International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Design and Development of a Responsive Online Footwear E-Commerce Website

[1]Prof. Mohammad Asif, [2]Yash Naginbhai Patel
[1]Assistant Professor, Department of Computer Science and Engineering,
Parul Institute of Technology, Parul University, Gujarat, India
[2]Students of Computer Science and Engineering,
Parul Institute of Engineering and Technology, Parul University, Gujarat, India

ABSTRACT

The rapid growth of digital commerce has transformed the global retail landscape, making ecommerce an indispensable channel for businesses across all sectors, including the footwear industry. This research paper presents the design, development, and evaluation of a fully responsive online footwear e-commerce website built using modern web technologies. The proposed system incorporates a user-friendly interface, product catalog management, shopping cart functionality, secure payment gateway integration, and an admin dashboard for inventory management. The development process follows an Agile methodology and employs HTML5, CSS3, JavaScript (React.js), Node.js, and MongoDB as the core technology stack. A responsive design approach using Bootstrap and CSS media queries ensures seamless accessibility across desktop, tablet, and mobile devices. The website achieves high usability scores in user testing and demonstrates strong performance benchmarks, with an average page load time under 2.5 seconds and a Google Lighthouse performance score above 85. The findings confirm that the proposed system effectively meets the functional and non-functional requirements for a modern e-commerce platform, providing a scalable foundation for future enhancements such as AI-based product recommendation and ARpowered virtual try-on features.

Keywords: *E-Commerce, Responsive Web Design, Footwear Website, Node.js, React.js, MongoDB, Bootstrap, Online Shopping, Web Development, UI/UX Design*

1. Introduction

The global footwear market is one of the fastest-growing segments of e-commerce, valued at over USD 365 billion in 2023 and projected to reach USD 530 billion by 2030 (Statista, 2024). The proliferation of smartphones and high-speed internet has accelerated the shift from physical retail to online shopping, compelling businesses to establish robust digital presences. Consumers today expect seamless, responsive, and visually appealing shopping experiences that function flawlessly across all devices.

Despite the clear market demand, many small and medium-sized footwear retailers still rely on outdated static websites or generic marketplace listings that fail to provide the immersive, brand-consistent experience customers expect. There is a significant gap between what customers demand and what traditional web solutions can deliver in terms of responsiveness, speed, and interactivity.

This research addresses that gap by documenting the end-to-end design and development of a responsive online footwear e-commerce website. The paper covers requirement analysis, system design, technology selection,

implementation, and testing, providing both a practical development guide and an evaluative academic contribution to the field of web engineering.

1.1 Research Objectives

The primary objectives of this research are:

- To design and develop a fully functional, responsive e-commerce website specifically tailored for footwear retail.
- To implement core e-commerce features including product catalog, search and filter, shopping cart, user authentication, and secure checkout.
- To ensure cross-device compatibility using responsive web design principles.
- To evaluate the system's performance, usability, and security against established web standards.
- To identify future enhancement opportunities including AI and AR technologies.

1.2 Scope of the Study

The scope of this research encompasses the front-end and back-end development of the e-commerce platform, database design, integration with a payment gateway (sandbox environment), and user testing. The study does not cover physical inventory management systems, logistics, or live financial transactions.

2. Literature Review

The field of e-commerce web development has been extensively studied, with researchers highlighting evolving best practices in user interface design, system architecture, and performance optimization.

Kumar and Sharma (2021) investigated the impact of responsive web design on consumer purchase intent and found that users were 67% more likely to complete a purchase on a mobile-optimized site compared to a non-responsive counterpart. Their study underscores the critical role of responsiveness in conversion rate optimization.

Zhang et al. (2020) conducted a comparative study of monolithic versus microservices architectures for ecommerce platforms. Their findings suggest that while monolithic architectures are simpler to develop initially, microservices offer superior scalability and fault isolation for high-traffic applications. This research adopts a modular monolithic approach as a balance between simplicity and scalability for a medium-scale footwear store.

The importance of page speed in e-commerce is well-documented. Google's research (2019) demonstrated that a one-second delay in mobile page load time can reduce conversions by up to 20%. Consequently, performance optimization—through image compression, lazy loading, and code minification—is treated as a first-class requirement in this project.

Concerning the footwear sector specifically, Patel and Mehta (2022) highlighted the significance of highquality product imagery, detailed size guides, and user reviews as primary drivers of purchase decisions in online footwear shopping. These insights directly inform the UX design decisions documented in this paper.

Prior work on MERN stack (MongoDB, Express.js, React.js, Node.js) e-commerce platforms, such as that by Ali and Hassan (2023), demonstrates the stack's suitability for building scalable, real-time web applications, supporting the technology choices made in this project.

