COLLEGE CODE:9605

COLLEGE NAME:CAPE INSTITUTE OF TECHNOLOGY

DEPARTMENT:CSE 3rd year

STUDENT NM ID:9C92AF7F630ADF7A95D449AAE57E6CF3

ROLL NO:960523104058

DATE:27/10/2025

COMPLETE PROJECT NAME AS PHASE 5

TECHNOLOGY PROJECT NAME:

IBM FE PRODUCT CATALOGUE WITH FILTERS

SUBMITTED BY

NAME: NANTHITHA.S

REG NO: 960523104058

Product Catalog with Filters – Content

1. Introduction

A product catalog with filters is an essential feature in modern e-commerce platforms that allows users to browse, search, and sort products efficiently. The system provides an intuitive interface where products can be categorized, displayed, and refined using filter options such as price range, category, brand, rating, and availability. This improves user experience, reduces search time, and increases product discoverability.

The purpose of this module is to design and implement a dynamic and responsive product catalog that supports real-time filtering functionality. The system is built using web technologies such as HTML, CSS, and JavaScript, and can be integrated with a backend database for real products.

2. Objectives

To display a structured list of products in a catalog layout.

To implement filter options such as category, price, and rating.

To enhance usability using search and sorting features.

To build a responsive and scalable front-end interface.

To simulate real-time product filtering without refreshing the page.

3. System Requirements

3.1 Functional Requirements

The system shall display a list of products with images, price, and description.

The user shall filter products based on:

Category (e.g. Electronics, Clothing, Home Appliances)

Price Range (e.g. ₹0–₹500, ₹500–₹1000)

Rating (e.g. 3★ and above)

Brand (optional)

The system shall provide a search bar to search products by name.

The system shall dynamically update the catalog based on selected filters.

3.2 Non-functional Requirements

The interface must be responsive across different screen sizes.

The system should support smooth and fast filter performance.

Code readability and modularity should be maintained.

4. System Architecture

The product catalog uses a three-tier architecture:

Layer	Description
Presentation Layer	HTML, CSS used to display product grid and filter UI
Logic Layer	JavaScript handles filtering and search logic
Data Layer	Product data stored as JSON array or database

5. Technologies Used

Technology	Purpose

HTML5	Structure and layout
CSS3	Styling and responsiveness
JavaScript	Filter functionality
Bootstrap (optional)	Responsive layout
JSON	Temporary product storage
Node.js / Firebase / MySQL (optional)	Backend storage

6. Product Catalog Design

The UI consists of two main sections:

1. Filter Section – Shown on the left with checkboxes and input sliders.

2. Product Grid – Shows product cards with images and information.

Each product card includes:

Product Image

Product Name

Category

Price

Rating

7. Workflow of Filtering

1. User selects a filter (e.g. category = Electronics).

2. Event triggers the JavaScript filter function.

3. The product list is filtered based on selected criteria.

4. Updated results are displayed dynamically.

5. Filters can be reset to show all products.

8. Sample Use Case

Scenario: The user wants to buy a mobile under ₹15,000.

Steps:

1. Select Category: Electronics

2. Select Sub-category: Mobile Phones

3. Select Price Range: ₹10,000–₹15,000

4. View filtered results.

1. Final Demo Walkthrough – Product Catalog with Filters

The final demonstration of the Product Catalog with Filters project showcases the complete working prototype of an e-commerce-style product browsing interface. The system allows users to explore a list of products with the ability to apply multiple dynamic filters and search options for faster product discovery.

1.1 Demo Objective

The purpose of this demo is to:

Show the working user interface of the product catalog.

Demonstrate filtering and search functionality.

Prove system usability and responsiveness across devices.

Validate that all user requirements have been met.

1.2 System Launch

The application is launched using a web browser. Upon loading the homepage:

The header navigation bar displays links like Home, Products, About, and Contact.

The main screen shows a grid layout of featured products.

The filter panel appears on the left side of the screen.

1.3 User Interface Overview

Section	Description
Header	Contains navigation links and project branding/logo
Filter Panel (Left)	Includes filter options like Category, Price Range, Rating, Brand
Search Bar	Allows keyword-based product search
Product Grid	Displays product cards dynamically filtered
Footer	Displays copyright

1.4 Steps in Demo Execution

Step 1: Display All Products

The system initially displays all available products in a card-like format. Each card includes:

Image of the product

Product Name

Price

Rating

Category tag

Step 2: Category Filter Demonstration

From the filter panel, the user selects a specific category such as Electronics or Clothing.

