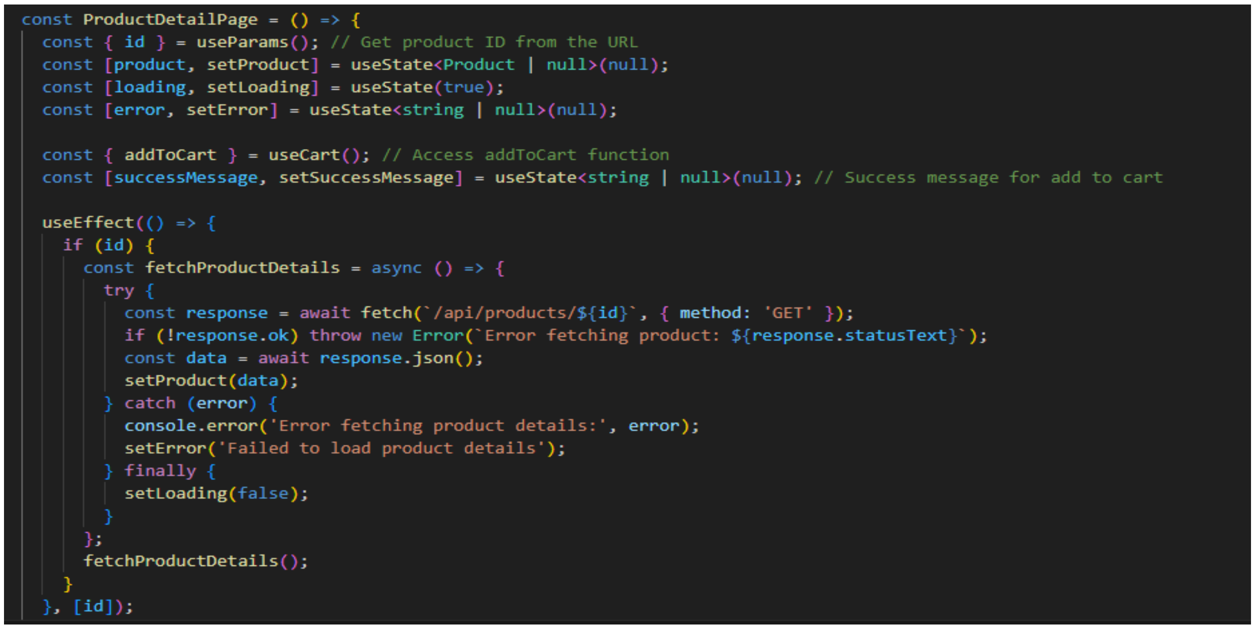
The Nike Court Vision Low Next Nature combines retro basketball style with modern sustainability. Made with at least 20% recycled content, it features a sleek design, durable materials, and ultimate comfort for everyday wear.

