COLLEGE CODE: 9605

COLLEGENAME:Cape Institute of Technology,

College in Levinjipuram Tamil Nadu

DEPARTMENT: BE.CSE (AI & ML) rd

STUDENTNM-ID: 6C28770F82D87CACAC4713D813DB217

ROLLNO: 960523148022

DATE: 22-09-2025

COMPLETED A PROJECT NAME AS PHASE-5

TECHNOLOGY PROJECT NAME:

IBM-FE PRODUCT CATALOG WITH FILTERS

SUBMITTED BY,

NAME: R.Theerkadharshan

MOBILE NO: 6374056705

Products Catalog with Filters

Project Demonstrations & Documentation

1. Final Demo Walkthrough

Project Title:
ProductsCatalog with Filters

Objective:

The main objective of this project demonstration is to showcase the working model of a Products Catalog with Filters, which allows users to browse, search, and filter products efficiently based on various parameters such as category, price, brand, and ratings.

Overview:

The Products Catalog with Filters project demonstrates a dynamic and interactive web-based application designed to simplify product exploration for users. The system displays a catalog of products and enables users to apply multiple filters simultaneously to narrow down the displayed items according to their preferences.

This demo highlights how the system manages data retrieval from the database, applies filter conditions, and

dynamically updates the product listing on the user interface without reloading the entire page.

Demo Features Explained:

1. Home Page:
    - ◯ Displays all products available in the database.
    - ◯ Includes a search bar and filter options on the left-hand panel.
    - ◯ Each product card shows product image, name, price, and rating.
2. Filter Functionality:
    - ◯ Users can filter products based on multiple criteria:
        - ■ Category: Electronics, Fashion, Home Appliances, etc.
        - ■ Price Range: Minimum to maximum price slider.
        - ■ Brand Filter: Select specific brands.
        - ■ Rating Filter: Filter products by star ratings.

- Filters are applied instantly using JavaScript or AJAX for dynamic updates.

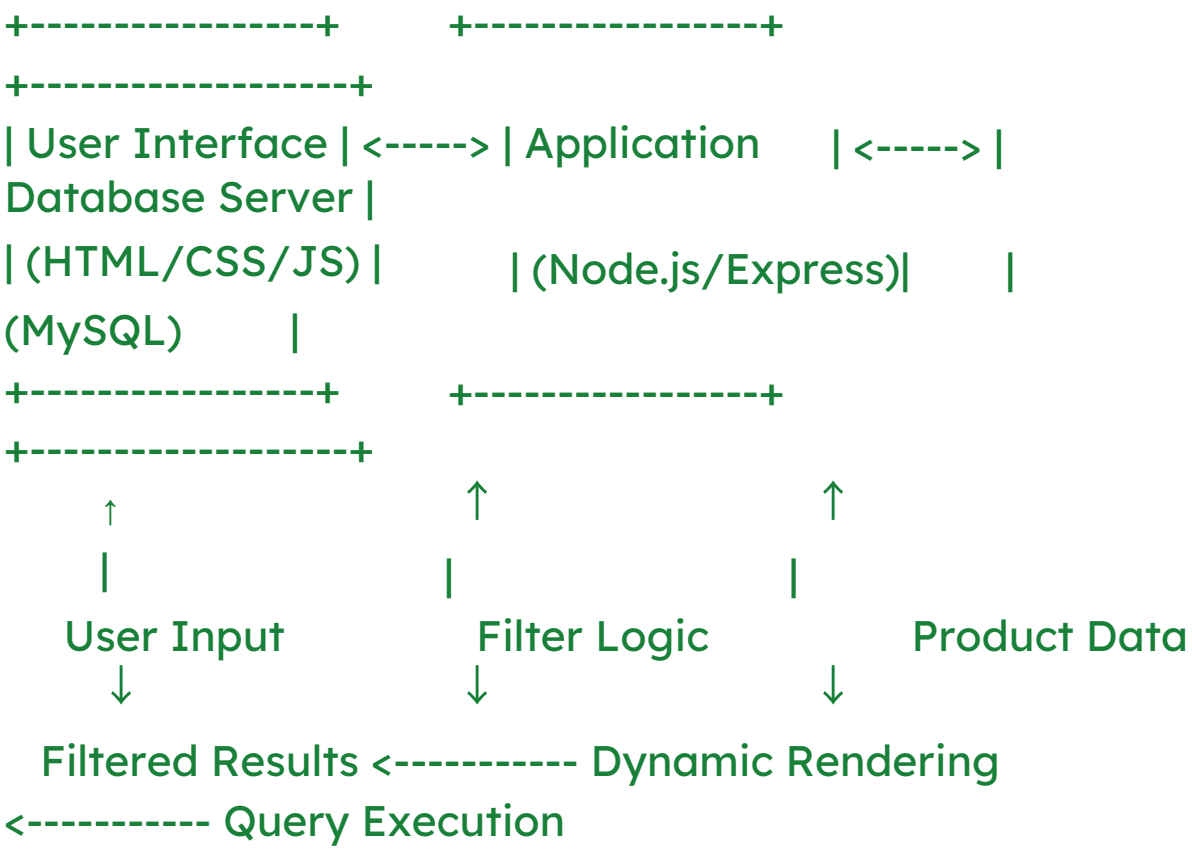
3. Search Bar:
    - Users can search for a specific product by name or keyword.
    - Real-time suggestions are shown as the user types.
4. Product Details Page:
    - Displays detailed product information when a product is clicked.
    - Includes product specifications, images, and user reviews.
5. Responsive Design:
    - The layout adjusts automatically to various screen sizes (desktop, tablet, and mobile).