3. System Requirements Analysis

3.1 Functional Requirements

The functional requirements were gathered through stakeholder interviews and competitive analysis of leading footwear e-commerce platforms including Zappos, Nike.com, and ASOS. The key functional requirements are categorized as follows:

User Module:

- User registration, login, and profile management with JWT-based authentication.
- Browse products by category (men, women, kids, sports, casual, formal).
- Advanced search with filters for size, color, brand, price range, and rating.
- Product detail pages with image gallery, size chart, and customer reviews.
- Add to cart, wishlist, and order tracking functionality.

Admin Module:

- Secure admin dashboard for product, category, and order management.
- Inventory tracking with low-stock alerts.
- Sales analytics and reporting dashboard.
- Customer management and order fulfillment workflow.

Payment and Checkout:

- Multi-step checkout with address management and order summary.
- Integration with Stripe payment gateway (sandbox mode).
- Order confirmation emails via NodeMailer.

3.2 Non-Functional Requirements

Requirement	Specification	Target Metric
Performance	Page load time for primary pages	< 2.5 seconds
Requirement	**Specification**	**Target Metric**
Scalability	Concurrent user support without degradation	Up to 500 concurrent users
Security	HTTPS, input sanitization, secure auth	OWASP Top 10 compliant
Usability	SUS (System Usability Scale) score	> 75 (Good)

Availability	System uptime percentage	> 99.5%
Responsiveness	Device compatibility	Mobile, tablet, desktop
Accessibility	WCAG compliance level	WCAG 2.1 Level AA

Table 1: Non-Functional Requirements Specification

4. System Design

4.1 System Architecture

The proposed system follows a three-tier client-server architecture consisting of the Presentation Layer (React.js front-end), the Application Layer (Node.js/Express.js REST API), and the Data Layer (MongoDB database). This architecture promotes separation of concerns, making each tier independently maintainable and scalable.

The front-end communicates with the back-end exclusively through RESTful API endpoints secured with HTTPS and authenticated using JSON Web Tokens (JWT). The back-end validates requests, applies business logic, interacts with the database through Mongoose ODM, and returns structured JSON responses. Static assets are served via a Content Delivery Network (CDN) to minimize latency for geographically distributed users.

4.2 Database Design

MongoDB was selected as the database system due to its flexible document-based schema, which accommodates the varied attribute sets of footwear products (e.g., sizes, colors, materials). The primary collections and their key fields are described below:

Collection	Key Fields
Users	_id, name, email, passwordHash, role, address[], wishlist[], createdAt
Products	_id, name, brand, category, description, price, discount, images[], sizes[], colors[], stock, ratings[], averageRating
Orders	_id, userId, items[], shippingAddress, paymentStatus, orderStatus, totalAmount, createdAt
Categories	_id, name, slug, parentCategory, image
Reviews	_id, productId, userId, rating, comment, createdAt
Collection	**Key Fields**
Cart	_id, userId, items[{productId, quantity, size, color}], updatedAt

Table 2: MongoDB Database Collections and Key Fields

4.3 UI/UX Design

The user interface was designed following Nielsen's 10 Usability Heuristics and Material Design principles. Wireframes were first created in Figma, followed by high-fidelity prototypes that underwent two rounds of user feedback before implementation. Key UI/UX decisions include:

- Sticky navigation bar with mega-menu for category browsing and search bar with live autocomplete suggestions.
- Hero banner carousel on the homepage featuring seasonal promotions with clear call-to-action buttons.
- Product listing pages with grid/list toggle, sortable columns, and AJAX-powered filters that update results without full page reloads.
- Detailed product pages with zoomable image gallery, size selector with availability indicators, and an Add to Cart button with instant visual feedback.
- Streamlined 3-step checkout flow (Cart Review → Delivery Details → Payment) to minimize cart abandonment.
- Breadcrumb navigation on all sub-pages to reduce user disorientation.

4.4 Responsive Design Strategy

Responsiveness was achieved through a mobile-first design approach using Bootstrap 5.3 as the CSS framework, supplemented by custom CSS media queries. The breakpoint strategy follows Bootstrap's standard grid system:

- Extra Small (< 576px): Single-column layout, collapsible navigation, stacked product cards.
- Small (576–768px): Two-column product grid, condensed header.
- Medium (768–992px): Three-column product grid, expanded sidebar.
- Large (≥ 992px): Four-column product grid, full navigation mega-menu, sidebar filters visible by default.

Touch-friendly elements, swipe-enabled carousels, and appropriately sized tap targets (minimum 44x44 CSS pixels as per WCAG guidelines) were implemented to optimize the mobile experience.