Category: Men's Shoes

Color: Black

Inventory: 35

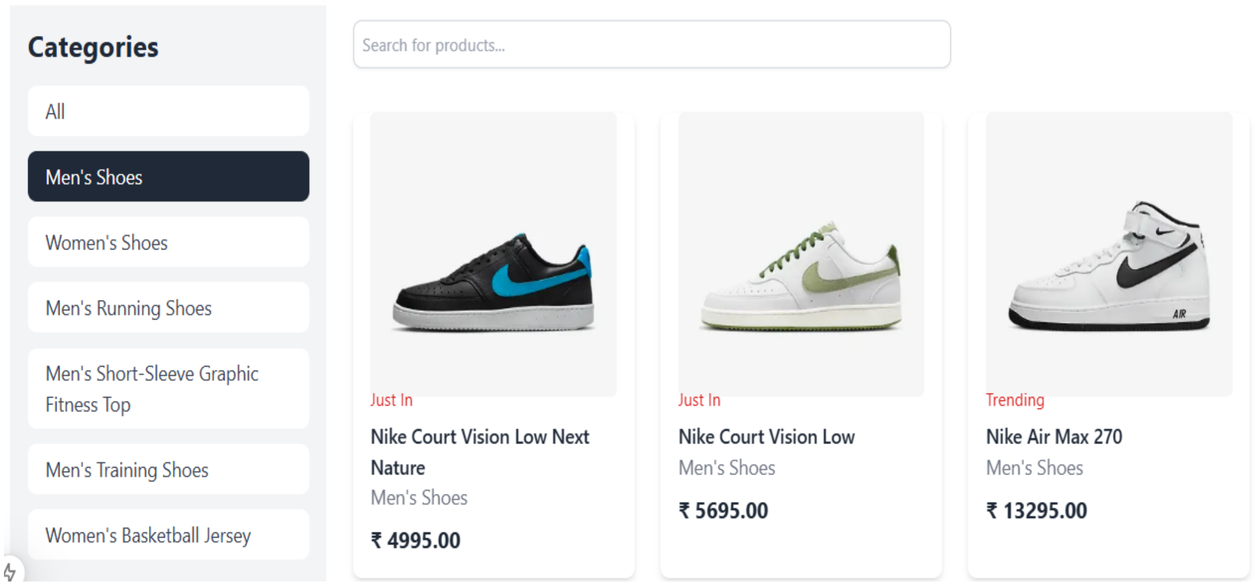
MRP: ₹ 4995.00

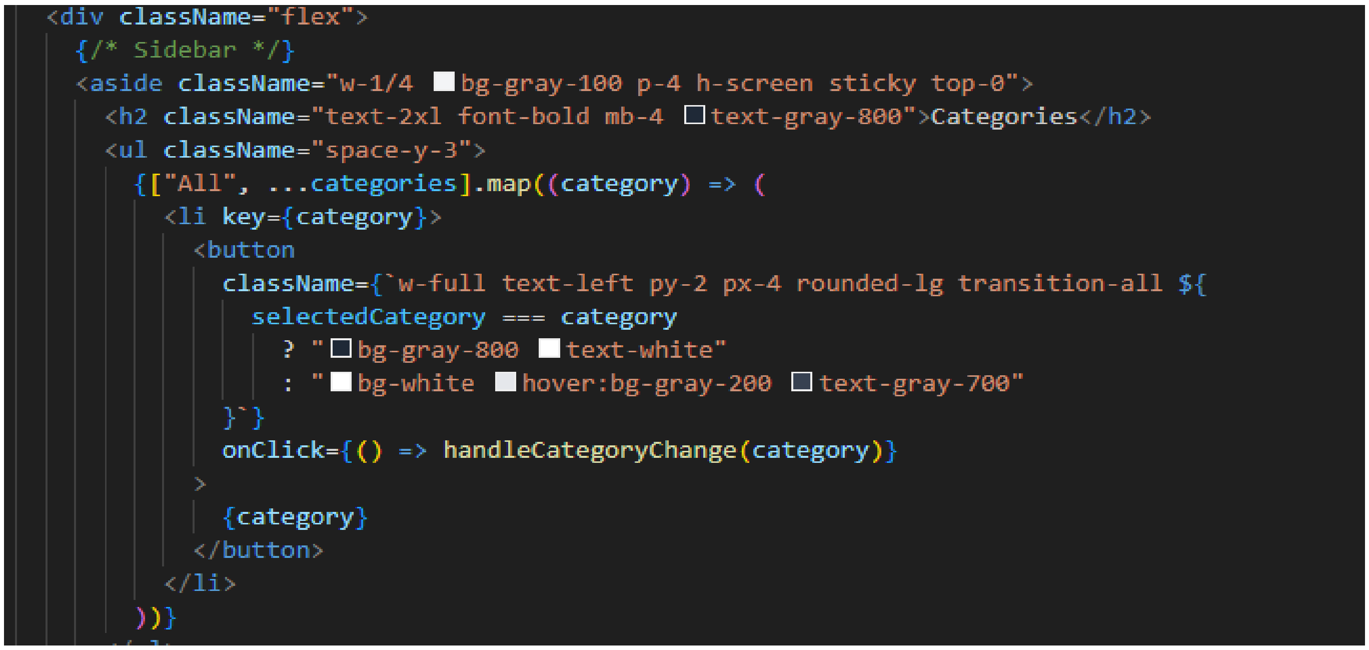


```
const ProductDetailPage = () => {
  const { id } = useParams(); // Get product ID from the URL
  const [product, setProduct] = useState<Product | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  const { addToCart } = useCart(); // Access addToCart function
  const [successMessage, setSuccessMessage] = useState<string | null>(null); // Success message for add to cart

  useEffect(() => {
    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]);
```

3. Category Filters:

- Users can filter products by categories such as "Men’s Running Shoes," "Women’s Sneakers," and more.
- Clicking on a category filter button dynamically updates the product list to show only the items within that category.

