A HYBRID RENDERING ARCHITECTURE FOR SEO-ORIENTED MULTILINGUAL E-COMMERCE IN EMERGING MARKETS

ODETA SHKRELI, JONIDA SHEHU

Department of Informatics, Faculty of Natural Sciences,

University of Tirana, Albania

e-mail: odeta.kromici@fshn.edu.al

Abstract

In emerging markets, e-commerce platforms are using Single Page Application (SPA) frontends to improve responsive user experiences. But when these architectures are combined with multilingual support and constrained infrastructure, they can conflict with search engine optimization (SEO) needs. This study presents a policy-driven hybrid architecture that balances the requirements of SEO and user experience through the adoption of a rendering policy that can switch between SSR, SSG/cache-assisted regeneration, and CSR based on route characteristics and runtime conditions. This is achieved based on the SEO value of the webpage, the requirements for personalization, and cacheability, and it can be adapted to changing conditions based on route characteristics, server load, and, where available, client hints or previously observed client conditions. As a result, the system for an eligible cacheable route can degrade from SSR to cached pre-rendered delivery. The architecture uses a schema-stable localization layer that externalizes translations into a dedicated store with keys on entity, field, and locale. The schema supports multilingual expansion without repeated schema changes. Composite indexing and batched retrieval provide predictable query performance, while deterministic fallback rules and caching reduce latency and ensure graceful behavior when translations are incomplete. We evaluate the proposed architecture against a CSR-only baseline and observe improvements in indexability-related and user-perceived performance metrics. The results suggest that policy-driven rendering combined with schema-stable localization can improve discoverability and performance while maintaining operational stability. The study therefore provides a practical architectural

approach for multilingual e-commerce platforms that operate under infrastructure and resource constraints.

Key words**:** *E-commerce, Hybrid Architecture, SEO, Emerging Markets, Multilingual Systems.*

Përmbledhje

Në tregjet në zhvillim, platformat e tregtisë elektronike po përdorin ndërfaqe Single Page Application (SPA) për të përmirësuar përvojën të përdoruesit. Megjithatë, kur këto arkitektura kombinojnë mbështetje shumëgjuhëshe dhe infrastrukturë të kufizuar, ato mund të bien ndesh me kërkesat e motorrëve të kërkimit SEO. Studimi paraqet një arkitekturë hibride të drejtuar nga politika, që balancon kërkesat e SEO-s dhe përvojën e përdoruesit përmes një politike renderimi, që kalon ndërmjet SSR, SSG/cache-assisted regeneration dhe CSR sipas karakteristikave të rrugëzimeve dhe kohës së ekzekutimit. Kjo arrihet mbi bazën e vlerës SEO të faqes, kërkesave për personalizim dhe mundësisë së ruajtjes në cache, dhe përshtatet ndaj kushteve në ndryshim, bazuar në karakteristikat e rrugëzimeve, ngarkesa e serverit dhe, kur disponohen, sinjalet nga klienti ose kushtet e vëzhguara. Si rezultat, për një rrugëzim të përshtatshëm për cache, sistemi mund të degradojë nga SSR në shpërndarje të parapërgatitur dhe të ruajtur në cache. Arkitektura përdor një shtresë lokalizimi me skemë të qëndrueshme, që i zhvendos përkthimet në një strukturë të veçantë, me çelësa sipas entitetit, fushës dhe lokalitetit gjuhësor. Skema mbështet zgjerimin shumëgjuhësh pa ndryshime strukturore. Vlerësimi tregon përmirësime në indeksueshmëri dhe performancën e perceptuar. Rezultatet sugjerojnë se renderimi i drejtuar nga politika, i kombinuar me lokalizimin me skemë të qëndrueshme, mund të përmirësojë zbulueshmërinë dhe performancën, duke ruajtur stabilitetin operacional. Në këtë mënyrë, studimi ofron një qasje praktike arkitekturore për platformat shumëgjuhëshe të tregtisë elektronike që veprojnë në kushte kufizimesh infrastrukturore dhe burimore.

