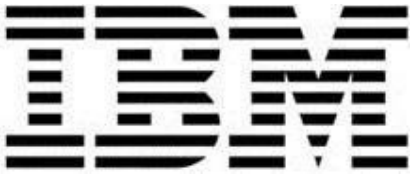

COLLEGE CODE: 9623

COLLEGE NAME: AMRITA COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

STUDENT NM-ID:34F54868237E7EAA94B4A5AC30E4BC09

ROLL NO: 962323104037

DATE: 22-09-2025

COMPLETED THE PROJECT NAMED AS PHASE 3

TECHNOLOGY PROJECT NAME: E-COMMERCE PRODUCT PAGE

SUBMITTED BY,

NAME:S.HARI SANKAR

MOBILE.NO:7904783190

E-Commerce Product Page – Detailed Project Documentation

1. Project Setup

Tech Stack Selection:

Frontend: React with Tailwind CSS for a fast, responsive UI. Tailwind allows reusable utility classes for clean code.
Backend: Node.js + Express for building RESTful APIs. Express provides routing and middleware flexibility.
Database: MongoDB for its schema-less design and ability to handle large product catalogs easily.
Version Control: GitHub for centralized collaboration, issue tracking, and CI/CD integration.
Package Manager: npm/yarn for dependency management.
Testing: Jest (unit testing), Cypress (end-to-end UI testing), Supertest (API testing).

Environment Setup:

• Install Node.js and npm on local machine.
• Create project folder with client and server subfolders for clear separation.
• Initialize package.json using npm init and set up scripts for dev, build, and start.
• Configure Tailwind CSS with PostCSS and autoprefixer for styling support.
• Use .env file for environment variables like database connection string and JWT secret.

Folder Structure:

/client /src /components (Reusable UI components) /pages (Page-level components) /api (Frontend API calls) package.json /server /routes (API routes) /controllers (Business logic) /models (MongoDB schemas) server.js package.json

2. Core Features Implementation

Product Listing Page – Fetch products from API and display in grid layout with price, image, and ratings.

Pagination and Infinite Scroll can be implemented for performance with large product lists.

Search & Filter – Implement debounce search on frontend and server-side filtering for better results.

Product Detail Page – Dynamic routing for product detail view, display images carousel, stock status.

Cart Management – Store cart data in React Context or Redux, sync with backend for logged-in users.

Authentication – Use JWT with HTTP-only cookies for secure login, implement protected routes.

Responsive Design – Mobile-first approach with Tailwind breakpoints (sm, md, lg, xl).

Accessibility – Follow WCAG guidelines, use semantic HTML tags for better SEO and screen-reader support

3. Data Storage (Local State / Database)

Frontend Local State: React Context API for small apps, Redux for larger projects.
Key States: cartItems, currentUser, filters, searchQuery.
Database Schema:
Products Collection – Fields: name, description, price, images[], category, stock, ratings, createdAt.
Users Collection – Fields: name, email, passwordHash, role (admin/customer), createdAt.
Orders Collection – Fields: userId, products[], totalAmount, paymentStatus, orderStatus, createdAt.
Indexes: Add indexes on product name and category for faster search queries.

4. Testing Core Features

Unit Tests:

• Test UI components using React Testing Library and Jest snapshots.
• Test functions like price calculation and cart quantity updates.
Integration Tests:
• Test API endpoints using Supertest to ensure routes return correct responses.
• Test database operations with an in-memory MongoDB server to avoid production data changes.

E2E Tests (Cypress):

• Simulate user flows – browsing products, adding to cart, logging in, and checkout process.
• Test responsiveness on different viewports (mobile, tablet, desktop).
CI/CD Integration:
• Run test suites automatically on GitHub Actions before merging pull requests

5. Version Control (GitHub)

Repository Setup: Create GitHub repo, add README.md with setup instructions

Branching Strategy:

- main → production-ready code.
- dev → integration branch for features.
- feature/* → each new feature in its own branch (e.g., feature/cart).

Workflow Example:

- Developer creates a new branch, implements feature, pushes code, opens PR.
- CI runs tests automatically, after approval PR is merged into dev.
- Periodically dev is merged into main after QA approval.

Commit Guidelines:

- Use Conventional Commits (feat:, fix:, refactor:, docs:, test:) for better changelogs.

Code Review:

- Mandatory peer review before merging to maintain code quality.
- Use GitHub Discussions or Issues for feature planning and bug tracking