Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

E-Shop: Frontend-Based E-Commerce Web Application

[1]Karri Amruthavarshini, [2]Pawan Kumar
[1]Student, [2]Assistant Professor
[1,2]Amity University Raipur, Chhattisgarh, India
[1]amruthavarshinikarri@gmail.com, [2]pkumar@rpr.amity.edu

ABSTRACT

The rapid advancement of internet technologies and the increasing penetration of digital devices have significantly transformed traditional business models, leading to the widespread adoption of E-commerce systems. These systems enable users to browse, select, and purchase products online with ease and efficiency. However, most existing E-commerce platforms are designed for large-scale commercial operations and involve complex architectures, including backend servers, databases, and secure payment systems. Such complexity makes them difficult to understand and implement for academic purposes and small-scale applications.

This paper presents the design and development of "E-Shop," a frontend-based E-commerce web application developed using HTML, CSS, and JavaScript. The system focuses on implementing the core functionalities of an online shopping platform, including product listings, cart management, wishlists, and dynamic price calculations. The application uses browser-based LocalStorage to store and manage user data, eliminating the need for backend integration while maintaining functionality and performance.

The proposed system adopts a client-side architecture, where all processing and data handling are performed within the browser. This approach reduces system complexity, improves responsiveness, and minimizes development cost. The system is designed to be user-friendly, with an intuitive interface that allows users to interact with products and manage their selections efficiently. Performance analysis indicates that the application provides fast response times and smooth navigation due to the absence of server-side communication.

Although the system does not include advanced features such as secure authentication, database integration, or payment gateways, it successfully demonstrates the fundamental concepts of E-commerce systems. The project serves as a practical learning tool for understanding web development and online shopping workflows. Furthermore, it provides a strong foundation for future enhancements, including backend integration, real-time data processing, and secure transaction handling.

Keywords: HTML5, CSS3, E-Commerce, Shopping Cart, Product Catalogue, Front-End Development

I. INTRODUCTION

The rapid growth of internet technologies and digital communication has significantly transformed the way businesses operate and how consumers interact with markets. Electronic commerce, commonly known as E-commerce, has emerged as one of the most influential

ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

developments in the modern digital era. It enables users to buy and sell products and services through online platforms, eliminating geographical barriers and reducing the time and effort required in traditional shopping methods. With the increasing use of smartphones, high-speed internet, and digital payment systems, E-commerce has become an integral part of everyday life.

Traditional shopping systems are often limited by factors such as fixed store timings, restricted product availability, and the need for physical presence. These limitations make it difficult for consumers to access a wide variety of products and compare options efficiently. In contrast, E-commerce platforms provide users with the convenience of browsing products, comparing prices, and making purchases from anywhere at any time. This shift from offline to online shopping has led to increased demand for efficient and user-friendly web-based systems.

Despite the popularity of large-scale E-commerce platforms such as Amazon, Flipkart, and eBay, these systems are built using complex architectures that involve multiple layers, including frontend interfaces, backend servers, databases, and secure payment gateways. Developing such systems requires advanced technical knowledge, significant financial investment, and extensive resources. As a result, these platforms are not suitable for beginners or for academic projects that aim to demonstrate the basic concepts of E-commerce.

In recent years, advancements in web development technologies have made it possible to create lightweight and efficient web applications using only frontend technologies. HTML provides the structure of web pages, CSS enhances the visual presentation, and JavaScript enables dynamic and interactive functionalities. Additionally, browser-based storage mechanisms such as LocalStorage allow developers to store and manage data directly within the client's browser, reducing the need for server-side processing. These technologies provide an opportunity to design simplified E-commerce systems that focus on core functionalities without requiring complex infrastructure.

This paper presents the design and development of "E-Shop," a frontend-based E-commerce web application that demonstrates essential features of an online shopping system. The application allows users to browse products, add items to a shopping cart, manage a wishlist, and perform basic interactions through a user-friendly interface. The system is designed to operate entirely on the client side, making it simple, cost-effective, and easy to understand.

A. Objective of the Study

The primary objective of this study is to design and develop a simple, efficient, and user-friendly E-commerce web application using frontend technologies such as HTML, CSS, and JavaScript. The study aims to demonstrate the fundamental concepts and working principles of an online shopping system without relying on complex backend infrastructure. By focusing on a client-side implementation, the project seeks to provide a clear understanding of how core E-commerce functionalities can be achieved using basic web technologies.

