



COLLEGE CODE : 9623

COLLEGE NAME : Amrita College of Engineering And Technology

DEPARTMENT : Computer Science and Engineering

STUDENT NM-ID : 9309BED1229D1E1AEC842B4AE35533F3

ROLL NO : 23CS048

DATE : 11-09-2025

Completed the project named as

Phase 2 Solution Design and Architecture

PROJECT NAME : PRODUCT CATALOG WITH FILTERS

SUBMITTED BY,

NAME : KANNAN K
MOBILE NO : 9994359062

Phase 2 – Solution Design & Architecture

1. Tech Stack Selection

Frontend: React.js (with Tailwind CSS for styling)

Backend: Node.js + Express.js (REST API)

Database: MongoDB (for flexible product catalog storage)

Authentication (if required): JWT-based auth

Hosting/Deployment:

Frontend → Vercel / Netlify

Backend → Render / AWS / Heroku

Database → MongoDB Atlas

2. UI Structure (Frontend)

Pages & Components:

Homepage / Product Catalogue Page:

- Header (Search bar, nav menu)
- Sidebar Filters (Category, Price Range, Brand)
- Product Grid (cards with image, title, price, rating)

Product Details Page:

- Product Image, Title, Price, Description, Rating, Add to

Wishlist/Cart

Admin Dashboard:

- Product Form (Add/Edit Product)
- Product List with edit/delete actions

3. API Schema Design (Node.js REST API)

Products Collection (MongoDB):

```
{ "_id": "ObjectId",  
  
  "name": "iPhone 14",  
  
  "description": "Latest Apple iPhone model",  
  
  "price": 899,  
  
  "category": "Mobile Phones",  
  
  "brand": "Apple",  
  
  "rating": 4.5,  
  
  "imageUrl":  
  
    "https://example.com/iphone.jpg",  
  
  "createdAt": "2025-09-08T00:00:00Z"  
  
}
```

API Endpoints:

- GET /api/products → Fetch all products (with filter/search/sort params)
- GET /api/products/:id → Fetch single product
- POST /api/products → Add product (Admin only)
- PUT /api/products/:id → Update product (Admin only)
- DELETE /api/products/:id → Delete product (Admin only)

4. Data Handling Approach

Filtering & Searching:

Handled via query params (e.g.,

`/api/products?category=mobile&brand=apple&price;[lte]=1000`

)

Sorting:

Handled with query params (sort=price_asc or

sort=rating_desc)

Pagination:

Backend: Skip + Limit in MongoDB

Frontend: Load more / page numbers

Caching (Optional for performance): Redis or frontend caching

with React Query

5. Component / Module Diagram

Frontend Components:

- App
- Header
- FilterSidebar
- ProductGrid
- ProductCard
- ProductDetails
- AdminDashboard (ProductForm, ProductList)

Backend Modules:

- server.js (Express setup)

- /routes/[productRoutes.js](#)
- /controllers/[productController.js](#)
- /models/[productModel.js](#)
- /middleware/auth.js (if authentication added)

6. Basic Flow Diagram

User Flow (Frontend + Backend):

User → UI (React) → REST API (Node.js) → MongoDB

Example:

1. User applies filters on UI → query params generated

2. React sends request to

`/api/products?category=shoes&price:[lte]=500`

3. Node.js (Express) queries MongoDB with filters

4. MongoDB returns filtered product list

5. API sends response → React renders ProductGrid