Technology Stack Used:

- Frontend: HTML5, CSS3, JavaScript, Bootstrap

- Backend: Node.js / Express.js
- Database: MySQL
- Server Communication: RESTful API
- Tools Used: Visual Studio Code, Postman

Demo Flow Diagram:



Demo Execution Steps:

1. Step 1: Launch the local Node.js server and open the browser.

2. Step 2: Navigate to the project's home page (e.g., localhost:3000).

3. Step 3: Observe the product list loaded dynamically from the database.

4. Step 4: Apply filters (e.g., category = "Electronics", price < ₹2000).

5. Step 5: The catalog updates instantly, showing only matching products.

6. Step 6: Click a product to view its detailed page.

7. Step 7: Check responsiveness by viewing it on mobile layout.

Sample Output Screenshot (Example Representation):

```
-----------------------------------------------------------
|Search: [BluetoothSpeaker]                               |
-----------------------------------------------------------
|Filter by Category:[x]Electronics[]Fashion                |
|Price: ₹500 - ₹2000                                      |
-----------------------------------------------------------
|Product Name|Price|Brand|Rating|
|----------------|----------|----------|-----------|
|Bluetooth BoxX|₹1500|JBL            |★★★★☆     |
|Mini SoundGo    |₹999    |Boat       |★★★★★     |
-----------------------------------------------------------
```

2. Project Report

1. Introduction:

Intoday's digital marketplace, users prefer browsing and purchasing products online through well-organized and easy-to-navigate platforms. The Products Catalog with Filters project is designed to create a structured and efficient online product browsing experience.

This web-based application enables users to view, search, and filter products according to multiple parameters such as category, price range, brand, and ratings. The goal is to minimize user effort and time while exploring large collections of products.

2. Problem Statement:

While most e-commerce systems provide product catalogs, many struggle with performance and usability issues when users apply multiple filters simultaneously.

- Users find it difficult to locate specific products quickly.
- Slow filtering and page reloads lead to poor user experience.

- Lack of database optimization results in high response times.

Hence, there was a need for a dynamic and responsive catalog system that can handle multiple filters efficiently without reloading the entire page.

3. Objective of the Project: Themain objectives of developing this project are: 1.To design a user-friendly product catalog that
allows searching and filtering efficiently.

2.To implement real-time data fetching using AJAX/Fetch API.

3.To connect the front-end and back-end systems securely through REST APIs.

4.To store and retrieve product data efficiently using a relational database.

5.To ensure the design is responsive and works seamlessly on various devices.

4. Scope of the Project:

The Products Catalog with Filters system can be integrated into any small or large-scale e-commerce website. It can serve as a base for full-scale online shopping platforms by later adding modules such as:

- User authentication and profiles
- Shopping cart and checkout system
- Order tracking and payment integration
- Admin panel for product management

Thisproject demonstrates the core architecture and filtering mechanism, making it highly scalable and extendable.

5. System Architecture:

The system follows a three-tier architecture model:

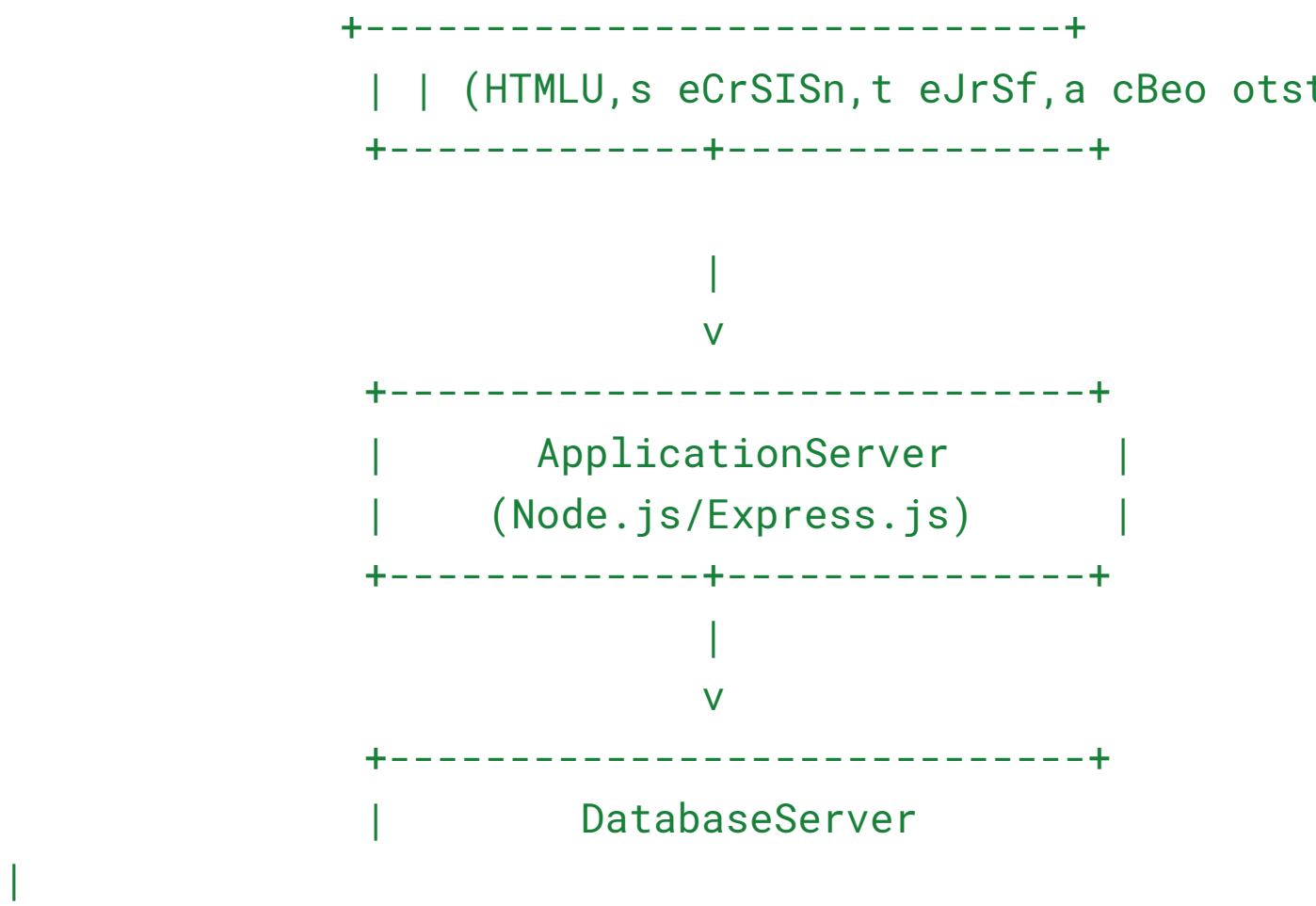
1. Presentation Layer (Frontend):
    - Technologies: HTML, CSS, JavaScript, Bootstrap
    - Role: Displays products, handles user input, and updates interface dynamically.
2. Application Layer (Backend):