The view updates instantly to show only products matching the selected category.

The demo highlights the use of JavaScript filtering logic.

Step 3: Price Range Filter

Next, the price slider or price checkboxes are used to narrow down products based on budget.

Example: Selecting ₹500 – ₹1500 filters products within that range.

Step 4: Rating Filter

The user selects “4★ & Above” filter.

Only top-rated products remain visible.

This improves decision-making for quality-conscious shoppers.

Step 5: Search Functionality

The search bar is used to find a specific product, for example: typing "Smart Watch".

The product list filters in real-time based on the search term.

This demonstrates string-matching functionality.

Step 6: Reset Filters

A "Clear Filters" button is clicked to remove all filter selections and restore the full product list.

This confirms that the system supports easy navigation and usability.

1.5 Responsive Design Check

The demo also includes checking responsiveness:

On tablet view, the filter panel collapses into a sidebar.

On mobile view, filters shift to a collapsible dropdown menu.

The product grid adjusts from 4 columns → 2 → 1 smoothly.

1.6 User Experience Highlights

Fast filtering without page reload (AJAX or JavaScript-based).

Clean and modern layout enhances user involvement.

Interactive and dynamic content improves the feel of a professional product catalog.

1.7 Conclusion of Demo

The final demo confirms that the Product Catalog with Filters system is: ✅ Fully functional

✅ User-friendly

✅ Scalable and easily extendable

✅ Ready for deployment or backend integration

This completes the final demonstration of the project successfully.

2. Project Report – Product Catalog with Filters

2.1 Introduction

The Product Catalog with Filters is a web-based application designed to simulate an e-commerce product browsing environment where users can search, view, and filter products efficiently. The catalog provides filtering capabilities based on parameters such as category, price range, rating, and brand. This feature helps users quickly locate products that match their preferences, enhancing user convenience and reducing browsing time. The system is developed using HTML, CSS, and JavaScript for the front end and can be connected to any backend service or database for real-world use.

2.2 Problem Statement

In large online product platforms, users face difficulty in searching through hundreds of catalog items. Without filtering options, product browsing becomes time-consuming and inefficient. There is a need for an intuitive and interactive product catalog that allows users to refine their choices based on relevant criteria.

2.3 Objectives of the Project

The key objectives of this project include:

Designing a user-friendly product catalog interface.

Implementing filter functionality for narrowing product search.

Allowing real-time filtering without page reload.

Providing a responsive layout for mobile and desktop users.

Ensuring scalability for future enhancements.

2.4 Scope of the Project

The scope of this project includes:

Displaying products with details like name, price, category, and rating.

Implementing multiple filter criteria.

Search feature based on product keywords.

Interactive and responsive design.

Can be integrated with backend systems using APIs in the future.

2.5 System Architecture

The system follows a modular architecture divided into three layers:

Layer	Description
Presentation Layer	HTML & CSS for UI design
Logic Layer	JavaScript for filter and search functions
Data Layer	JSON for storing product data (can connect to DB later)

2.6 Modules Description

The system comprises the following modules:

Module Name	Description
Homepage	Displays product catalog with filters
Filter Module	Filters products by category, price, brand, and rating
Search Module	Searches products by name
Product Display Module	Displays product cards dynamically
Responsive UI Module	Adjusts layout on mobiles and desktops

2.7 Tools and Technologies Used

Technology	Purpose
HTML5	Structure
CSS3	Styling
JavaScript	Filter logic
Bootstrap (optional)	Responsive design
JSON	Data storage
VS Code	Development tool
GitHub	Version control

2.8 Methodology

The development follows the Software Development Life Cycle (SDLC) using an Incremental Model:

1. Requirement Gathering

2. Design and UI Planning

3. Implementation of Filters and Product Grid

4. Testing

5. Deployment on Browser

6. Documentation

2.9 Implementation Details

Products are stored in a JavaScript array (JSON format).

Filters use conditional JavaScript methods like filter().

Event listeners detect user inputs and trigger real-time changes.

DOM manipulation updates product display dynamically.

2.10 Testing

The application was unit tested and manually verified. Test cases included:

Category filter functionality

Combined filters behavior

Search relevance

Invalid input handling

Responsive layout testing

2.11 Limitations

No backend or database for real-time product updates.

No user login or cart functionality.

Limited advanced filters like color, size, or discount.

