Royal International Global Journal of Advance and Applied Research
Peer Reviewed International, Open Access Journal.
ISSN: 2998-4459 | Website: https://rlgjaar.com Volume-2, Issue-8 | August - 2025

Original Article

A Comparative Study on the Role of React.js and Node.js in Modern E-commerce Web Applications

Dr. Nisar Nijam Shaikh[1] Dr: Jasmin Isahak Sayyed[2]
[1,2] Hon. Shri. Babanrao Pachpute Vichardhara Trust's Faculty of Management, Kashti

Manuscript ID:
RIGJAAR-2025-020818

ISSN: 2998-4459
Volume 2
Issue 8
Pp. 113-115
August 2025

Submitted: 08 July 2025
Revised: 13 July 2025
Accepted: 07 Aug. 2025
Published: 31 Aug. 2025

Correspondence Address:
Dr. Nisar Nijam Shaikh
Hon. Shri. Babanrao Pachpute Vichardhara Trust's Faculty of Management, Kashti
Email-
shaikhnisar50@gmail.com

Quick Response Code:

Web. https://rlgjaar.com

DOI:
10.5281/zenodo.23207841

DOI Link:

https://doi.org/10.5281/zenodo.23207841

Creative Commons

Abstract

The rapid growth of online shopping has created a need for faster, more scalable, and interactive websites. With global online sales expected to reach $8.1 trillion by 2026 (Statista, 2024), older website designs that reload whole pages often struggle to keep up. This paper looks at how two popular JavaScript tools—React.js for the frontend and Node.js for the backend—help improve e-commerce websites. By reviewing studies, industry reports, and testing results, we show how these tools make websites faster, handle more users at once, and help developers work more efficiently. Our findings show performance improvements of 30–50%, quicker page updates, better handling of many users, and faster development times. This research suggests that using the MERN stack (MongoDB, Express, React, Node.js) is a strong choice for building modern, high-traffic online stores.

Keywords: *React.js, Node.js, MERN Stack, E-commerce Development, JavaScript Frameworks, Web Performance, Scalability, Microservices*

Introduction

E-commerce is no longer just a business tool — it is now a global economic engine. Over 2.64 billion consumers are expected to shop online in 2024, with increasing expectations for speed, personalization, and reliability (Statista, 2024). In this environment, traditional server-rendered web architectures using PHP, JSP, or ASP.NET often struggle with latency, scalability, and responsiveness.

Modern development paradigms have shifted toward JavaScript-centric stacks that enable single-page applications (SPAs), real-time data synchronization, and dynamic user experiences. React.js, developed by Meta (Facebook), and Node.js, created by Ryan Dahl, are two foundational technologies enabling this shift. Their combined adoption — forming the MERN stack — allows full-stack development using a single language across the entire application lifecycle.

According to the 2024 Stack Overflow Developer Survey, React.js is the most used frontend library (used by 44% of developers), while Node.js is the most preferred backend runtime (used by 42%). Their popularity is driven by speed, flexibility, and the ability to build microservice-oriented, cloud-native applications.

This study explores the practical benefits and limitations of using React.js and Node.js in modern e-commerce web development, evaluating both theoretical foundations and empirical evidence.

Objectives of Research

This research is aimed at the following objectives:

1. To study the architectural principles and working models of React.js and Node.js in the context of web development.
2. To evaluate how these technologies improve performance, scalability, and user experience in e-commerce platforms.
3. To analyze the benefits of the unified JavaScript ecosystem using the MERN stack.
4. To explore recent trends in deployment strategies, including cloud-native and microservice architectures.
5. To identify challenges faced during implementation and propose future enhancements.
6.

Creative Commons (CC BY-NC-SA 4.0)
This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License, which allows others to remix, tweak, and build upon the work noncommercially, as long as appropriate credit is given and the new creations ae licensed under the idential terms.

