COLLEGE CODE: 9504

COLLEGE NAME: DR G U POPE COLLEGE OF ENGINEERING

DEPARTMENT: CSE

STUDENT NM-ID : 4122921B2EE9B7EB97042BD86E090B16

ROLL NO: 950423104047

DATE: 29/09/2025

Completed the project named as Phase__ 4

TECHNOLOGY PROJECT NAME : PRODUCT CATALOG WITH FILTERS

SUBMITTED BY,

NAME :THILLAI RAJA M

MOBILE NO :6369355154

Product Catalog with Filters — Enhancements & Deployment

1. Additional Features

- Wishlist / Favorites – Logged-in users can save products. Backend stores in a wishlist collection with userId and productId.
- Sorting Options – Allow sorting by price (asc/desc), ratings, and newest arrivals. Implemented via query params (/products?sort=price_asc).
- Advanced Filters – Filters like price range slider, brand, category, rating, and availability. Example API: /products?brand=Nike&rating;=4&minPrice;=500&maxPrice;=2000.
- Pagination or Infinite Scroll – Use page numbers (/products?page=2&limit;=20) or infinite scroll to load more results.
- Product Comparison – Select 2–3 products and show specifications side by side.
- User Authentication (Optional) – Secure login/signup with JWT to persist user cart and wishlist.

2. UI/UX Improvements

- Responsive Layout – Mobile-first design using CSS Grid or Flexbox. Breakpoints: Mobile (360–768px), Tablet (768–1024px), Desktop (1024px+).
- Search Bar with Suggestions – Autocomplete feature to suggest products as users type.
- Filter Panel – Sidebar filters for desktop, dropdown filters for mobile with Apply/Reset buttons.
- Product Card Design – Larger images, quick view button, add-to-cart shortcut, hover effects.
- Consistent Branding – Use consistent typography, colors, and spacing. Optional dark mode toggle.
- Loading States – Use skeleton loaders or animations to improve UX during data fetch.

3. API Enhancements

- Filtering & Sorting APIs – Use query params for efficient backend filtering and sorting.
- Search Endpoint – Implement full-text search (MongoDB $text, SQL LIKE). For advanced setups, use Elasticsearch or Meilisearch.
- Authentication APIs – Signup (/api/auth/signup), Login (/api/auth/login), JWT verification middleware.
- Pagination APIs – Always return paginated results to optimize load. Example response includes page, limit, totalProducts.
- Caching & CDN – Use Redis for caching frequent queries. Serve static assets/images through a CDN (Cloudflare/AWS CloudFront).

4. Performance & Security Checks

- Frontend Optimization – Lazy loading of images, code splitting, WebP format images, minified JS/CSS.
- Backend Optimization – Database indexing, asynchronous APIs, caching of frequent queries. Security
- Measures – Enforce HTTPS, JWT/OAuth authentication, input validation with Joi/Yup, sanitize inputs against SQL Injection/XSS.
- Testing Tools – Use Lighthouse (performance & accessibility), Postman (API testing), GTmetrix (site speed).

5. Testing of Enhancements

- Unit Testing – Test core functions like filters, sorting, and wishlist handling. Frameworks:
- Jest/Mocha. Integration Testing – Ensure frontend and backend interactions work seamlessly.
- Cross-Browser Testing – Validate compatibility across Chrome, Firefox, Edge, Safari.
- Responsive Testing – Test UI on different devices (mobile, tablet, desktop). User Acceptance
- Testing (UAT) – Collect feedback from sample users to refine before deployment.

6. Deployment (Netlify, Vercel, or Cloud Platform)

- Frontend Deployment – Deploy React/Next.js app to Netlify or Vercel. Configure API base URL
- via environment variables.
- Backend Deployment – Options: Heroku/Render (simple), AWS EC2/Lambda (scalable),
- Google Cloud Run/Firebase (serverless).
- Database Hosting – Cloud databases such as MongoDB Atlas, Firebase Firestore, or Supabase (PostgreSQL).
  CI/CD – Automate deployments with GitHub Actions: run tests, build, then deploy on push.
  Monitoring & Analytics – Use Sentry for error tracking, Google Analytics/Mixpanel for usage, Datadog/New Relic for performance monitoring.