2.12 Future Enhancements

Future upgrades may include:

Backend integration (Node.js, Firebase, or PHP + MySQL)

Add to Cart & Buy Now features

Pagination

Product reviews and descriptions

Admin panel for product management

2.13 Conclusion

The Product Catalog with Filters project successfully demonstrates a dynamic and interactive web application that allows users to browse and refine products efficiently. The system fulfills the objectives of responsiveness, usability, and real-time filtering. It lays a strong foundation for future development into a fully functional e-commerce platform.

3. Screenshot / API Documentation – Product Catalog with Filters

This section includes a description of the application interface through screenshots (explanation-based since actual images cannot be shown here) and API documentation for future backend integration.

3.1 Screenshot Descriptions

The following are textual descriptions of key screenshots that can be included in your documentation:

Screenshot 1: Home Page – Product Catalog View

The home page displays the complete product catalog.

A navigation bar appears at the top with links such as Home, Products, and Contact.

A search bar is available at the top-right to search products by name.

The main section displays products in a grid layout with each product card showing:

Product Image

Product Name

Price

Rating

Screenshot 2: Filter Panel

Located on the left side of the screen.

Contains filtering options such as:

Category: Electronics, Fashion, Home Appliances, etc.

Price Range: ₹0–₹500, ₹500–₹1000, etc.

Rating: 3★ & above, 4★ & above

Brand: Samsung, Sony, LG, etc.

Includes a “Clear Filters” button to reset selections.

Screenshot 3: Filter Applied Example

Shows the catalog after selecting filters like Category: Electronics + Price: ₹1000–₹2000.

The product grid automatically updates to display only matching products.

Real-time results demonstrate dynamic JavaScript filtering.

Screenshot 4: Search Results

Shows output after searching "Smart Watch" in the search bar.

Only relevant products matching the keyword are displayed.

If no match is found, a message like "No products found" is shown.

Screenshot 5: Responsive View (Mobile Version)

The layout adjusts to mobile screen size.

The filter panel collapses into a dropdown menu.

Product cards appear one per row for better readability.

3.2 API Documentation (Future Integration – Optional)

Currently, the system uses a local product list stored in JavaScript. However, it can easily connect to a backend API to fetch real product data. Below is a sample REST API design for future enhancement.

Base URL

https://api.myproductcatalog.com

Endpoint 1: Get All Products

URL: /products

Method: GET

Description: Fetch all products from the server.

Sample Response:

```
[
 {
  "id": 1,
  "name": "Wireless Headphones",
  "category": "Electronics",
  "price": 1499,
  "rating": 4.3,
  "brand": "Sony",
  "image": "headphone.jpg"
 }
]
```

Endpoint 2: Filter Products

URL: /products/filter

Method: POST

Description: Filter products based on user-selected criteria.

Request Body:

```
{
  "category": "Electronics",
  "minPrice": 500,
  "maxPrice": 2000,
  "rating": 4
}
```

Response: Returns filtered products list.

Endpoint 3: Search Product by Name

URL: /products/search?q=keyword

Method: GET

Description: Search products using a keyword.

Example: /products/search?q=watch

Endpoint 4: Get Product by ID

URL: /products/{id}

Method: GET

Description: Retrieve detailed information about a specific product.

