TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

Analisis Komparatif Kinerja Next.js, Nuxt.js, dan Remix.js dalam Implementasi Server Side Rendering Website Berita

Richie Reuben Hermanto, Mychael Maoeretz Engel*

Fakultas Teknologi Informasi, Program Studi Informatika, Universitas Ciputra, Surabaya, Indonesia
Email: [1]richiereubenh@gmail.com, [2,*]mychael.engel@ciputra.ac.id
Email Penulis Korespondensi: mychael.engel@ciputra.ac.id

Abstrak–Perkembangan pesat media berita online menuntut website memiliki performa tinggi agar mampu menyajikan informasi secara cepat, stabil, dan tetap optimal di mesin pencari. Penelitian ini bertujuan menganalisis serta membandingkan kinerja *framework* Next.js, Nuxt.js, dan Remix.js dalam implementasi *Server Side Rendering* (SSR) pada website berita. Metode yang digunakan adalah eksperimen komparatif dengan membangun tiga prototipe website berita identik menggunakan ketiga *framework*, kemudian diuji dengan Google Lighthouse untuk mengukur peforma *rendering* dan Apache JMeter untuk menilai ketahanan terhadap beban pengguna. Pengujian beban dilakukan dengan tiga skenario berbeda menggunakan 250, 500, dan 1000 *virtual users/threads* dengan *ramp-up period* 5 detik, durasi uji 300 detik, serta *loop count infinite* agar setiap pengguna terus mengulangi *request* hingga pengujian berakhir. Hasil pengujian Google Lighthouse menunjukkan bahwa Remix.js unggul pada metrik *First Contentful Paint* sebesar 2,14 detik, *Largest Contentful Paint* 2,9 detik, dan *Speed Indeks* sebesar 2,14 detik, sehingga headline dan gambar berita dapat ditampilkan lebih cepat serta meningkatkan persepsi kecepatan dan kenyamanan pengguna. Pada pengujian JMeter, Nuxt.js menampilkan kinerja terbaik dengan *response time* terendah, *throughput* tertinggi, dan *error rate* yang lebih stabil dibandingkan *framework* lainnya, menunjukkan ketahanan lebih baik menghadapi lonjakan trafik besar. Kesimpulan penelitian ini adalah Remix.js lebih optimal untuk meningkatkan pengalaman pengguna melalui kecepatan akses konten, sedangkan Nuxt.js lebih unggul dalam aspek skalabilitas dan reliabilitas ketika terjadi lonjakan trafik. Implikasi dari temuan ini adalah pemilihan *framework* SSR untuk website berita sebaiknya disesuaikan dengan prioritas utama, apakah menekankan pada kecepatan akses atau pada ketahanan sistem dalam kondisi beban tinggi.

Kata Kunci: *Server Side Rendering*; Website Berita; Next.js; Nuxt.js; Remix.js

Abstract–The rapid development of online news media requires websites to have high performance in order to deliver information quickly, stably, and optimally on search engines. The objective of this study is to analyze and compare the performance of the Next.js, Nuxt.js, and Remix.js frameworks in implementing Server Side Rendering (SSR) on news websites. The method used is a comparative experiment by building three identical news website prototypes using the three frameworks, then testing them with Google Lighthouse to measure rendering peformance and Apache JMeter to assess resilience to user load. The load testing was conducted in three scenarios using 250, 500, and 1000 virtual users/threads with a 5 second ramp-up period, a 300 second test duration, and an infinite loop count so that each virtual user continuously repeated requests until the test ended. Google Lighthouse test results show that Remix.js excels in the First Contentful Paint metric at 2.14 seconds, Largest Contentful Paint at 2.9 seconds, and Speed Index at 2.14 seconds, allowing news headlines and images to be displayed faster and improving user perception of speed and convenience. In JMeter testing, Nuxt.js showed the best performance with the lowest response time, highest throughput, and more stable error rate compared to other frameworks, demonstrating better resilience in the face of large traffic spikes. The conclusion of this study is that Remix.js is more optimal for improving user experience through content access speed, while Nuxt.js is superior in terms of scalability and reliability when traffic spikes occur. The implication of these findings is that the selection of an SSR framework for news websites should be tailored to the main priority, whether it is emphasizing access speed or system resilience under high load conditions.

Keywords: Server Side Rendering; News Website; Next.js; Nuxt.js; Remix.js

1. PENDAHULUAN

Di era masifnya perkembangan teknologi saat ini, media berita telah mengalami transformasi yang sangat signifikan. Media cetak yang sebelumnya mendominasi kini mulai tergeser, dengan mayoritas masyarakat beralih pada website berita online sebagai sumber informasi utama. Peningkatan konsumsi berita online berimplikasi pada bertambahnya jumlah media digital (Muhammad A Ayub, 2021). Pada tahun 2022, jumlah media digital di Indonesia mencapai 47.000 (Aryo Subarkah Eddyono, 2022). Fenomena tersebut menuntut website berita untuk memiliki performa yang tinggi agar dapat memberikan pengalaman pengguna yang optimal. Selain performa, penerapan *Search Engine Optimization* (SEO) juga memegang peran penting untuk meningkatkan *traffic* website berita yang pada akhirnya mendorong pertumbuhan pendapatan melalui iklan (Mulyadi & Suharman, 2024).

Terdapat beberapa teknik *rendering* yang dapat digunakan untuk mengoptimasi SEO pada website, di antaranya *Server Side Rendering* (SSR) dan *Static Site Generation* (SSG) (Maulana & Susetyo, 2025). Akan tetapi, penelitian sebelumnya menunjukkan bahwa SSG memiliki keterbatasan. SSG menawarkan kecepatan tinggi dan optimal untuk SEO, tetapi tidak cocok untuk website berita dengan konten yang berubah secara dinamis karena memerlukan *rebuild* untuk memperbarui data (Hanafi et al., 2024). Sehingga, pendekatan SSR lebih sesuai untuk diimplementasikan pada konteks website berita yang memiliki perubahan konten yang dinamis.

Javascript menjadi teknologi yang paling populer digunakan dalam pengembangan website. Saat ini, sekitar 97% website menggunakan Javascript, meningkat 88% dari satu dekade yang lalu (Ollila et al., 2022). Hal ini dikarenakan Javascript memiliki karakter yang mampu membangun antarmuka website yang interaktif dan mendukung proses pengembangan yang cepat (M Syahputra, 2025). Next.js dan Nuxt.js merupakan 2 *framework* Javascript yang

TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

mendukung teknik SSR dan paling populer digunakan didunia industri. Arsitektur Nuxt.js juga mendukung pemisahan kode secara otomatis dan proses hidrasi yang efisien, memastikan HTML yang di *render* di sisi server dapat bertransisi dengan mulus ke interaktivitas di sisi klien, sehingga menjadikan pendekatan ini sangat ideal untuk aplikasi yang membutuhkan konten dinamis sekaligus tetap mengedepankan visibilitas di mesin pencari (Rao, 2025). Next.js mengimplementasikan SSR secara kuat melalui fungsi *getServerSideProps*, yang memungkinkan pengambilan data setiap kali permintaan dilakukan. Hal ini menjadikan Next.js sangat cocok untuk aplikasi yang menyajikan data yang sering diperbarui pengguna (Rao, 2025). Sedangkan Remix.js adalah *framework* React yang memungkinkan pengembang mendefinisikan fungsi *loader* untuk mengambil data dari server, fungsi *action* untuk memproses perubahan data, serta komponen React untuk merender antarmuka pengguna dalam setiap *route* (Landgraf, 2023).

