COLLEGE CODE : 9623

COLLEGE NAME : Amrita College of Engineering And Technology

DEPARTMENT : Computer Science and Engineering

STUDENT NM-ID:736D7089F78C08CB69149C84D5EAE522

ROLL NO : 23CS057

DATE : 22-09-2025

Completed the project named as

Phase 3 — MVP Implementation

PROJECT NAME: PRODUCT CATALOG WITH FILTERS

SUBMITTED BY,

NAME : MATHESH I

MOBILE NO : 9042731728

Phase 3 — MVP Implementation

Topic: Product Catalog with Filters

1. Project Setup

The project is initialized using Node.js for backend REST API and React.js for frontend development. The folder structure includes separate directories for components, services, database models, and routes. GitHub is used for version control and collaborative development.

2. Core Features Implementation

a. Display of product catalog with name, image, description, and price.

b. Filter functionality by categories such as electronics, clothing, and groceries.

c. Sorting options like price (low to high, high to low) and popularity.

d. Search bar to find products by keywords.

e. Pagination for large product lists.

3. Data Storage (Local State / Database)

Product details are stored in a MongoDB database with fields like productID, name, category, price, description, imageURL, and stock availability. The frontend uses React state management to dynamically update filters and search results.

- **Purpose:** Store all product information persistently.

- **Data Model:** Each product is stored as a document with fields

such as productID, name, category, price, description, imageURL, and stock availability.

- **Filtering:** When users apply filters (e.g., by category, price range, availability), the backend translates these into MongoDB queries.

- **Querying:** MongoDB supports flexible querying, including range queries (price), equality (category), existence or quantity checks (stock > 0), and pattern matching (search by name).

- **Indexing:** To optimize performance, indexes are typically created on frequently filtered fields like category, price, and stock.

- **Result:** The database returns only the products matching the filter criteria, reducing the data sent to the frontend.

4. Testing Core Features

a. Unit testing of API endpoints for product retrieval and filtering.

b. Frontend testing of search and filter functions using React Testing Library.

c. Integration testing to ensure seamless communication between backend and frontend.

d. Manual testing of edge cases, such as empty product lists or invalid filters.

5. Version Control (GitHub)

GitHub repository is maintained with branches for development and production. Commits are pushed regularly with proper commit messages. Pull requests are reviewed and merged to ensure code quality. Issues and tasks are tracked using GitHub

Projects for smooth project management.

The Product Catalog with Filters project provides an intuitive interface for users to browse and find products easily. The combination of robust backend APIs and efficient frontend filtering ensures smooth user experience. Scalability is ensured by modular code structure, making it adaptable for future enhancements like advanced AI-based recommendations and personalized product suggestions.