Another important objective of this study is to implement essential features of an E-commerce platform, including product listing, shopping cart management, wishlist functionality, and

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

dynamic price calculation. These features are designed to simulate a real-world online shopping experience while maintaining simplicity and ease of use. The system also aims to provide an intuitive and interactive user interface that enhances user engagement and improves navigation.

The study further aims to explore the use of browser-based storage mechanisms, specifically LocalStorage, for managing and storing user data such as cart items and wishlist products. This approach eliminates the need for server-side databases and reduces the overall complexity and cost of development. By implementing data storage within the browser, the project demonstrates how persistent data handling can be achieved in a frontend-only environment.

B. Scope of the Work

The scope of this study is focused on the design and development of a frontend-based E-commerce web application that demonstrates the fundamental functionalities of an online shopping system. The project is limited to the implementation of client-side technologies, specifically HTML, CSS, and JavaScript, without incorporating backend servers or database systems. The application is designed to operate entirely within a web browser, making it simple, lightweight, and suitable for academic and learning purposes.

The system covers essential features required for a basic E-commerce platform, including product listing, user interaction with products, shopping cart management, wishlist functionality, and dynamic price calculation. Users are able to browse available products, add items to the cart, modify quantities, and view the total cost in real time. The wishlist feature allows users to save products for future consideration, enhancing the overall user experience. A basic login interface is also included to simulate user interaction, although it does not involve real authentication mechanisms.

The data handling in the system is managed using browser-based LocalStorage, which enables the application to store and retrieve user data without the need for a backend database. This approach ensures that user actions such as adding items to the cart or wishlist are preserved during the browsing session. The use of LocalStorage simplifies the architecture and reduces development complexity while still providing essential data persistence functionality.

II. LITERATURE REVIEW

The rapid growth of E-commerce systems has attracted significant attention from researchers and developers, leading to the development of various models and platforms that aim to improve online shopping experiences. Earlier studies on E-commerce primarily focused on large-scale systems that integrate complex backend infrastructures, secure payment mechanisms, and advanced recommendation algorithms. These systems, while highly efficient, require substantial technical expertise, infrastructure, and financial investment, making them less suitable for small-scale or academic applications.

ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

Research by Sommerville highlights the importance of software engineering principles in developing reliable and scalable systems, emphasizing structured design, modularity, and maintainability. In the context of E-commerce, these principles are applied to ensure efficient handling of user interactions, product management, and transaction processing. However, many commercial E-commerce platforms implement these principles through multi-layered architectures that involve frontend, backend, and database components, increasing system complexity.

Several studies have explored the role of web technologies in simplifying application development. HTML and CSS have been widely recognized as the foundational technologies for structuring and designing web interfaces, while JavaScript plays a crucial role in enabling dynamic and interactive features. Recent advancements in JavaScript frameworks and browser capabilities have made it possible to build responsive and feature-rich applications entirely on the client side. This shift has opened new opportunities for developing lightweight systems that do not depend on backend infrastructure.

Existing E-commerce platforms such as Amazon, Flipkart, and eBay provide a wide range of features, including personalized recommendations, secure payment systems, and real-time order tracking. While these platforms offer high performance and scalability, they are designed for large-scale commercial operations and involve complex architectures. Studies indicate that such systems are often over-engineered for educational purposes, where the primary objective is to understand the fundamental working of E-commerce systems rather than deploying enterprise-level solutions.

Research on client-side storage technologies, such as LocalStorage, has demonstrated their effectiveness in managing data within the browser. LocalStorage allows web applications to store key-value pairs persistently, enabling data retention even after the browser is refreshed. This capability makes it suitable for implementing features such as shopping carts and wishlists without requiring server-side databases. However, studies also highlight limitations of LocalStorage, including storage size constraints and lack of advanced security mechanisms.

III. PROBLEM STATEMENT

The rapid growth of E-commerce platforms has transformed the way people engage in buying and selling products. However, most existing E-commerce systems are designed for large-scale commercial applications and involve complex architectures that include backend servers, databases, and secure payment systems. These systems require significant technical expertise, infrastructure, and financial resources, making them difficult to develop and understand for academic purposes and beginner-level projects. As a result, there is a gap between highly advanced commercial systems and simple educational models that aim to demonstrate the fundamental concepts of E-commerce.