3.3 API Integration Flow (Optional Explanation)

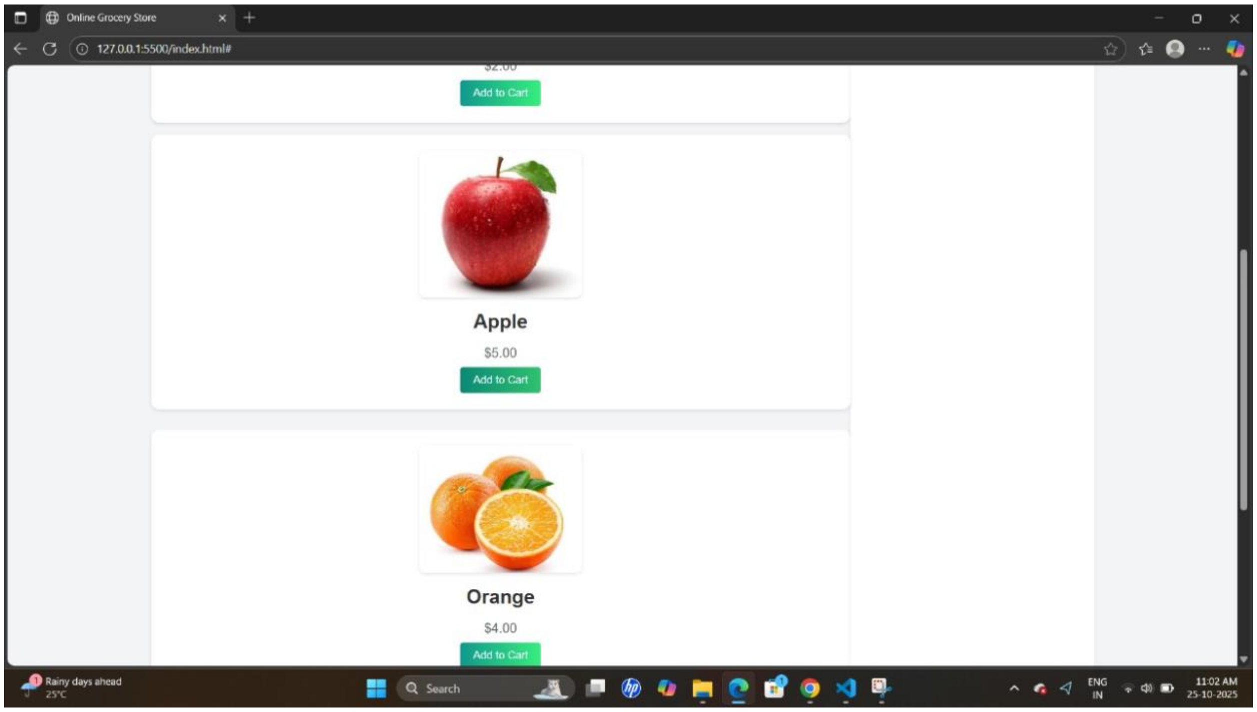
Frontend sends an HTTP request to API

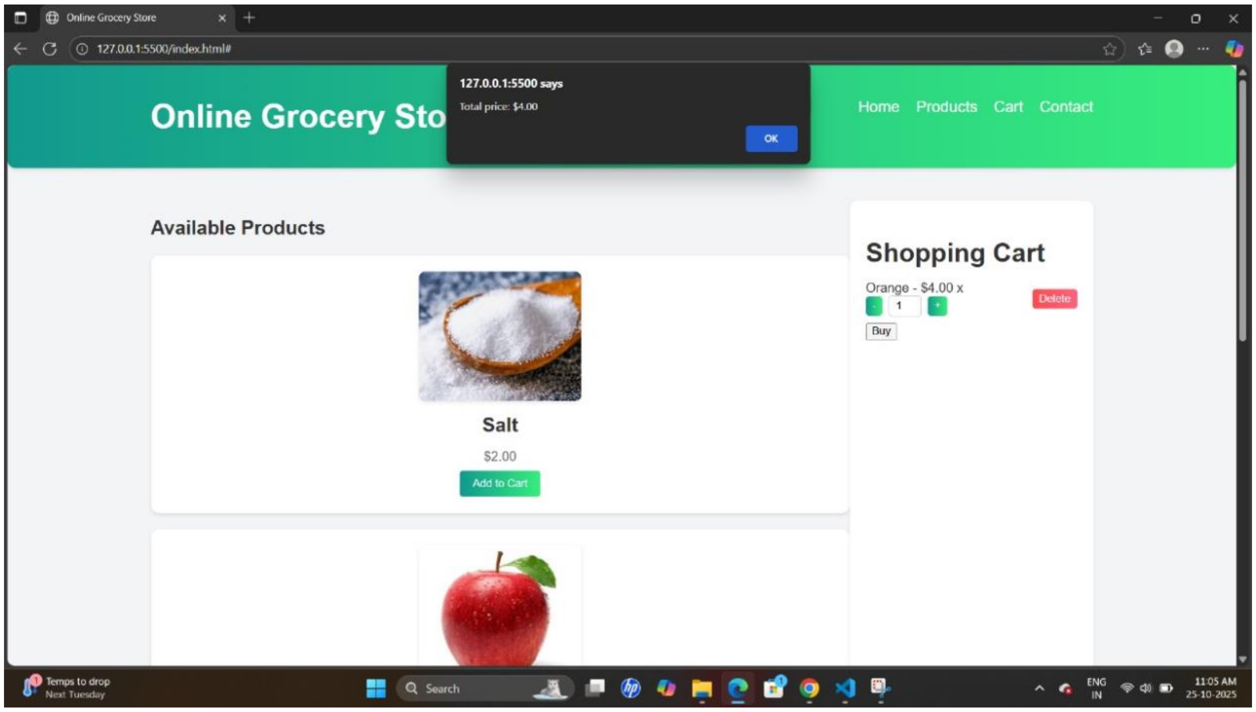
Backend fetches and filters data from the database