How to cite this article:
Shaikh, N. N., & Sayyed, J. I. (2025). A Comparative Study on the Role of React.js and Node.js in Modern E-commerce Web Applications. Royal International Global Journal of Advance and Applied Research, 2(8), 113–115. https://doi.org/10.5281/zenodo.23207841

Royal International Global Journal of Advance and Applied Research
Peer Reviewed International, Open Access Journal.
ISSN: 2998-4459 | Website: https://rlgjaar.com Volume-2, Issue-8 | August - 2025

Review of Literature

Several academic and industrial sources highlight the rising dominance of JavaScript-based full-stack solutions in e-commerce development. Key findings from prior studies include:

- IJRASET (2023): A study on React.js and Firebase integration showed a 35% improvement in page responsiveness using Virtual DOM and component reusability. The research emphasized modular architecture, contributing to a 25% reduction in development time.
- Phaedra Solutions (2024): This industry white paper highlighted that eliminating context switching between backend and frontend (through shared JavaScript) resulted in 30% faster API interactions and improved team productivity.
- IJIRMPS (2023): A benchmark comparison of backend frameworks revealed Node.js handles 1.5x more concurrent HTTP requests than PHP servers, thanks to its non-blocking I/O model.
- IJARSCT (2023): An analysis of frontend performance concluded that React.js offers up to 45% faster rendering times in dynamic UIs than traditional server-rendered HTML engines.
- IJGST (Khan & Gonsalvez, 2023): Research on MERN-based cloud applications showed 28% faster deployment cycles and reduced resource overhead compared to LAMP-based deployments.
- Netflix Tech Blog (2023): Netflix adopted Node.js to improve performance and reduce server load, reporting a 70% reduction in startup time for their web interface dashboards.
- PayPal Engineering (2023): PayPal's migration to Node.js led to a 35% drop in response time and a 33% improvement in server throughput across major services.

  These studies consistently show that React.js and Node.js deliver substantial performance and scalability benefits when applied to real-world web systems.

Research Methodology

1 Development Environment

Two prototype e-commerce applications were built for analysis:

- Modern Stack (MERN): MongoDB, Express.js, React.js, and Node.js
- Traditional Stack (LAMP): Linux, Apache, MySQL, and PHP

Both platforms were developed with features such as product catalogs, shopping carts, user registration, authentication, and payment integration.

2 Hosting and Infrastructure

Applications were deployed on AWS EC2 instances with uniform specifications:

- 2 vCPUs, 4 GB RAM, Ubuntu 22.04
- Load balancer (NGINX)
- MongoDB Atlas for the MERN stack

3 Metrics Measured

- Frontend Load Time: Measured using Google Lighthouse and GTmetrix
- API Latency: Measured using Apache JMeter
- Concurrent User Handling: Stress-tested using Locust
- Memory & CPU Usage: Tracked using PM2 and top/htop utilities
- Development Time: Documented over sprint cycles

4 Testing Method

Each system was subjected to 100, 500, 1000, and 2000 concurrent users. Response times and system loads were averaged over three iterations for statistical validity.

Results

1. Frontend Performance:
   React-based UI rendered 38% faster on average compared to the traditional PHP-based HTML views. The use of Virtual DOM and lazy loading contributed to improved perceived performance.
2. API Response Time:
   Node.js APIs averaged 130 ms under medium load, while PHP-based APIs took 220 ms. Under higher load, Node.js maintained sub-200 ms latency, whereas PHP exceeded 300 ms.
3. Scalability:
   Node.js successfully served 1600+ concurrent users without significant performance degradation. PHP-based servers became unstable beyond 1000 users due to thread-blocking architecture.
4. Resource Efficiency:
   Node.js showed 25–28% lower memory usage during peak load due to its event-driven model and single-threaded architecture, as opposed to Apache's multi-threaded behavior.
5. Development Duration:
   The MERN stack project was completed 26% faster due to unified JavaScript development, reusable components, and streamlined state management using Redux.
6. User Experience Feedback:
   A sample group of 50 test users preferred the React-based interface, with 84% describing it as "highly responsive" and 76% reporting smoother checkout flows.

Discussions

1 Unified Language Environment

Using JavaScript across both client and server reduces context switching and fosters collaboration among full-stack developers. This unification shortens onboarding time and simplifies debugging.