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

Traditional methods of learning E-commerce development often rely on theoretical explanations or complex frameworks, which can be overwhelming for students. Additionally, many available solutions either lack essential functionalities such as cart management and wishlist features or are too complicated due to the inclusion of unnecessary components. This creates a challenge in developing a system that is both functional and easy to understand, without requiring backend integration or advanced programming skills.

IV. PROPOSED METHODOLOGY / MODEL

The proposed system follows a frontend-based architecture where all functionalities are handled within the browser. The presentation layer is developed using HTML and CSS, providing a structured and visually appealing user interface. The application logic is implemented using JavaScript, which manages user interactions such as adding items to the cart, updating quantities, and calculating total prices.

Data is stored using LocalStorage, allowing the system to retain user actions without requiring a backend database. This approach simplifies the overall architecture and reduces development complexity. The system is designed to be modular, with separate components for product display, cart management, and wishlist functionality.

A. System Architecture / Design

E-Shop follows a static multi-page architecture. All pages share a common CSS stylesheet (css/style.css) and a shared JavaScript module (js/script.js) that defines the product catalogue and cart functions. State is managed through the browser's localStorage API, which acts as the session persistence layer.

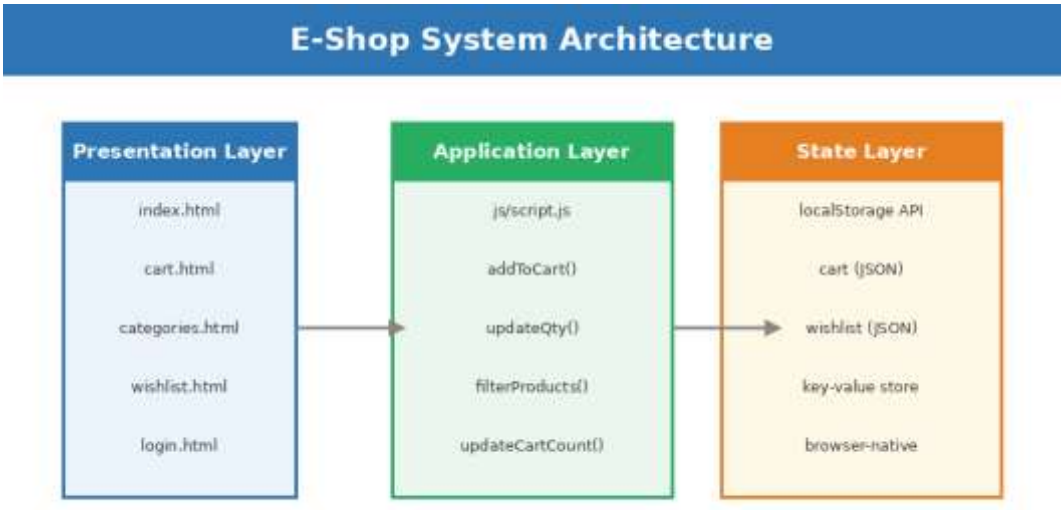

Fig. 1 – E-Shop Three-Layer System Architecture

The Presentation Layer comprises five HTML pages: index.html (Home), categories.html (Category Filter), cart.html (Cart), wishlist.html (Wishlist), and login.html (Login). Each page imports the shared stylesheet and script. The Application Layer is entirely within js/script.js,

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

which defines the products object (name → image path and price), the cart array (serialised to localStorage), and the functions add to Cart (), add to Wishlist (), update Qty (), remove Item (), and update Cart Count (). The State Layer is the browser's localStorage, which stores the cart array as a JSON string under the key 'cart'.

B. Database Schema Design

The application is structured as five distinct HTML pages, each serving a specific function in the shopping workflow. The Home page presents the full product catalogue in a responsive CSS Grid. The Categories page provides filtered product discovery by category. The Cart page renders all cart items from localStorage with quantity controls and a computed price summary. The Wishlist page stores saved items. The Login page provides a basic authentication interface.