Fjalë kyçe: *Tregti elektronike, arkitekturë hibride, SEO, ekonomi në zhvillim, sisteme shumëgjuhëshe.*

Introduction

The global e-commerce sector has grown rapidly in recent years. The most recent UNCTAD report estimated the value of worldwide e-commerce at roughly 27 trillion USD for 2022 (UNCTAD, 2024). E-commerce platforms

are becoming widely used due to their ability to connect the business strategy with the technological infrastructure, and societal and regulatory concerns (Laudon & Traver, 2023). Because of this, over the past years, there have been great advancements in the technology behind these platforms. Nowadays, e-commerce systems have become complex programs that enable consumers to interact with the platform and businesses to manage their sales and other aspects of their operations. Digital commerce has become firmly established in developed economies. In Albania, an emerging economy struggling with persistent structural and digital market limitations, adoption remains infrequent and inconsistent. This limited progress is primarily attributable to several factors, including disparities in digital infrastructure, payment systems, and consumer confidence.

The Institute of Statistics of Albania reported in 2023 that about 83.1% of Albanians aged 16 to 74 used the internet, INSTAT (2023). Although e-commerce platforms have gained traction among certain enterprises, numerous small and medium-sized businesses continue to leverage social media for online transactions. Conversely, these platforms, despite their practicality, are limited by their inherent functionalities and exhibit restricted visibility in online search results, thus representing incomplete solutions. As a result, local e-commerce systems are necessitated to operate within environments marked by infrastructural variability, leverage multilingual support, and adapt to the dynamic nature of digital ecosystems.

Modern e-commerce platforms increasingly use Single Page Application (SPA) architectures, often developed with frameworks like React, Angular, or Vue (Banks & Porcello, 2020). These architectural approaches facilitate adaptive interfaces and substantial client-side interactivity. Consequently, pages can undergo dynamic alterations without necessitating full reloads, thereby improving both user experience and application performance.

Conversely, single-page application (SPA) architectures present challenges for search engine optimization (SEO) strategies. In numerous SPA implementations, the server initially transmits a basic HTML structure, with the page's content subsequently generated through JavaScript execution within the user's browser (Mesbah, van Deursen, & Lenselink, 2012). Although modern search engines can process JavaScript with additional resources, this may impede indexing processes or reduce crawling efficiency (Google Search

Central, 2026). This is especially the case for e-commerce websites, where product pages should be indexable by search engines to allow for organic search traffic. Prior research shows that, for retail websites, the majority of search-engine traffic originates from organic rather than sponsored links (Baye et al., 2016). (Gallino, Karacaoglu, & Moreno, 2023), (Basalla et al., 2022) have also shown that delays in website speed can negatively affect customer behavior and conversion-related outcomes. Server-side rendering (SSR), static site generation (SSG), and hybrid rendering strategies are some of the technical solutions that have been attempted to resolve this challenge.

The main contribution of this study is the design of a hybrid rendering architecture and its evaluation through comparative analysis in a production multilingual e-commerce environment. The architecture introduces a policy-based mechanism that selects rendering strategies according to page characteristics and runtime conditions, supported by a schema-stable multilingual data model. The study further evaluates the system using performance, SEO indexability, and operational stability metrics.

Literature review

1. SEO challenges in Single-Page Applications

Search engine optimization is difficult with single-page applications. The main issue, as explained by the authors in their paper (Mesbah, van Deursen, & Lenselink, 2012), is the timing of content generation. SPAs generate content at runtime through JavaScript execution, whereas traditional web crawlers expect content in the initial HTML response. Despite the fact that Google has enhanced its capacity to execute JavaScript, there are still constraints in terms of crawl budget efficacy, allocation of resources, and execution timeouts (Google Search Central, 2026).

Various solutions have been developed to mitigate this dilemma:

- Server-Side Rendering (SSR): Frameworks like Next.js (for React) enable server-side rendering, generating complete HTML on the server before delivery to the browser (web.dev, 2026), but SSR increases server resource consumption and can delay Time to Interactive (TTI). Upgraded versions of these frameworks have streamlined the SSR process by utilizing Server Actions and sophisticated Render Patterns, enabling smoother data modifications without adding to the performance cost, which is usually

associated with traditional server-side rendering (Vercel Inc., 2024).

- Static Site Generation (SSG): This approach generates static HTML at build time, offering excellent performance but limited suitability for e-commerce, where prices and inventory change frequently (web.dev, 2026).

- Hybrid Approaches: Combining SSR for critical pages with client-side rendering for interactive components has emerged as a promising middle ground (web.dev, 2026). However, limited research provides validated implementations specifically for e-commerce platforms in emerging markets.