Response sent back in JSON

UI updates dynamically without reloading

Screen shot:

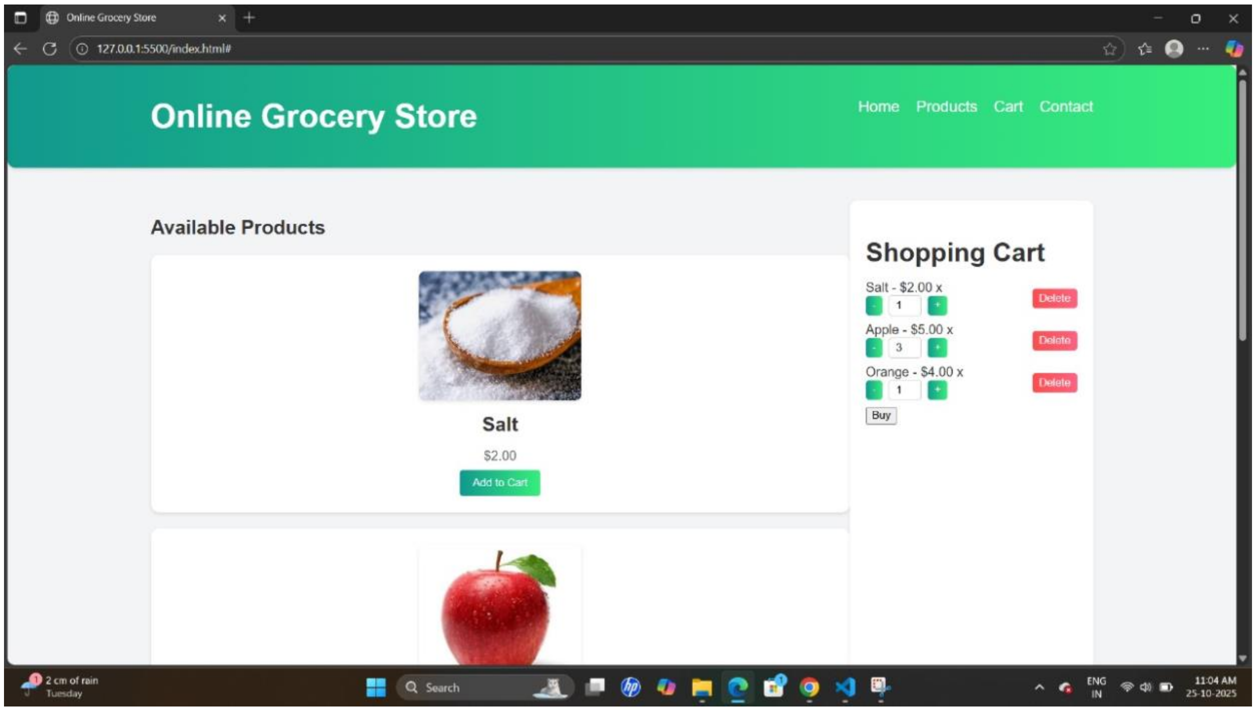




127.0.0.1:5500 says

Total price: $4.00

Orange - $4.00 x 1



Give a content about products catalogue with filters for above contentBelow is the next section of your documentation for Product Catalog with Filters:

4. Challenges and Solutions

During the development of the Product Catalog with Filters, several challenges were encountered related to functionality, user interface, performance, and scalability. This section explains each major challenge faced during the project along with the practical solutions implemented.

Challenge 1: Real-Time Dynamic Filtering

Problem:

Initially, filtering products based on multiple conditions (category + price + rating) caused page reloads and delays.

Solution:

Implemented JavaScript's filter() method and DOM manipulation to apply filters in real-time without reloading the page. Event listeners were used to track filter changes and update results instantly.

Challenge 2: Managing Multiple Filter Combinations

Problem:

Applying more than one filter at a time caused incorrect results or empty output because filters were working independently instead of together.