5. Technology Stack and Implementation

5.1 Technology Selection

Layer	Technology	Justification
Front-End	React.js 18	Component-based architecture, virtual DOM for performance, large ecosystem
Layer	**Technology**	**Justification**
Styling	Bootstrap 5.3 + CSS3	Rapid responsive grid system, pre-built UI components
State Mgmt.	Redux Toolkit	Centralized state for cart, auth, and product filters

Back-End	Node.js + Express.js	Non-blocking I/O, REST API development, large NPM ecosystem
Database	MongoDB + Mongoose	Flexible schema for varied product attributes, JSON-native
Auth	JWT + bcrypt.js	Stateless authentication, secure password hashing
Payment	Stripe API	Well-documented, sandbox testing, PCI-DSS compliant
Deployment	Vercel (FE) + Render (BE)	CI/CD integration, free tier suitable for demonstration
Version Ctrl.	Git + GitHub	Industry-standard, branch-based collaboration workflow

Table 3: Technology Stack Selection

5.2 Development Methodology

The project was developed using an Agile Scrum methodology with two-week sprint cycles over a total development period of twelve weeks. The sprints were organized as follows:

- Sprint 1–2: Project setup, requirement finalization, database schema design, and wireframing.
- Sprint 3–4: Back-end API development — user authentication, product CRUD, category management.
- Sprint 5–6: Front-end development — homepage, product listing, and product detail pages.
- Sprint 7–8: Shopping cart, wishlist, and user profile pages.
- Sprint 9–10: Checkout flow, Stripe integration, and order management.
- Sprint 11: Admin dashboard — product management, order processing, and analytics.
- Sprint 12: Testing, performance optimization, bug fixing, and deployment.

5.3 Key Implementation Details

Authentication and Authorization: JWT tokens with a 24-hour expiry are issued upon successful login. Protected routes on both the front-end (React Router private routes) and back-end (Express middleware) prevent unauthorized access. Refresh token rotation was implemented to extend sessions securely without requiring frequent re-logins.

Image Optimization: Product images are stored on Cloudinary CDN and served in WebP format where supported, with JPEG fallback. Lazy loading is applied to off-screen product images using the native loading='lazy' attribute, reducing initial page weight by approximately 60%.

Search and Filter: A debounced search input reduces unnecessary API calls during typing. Product filtering is implemented server-side using MongoDB aggregation pipelines, enabling efficient multi-parameter filtering on large product catalogs. Filter states are persisted in the URL query string for shareability and browser history support.

Payment Integration: Stripe Elements were used to render PCI-compliant payment input fields hosted by Stripe's servers, ensuring that sensitive card data never touches the application's servers. Payment intents are created server-side and confirmed client-side following Stripe's recommended flow.

6. Testing and Evaluation

6.1 Testing Strategy

A multi-layered testing strategy was adopted, covering unit testing, integration testing, end-to-end testing, usability testing, and performance testing. Jest and React Testing Library were used for unit and integration tests on the front-end, while Supertest was used for API endpoint testing. Cypress was employed for endto-end testing of critical user flows including registration, login, product search, add-to-cart, and checkout.

6.2 Performance Testing Results

Google Lighthouse audits were conducted on the three highest-traffic pages — the Homepage, Product Listing Page, and Product Detail Page — in both desktop and mobile modes. The results are summarized below:

Page	Performance	Accessibility	Best Practices	SEO
Homepage	91	94	96	98
Product Listing Page	87	92	95	96
Product Detail Page	89	95	96	97
Checkout Page	88	93	96	92

Table 4: Google Lighthouse Audit Scores (Desktop Mode)

6.3 Usability Testing

Usability testing was conducted with 20 participants (10 male, 10 female; ages 18–45) who were asked to complete five core tasks: (1) finding and viewing a specific product, (2) applying size and price filters, (3) adding a product to the cart and modifying quantity, (4) completing the checkout process, and (5) writing a product review. Task completion rates, error rates, and time-on-task were measured. Post-test, participants completed the System Usability Scale (SUS) questionnaire.

Results: The average SUS score was 82.5 out of 100, falling in the 'Excellent' category (Bangor et al., 2009). All five tasks achieved completion rates above 90%. The checkout flow recorded the highest error rate
(12%) due to unclear error messages on the payment form, which was subsequently improved. Average task completion time for purchasing a product end-to-end was 4 minutes 32 seconds, considered acceptable for an e-commerce context.