2. Web performance and core web vitals

The conversion rates of an online store are directly influenced by performance. Souders (2007) found that the frontend accounts for eighty percent of the total amount of time that end-users spend responding to applications and that images, CSS, and JavaScript are, in fact, the key bottlenecks. This idea was elaborated further by Grigorik (2013), while he incorporated issues related to network performance, with an emphasis on the role that Content Delivery Networks (CDNs) and HTTP/2 play. Three critical criteria for user-perceived performance are defined by Google's Core Web Vitals, as follows: loading performance is evaluated by Largest Contentful Paint (LCP) metric; responsiveness to user interactions is evaluated with the Interaction to Next Paint (INP) metric; and visual stability is evaluated with the Cumulative Layout Shift (CLS) metric (Google Search Central, 2025). These measurements are relevant to SEO because they form part of Google’s broader page experience signals (Google Search Central, 2025). According to Souders (2007), caching methods continue to be the most successful strategy for speed improvement. These strategies include browser caching, server-side caching, and caching for database queries.

Related work

Research relevant to this study spans multiple areas, including SEO compliance in JavaScript frameworks, server- and static-rendering strategies for web applications, multilingual website usability, and latency in e-commerce systems.

Ágh et al. (2024) discuss that successful search engine optimization within contemporary JavaScript frameworks necessitates more than just the

implementation of rendering strategies like server-side rendering (SSR), static site generation (SSG), and hybrid rendering; it also requires the comprehensive incorporation of metadata management, internal linking structures, image optimization techniques, the inclusion of indexing support files, and automated SEO testing protocols throughout the deployment process. Although (Vepsäläinen et al., 2023) do not focus specifically on SEO or e-commerce, their discussion of lighter web architectures is relevant to this study. Our hybrid rendering model reflects a similar design principle by reducing unnecessary client-side work and using server- or cache-assisted rendering for routes where content visibility, stability, and crawlability are important. While Vīksna et al. (2022) focus on assessing website multilingualism rather than rendering strategies, their work supports our study by showing that multilingual web presence can be systematically evaluated and remains an important challenge in digital environments. (Basalla et al., 2021) show that latency significantly affects how well e-commerce platforms perform. This impact goes beyond immediate user interactions, also influencing customer behaviour over time.

Existing studies usually address one of three concerns in isolation: rendering for crawlability, frontend performance optimization, or multilingual content handling. Fewer studies have examined a unified architecture that simultaneously optimizes rendering strategy, localization storage, and operational stability within a live e-commerce setting. This deficiency is especially pronounced in developing markets, where platforms must function amidst fluctuating infrastructural conditions while accommodating multilingual content and preserving search visibility. The architecture presented in this research addresses this issue through an integrated hybrid rendering approach, which was assessed on a production platform.

Proposed hybrid architecture

The proposed architecture follows a three-tier structure separating presentation, business logic, and data access concerns. Figure 1 illustrates the high-level architecture.