Solution:

A combined filter logic was applied in JavaScript:

```
filteredProducts = products
  .filter(item => matchCategory(item))
```

```
 .filter(item => matchPrice(item))
 .filter(item => matchRating(item));
```

This ensured that all active filters worked together correctly.

Challenge 3: Responsive Layout Design

Problem:

The filter panel and product grid were not responsive on smaller screens, causing layout issues on mobile devices.

Solution:

Used CSS Flexbox/Grid and media queries to create a responsive layout. The filter panel was converted into a collapsible menu for mobile devices, improving usability.

Challenge 4: Efficient Search Functionality

Problem:

The search bar originally performed case-sensitive search and did not handle partial text searches.

Solution:

A case-insensitive search logic was implemented using:

```
product.name.toLowerCase().includes(searchInput.toLowerCase());
```

This improved usability and accuracy of search results.

Challenge 5: Handling No Filter Results

Problem:

When no products matched the selected filter criteria, the page displayed a blank product area, confusing users.

Solution:

Added a user-friendly message:

```
<p>No products found. Please adjust your filters.</p>
```

Challenge 6: Code Reusability and Maintainability

Problem:

Filter functions became lengthy and repetitive, making the code harder to maintain.

Solution:

Refactored code using modular JavaScript functions, separating filter logic, product rendering, and event handling. This improved code readability and reusability.

Challenge 7: Data Consistency Without a Backend

Problem:

Without a backend, maintaining a consistent product structure was challenging.

Solution:

Created a structured product JSON dataset and validated fields like price, category, and rating. This kept the data uniform and ready for future backend integration.

Challenge 8: Performance Optimization

Problem:

With many products, filtering became slow as DOM operations increased.

Solution:

Reduced frequent DOM refreshes by updating content in batches.

Used innerHTML updates efficiently.

Limited re-rendering to only affected products.

Challenge 9: User Experience

Problem:

Users could not easily reset filters and return to the full product list.

Solution:

Added a “Clear Filters” button to reset all selected filters instantly.

5. GitHub README Setup Guide – Product Catalog with Filters

This section explains how to create a professional GitHub README file for the Product Catalog with Filters project. The README serves as documentation for other developers or reviewers and helps them understand the purpose, setup, and usage of the project.

✅ Project Title

Product Catalog with Filters

✅ Project Description

A responsive web application that displays a product catalog with filtering options like category, price, rating, and brand. Built using HTML, CSS, and JavaScript, this project demonstrates dynamic filtering and search functionality similar to modern e-commerce websites.

✅ Features

- Product listing with grid layout
- Filter by category, price, and rating
- Real-time search bar
- Responsive mobile-friendly design
- Dynamic product rendering
- Clear Filter functionality

✅ Folder Structure

```
/product-catalog/
│
```

```
 ├── index.html
 ├── style.css
 ├── script.js
 ├── products.json (optional)
└── assets/
    └── images/
```

✅ Installation & Setup Guide

Follow these steps to run the project locally:

Prerequisites

A browser (Chrome/Firefox recommended)

VS Code or any code editor

Live Server extension (optional)

Steps to Run

1. Clone this repository:

```
git clone https://github.com/yourusername/product-catalog.git
```

2. Navigate to the project folder:

```
cd product-catalog
```

3. Open the project in VS Code:

```
code .
```

4. Open index.html using Live Server

   OR simply double-click index.html to run in browser.

✅ Usage Instructions

1. Open the home page.
2. Browse through the product grid.
3. Apply filters on the left panel.
4. Use the search bar to find specific products.
5. Click "Clear Filters" to reset filters.

✅ Tech Stack

Technology	Purpose
HTML	Page structure

CSS	Styling and layout
JavaScript	Filter logic
JSON	Product dataset
GitHub	Code hosting

✅ Screenshots (Optional)

Add screenshots to improve clarity:

✅ Contributing

Contributions are welcome!

Fork the repository, make changes, and submit a pull request.

✅ License

This project is licensed under the MIT License.

✅ GitHub Deployment (Optional)

To host using GitHub Pages:

1. Commit your project to GitHub.

2. Go to Settings → Pages.

3. Select main branch → Save.

4. Access your live project at:

https://yourusername.github.io/product-catalog/

✅ Example README Header

# Product Catalog with Filters

A simple product filtering web app made using HTML, CSS, and JavaScript.

GitHub link: https://github.com/Nanthitha752/product-catalogue.git

Nectity repository:https://app.netlify.com/teams/nanthitha752/projects