COLLEGE CODE: 9504

COLLEGE NAME: DR G U POPE COLLEGE OF ENGINEERING

DEPARTMENT: CSE

STUDENT NM-ID: ADA87D0FB07244F94A4408108326B485

ROLL NO: 950423104036

DATE: 22/09/2025

Completed the project named as Phase 3

TECHNOLOGY PROJECT NAME : PRODUCT CATALOG WITH FILTERS

SUBMITTED BY,

NAME : SETRIC DINGSON.A

MOBILE NO: 9363378083

Product Catalog with Filters

MVP Implementation Plan

1. Project Setup

Thisphase involves laying the foundation for the development environment. The team selects React for the frontend, as it provides dynamic UI rendering and reusable components. For backend, Node.js with Express (or alternatively Django/Flask) is considered to handle APIs. Initially, data may be stored in local JSON or in-memory state, but future scalability allows MySQL or MongoDB.

Environment setup includes installing Node.js, Git, and required frameworks. Package manager (npm or yarn) ensures consistent dependency management. Configuration files such as .gitignore, README.md, and .env variables are also created.

The project follows a modular structure: components for UI elements, pages for different views, data folder for storing product information, and services for API calls.

Finally, a GitHub repository is initialized for version control. The first commit contains boilerplate project files, marking the official start of the implementation.

2. Core Features Implementation

The heart of the MVP liesindelivering theproductcatalog functionality. Product Catalog Display: A grid or list view presents product details including image, name, price, category, and a short description. This ensures customers quickly scan through available options. Filtering: Users can narrow down results using category filters (e.g., electronics, clothing, appliances) and price range filters (using sliders or numeric inputs). Sorting options are provided for price (low to high, high to low) and popularity. Search Feature: A keyword-based search bar is implemented. It supports case-insensitive and partial matches to improve usability.

Responsive Design: Catalog pages are optimized for mobile, tablet, and desktop ensuring consistent experience across devices.

3. Data Storage (Local State / Database)

TheMVP beginswithalocalstate approach,whereproduct data is stored in a JSON file or within React's state management system (useState/useReducer). This provides quick setup without external dependencies.

For extended implementation, a database layer can be introduced. SQL options like MySQL/PostgreSQL organize data into structured tables, while NoSQL databases like MongoDB allow flexible JSON document storage.

Backend APIs are proposed to make data dynamic: GET /products (all products), GET /products/:id (details of one product), and GET /products with filters (category, min, max price). This architecture allows easy scaling.

4. Testing Core Features

Testingensuresthe reliability and correctness of features.

Unit Testing: Using Jest, Mocha, or PyTest, individual functionalities are verified. Tests include rendering of products, filtering logic validation, sorting accuracy, and search functionality.
Manual Testing: The system is manually tested on multiple browsers and devices. Test cases include applying multiple filters, handling cases where no products match, and checking for broken images.

Bug Fixing: Identified issues are logged and resolved. This iterative cycle improves stability of the MVP before final submission.

5. Version Control (GitHub)

Version control is maintained using Git and GitHub. A branching strategy is followed where the main branch holds stable releases, dev is used for ongoing integration, and feature branches are created for new additions.

Commit messages follow best practices with prefixes like feat, fix, and test. This makes the history clean and understandable.
Optionally, GitHub Actions can be configured for automated testing and deployment pipelines, ensuring continuous integration and delivery.