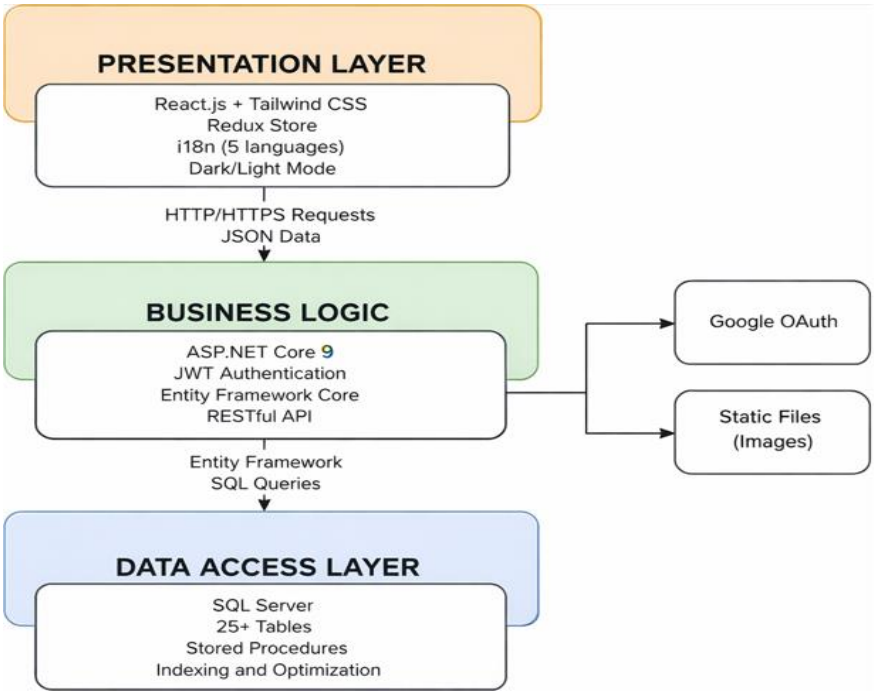

Figure 1. Three-tier Hybrid Architecture of the proposed platform

React was used because its component model allowed the same route structure to support multiple rendering modes while preserving content equivalence across CSR, SSR, and cache-assisted delivery paths. Centralized state management helped reduce the risk that observed differences between rendering modes would arise from divergent client-side state transitions rather than from the rendering policy itself. Tailwind CSS is used for styling because it supports responsive design and consistent user interface development.

The backend is implemented using ASP.NET Core 9, selected for its high performance and built-in security features. Data persistence is handled by SQL Server with Entity Framework Core as the object-relational mapping framework. Authentication uses JWT-based tokens and OAuth-based external authorization flows. React-i18next is used for interface localization, while dynamic multilingual content is handled through the backend translation layer.

1. Hybrid rendering strategy

The core innovation of the proposed architecture is selective rendering based on route characteristics, personalization needs, and runtime conditions, with limited crawler-specific handling where necessary. This strategy resolves the SPA-SEO dilemma by treating different pages differently:

- *Server-side rendered pages (for SEO-critical content):* Homepage, Category listing pages, Individual product pages, Collection landing pages.
- *Client-side rendered components (for interactive features):* Shopping cart management, User dashboard and profile, Wishlist functionality, Checkout process, Product filtering and sorting, Theme switching.

While this separation ensures that SEO-critical content is delivered in an indexable form, the architecture does not rely primarily on crawler-targeted dynamic rendering. Instead, it primarily uses route- and policy-based rendering selection, with crawler-specific pre-rendering retained only as a limited compatibility mechanism where necessary. Regular users therefore receive responses according to the rendering mode selected by the policy engine, including SSR, SSG/cache-assisted regeneration, or CSR depending on route characteristics and runtime conditions.

1.1. Rendering policy engine

The proposed platform implements a rendering policy engine that determines the rendering mode for each request, hence eliminating a static, hardcoded SSR/CSR route division. Maintaining interactive responsiveness in stateful or highly customized workflows is crucial, as is the provision of indexable HTML and consistent metadata for content critical to search engine optimization. This method aligns with search engine guidelines, which highlight the challenges of indexing and emphasize the need for careful management of JavaScript-based rendering to ensure content is easily found (Google Search Central, 2026).

Policy inputs. The decision is based on these factors: (i) SEO value, which represents the anticipated significance of organic discovery for the page (e.g., homepage, category listings, product pages), (ii) personalization requirements, which indicate if the response depends on user-specific context, and (iii) cache ability, which reflects the volatility of the content and its suitability for

caching. The policy also considers runtime limitations, including client-side factors like expected network round-trip time and device type, if available. Previous research on Ajax crawling and current search engine guidelines suggest that rendering choices can affect how easily content is found (Mesbah, van Deursen, & Lenselink, 2012; Google Search Central, 2026).

Policy outputs. The engine returns a rendering mode and an associated caching strategy. SEO-critical pages' default to SSR or SSG/cache-assisted regeneration to ensure that the initial response includes complete, indexable HTML and metadata. Where content is sufficiently cacheable, SSG/cache-assisted regeneration is preferred because it preserves many benefits of static delivery while allowing controlled regeneration, thereby reducing origin load and supporting large catalogs (Vercel Inc., 2025). Under elevated server load, the policy enables controlled degradation from SSR to SSG/cache-assisted regeneration for eligible routes, stabilizing tail latency without sacrificing indexability.

Rule examples. Table 1 summarizes representative rendering policy rules used by the system. Pages with high SEO value and high cacheability are usually prioritized for SSG/cache-assisted regeneration (or SSR when freshness constraints require it), whereas pages with high personalization favour CSR or SSR with limited caching. The rendering policy maintains content equivalence across modes for the same URL and locale.

This means that, instead of replacing page semantics, SSR or cache-assisted regeneration delivers the canonical content and metadata, and CSR progressively enhances interactivity after hydration. This avoids broad reliance on user-agent–based dynamic rendering, which is described in search documentation as a workaround rather than a recommended long-term strategy (Google Search Central, 2025).