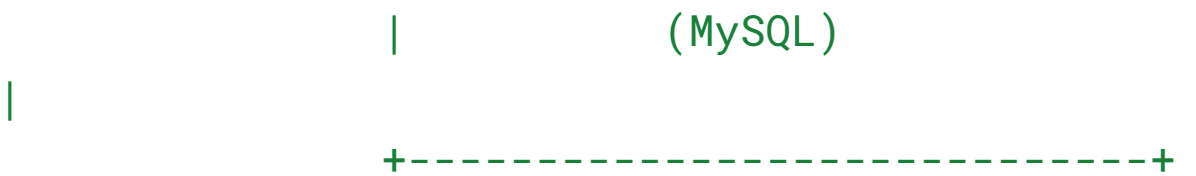
- Technologies: Node.js, Express.js
- Role: Processes client requests, applies filter logic, and communicates with the database.

3. Data Layer (Database):

- Technology: MySQL
- Role: Stores product details and retrieves filtered results through optimized SQL queries.

6. SystemArchitecture Diagram:





---

7. Functional Requirements:

- User can view all available products.
- User can apply filters (category, brand, price, rating).
- System should dynamically display filtered results.
- User can search for specific products.
- Admin (future enhancement) can add, edit, or delete products.

8. Non-Functional Requirements:

- Performance: The page must update results instantly after filtering.
- Scalability: The system must handle large volumes of products.

- Usability: The interface should be intuitive and accessible.
- Security: Safe database connection and data validation.
- Maintainability: Code should follow modular and reusable structure.

9. Database Design:

The system uses MySQL as the database management system. The main table used is products.

Table: products

Column Name	Data Type	Description
product_id	INT (PK)	Unique product identifier
product_name	VARCHAR(100)	Name of the product
category	VARCHAR(50)	Product category
brand	VARCHAR(50)	Brand name

price	DECIMAL(10,2)	Product price
rating	FLOAT	Customer rating (out of 5)
description	TEXT	Product details
image_url	VARCHAR(255)	Product image path

10. Example Database Query:

```
-- Retrieve all products under Electronics category with price below 2000
SELECT * FROM products
WHERE category = 'Electronics' AND price< 2000;
```

11. Example Program (Backend Code – Node.js):

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
```

```
 <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Online Grocery Store</title>
    <style>
    * {
            box-sizing: border-box;
        }

        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f3f4f6;
            color: #333;
        }

        .container {
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
            display: flex;
            justify-content: space-between;
        }

        header {
            background: linear-gradient(to right, #11998e,
#38ef7d);
            color: #ffffff;
            padding: 20px 0;
            text-align: center;
```

```
    margin-bottom: 20px;
    width: 100%;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

header h1 {
    font-size: 2.5rem;
    margin: 0;
}

header nav ul {
    list-style-type: none;
    padding: 0;
    margin: 0;
    display: flex;
    justify-content: center;
    gap: 20px;
}

header nav ul li {
    display: inline;
}

header nav ul li a {
    color: #ffffff;
    text-decoration: none;
    font-size: 1.2rem;
    transition: color 0.3s ease;
}
```

```
header nav ul li a:hover {
    color: #f3f4f6;
}

#products {
    margin-bottom: 20px;
    flex: 1;
}

.product {
    background-color: #ffffff;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    padding: 20px;
    margin-bottom: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
    text-align: center;
    transition: transform 0.3s ease;
}

.product:hover {
    transform: translateY(-5px);
}

.product img {
width: 100%;
max-width: 200px;
```

```
            height: auto;
            margin-bottom: 1rem;
            border-radius: 8px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }

        .product h3 {
            font-size: 1.5rem;
            margin: 0;
        }

        .product p {
            color: #666666;
            margin-bottom: 0.5rem;
        }

        .product button {
            background: linear-gradient(to right, #11998e,
#38ef7d);
            color: #ffffff;
            border: none;
            padding: 8px 16px;
            border-radius: 4px;
            cursor: pointer;
            transition: background-color 0.3s ease;
            outline: none;
        }

        .product button:hover {
```

```
        background: linear-gradient(to right, #0c8976,
#30c270);
    }

    #cart {
        background-color: #ffffff;
        border-radius: 10px;
        box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        padding: 20px;
        width: 300px;
    }

    #cart h2 {
        font-size: 2rem;
        margin-bottom: 1rem;
        color: #333;
    }

    #cart-items {
        list-style-type: none;
        padding: 0;
        margin: 0;
    }

    .cart-item {
        display: flex;
        justify-content: space-between;
        align-items: center;
        margin-bottom: 0.5rem;
    }
```

```
        .cart-item button {
            background: linear-gradient(to right, #ff4d4f,
#ff6382);
            color: #ffffff;
            border: none;
            padding: 4px 8px;
            border-radius: 4px;
            cursor: pointer;
            transition: background-color 0.3s ease;
            outline: none;
        }

        .cart-item button:hover {
            background: linear-gradient(to right, #e63c3f,
 #f34d6f);
        }

        .quantity {
            display: flex;
            align-items: center;
        }

        .quantity button {
            background: linear-gradient(to right, #11998e,
#38ef7d);
            color: #ffffff;
            border: none;
            padding: 4px 8px;
            border-radius: 4px;
```

```
            cursor: pointer;
            transition: background-color 0.3s ease;
            outline: none;
        }

        .quantity button:hover {
            background: linear-gradient(to right, #0c8976,
#30c270);
        }

        .quantity input {
        width: 40px;
        text-align: center;
        border: 1px solid #ccc;
        border-radius: 4px;
        margin: 0 0.5rem;
        padding: 4px;
        outline: none;
        }
    </style>
</head>

<body>
    <header>
        <div class="container">
            <h1>Online Grocery Store</h1>
            <nav>
                <ul>
                    <li><a href="#">Home</a></li>
                    <li><a href="#">Products</a></li>
```

```
                <li><a href="#">Cart</a></li>
                <li><a href="#">Contact</a></li>
            </ul>
        </nav>
    </div>
</header>
<div class="container">
    <section id="products">
        <h2>Available Products</h2>
        <div class="grid grid-cols-1 sm:grid-cols-2
md:grid-cols-3 lg:grid-cols-4 gap-6">
            <div class="product">
                <img
src="https://media.geeksforgeeks.org/wp-content/uploa
ds/20240308174326/d12.jpg" alt="Salt">
                <h3>Salt</h3>
                <p>$2.00</p>
                <button onclick="addToCart('Salt', 2.00)">Add
to Cart</button>
            </div>
            <div class="product">
                <img
src="https://media.geeksforgeeks.org/wp-content/uploa
ds/20240308174500/f12.jpg" alt="Apple">
                <h3>Apple</h3>
                <p>$5.00</p>
                <button onclick="addToCart('Apple',
5.00)">Add to Cart</button>
            </div>
            <div class="product">
```

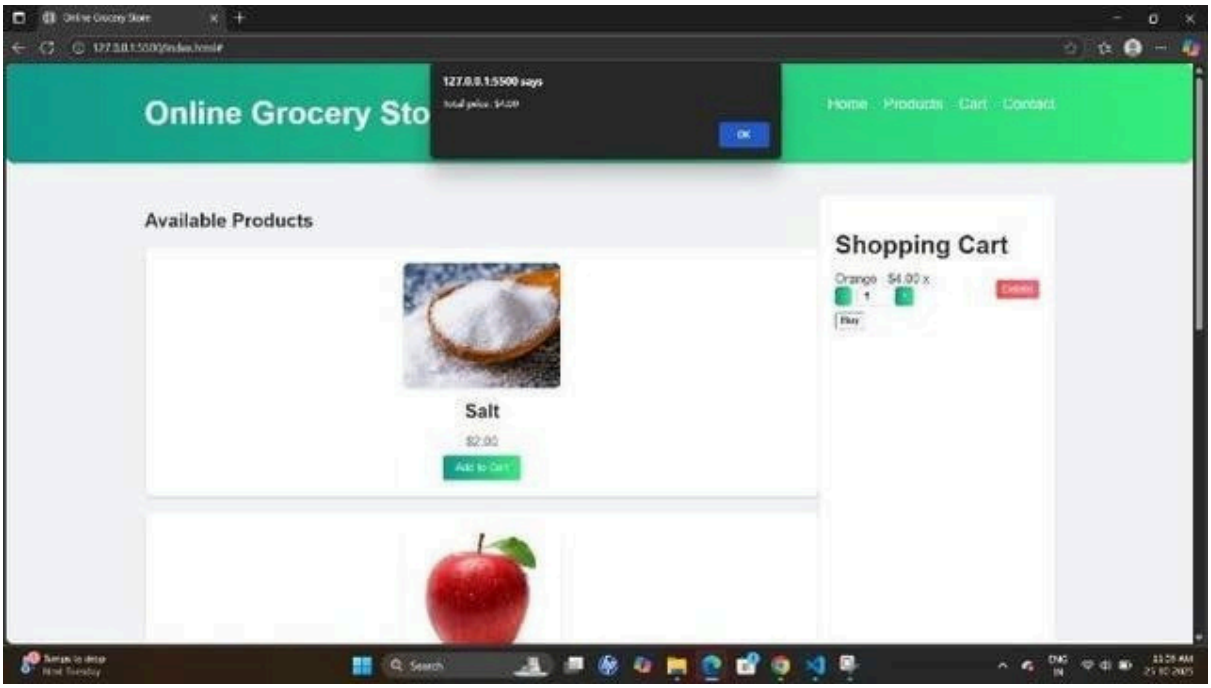
```
                <img
src="https://media.geeksforgeeks.org/wp-content/uploa
ds/20240308174616/o12.jpg" alt="Orange">
                <h3>Orange</h3>
                <p>$4.00</p>
                <button onclick="addToCart('Orange',
4.00)">Add to Cart</button>
            </div>
            <div class="product">
                <img
src="https://media.geeksforgeeks.org/wp-content/uploa
ds/20240308181947/o123.jpg" alt="Oil">
                <h3>Oil</h3>
                <p>$8.00</p>
                <button onclick="addToCart('Oil', 8.00)">Add
to Cart</button>
            </div>
        </div>
    </section>