Penelitian yang dilakukan oleh Roy Hanafi, Abd Haq, dan Ninik Agustin melakukan evaluasi terhadap tiga metode *rendering* meliputi, *Client-Side Rendering* (CSR), *Server Side Rendering* (SSR), dan *Static Site Generation* (SSG) pada *framework* Next.js untuk website "Filmku". Pengujian dilakukan pada tiga halaman meliputi, halaman autentikasi, halaman profil, dan homepage di tiga browser berbeda. Hasil menunjukkan SSG tercepat pada *initial load* di semua halaman, CSR lambat pada *initial load* tetapi unggul pada *subsequent loads*, dan SSR memberikan kestabilan performa meski membebani server dengan *resource* lebih besar (Hanafi et al., 2024).

Syuhda Fakhrunnisa dan Rahadian Kurniawan melakukan studi komparatif kinerja tiga *framework frontend* ReactJS, VueJS, dan Blade Templating Engine pada modul terintegrasi platform *talent pool*. Tiga modul yang diuji adalah Jobseeker dikembangkan menggunakan Blade, Modul Administrasi Keuangan dikembangkan menggunakan ReactJS, dan Platform Ujian Pelatihan Coding dikembangkan menggunakan VueJS. Hasil penelitian menunjukkan bahwa VueJS unggul dalam responsivitas, interaktivitas, dan kinerja *real time*, Blade Templating Engine lebih optimal untuk *rendering* awal konten statis, sedangkan ReactJS lebih sesuai untuk aplikasi enterprise dengan komponen dinamis yang kompleks. Temuan ini menegaskan bahwa pemilihan teknologi *frontend* perlu disesuaikan dengan karakteristik aplikasi, konteks penggunaan, serta kebutuhan pengguna akhir. (Fakhrunnisa & Kurniawan, 2024).

Berdasarkan penelitian yang dilakukan oleh Mangapul Siahaan dan Ryan Kenidy, dilakukan perbandingan kinerja *rendering* dari empat *framework* Javascript populer, yaitu React, Vue, Next.js, dan Nuxt.js. Studi ini menekankan pentingnya optimasi performa dalam pengembangan web modern dengan menguji *framework* tersebut menggunakan tiga alat pengujian, meliputi WebPage Test, PageSpeed Insight, dan GTMetrix. Hasil penelitian menunjukkan bahwa Vue memiliki performa terbaik secara keseluruhan dalam pengujian WebPage Test dengan rata-rata waktu rendering 0,9 detik, sementara React unggul di PageSpeed Insight dengan waktu tercepat 2,8 detik. Dalam pengujian menggunakan GTMetrix, Next.js tercatat sebagai yang tercepat dengan waktu 0,8 detik (Siahaan & Kenidy, 2023).

Merujuk pada penelitian sebelumnya, terdapat beberapa celah penelitian yang belum banyak dibahas. Pertama, belum ada penelitian yang secara khusus membandingkan performa *rendering framework* dalam konteks pengembangan website berita, yang memiliki karakteristik berupa kebutuhan akan waktu respon yang cepat untuk memastikan pembaca dapat mengakses informasi secara instan, strategi SEO yang efektif untuk memperbesar visibilitas pada mesin pencari, serta ketahanan sistem dalam menghadapi lonjakan jumlah pengguna secara bersamaan. Penelitian sebelumnya juga belum memperimbangkan secara eksplisit kebutuhan fungsional dari masing-masing website yang diuji, seperti rasio gambar dan teks, spesifikasi perangkat keras yang digunakan untuk fase pengembangan dan pengujian, hingga versi dari tiap *framework* yang digunakan yang tentunya dapat memengaruhi performa. Selain itu, belum ada penelitian yang secara jelas menentukan jenis perangkat atau *device* yang digunakan untuk menjalankan pengujian Google Lighthouse, apakah menggunakan skenario *mobile* atau *desktop*, dan tidak dijelaskan pula pengaturan atau simulasi koneksi jaringan yang digunakan saat pengujian, padahal kedua faktor tersebut berpengaruh signifikan terhadap hasil pengukuran performa. Belum terdapat juga penelitian yang mengkaji performa *framework* Javascript modern seperti Remix.js yang mulai mendapat perhatian dalam pengembangan aplikasi web berbasis *Server Side Rendering*. Untuk mengisi celah kekosongan tersebut, penelitian ini akan berfokus pada perbandingan performa *Server Side Rendering* (SSR) antara *framework* Javascript modern Next.js, Nuxt.js, dan Remix.js dalam konteks website berita.

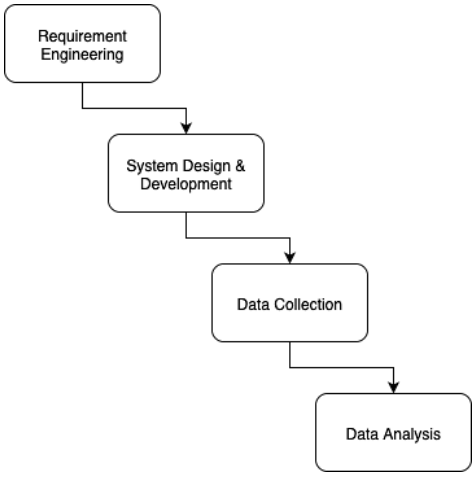
Tujuan dari penelitian ini adalah mengidentifikasi perbedaan performa *rendering* dan ketahanan terhadap beban pengguna website berita yang dikembangkan menggunakan *framework* Next.js, Nuxt.js dan Remix.js, serta memberikan rekomendasi berbasis data bagi pengembang web dan organisasi media dalam menentukan *framework* yang paling optimal untuk website berita. Melalui penelitian ini, diharapkan dapat memberikan pemahaman yang menyeluruh bagi pengembang dalam memilih *framework Server Side Rendering* yang tepat dan sesuai dengan kebutuhan spesifik website berita. Selain itu, hasil penelitian ini juga dapat menjadi rujukan berharga bagi perusahaan media dalam mengoptimalkan kinerja website mereka, sehingga mampu meningkatkan pengalaman pengguna sekaligus memperkuat daya saing di tengah industri digital yang semakin kompetitif.

2. METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan kuantitatif dengan metode eksperimental komparatif untuk menganalisis dan membandingkan kinerja *framework* Next.js, Nuxt.js, dan Remix.js dalam implementasi *Server Side Rendering* (SSR) pada pengembangan website berita. Pendekatan ini dipilih karena memungkinkan peneliti untuk melakukan pengamatan langsung terhadap performa masing-masing *framework* melalui pengujian terstandarisasi. Pendekatan ini relevan

TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

dengan tujuan penelitian, yaitu mengidentifikasi perbedaan peforma *rendering* dan ketahanan beban website yang dikembangkan dengan *framework* Next.js, Nuxt.js, dan Remix.js, serta memberikan rekomendasi berbasis data kepada pengembang web dan organisasi media tentang pilihan *framework* yang paling optimal untuk website berita. Dengan menerapkan pendekatan eksperimen terkontrol, penelitian ini memastikan replikasi yang akurat dan menghasilkan data objektif yang dapat diverifikasi. Penelitian dilakukan dalam 4 tahapan sistematis yang meliputi *Requirement Engineering*, *System Design & Development*, *Data Collection*, dan *Data Analysis* seperti yang ditunjukkan pada Gambar 1.



Gambar 1. Tahapan metode penelitian

2.1 Requirements Analysis

Requirement Analysis pada penelitian ini dilakukan untuk mengidentifikasi dan merumuskan kebutuhan dasar pengembangan prototipe website berita. Teknologi yang digunakan pada penelitian ini adalah Next.js, Nuxt.js, dan Remix.js yang akan dikembangkan pada lingkungan pengembangan yang sama. Selain itu, website juga di kembangkan dengan jumah data dan antarmuka yang identik, sehingga perbandingan kinerja yang dilakukan dapat bersifat objektif. Hal ini penting karena dengan mempertimbangkan variabel terkontrol seperti spesifikasi perangkat keras, jumlah data, hingga lingkungan pengembangan yang konsisten akan menghindarkan dari faktor eksternal yang dapat mempengaruhi hasil akhir penelitian (Anggraeni et al., 2024).

Berikut ini adalah kebutuhan yang telah diidentifikasi untuk mendukung proses pengembangan dan pengujian prototipe:

a. Data Requirements
   Pada penelitian ini, ketiga prototipe website berita yang dibandingkan akan menampilkan sebanyak 1000 data berita yang terdiri dari kombinasi antara teks dan gambar. Teks artikel mencakup judul dan tanggal publikasi, sedangkan gambar memiliki resolusi sebesar 123x124 pixel yang akan menjadi ilustrasi pendukung berita. Seluruh data berita diperoleh melalui API publik “Berita Indo API” yang dikembangkan oleh Satya Wikananda.
b. Hardware and Software Requirements
   Aplikasi akan dibangun dan diuji menggunakan spesifikasi perangkat keras sebagai berikut :

Tabel 1. Spesifikasi perangkat keras dan lunak pengujian

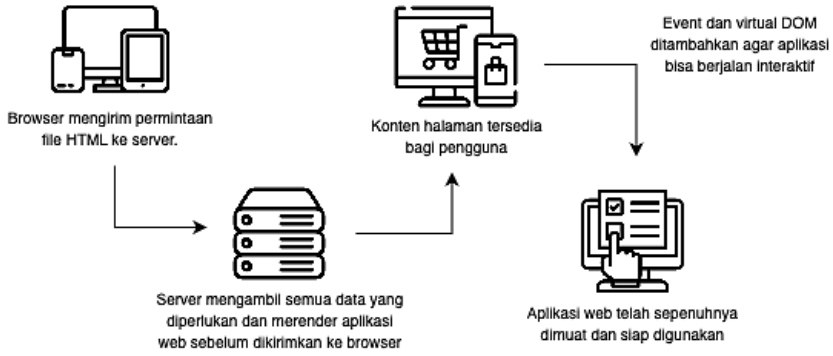
Kategori	Nama	Spesifikasi
Software	Laptop	MacBook Pro 14
	Chip	Apple M4 Pro
	CPU	14 Core
	Storage	1TB (SSD)
	Memory	24GB
Hardware	Sistem Operasi	macOS Sequoia version 15.5
	Browser	Google Chrome
	Code Editor	Visual Studio Code
	Tools Pengujian Kinerja Website	Google Lighthouse
	Tools Pengujian Beban Website	JMeter v5.6.3
	Runtime Environment	Node.js v23.11.0
	Web Application Framework 1	Next.js v15.3.4
	Web Application Framework 2	Nuxt.js v3.17.6
	Web Application Framework 3	Remix.js v2.16.8
	CSS Framework	Tailwind v4.1.11

TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

Tabel 1 menunjukkan spesifikasi perangkat keras dan perangkat lunak yang digunakan sebagai acuan untuk memastikan proses pembangunan dan pengujian aplikasi berlangsung pada lingkungan yang terstandarisasi dan konsisten.

2.2 System Design and Development

Sistem yang dikembangkan dalam penelitian ini menggunakan teknik atau pendekatan *Server Side Rendering* (SSR). SSR adalah metode di mana proses *rendering* halaman web dilakukan di sisi server sebelum dikirim ke browser pengguna. Artinya, HTML yang sudah lengkap dikirim langsung ke klien, sehingga mempercepat waktu tampilan awal dan meningkatkan performa serta SEO (Bhanuartha et al., 2025), terutama pada website dengan konten dinamis seperti website berita. Alur proses *Server Side Rendering* secara lebih jelas dapat dilihat pada Gambar 2.



Gambar 2. Tahapan *Server Side Rendering* pada website

Dalam implementasinya, prototipe website berita dikembangkan menggunakan tiga *framework* modern yang mendukung SSR, yaitu Next.js, Nuxt.js, dan Remix.js. Antarmuka pengguna dirancang secara identik pada setiap prototipe, dengan memanfaatkan Tailwind CSS sebagai kerangka kerja *utility first* untuk mempercepat pengembangan desain yang responsif dan konsisten. Konten berita yang digunakan dalam pengujian tidak diambil langsung dari API publik "Berita Indo API", melainkan terlebih dahulu dikonversi menjadi file JSON lokal yang kemudian di *fetch* oleh masing-masing prototipe website. Dengan pendekatan ini, kemampuan setiap *framework* dalam melakukan *render* halaman dapat diukur secara lebih akurat tanpa dipengaruhi faktor eksternal seperti latensi jaringan atau waktu respon server API.

2.3 Data Collection

Pada penelitian ini, data dikumpulkan melalui uji coba performa antara tiga *framework* dengan menggunakan dua *tools* utama, yaitu Google Lighthouse dan Apache JMeter. Setiap pengujian, baik menggunakan Google Lighthouse maupun JMeter, dilakukan sebanyak 5 kali untuk memastikan konsistensi hasil serta meminimalkan kemungkinan munculnya variabilitas acak yang dapat memengaruhi akurasi temuan. Seluruh pengujian dijalankan dengan parameter dan kondisi yang sama sehingga memastikan lingkungan pengujian konsisten dan tidak terpengaruh oleh variabel lain di luar skenario yang diuji.