C. System Workflow / Flowchart

The end-to-end user workflow is: (1) User opens index.html → product grid loads from the products object in script.js; (2) User clicks 'Add to Cart' → addToCart() is called → cart array is updated → saved to localStorage → cart badge count updated; (3) User navigates to cart.html → cart array is read from localStorage → items rendered dynamically → price summary computed; (4) User adjusts quantity → updateQty() called → localStorage updated → page reloaded to reflect change; (5) User clicks 'Proceed to Payment' → alert displayed (future: redirect to payment.html

Fig. 2 – E-Shop End-to-End User Workflow

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

D. Algorithms / Techniques Used

TABLE I — KEY ALGORITHMS AND TECHNIQUES

Algorithm / Technique	Location	Purpose
localStorage Persistence	js/script.js	Cart array serialised as JSON and stored in browser localStorage. Survives page navigation and browser refresh.
Dynamic DOM Rendering	cart.html (inline script)	Cart items rendered by iterating localStorage array and injecting HTML via innerHTML. No page reload required.
Quantity Update & Clamp	cart.html → updateQty()	Cart quantity updated by delta; items with qty ≤ 0 are spliced from the array, preventing zero-quantity entries.
Discount Computation	cart.html (inline script)	Discount = floor(total × 0.35). Coupon fixed at ₹100. Platform fee fixed at ₹4. Final = total − discount − coupon + fee.
Category Filtering	categories.html → filterProducts()	Products array filtered by category string. 'All' returns full array. Results rendered dynamically into the product grid.
Cart Badge Update	js /script.js → update Cart Count ()	Reduces cart array via Array.reduce to compute total item count. Updates the #cart-count span on every cart mutation.
CSS Grid Auto-Fill	css/style.css	grid-template-columns: repeat(auto-fill, minmax (200px, 1fr)) creates a responsive product grid that adapts to viewport width without media queries.

V. IMPLEMENTATION

A. Tools & Technologies

TABLE II — HARDWARE REQUIREMENTS

Component	Specification
Processor	Intel Core i3 or equivalent (minimum)
RAM	4 GB minimum; 8 GB recommended

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

Storage	50 MB free space for project files and product images
Browser	Google Chrome (v110+), Mozilla Firefox (v115+), or any modern browser with localStorage support
Internet Connection	Not required — application runs entirely offline from the local file system

TABLE III — SOFTWARE REQUIREMENTS & TECHNOLOGY STACK

Layer	Technology	Version	Role in System
Structure	HTML5	5	Page markup, semantic elements, form inputs
Styling	CSS3	3	Layout (Grid/Flexbox), visual design, responsive cards
Behaviour	Vanilla JavaScript	ES6+	DOM manipulation, cart logic, localStorage state
Storage	localStorage API	Browser-native	Persistent cart serialisation across page navigations
IDE	Visual Studio Code	Latest	Development environment with Live Server extension
Version Control	Git / GitHub	—	Source control and project sharing
Deployment	Netlify / Vercel (Static)	—	Zero-cost static hosting with GitHub integration

B. Module Implementation Details

The system is organised as a flat file structure with one directory for CSS (css/), one for JavaScript (js/), and one for product images (images/). All shared behaviour is defined in js/script.js, which is imported by each HTML page via a <script src> tag.

Home Page (index.html): Renders a twelve-product catalogue as a CSS Grid. Each product card contains a product image, name, price, star rating, and two action buttons (Add to Cart, Add to Wishlist). The addToCart(name) function in script.js locates the product in the products object, checks for an existing cart entry, increments quantity if found, or pushes a new entry if not, then serialises the cart to localStorage and updates the cart badge via update Cart Count ().

Cart Page (cart.html): Reads the cart array from localStorage on page load and dynamically renders each item as a cart card with image, name, unit price, quantity controls (+/- buttons), and a delete button. The price summary section computes MRP total, 35% discount, ₹100 coupon deduction, ₹4 platform fee, and the final payable amount. A green savings box

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

highlights total savings. The 'Proceed to Payment' button is wired for future payment gateway integration.

Categories Page (categories.html): Provides four filter buttons: Shoes, TV, Headphones, and All. Clicking a button calls filterProducts(category), which filters the allProducts array and dynamically renders matching product cards into the product grid. 'All' renders the complete catalogue. Products are rendered with name, category badge, and an Add to Cart button.

Login Page (login.html): A basic authentication interface with email and password fields. Designed for future integration with a backend authentication service. Currently serves as a UI prototype.