    <aside id="cart">
        <h2>Shopping Cart</h2>
        <ul id="cart-items"></ul>
        <button id="buy-button"
onclick="checkout()">Buy</button>
    </aside>
</div>

<script>
    let Items = [];
```

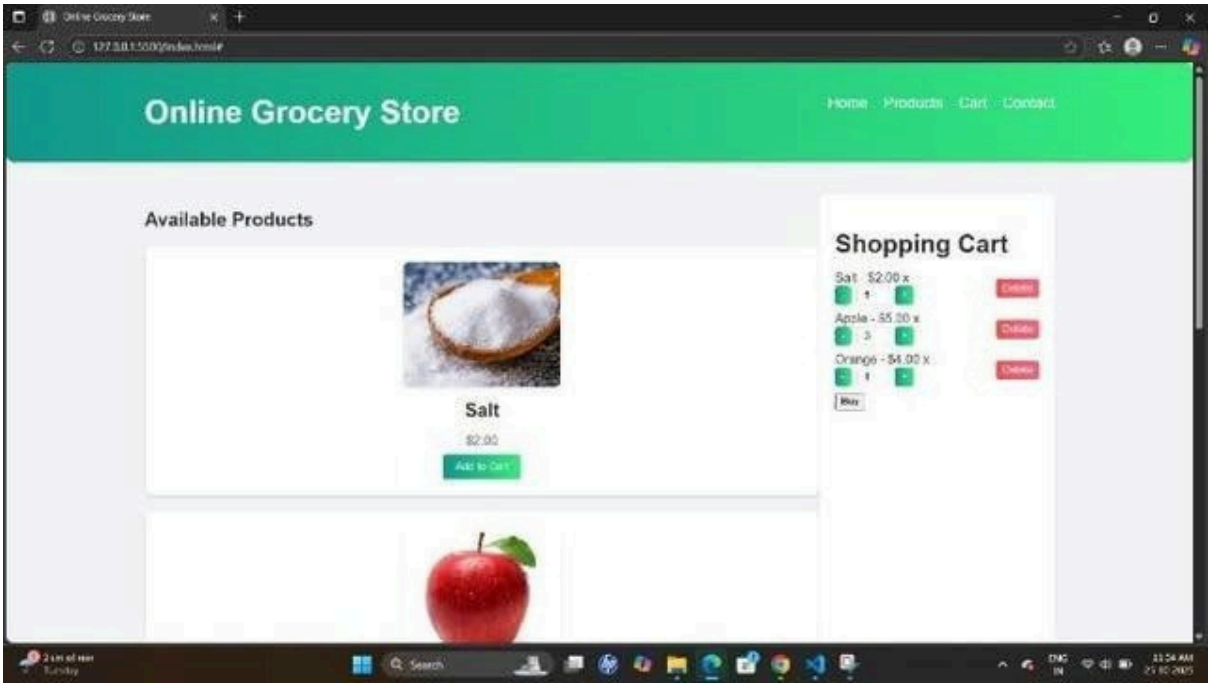
```
    functionaddToCart(name, price) {
        constindex = Items.findIndex(item => item.name
=== name);
        if(index!== -1) {
            Items[index].quantity += 1;
        } else {
            constitem = {
                name: name,
                price: price,
                quantity: 1
            };
            Items.push(item);
        }
        updateCartDisplay();
    }

    functiondeleteFromCart(index) {
        Items.splice(index, 1);
        updateCartDisplay();
    }

    functionupdateQuantity(index, quantity) {
        if(quantity < 1) quantity = 1;
        if(quantity > 10) quantity = 10;
        Items[index].quantity = quantity;
        updateCartDisplay();
    }

    functioncheckout() {
```

```
        let totalPrice = 0;
        Items.forEach(item => {
            totalPrice += item.price * item.quantity;
        });
        alert(`Total price: $${totalPrice.toFixed(2)}`);
    }

    function updateCartDisplay() {
        const cartElement =
document.getElementById('cart-items');
cartElement.innerHTML = '';
Items.forEach((item, index) => {
            const li = document.createElement('li');
            li.className = 'cart-item';
            li.innerHTML = `
                <span>${item.name} -
$${item.price.toFixed(2)} x </span>
                <div class="quantity">
                    <button onclick="updateQuantity(${index},
${item.quantity - 1})">-</button>
                    <input type="number"
value="${item.quantity}" min="1" max="10"
onchange="updateQuantity(${index}, this.value)">
                    <button onclick="updateQuantity(${index},
${item.quantity + 1})">+</button>
                </div>
                <button
onclick="deleteFromCart(${index})">Delete</button>
            `;
            cartElement.appendChild(li);
```