Google Lighthouse adalah alat *open source* otomatis yang digunakan untuk mengukur dan mengaudit kualitas halaman web, termasuk performa, aksesibilitas, dan aspek aplikasi web progresif (Muna et al., 2022). Pada Google Lighthouse terdapat beberapa metrik yang digunakan untuk mengevaluasi performa ketiga website berita, meliputi *First Contentful Paint* yang mengukur waktu hingga konten pertama ditampilkan, *Largest Contentful Paint* yang menunjukkan waktu render elemen terbesar di layar, *Cumulative Layout Shift* yang menilai kestabilan tata letak halaman, *Total Blocking Time* yang menghitung total waktu halaman tidak responsif terhadap interaksi pengguna, serta *Speed Index* yang mengukur kecepatan tampilan visual konten selama proses pemuatan (Cahyono & Kamarudin, 2024). Penelitian ini menggunakan acuan kategori peforma metrik Google Lighthouse sebagaimana dirangkum dalam Tabel 2. Pengujian menggunakan Google Lighthouse pada penelitian ini dilakukan dengan pengaturan *device* mobile dan *network fast 4G* untuk memastikan kondisi pengujian konsisten pada setiap *framework*.

Tabel 2. Kategori tingkat kinerja Google Lighthouse

Metrik	Tingkat Kinerja
First Contentful Paint (FCP)	Cepat, indicator hijau (0-1,8 detik) Sedang, indicator oranye (1,8-3 detik) Lambat, indicator merah (> 3 detik)
Largest Contentful Paint (LCP)	Cepat, indikator hijau (0-2,5 detik) Sedang, indikator oranye (2,5-4 detik) Lambat, indikator merah (> 4 detik)
Cumulative Layout Shift (CLS)	Cepat, indikator hijau (0-0,1) Sedang, indikator oranye (0,1-0,25)

TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

Metrik	Tingkat Kinerja
Total Blocking Time (TBT)	Lambat, indikator merah (> 0,25)
	Cepat, indikator hijau (0-200 milidetik)
	Sedang, indikator oranye (200-600 milidetik)
	Lambat, indikator merah (> 600 milidetik)
Speed Index (SI)	Cepat, indikator hijau (0-3,4 detik)
	Sedang, indikator oranye (3,4-5,8 detik)
	Lambat, indikator merah (> 5,8 detik)

Pengujian menggunakan Apache JMeter dilakukan untuk mensimulasikan banyak pengguna yang mengakses website secara bersamaan, sehingga memungkinkan evaluasi terhadap ketahanan dan kinerja sistem (Ismail et al., 2023). JMeter mengirimkan banyak *request* dari sisi pengguna secara paralel untuk mengukur *average response time*, *error rate*, dan *throughput*.

Pengujian beban dilakukan dengan tiga skenario berbeda menggunakan 250, 500, dan 1000 *virtual users/threads* yang dijalankan untuk tiga *framework*, yaitu Nuxt.js, Next.js, dan Remix.js. *Ramp-up* period ditetapkan 5 detik, sedangkan durasi pengujian ditetapkan 300 detik untuk memastikan beban berlangsung stabil. *Loop Count* diatur ke pengaturan *infinite* membuat setiap pengguna terus mengulang *request* hingga durasi pengujian berakhir.

2.2 Data Analysis

Analisis data dalam penelitian ini dilakukan dengan cara membandingkan hasil pengujian performa dari masing-masing *framework*, yaitu Next.js, Nuxt.js, dan Remix.js. Setiap metrik dilakukan pengujian sebanyak lima kali, kemudian hasilnya dirata-ratakan untuk setiap framework agar memperoleh nilai yang lebih konsisten dan representatif. Data yang diperoleh dari Google Lighthouse dianalisis untuk mengevaluasi aspek performa berdasarkan lima metrik utama, yaitu *First Contentful Paint* (FCP), *Largest Contentful Paint* (LCP), *Cumulative Layout Shift* (CLS), *Total Blocking Time* (TBT), dan *Speed Index* (SI). Nilai rata-rata dari masing-masing metrik diperoleh secara numerik dari hasil pengujian pada setiap *framework* dan dibandingkan secara langsung untuk melihat *framework* mana yang menghasilkan performa terbaik dalam hal kecepatan muat dan stabilitas visual halaman. Sementara itu, data dari Apache JMeter digunakan untuk menganalisis ketahanan website terhadap peningkatan beban pengguna. Tiga skenario beban yang digunakan, yaitu 250, 500, dan 1000 pengguna virtual, menghasilkan data numerik berupa *response time*, *error rate*, dan *throughput*.

Data yang terkumpul dianalisis untuk menguji signifikansi perbedaan menggunakan metode ANOVA. ANOVA adalah metode statistik yang digunakan untuk menguji perbedaan rata-rata antar beberapa kelompok data. Tujuan utamanya adalah menilai apakah perbedaan yang muncul di dalam data disebabkan oleh faktor yang diuji atau sekadar muncul akibat variasi acak (Nainggolan et al., 2025). Metrik Lighthouse diuji menggunakan metode *one way* ANOVA, meliputi FCP, LCP, dan SI dengan ambang batas signifikansi (α) sebesar 0,05. Sementara itu, metrik TBT dan CLS tidak dianalisis lebih lanjut menggunakan ANOVA karena hasil pengujian menujukkan nilai yang sangat kecil dan relatif konstan pada setiap iterasi, sehingga tidak menunjukkan perbedaan yang signifikan antar *framework*. Adapun metrik JMeter, meliputi *average response time*, *throughput*, dan *error rate* dianalisis menggunakan *two way* ANOVA, untuk menguji pengaruh dua faktor sekaligus, yaitu *framework* dan jumlah *thread*, serta interaksi keduanya, dengan ambang batas signifikansi (α) sebesar 0,05. Hasil dari pengujian Lighthouse dan JMeter ini menjadi dasar dalam menentukan *framework* yang optimal untuk pengembangan website berita dengan pendekatan *Server Side Rendering*.

3. HASIL DAN PEMBAHASAN

3.1 Hasil

3.1.1 Hasil Pengujian Google Lighthouse

Pengujian performa menggunakan Google Lighthouse dilakukan untuk mengukur kecepatan render dan stabilitas layout dari tiga *framework* pada implementasi *Server Side Rendering*. Pengujian ini berfokus pada lima metrik utama, yaitu *First Contentful Paint* (FCP), *Largest Contentful Paint* (LCP), *Cumulative Layout Shift* (CLS), *Total Blocking Time* (TBT), dan *Speed Index* (SI). Hasil yang diperoleh disajikan pada Tabel 2 hingga Tabel 6 sebagai berikut.

Tabel 3. Hasil pengujian performa berdasarkan metrik FCP (s)

Iterasi	Next.js	Nuxt.js	Remix.js
1	5,3	9,2	2,2
2	5,3	9,2	2,2
3	5,3	9,2	2,1
4	5,3	9,2	2,1
5	5,3	9,2	2,1

Hasil pengujian pada Tabel 3 menunjukkan bahwa nilai FCP pada setiap *framework* relatif stabil di seluruh iterasi. Remix.js memiliki waktu dengan rentang 2,1 hingga 2,2 detik, Next.js konsisten pada 5,3 detik di semua iterasi,

TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

sedangkan Nuxt.js memiliki waktu sebesar 9,2 detik. Perbedaan ini memperlihatkan variasi kinerja antar *framework* dalam menampilkan konten awal halaman.