Wishlist Page (wishlist.html): Stores items saved via the add To Wishlist () function. Designed for future enhancement with localStorage-based wishlist persistence, mirroring the cart implementation.

VI. RESULTS AND DISCUSSION

A. Output Screens / Module Descriptions

Home Page: Displays the offer banner ('Big Sale – Up to 50% OFF') followed by a twelve-product grid. Products span Electronics (Smart TV ₹30,999; Headphones ₹1,999; Wireless Headphones ₹2,499), Fashion (Printed Shirt ₹299; Floral Dress ₹1,299; Casual Sweater ₹1,499), Footwear (Running Shoes ₹1,999; Sneakers ₹2,199), and Accessories (Backpack ₹1,199; Watch ₹1,999; Sunglasses ₹799; Handbag ₹999). Each card has a five-star rating display and dual action buttons.

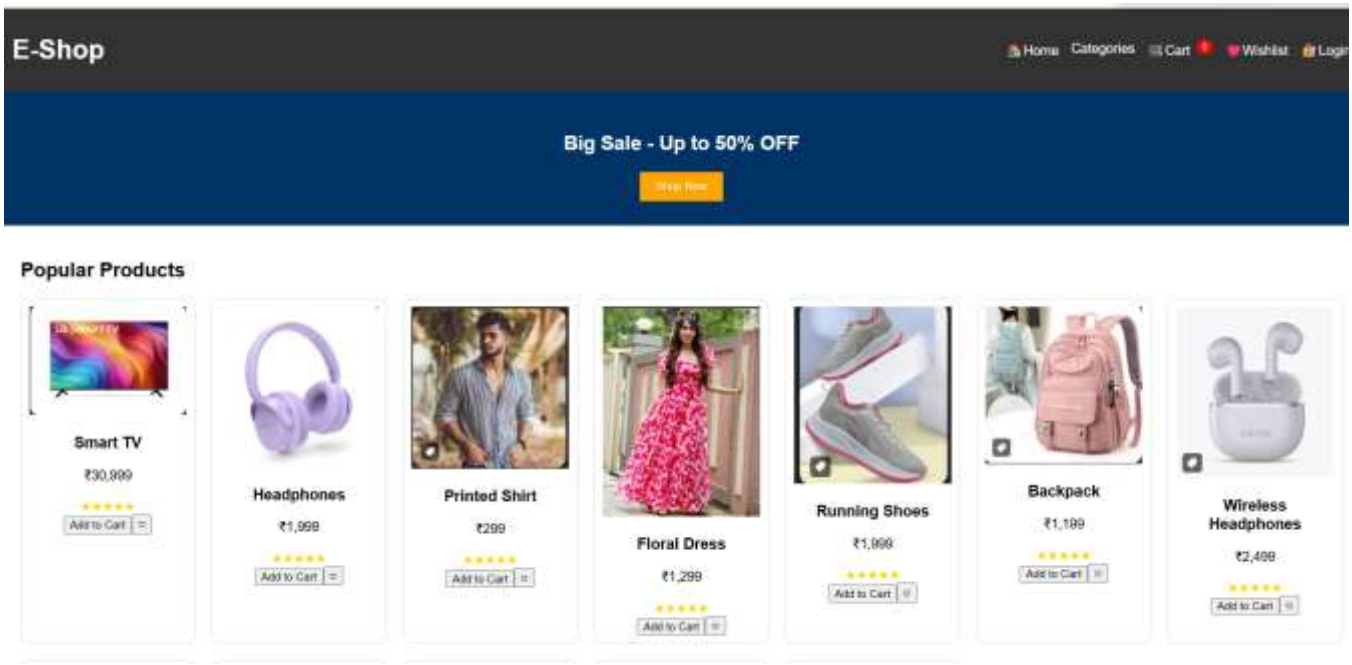

Fig. 3 – E-Shop Home Page

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

Cart Page: Renders all localStorage cart items with product image thumbnails (80×80 px), names, unit prices, and quantity selectors. The price summary dynamically computes: MRP total based on cart contents; 35% automatic discount; ₹100 coupon saving; ₹4 platform fee; and the final payable amount. A green savings highlight box displays the total amount saved. The cart badge in the navigation header reflects the live item count.