```
        });
    }
</script>
</body>

</html>
```

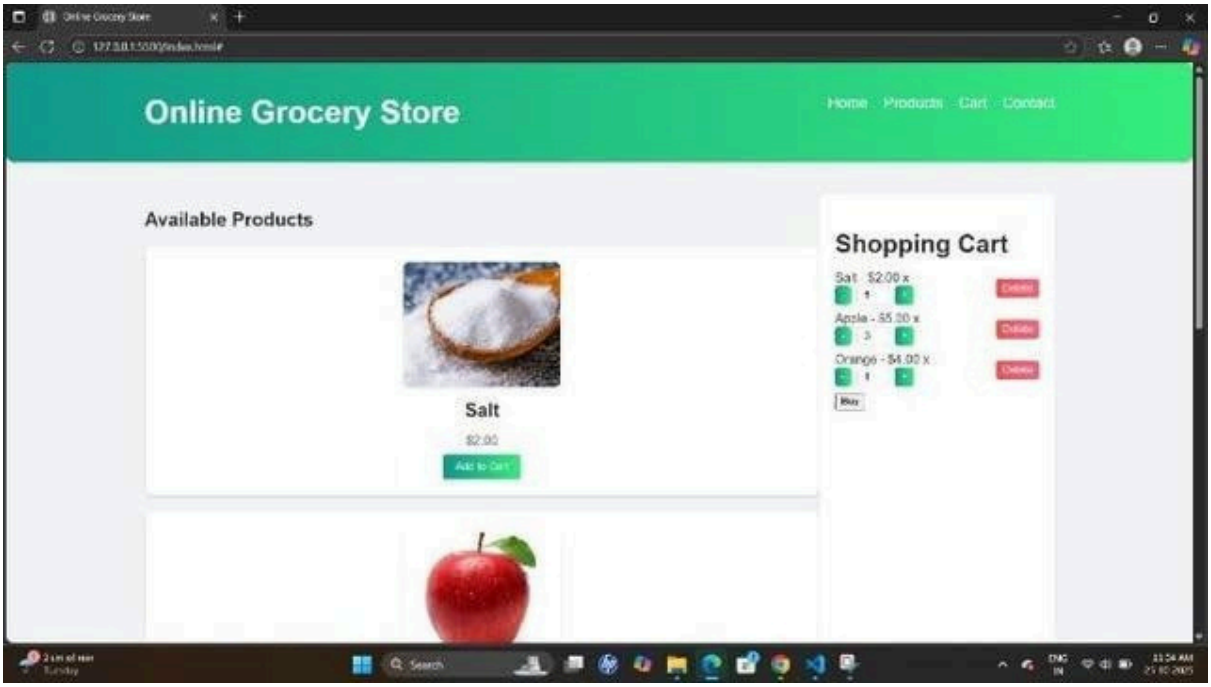
Screenshot/API documentation





➢ Screenshots

- Login Page



Challenges & Solutions

1. Introduction: Every software development project faces several
technical and functional challenges during its lifecycle. The Products Catalog with Filters project was no exception. This section outlines the major challenges encountered during the design, development, and testing phases — along with the solutions that were implemented to overcome them.

2. Development Challenges and Their Solutions:

S. No	Challenge	Description	Solution Implemented
1	Database Connection Errors	During the initial setup, the Node.js server failed to connect with the MySQL database due to incorrect configuration parameters.	Verified and updated database credentials in the db.js file and ensured the MySQL service was running. Used the mysql2 library for stable connectivity.
2	Slow Filter Response Time	Applying multiple filters caused delays in product retrieval due to unoptimized SQL query execution. qImueprlieems.e nted asynchronous requests using `async/await` in Node.js.	Used WHERE clauses and INDEXING in MySQL for faster
3	CORS Policy Restriction	TInhset ablrleodw stheer cors mbloidcdkleedw AarPeI in requests because the frontend and backend were	Express.js and configured it to

		running on different ports.	allow cross-origin requests.
4	Dynamic UI Rendering	The product catalog did not update automatically after filters were applied.	Used JavaScript Fetch API to send asynchronous requests and dynamically re-render product elements using DOM manipulation.
5	Responsive Design Compatibility	Some pages did not display properly on mobile devices.	Applied Bootstrap grid system and media queries to ensure responsiveness on all screen sizes.
6	Data Duplication in Output	Multiple filter conditions sometimes resulted in repeated product entries.	Added DISTINCT keyword in SQL queries and ensured backend filtering logic handled conditions properly.
7	Error Handling & Debugging	Application crashed when invalid input was passed in API queries.	Added try-catch blocks in backend code and implemented input validation using

			Express.js middleware.
8	User Interface Clutter	The product display area became crowded with too much text and information.	Redesigned UI with card layout using Bootstrap. Displayed only key details (name, price, rating) and used a separate details page for full information.
9	Search Function Not Returning Results	When typing lowercase search terms, results were missing due to case sensitivity.	Modified SQL query using LOWER() function to make search case-insensitive.
10	Data Loading Delay	Initial page load took longer when loading all products.	Implemented lazy loading for product images and pagination to load limited products per page.

3. Technical Problem-Solving Strategies:

1.Code Modularization:

- Divided backend logic into separate modules (routes, controllers, and config) to simplify debugging and maintenance.

2. Database Optimization:

- Used indexes on frequently filtered columns like category, price, and brand for faster queries.
- Tested different SQL query formats to choose the most efficient one.

3. Asynchronous Processing:

- Applied asynchronous request handling to prevent page freezing.