Tabel 4. Hasil pengujian performa berdasarkan metrik SI (s)

Iterasi	Next.js	Nuxt.js	Remix.js
1	5,3	9,2	2,2
2	5,3	9,2	2,2
3	5,3	9,2	2,1
4	5,3	9,2	2,1
5	5,3	9,2	2,1

Hasil pengujian metrik SI pada Tabel 4 memperlihatkan tren yang sejalan dengan metrik FCP. Remix.js kembali mencatat waktu dengan rentang 2,1 hingga 2,2 detik, diikuti oleh Next.js dengan nilai tetap 5,3 detik, dan Nuxt.js pada 9,2 detik. Seluruh *framework* menunjukkan konsistensi hasil tanpa fluktuasi antar iterasi, menandakan kestabilan performa dalam memuat keseluruhan tampilan halaman secara visual.

Tabel 5. Hasil pengujian performa berdasarkan metrik LCP (s)

Iterasi	Next.js	Nuxt.js	Remix.js
1	7,1	10,0	2,9
2	6,7	11,0	3,0
3	6,8	10,0	2,6
4	7,8	10,0	2,6
5	6,8	10,1	3,4

Pada Tabel 5, hasil pengujian LCP menunjukkan sedikit variasi antar iterasi. Remix.js mencatat waktu dengan rentang 2,6 hingga 3,4 detik, diikuti oleh Next.js pada 6,7 hingga 7,8 detik, dan Nuxt.js pada 10,0 hingga 11,0 detik. Pola ini menunjukkan perbedaan kecepatan dalam menampilkan elemen utama halaman seperti gambar atau *headline*.

Tabel 6. Hasil pengujian performa berdasarkan metrik TBT (ms)

Iterasi	Next.js	Nuxt.js	Remix.js
1	40	0	0
2	30	0	0
3	20	0	0
4	30	0	0
5	20	0	0

Pada Tabel 6, nilai TBT menunjukkan hasil yang stabil. Nuxt.js dan Remix.js memperoleh nilai 0 ms di seluruh iterasi, yang menandakan tidak adanya waktu blokir selama proses render. Sementara itu, Next.js mencatat waktu dengan rentang 20 hingga 40 ms, masih tergolong cepat dan tidak menunjukkan variasi besar antar pengujian.

Tabel 7. Hasil pengujian performa berdasarkan metrik CLS

Iterasi	Next.js	Nuxt.js	Remix.js
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0

Nilai pada metrik CLS pada Tabel 7 menunjukkan pola yang sangat stabil dengan nilai 0 di seluruh iterasi. Ketiga *framework* mempertahankan hasil yang identik pada setiap pengujian, tanpa adanya fluktuasi nilai antar iterasi.

3.1.2 Hasil Pengujian JMeter

Pengujian dengan Apache JMeter dilakukan untuk mengevaluasi ketahanan beban masing-masing *framework* dengan skenario 250, 500, dan 1000 *threads*. Parameter yang diukur meliputi *average response time*, *throughput*, dan *error rate*. Hasil yang diperoleh disajikan pada Tabel 7 hingga Tabel 10 sebagai berikut.

Tabel 8. Hasil pengujian beban berdasarkan metrik Response Time (ms)

Threads	Iterasi	Next.js	Nuxt.js	Remix.js
250	1	13046	1931	2963
	2	13173	1927	2963
	3	13242	1947	2970
	4	13154	1945	2966
	5	13299	1920	2957

TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

Threads	Iterasi	Next.js	Nuxt.js	Remix.js
500	1	14724	3470	5102
	2	14728	3470	5156
	3	14758	3466	5092
	4	14756	3451	5170
	5	14788	3486	5099
1000	1	15286	5740	7794
	2	15301	5713	7767
	3	15306	5703	7803
	4	15331	5735	7804
	5	15326	5723	7805

Hasil pengujian pada Tabel 8 menunjukkan bahwa nilai *response time* pada masing-masing *framework* meningkat seiring bertambahnya jumlah *threads*, menunjukkan pengaruh langsung antara beban pengguna dan waktu respons. Tren ini menunjukkan pola peningkatan linier pada seluruh *framework* ketika beban pengguna bertambah, namun dengan tingkat kestabilan yang berbeda-beda.

Tabel 9. Hasil pengujian beban berdasarkan metrik Throughput (req/sec)

Threads	Iterasi	Next.js	Nuxt.js	Remix.js
250	1	14,95239	103,32827	67,24939
	2	14,81409	103,47063	67,25367
	3	14,72700	102,40749	67,05733
	4	14,81991	102,59310	67,17958
	5	14,66563	103,91037	67,37555
500	1	26,12846	114,64533	77,77360
	2	26,07420	114,61705	76,91705
	3	26,01630	114,78313	77,90660
	4	26,04376	115,29829	76,73338
	5	25,98028	114,16019	77,79768
1000	1	49,85788	138,00376	101,24858
	2	49,87386	138,63699	101,62490
	3	49,88861	138,97222	101,13190
	4	49,80336	138,02031	101,12784
	5	49,80237	138,47296	101,14766

Pada Tabel 9, nilai *throughput* di ketiga *framework* menunjukkan peningkatan konsisten ketika jumlah *threads* bertambah. Pola ini memperlihatkan hubungan langsung antara peningkatan jumlah *threads* dengan kapasitas penanganan permintaan pada masing-masing *framework*.

Tabel 10. Hasil pengujian beban berdasarkan metrik Error Rate (%)

Threads	Iterasi	Next.js	Nuxt.js	Remix.js
250	1	25.43%	1.38%	0.81%
	2	25.54%	1.36%	0.77%
	3	25.55%	1.42%	0.79%
	4	25.31%	1.42%	0.74%
	5	26.07%	1.41%	0.78%
500	1	57.74%	11.03%	14.48%
	2	57.88%	11.03%	14.73%
	3	57.90%	11.03%	14.53%
	4	57.95%	10.98%	14.65%
	5	58.01%	11.08%	14.53%
1000	1	78.03%	26.65%	34.85%
	2	78.08%	26.52%	34.78%
	3	78.15%	26.54%	34.88%
	4	78.14%	26.62%	34.92%
	5	78.11%	26.63%	34.92%

Pada Tabel 10, metrik *error rate* menunjukkan pola kenaikan seiring peningkatan jumlah *threads* untuk semua *framework*. Pola ini menegaskan bahwa seluruh *framework* mengalami peningkatan tingkat kesalahan ketika beban pengguna bertambah, namun dengan perbedaan tingkat kestabilan antar *framework*.

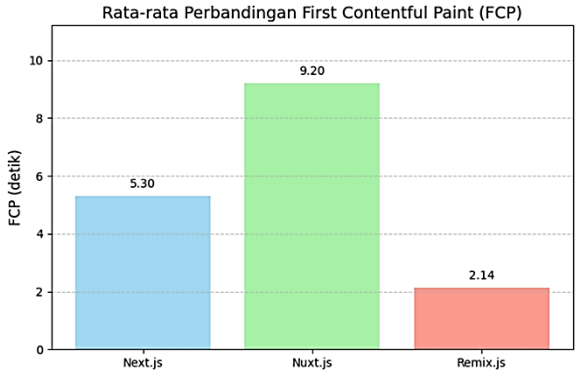
TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

3.2 Pembahasan

Setelah pengujian dilakukan sebanyak 5 kali pada masing-masing *framework*, nilai yang diperoleh kemudian dirata-ratakan untuk memberikan hasil yang lebih representatif. Rata-rata akan digunakan sebagai dasar perbandingan performa antar *framework* pada setiap metrik yang diuji.