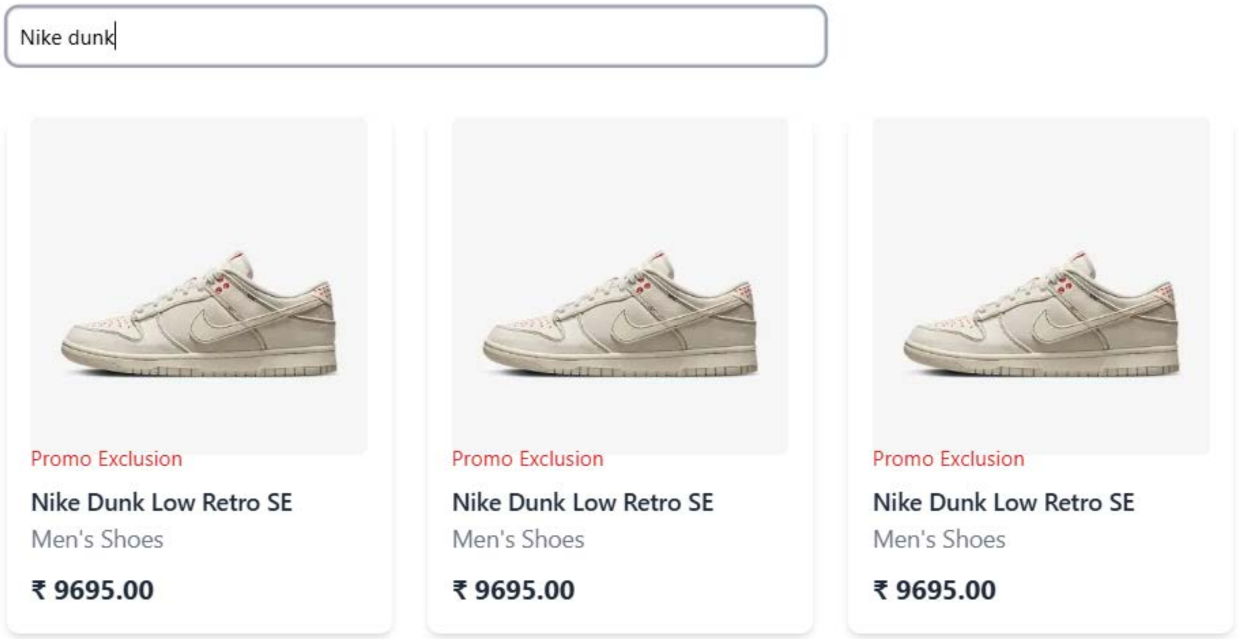


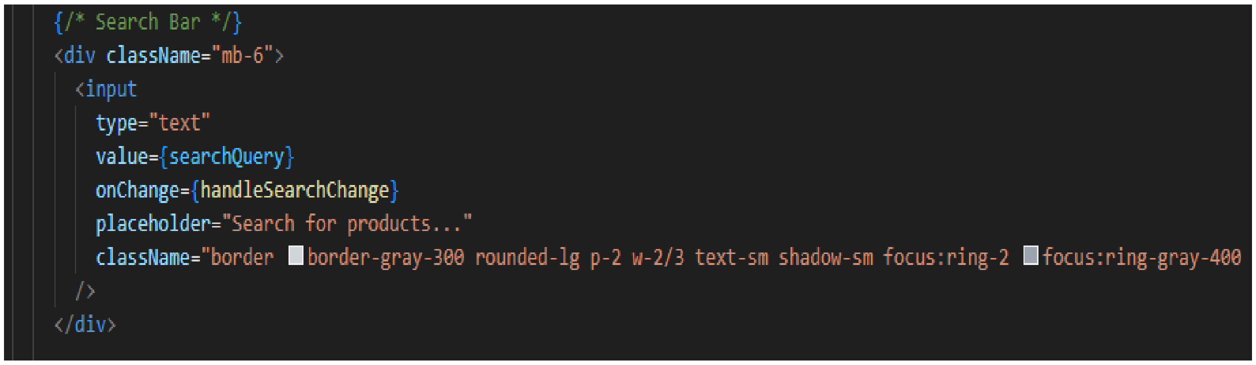


```
<div className="flex">
  {/* Sidebar */}
  <aside className="w-1/4 bg-gray-100 p-4 h-screen sticky top-0">
    <h2 className="text-2xl font-bold mb-4 text-gray-800">Categories</h2>
    <ul className="space-y-3">
      {["All", ...categories].map((category) => (
        <li key={category}>
          <button
            className={`w-full text-left py-2 px-4 rounded-lg transition-all ${
              selectedCategory === category
                ? "bg-gray-800 text-white"
                : "bg-white hover:bg-gray-200 text-gray-700"
            }`}
            onClick={() => handleCategoryChange(category)}
          >
            {category}
          </button>
        </li>
      ))}
```