Table 1. Representative Rendering Policy Rules

Page Characteristics	**Selected Rendering Mode**	**Rationale**
High SEO value, high cacheability, low personalization	SSG/cache-assisted regeneration	Maximizes caching efficiency while preserving indexable HTML
High SEO value, low cacheability (frequent updates)	SSR	Ensures fresh content and accurate metadata
Medium SEO value, high interactivity, authenticated user	CSR	Optimizes interaction and isolates user-specific state
High personalization on the SEO-relevant route	SSR with limited caching	Maintains indexable HTML while supporting personalized elements
High server load and cacheable page	Shift from SSR to SSG/cache-assisted regeneration	Stabilizes p95 and p99 latency through caching
Constrained client device or high network latency	SSR / cache-assisted regeneration with reduced JavaScript	Improves first render on slower devices

Implementation considerations. To avert N+1 retrieval patterns in server-rendered responses, localization and catalogue data are fetched in batches, while translations are retrieved through a cached localization layer. The policy engine records the specific mode and the triggering signals (e.g., cacheability flags, load state) as operational telemetry, enabling iterative adjustment of thresholds based on observed delay, failure rates, and exception patterns.

Multilingual database design

Prior multilingual web systems often rely on language-specific columns, duplicated content structures, or CMS-level translation layers, which can become difficult to scale as supported locales expand. Research on internationalized data design suggests the importance of decoupling language variants from core entity schemas, particularly in content-rich systems with heterogeneous translation completeness. Therefore, the flexible multilingual implementation, which avoids the need to change the schema when adding new languages, represents a significant architectural improvement. Table 2 shows how a general translation table is used, instead of creating separate columns or tables for each language for every entity.

Table 2. Translations Table Schema

Column	Type	Description	Example
id	INT	Primary key	1
entity_type	NVARCHAR(50)	Entity being translated	'product'
entity_id	INT	ID of the entity	42
field_name	NVARCHAR(50)	Field being translated	'name'
language_code	NVARCHAR(5)	Target language	'sq'
value	NVARCHAR(MAX)	Translated text	'Këpucë lëkure'

A hybrid strategy that combines file-based interface localization with database-driven translation of dynamic content is used to achieve multilingual support. When a key is missing, the implementation goes back to English. Static user interface elements, such as buttons, labels, and navigation text, are

preserved in JSON translation resources and loaded dynamically according to the current locale. The implementation supports five different languages. Dynamic content, on the other hand, is kept in a generic translations table in accordance with an entity–field–language–value structure. Dynamic content includes product names, product descriptions, category titles, and other entity features. When localized content is requested, the API retrieves the corresponding translation (e.g., through an indexed lookup or join) and applies a default-language fallback when a translation is unavailable. The initial language selection can be inferred from the preferences of the browser on the first visit, but users have the ability to override it by using a language selector. The locale that is selected is saved by using a cookie or local storage, and optionally by using a user profile setting for authenticated users. This ensures that the language remains in a consistent manner throughout each visit. This design provides advantages across several dimensions, including adaptability, scope, operational efficiency, durability, and ease of upkeep. The design facilitates the integration of new languages without necessitating schema alterations; furthermore, translations can be linked to any specified entity attribute. Composite indexing mechanisms enable efficient data retrieval, while a fallback mechanism to a default language enhances system reliability in the absence of specific translations also, a unified translation table simplifies backup and migration.

Analysis & discussion

The system was deployed on a medium-scale commercial e-commerce platform serving the Albanian market. The platform supports approximately several thousand products and multiple languages. The proposed hybrid rendering architecture was evaluated using a controlled comparative deployment in a production e-commerce environment. Two configurations were tested sequentially on the same platform, using identical infrastructure, URL structure, and content source:

- CSR-only baseline: all pages rendered via client-side rendering.
- Hybrid policy configuration: rendering mode dynamically selected (SSR, SSG/cache-assisted regeneration, and CSR) according to SEO relevance, cacheability, personalization needs, and server load.

For comparability, both phases used the same sitemap and the same 100 URLs (10 collection, 10 category, and 80 product pages), with measurements collected over two 14-day windows under comparable traffic and without structural catalog changes. Evaluation was conducted across three dimensions including user performance, search-engine indexability and render integrity, and operational stability. Given the non-normal distribution of performance data, results are reported using medians and p75 values, consistent with Google Core Web Vitals methodology.

