International Journal of Scientific Engineering and Research (IJSER)
ISSN (Online): 2347-3878
SJIF (2025): 8.036

Beyond Flexibility: Evaluating Performance, SEO, and Architectural Trade-Offs in React and Next.js E-commerce Applications

Samyak Joshi[1], Pankaj Raghuvanshi[2]

[1]Student, M. Tech., Department of CS & E, Alpine Institute of Technology, Dewas Road, Ujjain (M.P.) India
Email: *Joshi.samyak20[at]gmail.com*

[2]Assistant Professor, Department of CS&E, Alpine Institute of Technology, Dewas Road, Ujjain (M.P.) India
Email: *pankajsingh.it22[at]gmail.com*

Abstract: ***The choice of architecture for web application development greatly affects the performance, scalability, and maintenance of contemporary online shopping platforms. The paper provides comparative research of two types of web application architectures: a client-side application created with React and a full-stack solution developed with Next.js, based on the same e-commerce application. Web application performance is measured using such web metrics as LCP (Largest Contentful Paint), FCP (First Contentful Paint), and TTFB (Time to First Byte), as well as scalability, developer experience, and SEO-friendliness. As shown by the research, although React and Next.js have similar performance under warm conditions, React is inferior in its cold performance and does not allow for search engine optimization at all.***

Keywords: LCP (Largest Contentful Paint), FCP (First Contentful Paint), and TTFB (Time to First Byte), SEO (Search Engine Optimization)

1. Introduction

Innovation in web technologies has completely transformed the way modern-day web applications are developed, and e-commerce web applications are at the vanguard. As more people use the Internet to make purchases, there is an increasing need for web applications which not only offer robust functionality but are also high-performance, scalable, and easily searchable by search engines. Web application architecture plays a crucial role in achieving all these objectives [1].

During the initial stages, many of the websites developed were of the MPA (Multi Page Architecture) type, whereby the entire page had to reload every time an action was performed. This approach was simple and was favored by search engines, but the problem was that it led to a lot of delay and minimal interaction capabilities. However, the introduction of SPAs, using frameworks such as React to develop them, changed everything about the development of web applications as they were developed through reloading of pages via Virtual DOM (Document Object Model). React is owned by Facebook and uses a component-based architecture [2].

Despite React SPAs being widely used, they mostly depend on CSR (Client Side Rendering) for their functioning, and it is accompanied by certain drawbacks. The browser must download, parse, and execute JavaScript code in order for anything to be displayed on the page. It makes the initial loading time relatively high, resulting in low scores of important metrics, such as FCP and LCP, which are the metrics users perceive [3]. Additionally, CSR may create difficulties in terms of SEO, as crawlers cannot index dynamically created content [4].

To overcome the limitations, hybrid technologies such as Next.js have emerged. These include enhancements to React, such as Incremental Static Regeneration (ISR), server-side rendering (SSR), and static site generation (SSG). Pre-rendering of content through SSR or SSG reduces loading time and improves SEO. Next.js provides an integrated solution for web applications: one complete framework for web application development that includes routing, API endpoints, image optimization, and other optimizations [5].

It is worth noting that recent research emphasizes the significance of web performance measures commonly known as Core Web Vitals. Such key indicators as LCP (Largest Contentful Paint), FCP (First Contentful Paint), and TTFB (Time to First Byte) play an important role when it comes to rankings provided by search engines and serve as critical criteria for assessing user experience [6]. The findings of the research indicate that minor changes in such indicators may significantly contribute to success of an app [7].

The usage of Next.js comes with certain compromises as well. With improved performance and search engine optimization, Next.js comes with a stronger opinion regarding the architecture used by default, which constrains components to a certain extent and leaves you less room for shaping the application. By contrast, React allows greater freedom in creating an architecture tailored to specific requirements [8].