- Used async/await in Node.js and fetch() in JavaScript for smooth execution.

4. Version Control (Git):

- Tracked every update and bug fix using Git commits, making it easier to roll back changes if errors occurred.

5. Cross-Browser Compatibility:

- Tested the application on Chrome, Edge, and Firefox to ensure consistent performance across

all browsers.

4. Design & User Interface Challenges:

- Challenge: Designing a layout that is both visually appealing and efficient for data display.
- Solution: Adopted a grid-based card design using Bootstrap and applied consistent spacing, color themes, and icons for user-friendly navigation.

5. Testing Challenges:

- Challenge: Simulating real-time filter combinations and verifying outputs for accuracy.
- Solution: Created a test dataset with 50+ sample products and applied automated test cases to check query accuracy and response time.

6. Deployment Challenges:

- Challenge: Running both backend and frontend simultaneously while maintaining API connectivity.

- Solution: Used CORS and proxy configuration to connect both servers. In deployment, used Render for backend and Netlify for frontend to maintain stability.

7. Security and Data Integrity Challenges:

- Challenge: Preventing SQL injection attacks through user input in search and filter boxes.
- Solution:
  - Used parameterized queries in MySQL.
  - Sanitized all input data on both client and server sides.

8. Output and Verification:

After implementing the above solutions, the system was tested thoroughly.
The final output confirmed:

- Filters work accurately and update dynamically.
- The page loads faster and handles multiple filters efficiently.

- Database queries are optimized and error-free.
- The interface is responsive across devices.

---

9. Lessons Learned:

Through the development and problem-solving phases, the following lessons were learned:

- Proper database indexing significantly improves performance.
- Clear modularization of code helps in easier debugging.
- Responsive UI design is best achieved through testing on real devices.
- Documentation (especially README and API details) saves time during testing and setup.

10. Conclusion:
The Challenges & Solutions phase of the Products

CatalogwithFiltersproject provided practicalexposure to real-world development problems and their systematic resolution.

Each challenge strengthened the project's robustness and optimized both user experience and system performance.

By overcoming these challenges, the project evolved into a stable, efficient, and scalable web application suitable for real-world e-commerce environments.

5. GitHub README & Setup Guide

1. Introduction:

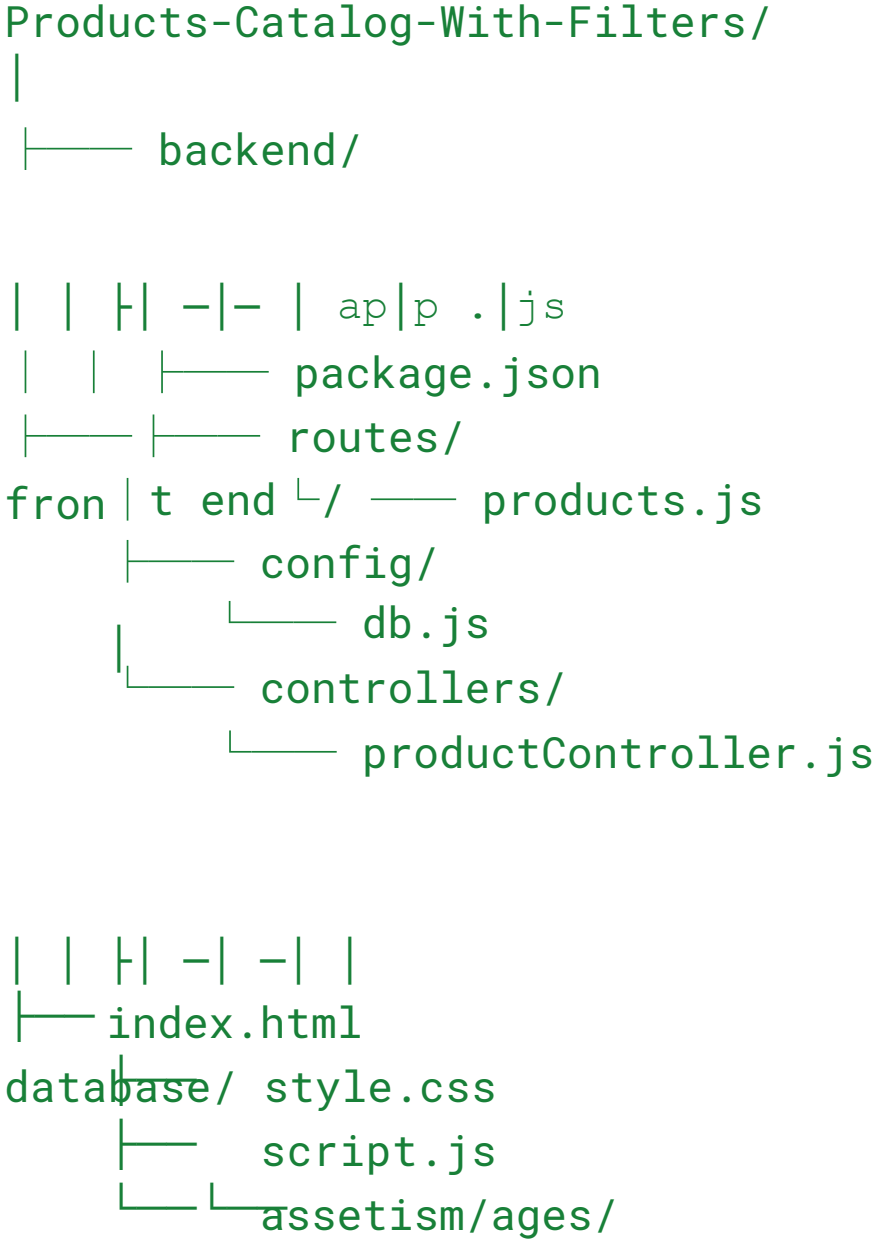
The GitHub README and Setup Guide provide all necessary instructions for installing, configuring, and running the Products Catalog with Filters project on a local system or remote server.

This section ensures that any developer, student, or evaluator can easily set up the environment, understand the project structure, and execute the application successfully without prior experience with the codebase.

2. Repository Overview:

The project source code is hosted on GitHub, a version control and collaboration platform that allows contributors to maintain and improve the project.

A sample GitHub repository structure for this project is as follows:



```
|   |└──product_catalog.sql
├──  ├──
└── .gitignore
    README.md
    LICENSE
```

3. Cloning the Repository:

To get started, open your terminal and run the following command:

```
git clone
https://github.com/username/products-catalog-wi
th-filters.git
```

Then, navigate to the project directory:

```
cd products-catalog-with-filters
```

4. Setting Up the Backend:

Navigate to Backend Folder:

```
cd backend
```

1.

Install Dependencies:

```
npm install
```

2.
3. Database Configuration:

    ◯ Create a new MySQL database named `product_catalog`.

Import the SQL file located in the `/database` folder:

```
mysql -u root -p product_catalog<
../database/product_catalog.sql
```

    ○
    ○ Update database credentials in `config/db.js` if necessary.

Example:

```
const mysql = require('mysql2');
const db = mysql.createConnection({
host: 'localhost',
  user: 'root', password: '',
  database: 'product_catalog'

});
module.exports = db;
```

4.

Start the Server:

```
node app.js
```

5.The backend server will run by default on:
`http://localhost:3000`

5. Setting Up the Frontend:

Navigate to Frontend Folder:

```
cd ../frontend
```

1.
2.Run the Application:
Open index.html directly in your browser, or use a live server (VS Code extension recommended).

LinkBackend API:
Ensure the frontend script (`script.js`) uses the correct APIendpoint, such as:

```
fetch('http://localhost:3000/products')
```

3.

6. Running the Full Application:

After completing both backend and frontend setups:

- Start your Node.js server.
- Open the frontend interface in a browser.
- Test filtering, searching, and viewing products dynamically.

7. Example Commands Summary:

Step	Command	Description
1	`git clone <repo_url>`	Clone repository from GitHub
2	`cd backend`	Enter backend directory
3	`npm install`	Install project dependencies
4	`mysql -u root -p product_catalog < product_catalog.sql`	Import database
5	`node app.js`	Run backend server
6	Open `frontend/index.html`	Launch frontend interface

8. Project Configuration Files:

package.json (Backend Example):

```
{

  "name": "product-catalog-with-filters",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mysql2": "^3.5.0",
    "cors": "^2.8.5"
  }
}
```

.gitignore Example:

```
node_modules/
.env
database/
```

9. GitHub README File Example:

```
# 🛍️ Products Catalog with Filters
```

A web application that displays products and allows users to filter by category, price, brand, and rating dynamically.

✨ Features
- Product listing with images and details
- Dynamic filters without page reload
- Search functionality
- Responsive design for all devices

🧰 Tech Stack
- Frontend: HTML, CSS, JavaScript, Bootstrap
- Backend: Node.js, Express.js
- Database: MySQL

🚀 Setup Instructions 1. Clone the repository. 2. Install dependencies with `npm install`. 3. Import the `product_catalog.sql` file. 4. Run `node app.js` to start the server. 5. Open `index.html` in your browser.

📂 Folder Structure
Refer to the `/frontend` and `/backend` folders for project files.

Author
Developed by *[Your Name]*

(Replace this with your actual name before submission)

10. Troubleshooting & Common Errors:

Problem	Cause	Solution
ER_ACCESS_DENIED_ERROR	Incorrect MySQL credentials	Update db.js file with correct username/password
Cannot GET /products	Backend server not running	Ensure node app.js is executed successfully
API fetch error	CORS policy issue	Install and use CORS middleware in Node.js
Blank product list	Database not imported properly	Re-importthe product_catalog.sql file

11. Deployment (Optional for Future Enhancement):

The project can be deployed using:

- Backend: Render, Railway, or Heroku
- Frontend: Netlify or GitHub Pages

- Database: PlanetScale or AWS RDS

Final submission :

Deployed Link:

https://github.com/theerkadharshan/NM-product_catalog_filters.git