2 Real-time Processing and Async Architecture

Node.js shines in real-time features such as live order tracking, chatbots, or dynamic notifications. Its event-driven nature allows handling of thousands of simultaneous socket connections — critical for flash sales or limited-time offers.

3 Modular and Scalable UI Architecture

React's component-based design enables large teams to work on independent modules simultaneously, increasing scalability. Features like code-splitting, React.lazy, and Suspense enhance performance for large applications.

4 Microservices and DevOps Readiness

MERN-based applications align well with microservices architecture. Each service (e.g., user auth, payment, inventory) can be containerized using Docker and orchestrated using Kubernetes, simplifying deployment and scaling.

5 SEO and Server-side Rendering (SSR)

Although React is client-side by default, frameworks like Next.js enable SSR, which improves SEO and initial load time. Node.js complements SSR by efficiently handling server load.

6 Security and Middleware

Node.js supports robust middleware (like Helmet, CORS, JWT) for securing APIs. React enhances frontend security via controlled component rendering and XSS prevention. However, both require regular patching and vulnerability scans.

7 Challenges and Future Work

State Management Complexity: React applications need careful state handling using Redux or Context API to avoid prop-drilling or stale data.

Callback Hell in Node.js: Poorly managed async flows can lead to complex nested callbacks, though mitigated by Promises and async/await.

Serverless Integration: Future adoption of AWS Lambda, Firebase Functions, and GraphQL can further optimize resource usage and developer productivity.

Conclusion

This research demonstrates that React.js and Node.js together offer a powerful foundation for building modern e-commerce web applications. Their unified JavaScript environment, asynchronous and component-based architecture, and cloud readiness make them superior alternatives to legacy stacks. Empirical testing and literature studies confirm significant improvements in performance, scalability, and development efficiency.

As online commerce trends lean toward personalization, real-time engagement, and cross-platform functionality, the MERN stack positions itself as a future-proof solution. While challenges such as state complexity and memory leaks must be addressed, the advantages far outweigh the limitations. Future enhancements may include AI-based product recommendation engines, serverless deployments, and integration with edge computing for ultra-fast global access.

Acknowledgement

The authors sincerely express their gratitude to all those who provided valuable guidance, encouragement, and support in completing this research paper on "A Comparative Study on the Role of React.js and Node.js in Modern E-commerce Web Applications." The study examines the application of React.js and Node.js in improving the performance, scalability, responsiveness, and development efficiency of modern e-commerce platforms.

The authors gratefully acknowledge the valuable contributions of researchers, academic publications, industry reports, and technology organizations whose studies and technical insights provided an important foundation for this research. The authors also appreciate the support of the testing and development resources used to evaluate frontend performance, API latency, concurrent-user handling, resource utilization, and development time.

The authors are thankful to their respective institutions, colleagues, teachers, researchers, and well-wishers for their valuable suggestions, academic guidance, encouragement, and cooperation throughout the preparation of this research work. The authors particularly acknowledge the support received from Hon. Shri. Babanrao Pachpute Vichardhara Trust's Faculty of Management and Faculty of Engineering, Kashti.

Financial support and sponsorship

Nil.

Conflicts of interest

The authors declare that there are no conflicts of interest regarding the publication of thesis paper.

References

1. IJRASET (2023). Developing an E-commerce Web Application with React.js and Firebase. IJRASET Journal.
2. Phaedra Solutions (2024). Advantages of MERN Stack in Modern Web Development.
3. IJIRMPS (2023). Backend Framework Comparison for Web Scalability.
4. IJARSCT (2023). Performance Study on React-based Frontend Systems.
5. Khan, A., & Gonsalvez, J. (2023). MERN Stack Efficiency in Cloud-hosted Applications, IJGST.
6. Statista (2024). Global Online Shopping Report: Forecast to 2026.
7. Stack Overflow (2024). Developer Survey Report.
8. Netflix Tech Blog (2023). Node.js in Action: Performance Improvements and Architecture Shift.
9. PayPal Engineering (2023). From Java to JavaScript: Building a Better Payments Platform.
10. Node.js Foundation (2024). Node.js Performance Metrics and Case Studies.