1. User performance evaluation

Performance was measured using synthetic testing (Google Lighthouse) and Real User Monitoring (RUM) via the Web Vitals library. The following metrics were evaluated: Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS), Interaction to Next Paint (INP), and Time to First Byte (TTFB).

1.1. Synthetic (lab) results

Tests were executed under mobile emulation (4G, mid-tier device), consistent with standard Lighthouse configuration.

Table 3. Synthetic Performance Comparison (Median Values)

Metric	CSR-only	Hybrid Policy	Effect
LCP (ms)	3210	1950	39% improvement
CLS	0.17	0.05	Reduced layout instability
INP (ms)	310	205	34% improvement
TTFB (ms)	740	480	35% reduction

The largest improvements appear in LCP and CLS, while INP also improves due to reduced client-side processing.

1.2. Real User Monitoring (RUM)

RUM data was collected from approximately 1000 user sessions during each evaluation window. Performance metrics were recorded using the Web Vitals

library embedded in the client application. Results are reported using p75 values, consistent with Core Web Vitals evaluation standards.

Table 4. Real User Performance (p75)

Metric	CSR-only	Hybrid Policy	Improvement
LCP	2980 ms	2140 ms	28%
CLS	0.15	0.06	Reduced instability
INP	285 ms	215 ms	25%
TTFB	710 ms	520 ms	27%

In real-user environments, improvements are generally smaller than lab results due to device variability and network conditions. In the observation window, the hybrid configuration reduced aggregate p75 LCP to 2140 ms, with the largest gains observed on SEO-critical routes.

2. SEO effectiveness and indexability

Search performance was evaluated using Google Search Console Page Indexing reports, URL Inspection tool validation, HTML structure validation via headless crawler

2.1. Indexing coverage

Indexing status was assessed 30 days after sitemap submission in each configuration.

Table 5. Indexing Coverage

Metric	CSR-only	Hybrid Policy
URLs Submitted	100	100
Successfully Indexed	66	92
Indexing Rate	66%	92%
Average Indexing Latency	17 days	7 days

The hybrid configuration improved indexing consistency and reduced average indexing latency.

2.2. HTML render integrity

To verify crawler-visible content, HTML was extracted using a headless browser and checked for the title, canonical link, meta description, and main product or category content container.

Table 6. HTML Metadata and Content Availability

Validation Metric	CSR-only	Hybrid Policy
Title present in initial HTML	76%	100%
Canonical tag present	82%	100%
Primary content visible without JS	68%	97%

CSR-only implementations often rely on JavaScript to render page content, while hybrid rendering delivers metadata and primary content directly in the initial HTML response.

3. Operational atability under load

To evaluate infrastructure resilience, controlled load simulations were conducted using concurrent traffic scenarios. Two modes were compared such as Static SSR (no adaptive policy) and Hybrid adaptive policy (SSR with degradation to SSG/cache-assisted regeneration for cacheable content). Some metrics analysed were p95 latency, p99 latency, error rate, and cache hit ratio.

Table 7. Load Stability Comparison

Metric	Static SSR	Hybrid Policy
p95 Latency	1480 ms	890 ms
p99 Latency	2840 ms	1320 ms
Error Rate	3.1%	0.8%
Cache Hit Ratio	34%	79%

Under load, the adaptive policy reduced tail latency by dynamically shifting eligible pages to SSG/cache-assisted regeneration, increasing cache utilization while preserving indexable HTML responses.

Conclusions

The evaluation demonstrates that the hybrid rendering architecture produces consistent improvements across all measured dimensions. Regarding user performance, LCP and CLS demonstrated the most significant gains, confirming that pre-rendered HTML substantially reduces content instability and delays caused by JavaScript hydration.

Improvements in INP were more moderate but still meaningful, suggesting that responsiveness benefits primarily from reduced main-thread blocking during initial page load. Indexing performance improved both in coverage and latency. We also found that reliance on crawler-side JavaScript execution is reduced by metadata and primary content within the initial HTML response. This finding supports current search-engine guidance for server-rendered content for SEO-critical pages. Operational testing also indicated that adaptive rendering policies improve tail latency (p95 and p99) under load. Static SSR configurations demonstrated performance degradation during traffic spikes, whereas the hybrid policy preserved stability by increasing cache utilization through SSG/cache-assisted regeneration, where applicable.