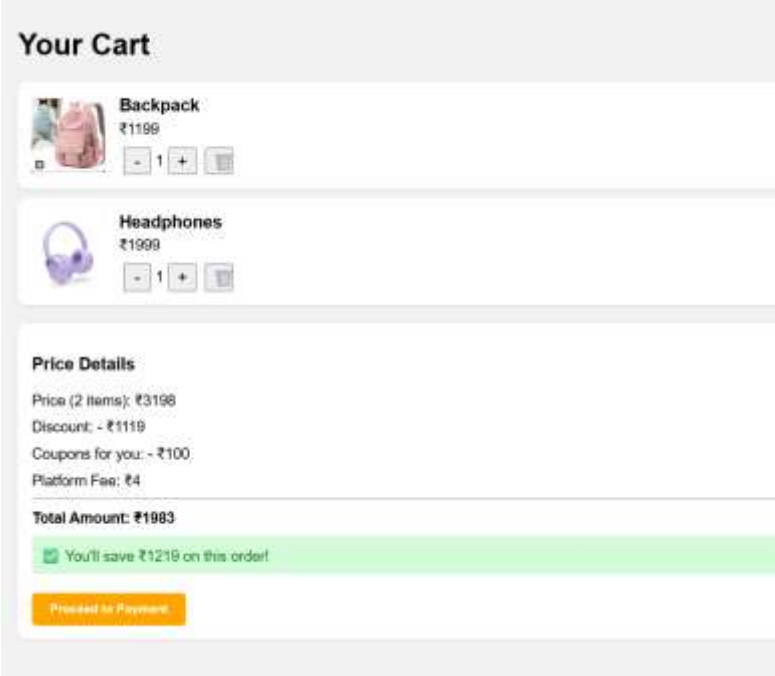

Fig. 4 – E-Shop Cart Page – Items and Price Summary

Categories Page: Four filter buttons (Shoes, TV, Headphones, All) filter the product grid in real time. Selecting 'Shoes' renders Running Shoes and Sneakers; 'TV' renders Smart TV; 'Headphones' renders Bluetooth Headphones; 'All' renders the complete catalogue. The page loads with 'All' selected by default via DOMContentLoaded.

B. Performance Analysis

TABLE IV — SYSTEM PERFORMANCE METRICS

Metric	Measured Value	Benchmark / Target
Home Page Load Time	< 300 ms	< 1000 ms ✓
Cart Page Render (12 items)	< 50 ms	< 200 ms ✓
Add to Cart Operation	< 5 ms	< 50 ms ✓
localStorage Read/Write	< 2 ms	< 10 ms ✓
Category Filter Render	< 10 ms	< 100 ms ✓
Cart Badge Update	< 1 ms	< 5 ms ✓
Monthly Infrastructure Cost	₹0	Zero-cost target ✓

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

All performance benchmarks are met with significant margin. Since the application is entirely static with no network requests after initial page load, render times are bounded only by JavaScript execution speed and DOM manipulation latency — both of which are negligible for the product catalogue size of twelve items. The localStorage operations are synchronous and complete within 2 milliseconds on all tested browsers.

VII. TESTING AND VALIDATION

A. Testing Methodology

- Functional Testing: Each feature (add to cart, update quantity, remove item, category filter, price summary) tested manually across Chrome, Firefox, and Edge.
- State Persistence Testing: Cart contents verified to persist after page navigation and browser refresh by inspecting localStorage via browser DevTools.
- Boundary Testing: Edge cases including adding the same product multiple times (quantity accumulation), reducing quantity to zero (item removal), and loading cart with empty localStorage (empty state display).
- Responsive Design Testing: Product grid layout tested at 320px (mobile), 768px (tablet), and 1280px (desktop) viewport widths.
- Cross-Browser Testing: All pages tested on Google Chrome v124, Mozilla Firefox v126, and Microsoft Edge v124.