4. **Search Bar for Real-Time Product Search:**

- The search bar allows users to search for products by name, description, or other relevant keywords.
- As users type, the list of products updates instantly based on the search query using state management with React hooks.

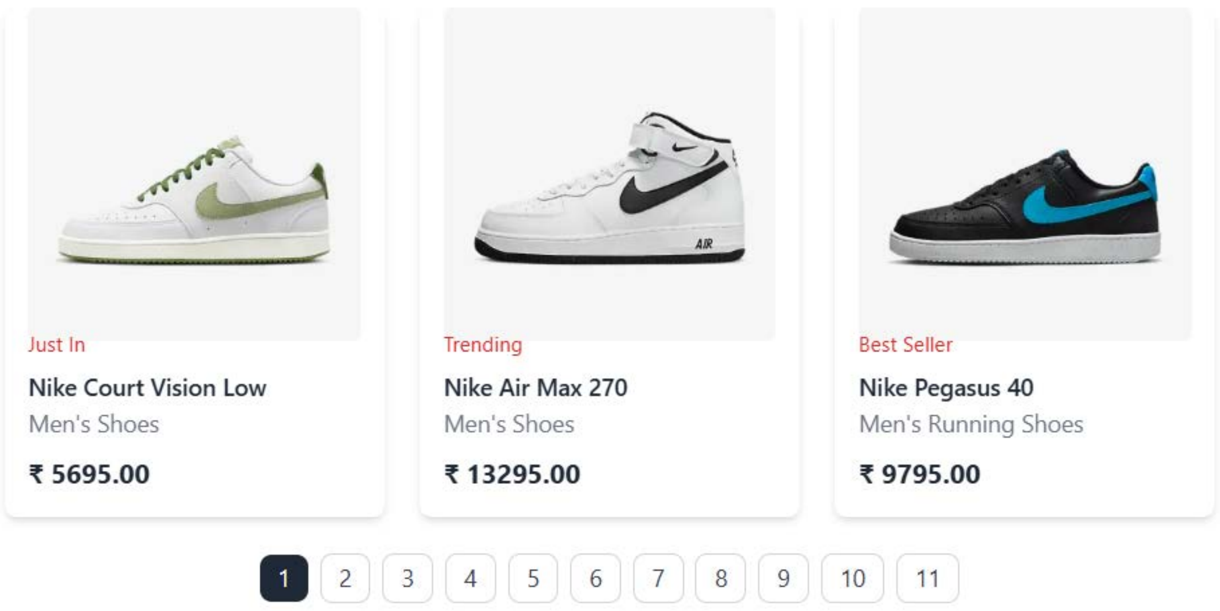


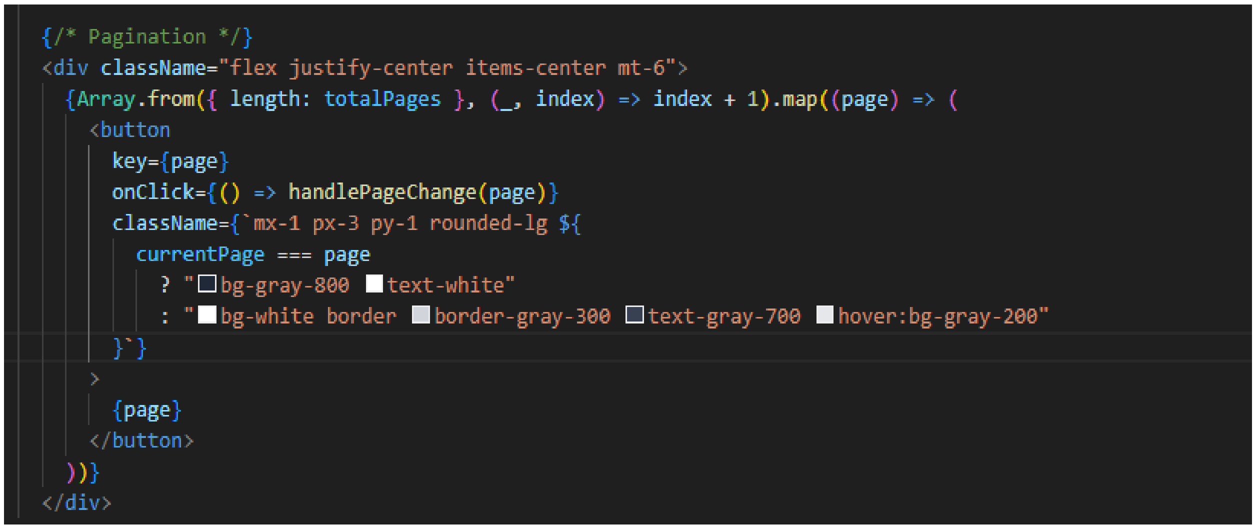


```
{/* Search Bar */}
<div className="mb-6">
  <input
    type="text"
    value={searchQuery}
    onChange={handleSearchChange}
    placeholder="Search for products..."
    className="border border-gray-300 rounded-lg p-2 w-2/3 text-sm shadow-sm focus:ring-2 focus:ring-gray-400
  />
</div>
```

5. Pagination for Optimized Navigation:

- Pagination breaks large lists of products into smaller, more manageable sections.
- Users can navigate through pages using "Next" and "Previous" buttons, enhancing the browsing experience and improving page load times.





```
{/* Pagination */}
<div className="flex justify-center items-center mt-6">
  {Array.from({ length: totalPages }, (_, index) => index + 1).map((page) => (
    <button
      key={page}
      onClick={() => handlePageChange(page)}
      className={`mx-1 px-3 py-1 rounded-lg ${
        currentPage === page
          ? "bg-gray-800 text-white"
          : "bg-white border border-gray-300 text-gray-700 hover:bg-gray-200"
      }`}
    >
      {page}
    </button>
  ))}
</div>
```

Code Implementation Highlights:

- **API Integration & Dynamic Data Rendering:**

  The products are fetched from an API using React's useEffect hook, ensuring the listing page always displays the most up-to-date information.

- **Modular Components:**

  Components like ProductCard, ProductListing, and CategoryFilter are reusable to ensure maintainability and a clean codebase.

- **Dynamic Routing with React Router:**

  React Router's useParams hook is used to dynamically route users to specific product pages based on their unique ID.

- **State Management with React Hooks:**