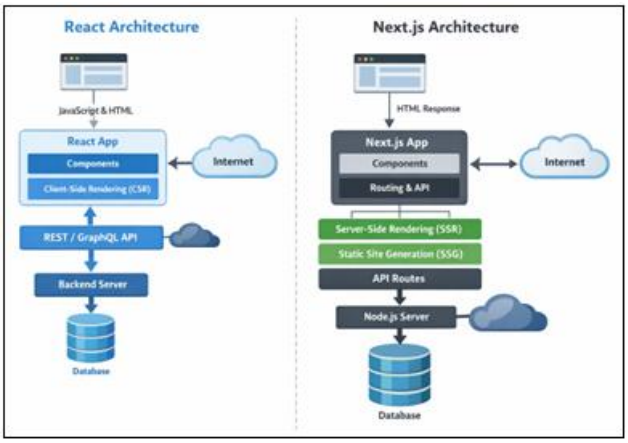

Figure 1: Architecture of React and Next.js

Aside from performance and SEO, there is one more important factor that makes frameworks attractive i.e. developer experience (DX). Developer experience includes many aspects which include ease of installation, learning curve, debuggability, scalability, and maintainability, production readiness. Frameworks that come bundled with some utilities and best practices, such as Next.js, could greatly increase developers' efficiency by eliminating additional setup processes [9]. In turn, frameworks that provide high flexibility like React would require more tooling, which may complicate development but give additional freedom.

As per recent advancements in web engineering, there have been developments in hybrid rendering models that incorporate CSR, SSR, and SSG techniques to achieve the optimal compromise between efficiency and interactivity. From research studies, it is evident that hybrid models can solve the problems associated with each individual rendering technique in terms of scalability and efficiency for complex web apps [10]. One such popular framework that uses a hybrid model is Next.js.

There have been increasing numbers of studies conducted, yet very few of them offer an integrated analysis of performance, SEO optimization, and developer experiences in one study design. Many of the current research studies focus exclusively on performance or SEO optimization without taking into consideration how architecture decisions will impact all of the three areas at the same time. Another issue is the lack of evidence from experiments conducted under similar application environments. There are three major contributions of the research. Firstly, it provides a systematic comparison of React vs. Next.js based on identical performance parameters. Secondly, it investigates how various methods of rendering influence SEO and the UX (User Experience). Finally, it explains the trade-offs of flexible architecture and optimized framework-based development. These contributions can help stakeholders select the best web development frameworks to implement their projects and ensure that the application is optimized for performance and SEO. The research can be extended to advanced optimization and web technologies.

2. Review of related work

2.1 Overview of Previous Work

Current advancements in web development frameworks include the improvement of application performance, SEO, and developer efficiency. In recent years, as applications become increasingly advanced, academics have explored various rendering techniques including client-side rendering (CSR), server-side rendering (SSR), and static site generation (SSG). React and Next.js frameworks have attracted much attention because of their ability to handle current challenges. According to the existing body of literature, there is a noticeable trend in utilizing mixed rendering techniques which can offer the best of both worlds [11], [12].

2.2 Summary of Recent Research Works

According to Patel et al., the use of Next.js enhances web performance and SEO through code splitting, image optimization, and incremental static regeneration (ISR), which not only improve page load time but also the ability of websites to be crawled. In terms of performance improvements, Next.js helps to reduce JavaScript bundle size while offering a better user experience through optimized content delivery [13].

Using a Next.js application, Hanafi et al. compared the performance of three types of rendering techniques – client-side rendering, server-side rendering, and static site generation – and determined that SSG is the fastest of all techniques in terms of page load time and the user experience [14].

Pati and Zaki conducted a direct comparison of React and Next.js by implementing similar applications in varying network conditions. On low-bandwidth networks, Next.js showed superior metrics of performance (such as FCP and TTI). Additionally, Next.js provided for more accessible and user-friendly apps since there was no need for heavy client-side processing [15].

The authors Kumar & Priya (2025) investigated how SEO works in the modern website development, highlighting that web applications created in React and based on the approach of client-side rendering cannot be easily indexed by crawlers. They discovered that Next.js significantly improves SEO thanks to the implementation of SSR and pre-rendering of HTML pages [16].

Brown & Johnson (2022) studied different options of enhancing performance in Next.js applications, highlighting the benefits provided by SSR and SSG technologies. It is possible to claim that hybrid rendering not only allows developers to ensure faster loading speed but also retains interactivity, making Next.js an effective option for scalable apps [17].

Chen (2025) proposed a highly sophisticated technique of rendering that involves the use of modular and adaptive hydration in the context of React and Next.js frameworks. The results of the study suggest that reducing the amount of

superfluous JavaScript execution and applying adaptive hydration can positively affect TTI (Time to Interactive) and FID (First Input Delay) [18].