6.4 Security Testing

Security testing was conducted using OWASP ZAP (Zed Attack Proxy) to identify common web vulnerabilities. The following measures were verified and confirmed effective:

- SQL/NoSQL Injection Prevention: All user inputs are validated and sanitized using expressvalidator and mongoose-sanitize before database interaction.
- Cross-Site Scripting (XSS): React's default JSX escaping mitigates most XSS risks; additionally, Content Security Policy (CSP) headers are set server-side.
- Cross-Site Request Forgery (CSRF): SameSite cookie attributes and JWT-based authentication (no session cookies) mitigate CSRF risks.
- Brute Force Protection: Login endpoint rate-limiting using express-rate-limit (5 attempts per 15minute window).
- HTTPS Enforcement: All traffic is served over TLS 1.2+ with automatic HTTP-to-HTTPS redirection.

7. Results and Discussion

The developed footwear e-commerce website successfully meets all defined functional and non-functional requirements. The MERN stack proved to be a highly effective choice for this application, enabling rapid development of both the API and the front-end while maintaining code consistency through a shared JavaScript language across the full stack.

The mobile-first responsive design approach, combined with Bootstrap 5.3, ensured excellent cross-device compatibility verified across Chrome, Firefox, Safari, and Edge on iOS, Android, and desktop platforms. The choice of a CDN for static asset delivery, combined with image optimization through Cloudinary and aggressive caching strategies, produced Lighthouse performance scores consistently above 85, meeting the defined target.

The SUS score of 82.5 confirms that the interface is intuitive and accessible to non-technical users. Qualitative feedback from testers highlighted the product image gallery, the filter-without-reload functionality, and the visual order tracking timeline as particularly appreciated features.

A notable challenge encountered was the management of complex filter state in React, which initially caused inconsistencies between the UI filter controls and the actual query sent to the API. This was resolved by centralizing all filter state in Redux and deriving API query parameters from the Redux store in a single, tested selector function, significantly reducing state-related bugs.

Performance bottlenecks on the product listing page, initially caused by loading all product images eagerly, were resolved through implementing intersection observer-based lazy loading and pagination (12 products per page with infinite scroll). This reduced the initial page data transfer from 4.2 MB to 680 KB, dramatically improving mobile load times.

8. Conclusion and Future Work

8.1 Conclusion

This research has demonstrated the successful design and development of a responsive, feature-complete online footwear e-commerce website using the MERN stack and Bootstrap. The system meets all defined functional requirements—including product catalog management, user authentication, shopping cart, secure payment, and admin dashboard—and achieves strong performance, usability, and security benchmarks. The Agile development methodology facilitated iterative improvement and timely delivery within the twelve-week internship timeline.

The project contributes to the body of knowledge in web engineering by providing a detailed, replicable account of MERN stack e-commerce development with specific application to the footwear retail domain, including domain-specific UX considerations such as size guides, multi-angle product imagery, and material filter options.

8.2 Future Work

Several enhancements are identified for future development:

- AI-Powered Product Recommendations: Integration of a collaborative filtering recommendation engine to personalize product suggestions based on browsing history, purchase patterns, and user similarity.
- Augmented Reality (AR) Virtual Try-On: Implementation of WebAR technology to allow users to visualize footwear on their feet using their smartphone camera, reducing return rates and increasing purchase confidence.
- Progressive Web App (PWA) Conversion: Adding a service worker and web app manifest to enable offline browsing, push notifications, and home screen installation, further closing the gap between web and native mobile app experiences.
- Multi-Vendor Marketplace: Extending the platform architecture to support multiple sellers with individual dashboards, commission management, and independent storefronts.
- Microservices Migration: Decomposing the monolithic API into microservices (product service, order service, user service, notification service) to improve horizontal scalability for high-traffic scenarios.

References

Ali, M., & Hassan, R. (2023). Building scalable e-commerce applications with the MERN stack: A systematic review. Journal of Web Engineering, 22(3), 145–172.

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. Journal of Usability Studies, 4(3), 114–123.

Google. (2019). The mobile speed imperative. Think with Google. https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-newindustry-benchmarks/

Kumar, A., & Sharma, P. (2021). Influence of responsive web design on consumer purchase intent in mobile e-commerce. International Journal of Electronic Commerce, 15(2), 88–104.

Nielsen, J. (1994). 10 usability heuristics for user interface design. Nielsen Norman Group. https://www.nngroup.com/articles/ten-usability-heuristics/

Patel, N., & Mehta, S. (2022). Consumer behavior in online footwear shopping: Key influencing factors. Journal of Retailing and Consumer Services, 64, 102–116.

Statista. (2024). Footwear market worldwide — statistics & facts. Statista Research Department. https://www.statista.com/topics/1185/footwear/

W3C. (2018). Web content accessibility guidelines (WCAG) 2.1. World Wide Web Consortium. https://www.w3.org/TR/WCAG21/

Zhang, Y., Li, W., & Chen, H. (2020). Microservices vs. monolithic architecture in e-commerce systems: A performance and maintainability analysis. IEEE Transactions on Software Engineering, 46(8), 823–840.