College Code : 9509

College Name : Holy Cross Engineering College

Department : Computer Science Engineering

Student NM id :E11EF02A38085BF808C481335F5D08E1

Roll No : 950923104027

Date : 22.09.2025

Project Name : IBM-FE-Product Catalog with Filters

Completed the project named as

Sumbitted By,

Mareeswaran C

9150430858

Introduction

This document outlines the strategic plan for Phase 3: the implementation of a Minimum Viable Product (MVP) for a "Product Catalog with Filters." The primary objective is to develop a functional, core version of the application within a strict 8-week deadline. This plan breaks down the project into five key areas: Project Setup, Core Features Implementation, Data Storage, Testing, and Version Control. By following this structured approach, we will ensure a focused development process, prioritizing essential features to deliver a high-quality, testable product by Week 8.

Project Setup (Week 1)

1. **Choose Your Stack**: For rapid development, a component-based framework is ideal.
    - **Framework**: **React** (using Create React App) or **Next.js** for a more robust foundation.
    - **Styling**: **Tailwind CSS** or a component library like **Material-UI (MUI)** for pre-built, customizable UI elements.
    - **Language**: **TypeScript** for type safety, which helps catch errors early.
2. **Initialize Project**:
    - Open your terminal and run the command for your chosen framework.
        - For Next.js + TypeScript + Tailwind CSS (Recommended):

    Bash

    npx create-next-app@latest product-catalog-mvp --ts --tailwind --eslint
3. **Structure Your Folders**: Create a logical folder structure inside your src or root directory.
4. /src
5. ├── /components # Reusable UI components (ProductCard, FilterSidebar, etc.)
6. ├── /pages # Main pages/views (HomePage, ProductDetailPage)
7. ├── /data # Mock product data (e.g., products.json)
8. ├── /hooks # Custom React hooks (e.g., useFilters)

9. └── /styles # Global CSS styles

Core Features Implementation (Weeks 2-4)

The goal is to build the essential user-facing features.

1. **Product Display**:
    - **ProductCard Component**: Create a component to display a single product's image, name, price, and a short description.
    - **ProductList Component**: Create a grid or list view that maps over your product data and renders a ProductCard for each item.
2. **Filtering Logic**:
    - **FilterSidebar Component**: Build a sidebar with various filter options. For the MVP, include:
        - **Search Bar**: A text input to filter products by name.
        - **Category Filter**: Checkboxes or a dropdown to select product categories (e.g., "Electronics," "Apparel").
        - **Price Range Filter**: A slider or two input fields (min/max) to filter by price.
    - **State Management for Filters**: Use React's useState hook at the parent component level (e.g., HomePage) to manage the active filter values.
3. **Connecting Filters to the List**:
    - In the parent component, create a function that filters the master product list based on the current state of the filters.
    - Pass the filtered list of products as a prop to your ProductList component. The list will re-render automatically whenever the filter state changes.

Data Storage (Week 5)

For an MVP, local data is sufficient and fast to implement.

1. **Local State**:

- **Mock Data**: Create a products.json file in your /data directory. Populate it with 15-20 sample products, ensuring they have properties that match your filters (e.g., name, price, category).

JSON

```
[
  {
  "id": 1,
   "name": "Wireless Headphones",
   "price": 99.99,
  "category": "Electronics",
  "imageUrl": "..."
  },
  ]
```

- **Data Fetching**: In your main page component, import this JSON file directly or use a useEffect hook to "fetch" and load the data into the component's state when it first mounts.

2. **State Management**:
   - For this MVP, React's built-in useState and useContext hooks are perfect.
   - useState: Manage the list of all products, the filtered products, and the active filter settings.
   - useContext (Optional): If you need to share filter state across deeply nested components without passing props down manually ("prop drilling"), the Context API is a great solution.

Testing Core Features (Week 6)

Focus on ensuring the most critical functionalities work as expected.

1. **Setup Testing Environment**:

- Use **Jest** as the test runner and **React Testing Library** for testing components from a user's perspective. These are often included by default with create-react-app or create-next-app.

2. **What to Test**:
    - **Component Rendering**: Does the ProductList render the correct number of ProductCard components?
    - **Filtering Logic**:
        - When a user types in the search bar, does the product list update to show only matching items?
        - When a category checkbox is clicked, are only products from that category displayed?
        - Does the price range filter correctly exclude products outside the selected range?
    - **User Interactions**: Simulate user events like clicks and text input to verify the application's response.

Version Control (Ongoing from Week 1)

Proper use of Git and GitHub is crucial for tracking changes and collaboration.

1. **Initialize Git Repository**:
    - If not already done by the project setup command, run git init in your project folder.
2. **Create a GitHub Repository**:
    - Go to GitHub and create a new, empty repository.
    - Follow the instructions to link your local repository to the remote one on GitHub.

      Bash

      git remote add origin <your-github-repo-url.git>

      git branch -M main

      git push -u origin main

3. **Branching Strategy (Git Flow)**:
    - **main Branch**: This branch should always represent your stable, production-ready code.
    - **develop Branch**: Create a develop branch from main. This will be your primary branch for integrating new features.
    - **Feature Branches**: For each new feature (e.g., feature/filter-sidebar, feature/product-grid), create a new branch from develop.

    Bash

    # From the develop branch

    git checkout -b feature/filter-sidebar

4. **Commit and Push Workflow**:
    - Work on your feature branch, making small, logical commits with clear messages (e.g., git commit -m "feat: create basic filter sidebar component").
    - Push your feature branch to GitHub regularly: git push origin feature/filter-sidebar.
    - When a feature is complete, open a **Pull Request (PR)** on GitHub to merge it into the develop branch. This allows for code review before merging.

Conclusion

By systematically executing the steps laid out in this plan, we will successfully deliver a robust MVP of the Product Catalog with Filters application by the Week 8 deadline. The focus on a modern tech stack, essential core features, and a clean local data model ensures rapid development and a solid foundation. The integrated testing and version control practices will maintain code quality and streamline the development workflow. This MVP will not only meet the immediate project requirements but will also serve as a scalable base for future enhancements, such as integrating with a live API, adding user authentication, or expanding filtering capabilities in subsequent phases.