The benchmark of current JavaScript frameworks by Zhang et al. (2020) found that JavaScript frameworks with server-side rendering gained advantages over pure client-side frameworks in terms of page loading speed and effectiveness of SEO [19]. Recent research with an emphasis on industry issues (Joshi et al., 2026) states that React retains its lead in developing dynamic UI due to its flexible architecture, whereas Next.js becomes more common when high-performance applications requiring good SEO are needed. This comparison reflects the compromise between flexibility and optimization. [20]

2.3 Important Findings from the Literature

A few key findings from literature review include the following:

- Client-side rendering using React is likely to slow down loading times and makes SEO challenging.
- Performance can be improved using server-side rendering, static site generation, and dynamic hybrid rendering in Next.js.
- Important measures of website performance that must be considered for assessing user experience include FCP, LCP, and TTI.
- SEO benefits are significant due to the presence of pre-rendering and optimized content.
- The developer experience with Next.js is enhanced because of several built-in functions and minimal configuration required.

2.4 Identifying the Gaps in Research

Despite the achievements noted, there remain some gaps in the literature, including the following:

- Lack of studies that provide an integrated evaluation that would address performance, SEO, and the developer experience together.
- The focus is placed more on performance measures than the overall evaluation of the system.
- Lack of experimental studies with controlled conditions and identical application scenarios.
- No quantifiable benchmarks exist for evaluating the developer experience.

These gaps highlight the need for a comprehensive comparative approach, which will be provided by the current study.

3. Proposed Comparative Methodology

This section presents the methodology adopted to perform a comparative analysis between React-based and Next.js-based e-commerce architectures. The study is designed to ensure experimental fairness by maintaining identical application logic, user interface components, and backend configurations across both implementations. The methodology focuses on evaluating performance, search engine optimization (SEO), scalability, and developer experience using standardized metrics and controlled testing conditions.

3.1 System Design and Implementation

Two different instances of an e-commerce web application were created for experimentation purposes:

- React Application (CSR Architecture): Created using React (Vite), with CSR architecture where the web content is rendered by the browser once the JavaScript code is loaded by the client.
- Next.js Application (Hybrid Architecture): Using Next.js for creating the application with SSR and SSG capabilities, enabling pages to be pre-rendered with hydration to add intractability on client side.

Both applications have:

- Same UI components (products listing, cart, and checkout)
- Same business logic and process flow
- Same back-end API and database configurations
- Same testing data set

This way, all differences will only come out of the architectural approach used and not the actual application logic.

3.2 Experimental Environment

The experimental setup aimed at minimizing any form of bias in comparing the systems tested. The stack comprised JavaScript (ES6+), React and Next.js powered by a REST API that provided common endpoints along with a shared database. In order to eliminate the effect that any bias caused by the setup would have, we executed both frameworks in their production build on equivalent hosting platforms. Data was collected from an automated, browser-based test environment. Tests were done in two different network environments: cold start and warm load.

3.3 Evaluation Criteria for Performance Assessment

We examined how well both applications worked based on the criteria below:

- Largest Contentful Paint (LCP): measures the speed of the largest component loading
- First Contentful Paint (FCP): reveals the first rendering time
- Time to First Byte (TTFB): represents the responsiveness of the server
- Time to Interactive (TTI): measures when the page is interactive

We used automated profiling of browsers in order to obtain these values.

3.4 SEO Metrics

We measured the SEO efficiency based on the following metrics:

- Page Crawlability: Search engine ability to scan and index pages
- Meta Data Approach: Dynamic or pre-rendering meta tags
- HTML Content Deliverability: Server-side or server-rendered HTML content

- SEO Scores from Lighthouse Tool

Next.js benefits with respect to SSR/SSG, React Limitations due to CSR

3.5 Scalability Test

In order to test the scalability of our web page architecture, we have analyzed the way multiple users could interact with it through load testing, which was performed in correlation with user requests, observing the time taken for responding as well as overall performance of the system.