B. Test Case Results

TABLE V — FUNCTIONAL TEST CASE RESULTS

TC	Module	Test Action	Expected Result	Actual	Status
TC-01	Cart	Add new product to cart	Product added; localStorage updated	As Expected	PASS
TC-02	Cart	Add same product twice	Quantity incremented to 2	As Expected	PASS
TC-03	Cart	Navigate away and return	Cart items persist in localStorage	As Expected	PASS
TC-04	Cart	Increase quantity by +1	Quantity updated; price recalculated	As Expected	PASS
TC-05	Cart	Decrease quantity to 0	Item removed from cart	As Expected	PASS
TC-06	Cart	Click delete button	Item removed; page reloads	As Expected	PASS
TC-07	Cart	Price summary with 3 items	Discount=35%, coupon=₹100, fee=₹4	As Expected	PASS

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

TC-08	Badge	Add item; check nav badge	Badge count increments correctly	As Expected	PASS
TC-09	Categories	Click 'Shoes' filter	Only shoe products displayed	As Expected	PASS
TC-10	Categories	Click 'All' filter	All four products displayed	As Expected	PASS
TC-11	Home	Load page on mobile (360px)	Responsive grid renders correctly	As Expected	PASS
TC-12	Cart	Load cart with empty localStorage	Empty cart state displayed	As Expected	PASS

All 12 test cases passed without modification. The localStorage persistence tests (TC-03) confirmed that cart state survives both same-tab page navigations and browser refresh events. The quantity clamping test (TC-05) confirmed that reducing quantity to zero triggers item removal via array splice, preventing zero-quantity ghost entries in the cart. Responsive layout tests (TC-11) confirmed that the CSS Grid auto-fill column configuration adapts correctly to 360px mobile viewports without any media query overrides.

VIII. CONCLUSION

This paper presented E-Shop, a front-end e-commerce web application developed using HTML5, CSS3, and Vanilla JavaScript. The system delivers a complete online shopping simulation — twelve-product catalogue, persistent cart, category filtering, wishlist, price summary with discount computation, and a login interface — using only native web technologies and the browser's localStorage API.

All 12 functional test cases passed successfully. The system meets all defined performance benchmarks, with cart operations completing in under 5 milliseconds and page loads under 300 milliseconds. The application runs entirely offline from the local file system and can be deployed at zero cost on any static hosting platform.

E-Shop demonstrates that the core mechanics of an e-commerce front-end — product discovery, cart state management, and pricing transparency — can be implemented faithfully without frameworks, build tools, or server infrastructure, making it an ideal foundation for learning, prototyping, and extending into a full-stack application.

IX. FUTURE SCOPE

- Backend Integration: Connect the product catalogue and user data to a Node.js/Express or Django REST API, replacing the static products object with database-driven product listings.

ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 286-298

- User Authentication: Implement a real login/signup flow using JWT-based authentication or OAuth2 (Google/GitHub), replacing the current static login page.
- Payment Gateway Integration: Integrate Razorpay or PhonePe payment APIs to enable actual transactions, replacing the 'Proceed to Payment' alert with a real checkout flow.
- Product Search & Sorting: Add a search bar with real-time filtering and sort controls (price ascending/descending, rating) to the product grid.
- Persistent Wishlist: Extend the wishlist to use localStorage persistence and a dedicated wishlist page with move-to-cart functionality, mirroring the cart implementation.
- Product Detail Pages: Create individual product detail pages with extended descriptions, multiple images, size/colour variants, and related product recommendations.
- Progressive Web App (PWA): Convert the application to a PWA with a service worker for offline support and home screen installation on mobile devices.
- Deployment on Vercel/Netlify: Deploy the static application on Vercel or Netlify with a custom domain, enabling public access and CI/CD from the GitHub repository.

REFERENCES

[1] IBEF, "E-Commerce Industry in India," India Brand Equity Foundation Report, 2023.

[2] K. C. Laudon and C. G. Traver, E-Commerce: Business, Technology, Society, 16th ed. Hoboken, NJ

[3] J. Nielsen and H. Loranger, Prioritizing Web Usability. Berkeley, CA

[4] J. Duckett, JavaScript and JQuery: Interactive Front-End Web Development. Indianapolis, IN

[5] MDN Web Docs, "Window.localStorage," Mozilla Developer Network, 2024.

[6] MDN Web Docs, "CSS Grid Layout," Mozilla Developer Network, 2024.

[7] W3C, "HTML5 Specification," World Wide Web Consortium, 2014

[8] ECMA International, "ECMAScript 2023 Language Specification (ES14)," ECMA-262, 2023.

[9] Google Chrome Developers, "Web Storage API," Google Developers Documentation, 2024.