3.2.1 Analisis Pengujian Google Lighthouse

a. First Contentful Paint (FCP)



Gambar 3. Grafik perbandingan rata-rata FCP pada pengujian Google Lighthouse

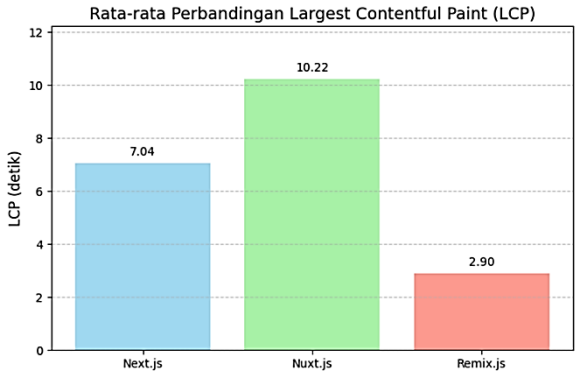
Tabel 11. Hasil uji ANOVA pada metrik FCP

Framework	Std. Dev	Min	Max	F	p-value
Next.js	0,000	5,30	5,30		
Nuxt.js	0,000	9,20	9,20	62532,67	0,000
Remix.js	0,055	2,10	2,20		

First Contentful Paint mengukur waktu sejak halaman mulai dimuat hingga konten pertama muncul di layar. Hasil pengujian statistik pada tabel 11 menunjukkan terdapat perbedaan signifikan antar *framework* dengan p-value 0,000 yang lebih kecil dari ambang signifikansi 0,05. Hasil pengujian yang tersaji pada gambar 3 memperlihatkan bahwa Remix.js terbukti paling unggul dengan waktu rata-rata 2,14 detik, diikuti Next.js dengan 5,3 detik, dan Nuxt.js sebagai yang terlama dengan 9,2 detik.

Perbedaan ini berdampak langsung pada pengalaman pengguna dalam mengakses website berita. Berdasarkan kategori kinerja Google Lighthouse, Remix.js masuk pada kategori sedang, namun tetap menjadi yang paling cepat dibanding *framework* lainnya sehingga pengguna dapat segera melihat elemen awal halaman dan merasakan kenyamanan dalam mengakses informasi. Next.js dengan waktu dan Nuxt.js sama-sama masuk kategori lambat**.** Kondisi ini membuat pengguna berpotensi merasakan keterlambatan, sehingga memberikan risiko yang lebih besar dalam menurunkan kepuasan, serta mengurangi loyalitas pembaca website berita (Hidayat & Nasution, 2024).

b. Largest Contentful Paint (LCP)



Gambar 4. Grafik perbandingan rata-rata LCP pada pengujian Google Lighthouse

Tabel 12. Hasil uji ANOVA pada metrik LCP

Framework	Std. Dev	Min	Max	F	p-value
Next.js	0,451	6,70	7,80	400,17	0,000
Nuxt.js	0,438	10,0	11,0		

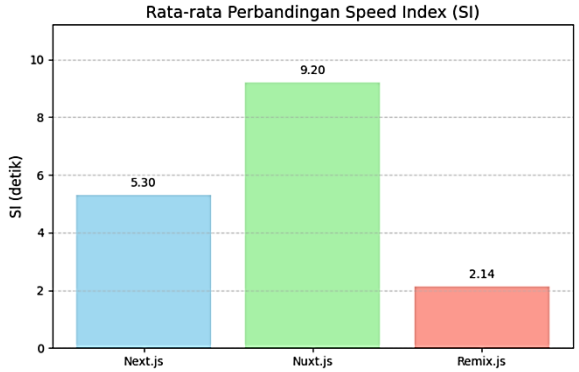
TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

Framework	Std. Dev	Min	Max	F	p-value
Remix.js	0,332	2,60	3,40		

Largest Contentful Paint mengukur waktu yang diperlukan untuk merender elemen konten terbesar dalam viewport. Hasil pengujian statistik pada Tabel 12 menunjukkan terdapat perbedaan signifikan antar *framework* dengan p-value 0,000 yang lebih kecil dari ambang signifikansi 0,05. Hasil pengujian yang tersaji pada Gambar 4 menempatkan Remix.js kembali di posisi terbaik dengan waktu rata-rata 2,9 detik, disusul Next.js dengan rata-rata 7,04 detik, dan Nuxt.js dengan waktu rata-rata 10,22 detik.

Perbedaan ini berdampak langsung pada pengalaman pengguna website berita. Pada kasus ini Remix.js memiliki LCP yang paling cepat, membuat elemen utama seperti gambar berita atau *headline* segera muncul, sehingga pengguna lebih cepat merasa halaman telah siap digunakan. Next.js dan Nuxt.js yang tergolong memiliki waktu LCP lambat berpotensi membuat halaman terlihat lambat selesai dimuat, sehingga menurunkan persepsi performa website, dan meningkatkan risiko pengguna meninggalkan halaman lebih awal.

c. Speed Index (SI)



Gambar 5. Grafik perbandingan rata-rata SI pada pengujian Google Lighthouse

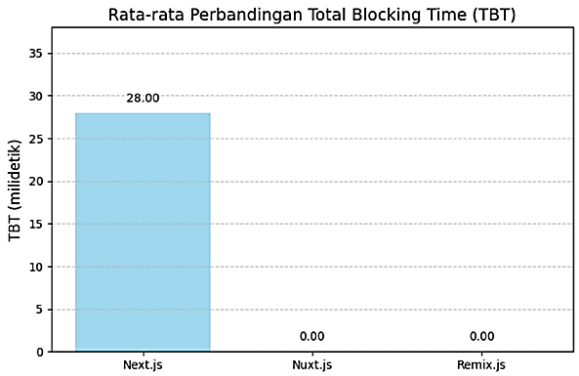
Tabel 13. Hasil uji ANOVA pada metrik SI

Framework	Std. Dev	Min	Max	F	p-value
Next.js	0,000	5,30	5,30		
Nuxt.js	0,000	9,20	9,20	62532,67	0,000
Remix.js	0,055	2,10	2,20		

Speed Index mengukur kecepatan tampilan konten secara visual. Hasil pengujian statistik pada Tabel 13 menunjukkan terdapat perbedaan signifikan antar *framework* dengan p-value 0,000 yang lebih kecil dari ambang signifikansi 0,05. Remix.js menunjukkan waktu tercepat dengan rata-rata 2,14 detik, disusul oleh Next.js dengan 5,3 detik, serta Nuxt.js yang paling lambat dengan 9,2 detik seperti yang terlihat pada gambar 5.

