College Code : 9509

College Name : Holy Cross Engineering College

Department : Computer Science Engineering

Student NM id :55B8986C89DFB599FDD4B45A587D1632

Roll No : 950923104019

Date : 06.10.2025

Project Name : IBM-FE-Product Catalog with Filters

Completed the project named as

Sumbitted By,

John Samuel J

9894703759

1. Introduction

In the rapidly evolving digital age, e-commerce platforms have transformed the way people shop. One of the key features of any online shopping website is the **product catalog** that enables customers to **view, search, and filter** products efficiently.

The project titled **"Product Catalog with Filters"** focuses on creating a simple yet functional e-commerce front-end application that demonstrates these features using **React**, **Vite**, and **TailwindCSS**.

The application allows users to:

- View a list of available products.
- Apply dynamic filters such as category, price, and brand.
- View product details in an organized and visually appealing layout.
- Experience seamless filtering without reloading the page.

This project emphasizes **front-end design, state management, responsiveness, and usability**, giving an insight into how real-world e-commerce websites are structured.

---

2. Project Objectives

The "Product Catalog with Filters" project aims to design and develop a user-friendly, responsive, and dynamic e-commerce front-end web application using React and TailwindCSS. The project objectives have been defined to cover both technical learning outcomes and user experience goals.

Below are the detailed objectives with explanations:

Objective 1: To develop a responsive and interactive product catalog interface using React and TailwindCSS

The first goal of this project is to build a modern, responsive web interface that displays a collection of products in a visually appealing and organized layout.
Using React, the application is divided into reusable components such as Header, ProductList, ProductCard, and FilterBar. This modular structure allows scalability and reusability of code.

To ensure a seamless user experience across all devices, TailwindCSS is used for styling. It provides utility-first classes that simplify the process of creating adaptive designs.
The catalog automatically adjusts its grid layout for mobile, tablet, and desktop screen sizes, ensuring accessibility and ease of navigation for all users.

Objective 2: To implement dynamic filtering functionality for efficient product search and navigation

One of the main features of any e-commerce platform is the ability for users to filter and refine product results easily. This project's second key objective is to develop a real-time filtering system that allows users to view only the products that meet their selected criteria (e.g., category, price range, brand, or rating).

By leveraging React hooks (useState, useEffect), the filtering logic updates the product list dynamically — without reloading the page. The user interface responds instantly as filters are applied or cleared, ensuring an engaging and efficient browsing experience.

The filtering process involves:

1. Maintaining a global product list (in products.js or fetched from an API).
2. Tracking the selected filter values using useState.
3. Applying filter logic using Array.filter() methods.
4. Updating the UI automatically when state changes.

Objective 3: To enhance user experience through intuitive UI and clear product categorization

This project also focuses on user experience design (UX). The interface is intentionally clean and intuitive so that users can quickly understand how to interact with it.
Each product includes essential details such as the image, name, and price, presented with adequate spacing and visual hierarchy.

Users can navigate, browse, and apply filters with minimal effort — reducing cognitive load and enhancing usability.

Objective 4: To apply component-based architecture for scalability and maintainability

Using React's component-driven design, each part of the UI is developed independently and can be reused across multiple pages or modules.
This makes the project easy to maintain, extendable, and suitable for future upgrades — such as adding backend APIs or integrating shopping cart features.

Objective 5: To deploy the web application and maintain version control

The final goal is to host the project online for public access and demonstration.
Using platforms like GitHub Pages, Netlify, or Vercel, the project is deployed with a live link that showcases its functionality.
Git and GitHub are used throughout development for version control, ensuring that every update is properly tracked and documented.

3. Tools and Technologies Used

Category	Tool / Framework	Purpose / Usage
Frontend Framework	React (with Vite)	Building user interface and component-based structure
CSS Framework	TailwindCSS	Styling and responsive design
Language	JavaScript (ES6)	Functional logic, event handling
Package Manager	npm (Node Package Manager)	Managing dependencies
IDE	Visual Studio Code	Writing and testing code
Version Control	Git & GitHub	Code versioning and collaboration
Deployment Platform	Netlify / Vercel	Hosting and deployment
Browser Tools	Chrome DevTools	Debugging and testing responsiveness

---

4. System Design and Architecture

System Overview

The project follows a **client-side architecture** built entirely with React.
It is divided into several reusable components that handle different responsibilities.

High-level design:

App.jsx

├— Header.jsx

├— FilterBar.jsx

├— ProductList.jsx

| └— ProductCard.jsx

├— Footer.jsx

└— data/products.js

Each component performs a specific role:

- **Header:** Displays the project title or logo.
- **FilterBar:** Contains dropdowns and buttons for category or price filters.
- **ProductList:** Dynamically displays products based on selected filters.
- **ProductCard:** Renders individual product details.
- **Footer:** Displays copyright.

Data Flow

1. The base product data is stored in a JavaScript file (products.js).
2. React's **useState** and **useEffect** hooks manage dynamic filtering.
3. When a user selects a category or price range, the filter state updates.
4. The updated state triggers a re-render, showing filtered results instantly.