In conclusion, we found that the results suggest that the proposed policy-driven rendering and caching architecture can balance SEO visibility, user experience, and infrastructure efficiency within emerging-market e-commerce deployments.

Limitations

Several limitations must be acknowledged.

- Results are based on a single production deployment of one e-commerce platform and may not generalize to platforms with substantially different infrastructure or traffic profiles.

- Indexing latency is partially influenced by external crawler scheduling decisions beyond system control. Because configurations were tested

sequentially rather than simultaneously, unobserved temporal factors may have influenced results.

- Real-user metrics depend on device capability distribution, which may vary over time.
- This study evaluates indexing coverage and latency, but it doesn't directly measure the long-term ranking position or the impact on organic revenue.

Despite these limitations, the consistent improvements seen in performance, indexing, and stability suggest that the hybrid rendering method is both technically sound and practically useful.

References

Ágh, J., Makarova, I., & Dakić, P. (2024). Enhancing SEO compliance of JavaScript web frameworks through automated testing inside CI/CD pipelines. In Software Quality: Analysis, Monitoring, Improvement, and Applications (SQAMIA 2024 Workshop Proceedings).

Banks, A., & Porcello, E. (2020). Learning React: Modern Patterns for Developing React Apps. O'Reilly Media.

Basalla, M., Schneider, J., Luksik, M., Jaakonmäki, R., & vom Brocke, J. (2021). On latency of e-commerce platforms. Journal of Organizational Computing and Electronic Commerce, 31(1), 1–17. DOI: 10.1080/10919392.2021.1882240.

Baye, M. R., De los Santos, B., & Wildenbeest, M. R. (2016). Search engine optimization: What drives organic traffic to retail sites? Journal of Economics & Management Strategy, 25(1), 6-31.

Gallino, S., Karacaoglu, N., & Moreno, A. (2023). Need for speed: The impact of in-process delays on customer behavior in online retail. Operations Research.

https://doi.org/10.1287/opre.2022.2262

Google Search Central. (2025). Understanding Core Web Vitals and Google Search results. https://developers.google.com/search/docs/appearance/core-web-vitals

Google Search Central. (2025). Dynamic rendering as a workaround.

https://developers.google.com/search/docs/crawling-indexing/javascript/dynamic-rendering

Google Search Central. (2026). Understand JavaScript SEO basics.

https://developers.google.com/search/docs/crawling-indexing/javascript/javascript-seo-basics

Grigorik, I. (2013). High Performance Browser Networking: What every web developer should know about networking and web performance. O'Reilly Media.

INSTAT. (2023). Survey on Information and Communication Technology (ICT) usage in Households and by Individuals in 2023. Albanian Institute of Statistics.

Laudon, K. C., & Traver, C. G. (2023). E-commerce 2023-2024: Business, Technology and Society. Pearson Education.

Mesbah, A., van Deursen, A., & Lenselink, S. (2012). Crawling Ajax-based web applications through dynamic analysis of user interface state changes. ACM Transactions on the Web, 6(1), Article 3.

Souders, S. (2007). High Performance Web Sites: Essential Knowledge for Front-End Engineers. O'Reilly Media.

UNCTAD. (2024). Digital Economy Report 2024: Shaping an environmentally sustainable and inclusive digital future. United Nations Conference on Trade and Development.

Vepsäläinen, J., Hellas, A., & Vuorimaa, P. (2023). The rise of disappearing frameworks in web development. In Web Engineering: 23rd International Conference, ICWE 2023 (pp. 319–326). Springer. DOI: 10.1007/978-3-031-34444-2_23

Vercel Inc. (2024). Server Actions and Mutations. Next.js Documentation.

https://nextjs.org/docs/14/app/building-your-application/data-fetching/server-actions-and-mutations

Vercel Inc. (2025). Incremental Static Regeneration (ISR). Next.js Documentation. https://nextjs.org/docs/pages/guides/incremental-static-regeneration

web.dev. (2026). Rendering on the Web. https://web.dev/articles/rendering-on-the-web

Vīksna, R., Skadiņa, I., Skadiņš, R., Vasiļjevs, A., & Rozis, R. (2022). Assessing the multilinguality of publicly accessible websites. In Proceedings of the Thirteenth Language Resources and Evaluation Conference (LREC 2022) (pp. 2108–2116). European Language Resources Association