3.6 Developer Experience Evaluation

The developers' experience was evaluated both qualitatively and quantitatively in terms of:

- The difficulty of setting up the platform
- The maintainability of the code base
- The learning curve of the software
- The ratio between the native functionality and dependency on third-party tools
- The perceived performance speed in debugging and developing

3.7 Data Collection and Analysis

Every experiment was performed multiple times. Average accuracy results were calculated, comparative diagrams and tables were generated, and statistical coherence was verified throughout all test scenarios.

4. Results and Analysis

Two apps were created using React (Vite) and Next.js technologies. For an impartial comparison, the apps have the same user interface, business logic, backend APIs, and database. Features included in each app are products listing page, detail page, cart handling, and checkout page. The React app was built following the CSR approach, whereas the Next.js app was constructed following SSR and SSG techniques. Each app was deployed in the production environment and benchmarked.

4.1 Performance Benchmarking Results

The benchmarks include several web performance metrics such as LCP, FCP, and TTFB. We assessed both cold-start and warm-load situations.

Key Observations:

- Next.js rendered pages with less LCP and FCP values as a result of pre-rendered content.
- React app took more time during the first loading as all pages are rendered by the browser.
- During warm load, both apps showed the same performance levels because of caching.
- Higher TTFB times were recorded with Next.js app due to server-side rendering.

4.2 Comparison of SEO Effectiveness

SEO was measured based on automated analysis and manual review of the resulting HTML code.

Our findings:
Better SEO performance of Next.js due to server-side rendering and advanced metadata management. A crawlability problem of the React-based website as the content is generated on the fly after executing JavaScript.

Pre-rendering enabled by Next.js helped improve indexing and ranking by search engines.

4.3 Analysis of Developer Experience

Developer experience was evaluated in terms of workflow, configuration, and long-term maintenance aspects.

Main insights:

- More flexible approach to web architecture provided by React.
- Out-of-the-box solutions for various needs offered by Next.js.
- Easier configuration process for Next.js, which required additional libraries in the case of React.
- Developers could easily debug and scale both websites, though Next.js favored structured design.