UI and UX Design

The UI design focuses on:

- **Simplicity:** Clean, minimal interface.
- **Responsiveness:** Adjusts automatically to mobile and desktop screens.
- **Usability:** Filters and buttons are easy to understand and accessible.

The UX ensures that users can filter products **without page reloads**, providing a smooth browsing experience.

---

5. Implementation Details

Step 1: Setting Up the Environment

npm create vite@latest e-commerce-platform --template react

cd e-commerce-platform

npm install

Initialize TailwindCSS:

npm install -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

Then configure the tailwind.config.js file to include:

content: ["./index.html", "./src/**/*.{js,jsx}"],

Step 2: Creating the Data File

File: src/data/products.js

```
export const products = [
  {
    id: 1,
    name: "Wireless Headphones",
    category: "Electronics",
    price: 1299,
    image: "/images/headphones.jpg",
  },
  {
    id: 2,
    name: "Cotton T-Shirt",
    category: "Clothing",
    price: 499,
    image: "/images/tshirt.jpg",
  },
  ...
];
```

Step 3: Building the Core Components

Header.jsx

Displays the app logo and title.

```
<header className="bg-white shadow-md p-4 text-2xl font-bold text-center text-indigo-600">
  Product Catalog with Filters
</header>
```

FilterBar.jsx

Contains filter controls:

```
<select
  value={selectedCategory}
  onChange={(e) => setSelectedCategory(e.target.value)}
  className="border rounded-md px-3 py-2"
>
  <option value="All">All</option>
  <option value="Electronics">Electronics</option>
  <option value="Clothing">Clothing</option>
  <option value="Groceries">Groceries</option>
</select>
```

ProductCard.jsx

Displays product image, name, and price:

```
<div className="shadow-md p-4 rounded-lg">
  <img src={product.image} alt={product.name} className="w-full h-48 object-cover rounded" />
  <h3 className="text-lg font-semibold mt-2">{product.name}</h3>
  <p className="text-indigo-600 font-bold">₹{product.price}</p>
</div>
```

Step 4: Implementing Filtering Logic

In ProductList.jsx:

```
const [category, setCategory] = useState("All");
const [filtered, setFiltered] = useState(products);

useEffect(() => {
  if (category === "All") setFiltered(products);
  else setFiltered(products.filter(p => p.category === category));
}, [category]);
```

Step 5: Testing and Validation

Test Scenarios:

1. Selecting “Electronics” should show only electronic items.
2. “All” should reset and show every product.
3. UI should adapt correctly to mobile screens.
4. No reload occurs while filtering.

Step 6: Deployment

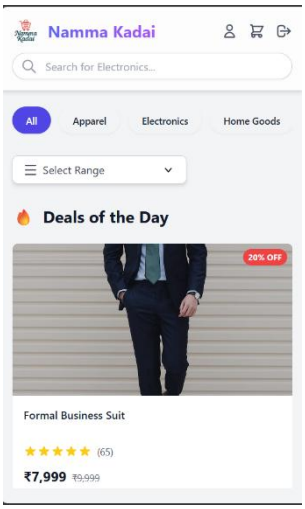
1. Build the production version:
2. npm run build
3. Deploy on **Netlify** or **Vercel**.
4. Push the source code to **GitHub** and link it with the deployment URL.

---

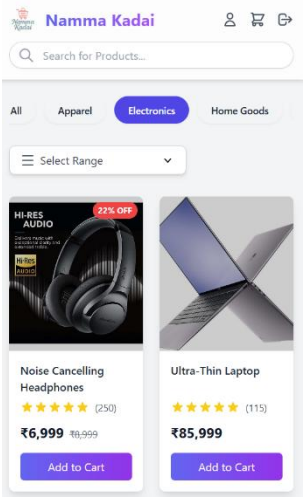
5. Screenshots

Home Page

without apply any Filters



With apply Filters



---

7. Challenges and Solutions

Challenge	Solution / Approach
Managing multiple filters simultaneously	Used combined state objects and useEffect hooks to synchronize updates
State re-rendering issues	Optimized with dependency arrays and memoized components
Tailwind responsive design issues	Applied responsive classes (sm:, md:, lg:) for layout consistency
Hosting image assets	Used public folder to store product images accessible by Vite

---

8. Results and Output

The final application successfully:

- Displays a dynamic product list.
- Filters products instantly based on user selections.
- Runs smoothly across all screen sizes.
- Demonstrates clean design and modular React code.

The deployed application link and source code are available on GitHub.

---

9. Conclusion

The **Product Catalog with Filters** project demonstrates how to create a dynamic and user-friendly front-end e-commerce interface. It provides a real-world example of **modern web development** practices using **React** and **TailwindCSS**.

During development, I learned how to:

- Build and structure React components.
- Manage dynamic data using hooks and props.
- Design responsive layouts using TailwindCSS utilities.
- Deploy React applications using Netlify and GitHub.

This project strengthened my understanding of **frontend frameworks**, **state management**, and **UI responsiveness**, and it serves as a strong foundation for future e-commerce or product-based web applications.

10. Future Enhancements

- Integrate real backend API (Node.js or Firebase).
- Add “Add to Cart” and “Wishlist” features.
- Include sorting (low-high, high-low).
- Implement user authentication and login system.
- Add reviews and product rating system.