Berdasarkan hasil pengujian, Remix.js yang termasuk dalam kategori cepat mampu menampilkan halaman secara utuh dengan sangat singkat. Hal ini memberikan kesan responsif dan nyaman bagi pengguna, sehingga pengalaman mengakses situs terasa lebih baik serta meningkatkan kemungkinan pengguna untuk bertahan lebih lama. Next.js, yang berada pada kategori sedang, masih menawarkan performa yang cukup baik dan dapat diterima. Meski tidak secepat Remix.js, waktu tampilnya relatif stabil, hanya saja pengguna mungkin merasakan sedikit keterlambatan, terutama ketika menggunakan jaringan yang kurang optimal. Sementara itu, Nuxt.js berada pada kategori lambat dengan waktu tampilan yang jauh lebih panjang. Kondisi ini berpotensi menurunkan kepuasan dan loyalitas pengguna karena harus menunggu lebih lama (Hidayat & Nasution, 2024).

d. Total Blocking Time (TBT)



Gambar 6. Grafik perbandingan rata-rata TBT pada pengujian Google Lighthouse

TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

Total Blocking Time mengukur total waktu di mana halaman terblokir dan tidak dapat merespons interaksi pengguna. Pada Gambar 6 dapat terlihat bahwa Nuxt.js dan Remix.js mencatat nilai 0 ms, menandakan bahwa proses *render* dan eksekusi Javascript mereka tidak menghambat interaksi pengguna. Next.js mencatat rata-rata 28 ms, yang meskipun tergolong cepat, menunjukkan adanya eksekusi *task* Javascript kecil yang sempat memblokir *thread* utama dalam waktu singkat. Perbedaan ini dapat disebabkan oleh strategi optimalisasi *bundle* Javascript yang lebih efisien di Nuxt.js dan Remix.js.

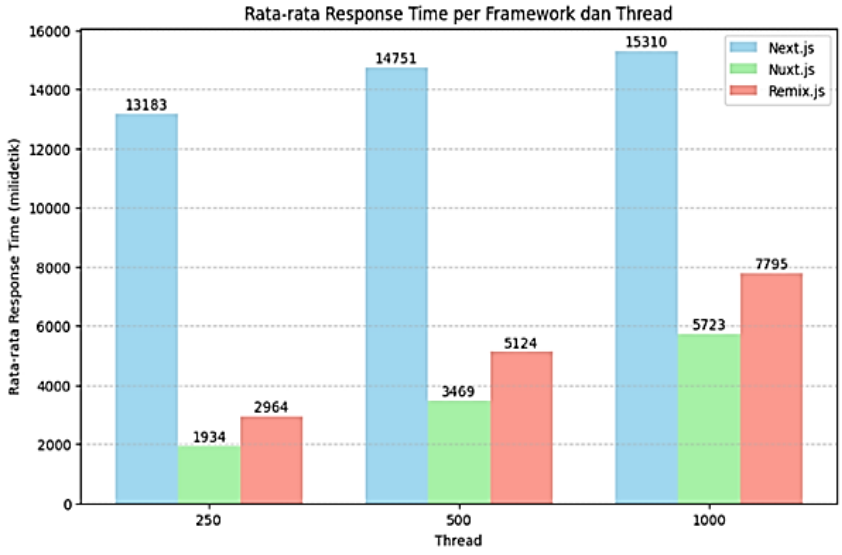
e. Cumulative Layout Shift (CLS)

Cumulative Layout Shift mengukur kestabilan visual selama proses pemuatan halaman. Ketiga *framework* mencatat nilai 0, hal ini menunjukkan bahwa tidak ada pergeseran tata letak yang signifikan saat halaman dimuat. Hal ini menandakan bahwa semua *framework* telah mengimplementasikan praktik terbaik dalam penentuan dimensi elemen, seperti gambar dan komponen UI, sehingga tidak menyebabkan perubahan posisi konten saat proses *render* berlangsung.

Berdasarkan hasil pengujian menggunakan Google Lighthouse, Remix.js menunjukkan keunggulan pada seluruh metrik performa. *Framework* ini mencatat waktu terendah pada FCP, LCP, dan SI, memiliki performa yang setara dengan Nuxt.js pada metrik interaktivitas (TBT), serta sama-sama stabil dengan semua *framework* pada kestabilan layout (CLS). Remix dirancang dengan menggunakan prinsip *Web Standards* seperti *loader/action*, formulir standar HTML, serta *progressive enhancement*, sehingga *rendering* awal sebagian besar diselesaikan di server (Landgraf, 2023). Pengurangan beban JavaScript pada klien misalnya *bundle* yang lebih kecil, serta logika interaktif yang hanya dijalankan bila diperlukan memungkinkan FCP, LCP, dan SI mencapai nilai lebih baik karena konten visual bisa muncul lebih cepat dan lebih lengkap sebelum *client side hydration* atau script berat dijalankan. Dengan hasil tersebut, Remix.js dapat dikatakan sebagai *framework* dengan performa paling konsisten dan unggul secara menyeluruh dalam implementasi SSR, khususnya pada aspek kecepatan akses.

3.2.2 Analisis Pengujian JMeter

a. Response Time



Gambar 7. Grafik perbandingan rata-rata Response Time pada pengujian Jmeter

Tabel 14. Hasil uji ANOVA pada metrik Response Time

Source	Sum Sq	df	F-value	p-value
Threads	96,258,399.24	2	35,209.04	< 0.001
Framework	1,001,584,515.24	2	366,355.92	< 0.001
Threads × Framework	10,780,150.76	4	1,971.56	< 0.001
Residual	49,210.40	36	–	–

Response time mengukur rata-rata waktu yang dibutuhkan server untuk merespons permintaan dari pengguna. Hasil uji statistik pada Tabel 14 menunjukkan terdapat perbedaan signifikan antar *framework* dan jumlah *thread* dengan nilai p < 0,001 yang lebih kecil dari ambang signifikansi 0,05. Dengan demikian, perbedaan waktu respons antar *framework* pada berbagai tingkat beban dapat dinyatakan signifikan secara statistik.