4.4 Comparative Performance Tables and Graphs

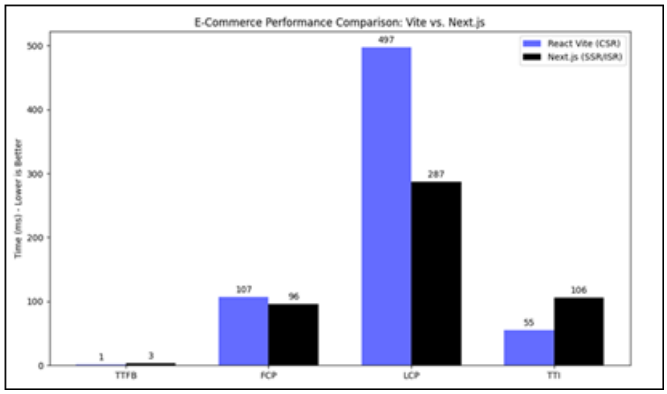

Figure 2: E-commerce performance comparison

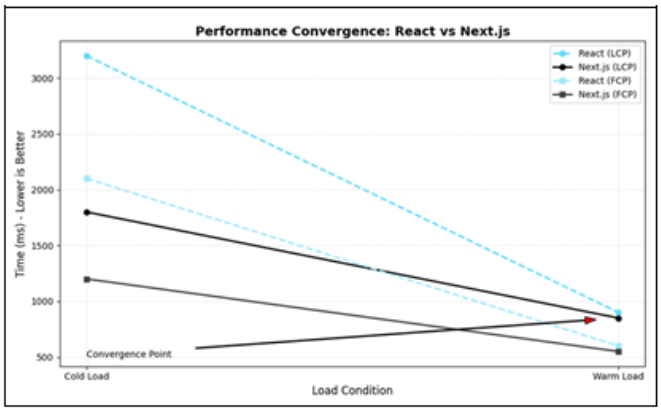

Figure 3: Performance Convergence

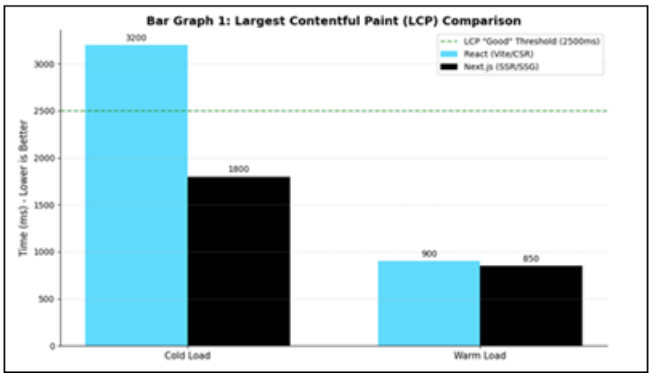

Figure 4: LCP Comparison

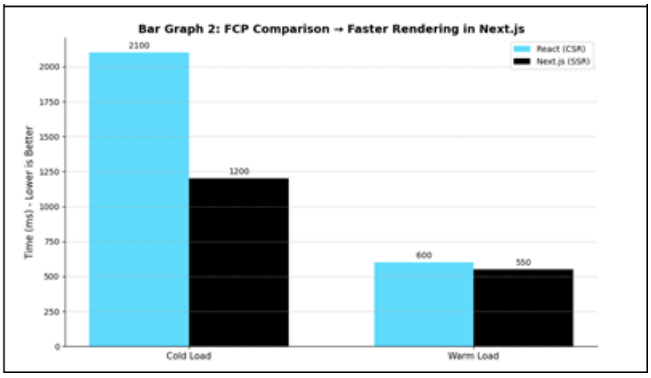

Figure 5: FCP Comparison

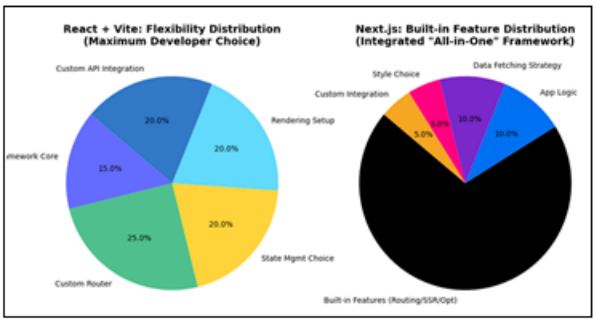

Figure 6: Maximum Developer Choice and Build-in feature Distribution

Table 1: Summarized comparative analysis

Metric	React (CSR)	Next.js (SSR/SSG)
LCP	Higher	Lower
FCP	Higher	Lower
TTFB	Lower	Slightly Higher
SEO	Moderate	High
DX	Flexible	Structured

4.5 Discussion of Results

With respect to results, it is evident that your selection of architecture influences the performance of the system as well as its effectiveness when used in a search engine. In terms of performance, Next.js comes out ahead of React right from the start due to the use of server-side rendering and static site generation, which significantly reduced dependence on client-side rendering. Consequently, Next.js is better suited to SEO websites, including online shops.

In contrast, React is more advantageous in circumstances requiring a large amount of customization and where SEO is not a key issue. In the end, your decision will be influenced by which aspects are more important to you: performance or flexibility.

5. Conclusions and Future Work

The research evaluated React and Next.js frameworks' performance, SEO optimization capabilities, scalability, and overall user experience during development. According to the results, both solutions enable developers to create scalable web applications with functional architecture. Still, the difference between client-side and server-side approaches affects their performance.

The React application demonstrated high levels of flexibility and modularity but showed poor performance regarding initial load time and SEO optimization because the framework requires more time to render content than Next.js. In contrast, the solution based on Next.js showed higher efficiency in terms of cold-start performance, providing faster loading time with lower values of LCP and FCP. Additionally, SEO optimization was better in the Next.js case since pre-rendering generates HTML content.

Regarding scalability, both solutions worked well with moderate traffic volume, although Next.js provided more stable performance because the approach enables developers to optimize the page rendering process. Moreover, React allows users to customize the development process freely, while Next.js focuses on providing a compact and optimized working environment.

Summing up, there is a trade-off between the two frameworks concerning flexibility and performance. React is an excellent choice for creating custom web pages with complex architecture, while Next.js prioritizes the performance of web applications.