  useState and useEffect are utilized for managing search queries, filtering, and handling API responses in real time.

Challenges and Solutions:

1. **Challenge:**

   Managing real-time data fetching and ensuring product data stays updated.

   - **Solution:** Employed useEffect for triggering API calls during component mounting, ensuring the product data is always current.
2. **Challenge:**

   Handling large product datasets and maintaining fast page loads.

   - **Solution:** Introduced pagination to break the product listings into smaller chunks, improving performance and user experience.
3. **Challenge:**

Implementing dynamic routing for each product's detail page.

- **Solution:** Used React Router’s useParams hook to dynamically load product details based on the unique product ID.

Best Practices:

1. **Reusability of Components:**

   Components were built to be reusable, making the code modular and reducing redundancy across the platform.

2. **Efficient State Handling:**

   React hooks allowed for clean management of state, keeping the interface responsive and interactive.

3. **Modular Code Structure:**

   By ensuring that each component handles one specific task, the application remains easy to debug and maintain.

Conclusion:

In conclusion, the dynamic frontend components for the Shoe World eCommerce platform successfully enhanced the overall user experience by providing responsive, real-time product listings, seamless navigation, and efficient filtering and searching capabilities. Through the use of modern React features like hooks and routing, along with strategies like pagination and dynamic data fetching, the project addressed common challenges and optimized performance. The result is a highly interactive, user-friendly platform that ensures users can easily browse and purchase products. With modular components and clean code practices, the website is well-positioned for future enhancements and scalability.