There are several directions to explore without changing the goals and outcomes of this research. For example:

- Expanding the technology pool: Inclusion of technologies like Angular, Vue.js, and Svelte can provide additional perspectives on comparison.
- Investigating performance characteristics: Integration of metrics like Cumulative Layout Shift (CLS), and Interaction to Next Paint (INP) will allow a thorough examination.
- Conducting practical evaluations: Using real-life deployments under production loads and CDN conditions will increase applicability.
- Examining hybrid performance optimization techniques: Hybrid optimization methods can be applied and compared in terms of performance.
- Evaluating the security and costs of each approach: Security considerations and the infrastructure costs for each architectures should also be considered.
- Applying artificial intelligence optimization techniques: Performance optimization can be done with the help of artificial intelligence.

References

[1] A. Mesbah and A. van Deursen, "Migrating multi-page web applications to single-page AJAX interfaces," *Proc. European Conf. Software Maintenance*, 2007.
[2] J. Walke, "React: Declarative, efficient, and flexible JavaScript library," Facebook Engineering, 2013.
[3] J. Miller, "Rendering on the Web," Google Developers, 2019.
[4] H. Nguyen et al., "SEO challenges in JavaScript-based web applications," *IEEE Internet Computing*, vol. 24, no. 5, pp. 34–41, 2020.
[5] G. Rauch, "Next.js: The React framework for production," Vercel, 2024.
[6] Google, "Core Web Vitals," 2023.
[7] S. Souders, *High Performance Web Sites*, O'Reilly Media, 2007.
[8] A. Osmani, *Learning JavaScript Design Patterns*, O'Reilly Media, 2017.
[9] R. Patel et al., "Developer experience in modern front-end frameworks," *Journal of Web Engineering*, vol. 19, no. 3, 2021.
[10] Y. Kumar et al., "Hybrid rendering techniques in modern web frameworks," *International Journal of Web Engineering*, 2022.
[11] Y. Zhang et al., "Performance comparison of modern JavaScript frameworks," *IEEE Access*, vol. 8, pp. 12345–12356, 2020.
[12] M. Bostock et al., "Client-side vs server-side rendering: A comparative study," *IEEE Software*, vol. 36, no. 4, pp. 45–52, 2019.
[13] V. Patel, "Analyzing the Impact of Next.js on Site Performance and SEO," *Int. J. Computer Applications Technology and Research*, vol. 12, no. 10, pp. 24–27, 2023.
[14] R. Hanafi, A. Haq, and N. Agustin, "Comparison of Web Page Rendering Methods Based on Next.js Framework Using Page Loading Time Test," *TEKNIKA*, vol. 13, no. 1, pp. 102–108, 2024.
[15] S. Pati and Y. Zaki, "Evaluating the Efficacy of Next.js: A Comparative Analysis with React.js on Performance, SEO, and Global Network Equity," *arXiv preprint arXiv:2502.15707*, 2025.
[16] R. R. Kumar and A. Priya, "SEO Optimization in Web Development: How Next.js Helps," *IJRASET*, 2025.
[17] T. Brown and L. Johnson, "Building High-Performance Web Applications with Next.js," 2022.
[18] K. Chen, "Improving Front-end Performance through Modular Rendering and Adaptive Hydration in React Applications," *arXiv preprint*, 2025.
[19] Y. Zhang et al., "Performance Comparison of Modern JavaScript Frameworks," *IEEE Access*, vol. 8, 2020.
[20] S. Joshi and P. Raghuvanshi, "From React to Next.js: A Comparative Review of Performance, SEO, and Developer Experience," *IJSREM*, 2026.

Author Profile

Samyak Joshi is a postgraduate student (M.Tech.) in the Department of Computer Science and Engineering at Alpine Institute of Technology, Ujjain. His academic interests include Frontend Technology, Agentic AI. He is actively engaged in coursework, technical projects, and research-oriented learning as part of his postgraduate studies. His current work focuses on applying computational techniques to solve real-world problems, and he is keenly interested in emerging technologies and interdisciplinary research.

Pankaj Raghuvanshi is an Assistant Professor in the Department of Computer Science and Engineering at Alpine Institute of Technology. He is actively involved in teaching undergraduate engineering courses and contributing to academic research. His research interests primarily include computer networks, mobile ad hoc networks (MANETs), and emerging areas of data communication and analysis. He has authored research publications in various peer-reviewed journals.