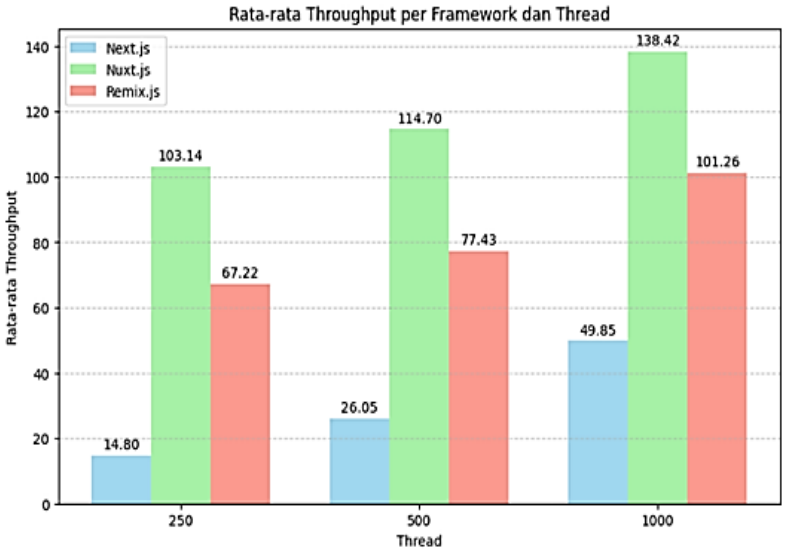
Berdasarkan hasil rata-rata pengujian yang ditampilkan pada gambar 7, pada skenario 250 *thread*, Nuxt.js mencatat waktu tercepat sebesar 1934 ms, diikuti oleh Remix.js dengan 2964 ms, sedangkan Next.js jauh lebih tinggi pada 13183 ms. Pada 500 *thread*, Nuxt.js masih unggul dengan 3469 ms, Remix.js berada di posisi kedua dengan 51234 ms, dan Next.js mencatat 14751 ms. Sementara pada 1000 *thread*, Nuxt.js tetap stabil dengan 5723 ms, Remix.js meningkat ke 7795 ms, sedangkan Next.js mencapai waktu terlama yakni 15310 ms. Pola ini

TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

menunjukkan bahwa Nuxt.js lebih konsisten menjaga performa meskipun jumlah *thread* meningkat, sedangkan Next.js paling rentan mengalami lonjakan waktu respon.

Perbedaan hasil pengujian ini berdampak langsung pada performa website, khususnya dalam konteks website berita yang membutuhkan waktu akses cepat untuk menjaga kepuasan pengguna. *Framework* dengan *response time* rendah, seperti Nuxt.js, memungkinkan berita ditampilkan lebih cepat meskipun terjadi lonjakan trafik, sehingga pengguna tetap mendapatkan pengalaman yang lancar tanpa keterlambatan yang signifikan. Remix.js memberikan performa menengah yang masih cukup baik untuk memastikan akses berita tetap relatif cepat, meski tidak seefisien Nuxt.js dalam skala besar. Sebaliknya, Next.js dengan *response time* yang jauh lebih tinggi berpotensi menurunkan pengalaman pengguna, terutama pada kondisi beban tinggi, karena halaman berita akan memerlukan waktu lebih lama untuk dimuat. Hal ini dapat mengurangi tingkat kenyamanan pengguna, meningkatkan kemungkinan *bounce rate*, serta menurunkan daya saing website berita dalam menghadapi persaingan digital yang menuntut akses informasi secara instan.

b. Throughput



Gambar 8. Grafik perbandingan rata-rata Throughput pada pengujian Jmeter

Tabel 15. Hasil uji ANOVA pada metrik Throughput

Source	Sum Sq	df	F-value	p-value
Threads	9,484.33	2	38,304.86	< 0.001
Framework	59,333.71	2	239,634.24	< 0.001
Threads × Framework	3.16	4	6.39	0.00054
Residual	4.46	36	–	–

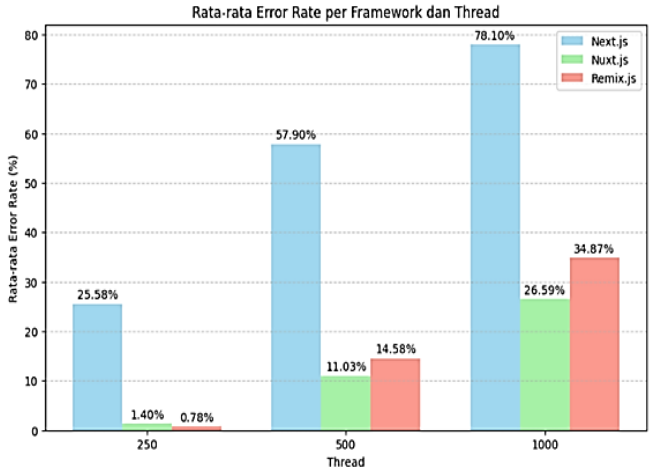
Throughput mengukur jumlah *request* yang berhasil diproses server per detik. Hasil uji statistik pada Tabel 15 menunjukkan adanya perbedaan signifikan antar *framework* dan jumlah *thread*. Faktor *thread* berpengaruh signifikan terhadap waktu respons dengan $p < 0{,}001$, faktor *framework* juga berpengaruh signifikan dengan $p < 0{,}001$, dan interaksi antara keduanya turut signifikan dengan $p = 0{,}00054$. Seluruh nilai p lebih kecil dari ambang signifikansi 0,05 sehingga hasil dapat dinyatakan signifikan.

Berdasarkan hasil rata-rata pengujian yang tersaji pada gambar 8, pada skenario 250 *thread*, Nuxt.js mencatat *throughput* tertinggi dengan 103,14 request/detik, diikuti Remix.js dengan 67,22 request/detik, sementara Next.js hanya mampu mencapai 14,80 request/detik. Pada 500 *thread*, Nuxt.js tetap unggul dengan 114,70 request/detik, Remix.js meningkat ke 77,43 request/detik, dan Next.js tertinggal jauh dengan 26,05 request/detik. Ketika jumlah *thread* ditingkatkan menjadi 1000, Nuxt.js masih dominan dengan 138,42 request/detik, Remix.js bertahan di posisi kedua dengan 101,26 request/detik, sedangkan Next.js masih menjadi yang terendah dengan 49,85 request/detik. Pola ini menunjukkan bahwa Nuxt.js mampu menjaga *throughput* tetap tinggi seiring peningkatan jumlah pengguna, sementara Next.js paling terbatas dalam hal skalabilitas.

Perbedaan *throughput* ini memiliki dampak yang signifikan terhadap performa website berita, terutama ketika dihadapkan pada jumlah pengguna yang besar dan akses secara bersamaan. Nuxt.js dengan *throughput* tertinggi mampu memproses lebih banyak request per detik, sehingga lebih andal dalam menjaga stabilitas layanan ketika terjadi lonjakan trafik, misalnya saat ada berita viral atau *breaking news*. Remix.js berada pada posisi kedua yang masih cukup mumpuni untuk melayani pengguna dengan lancar, meskipun kapasitasnya sedikit lebih terbatas dibanding Nuxt.js. Sebaliknya, Next.js dengan *throughput* terendah berpotensi mengalami *bottleneck* lebih cepat ketika trafik meningkat dibandingkan *framework* lainnya, yang dapat mengakibatkan keterlambatan respon server, hingga menurunnya kecepatan akses halaman berita.

TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

c. Error Rate



Gambar 9. Grafik perbandingan rata-rata Error Rate pada pengujian Jmeter

Tabel 16. Hasil uji ANOVA pada metrik Error Rate

Source	Sum Sq	df	F-value	p-value
Threads	10,417.55	2	400,709.21	< 0.001
Framework	15,302.28	2	588,599.54	< 0.001
Threads × Framework	1,157.86	4	22,268.51	< 0.001
Residual	0.47	36	–	–

Error rate mengukur persentase *request* yang gagal diproses server saat menerima beban tertentu. Hasil uji statistik pada Tabel 16 menunjukkan adanya perbedaan signifikan antar *framework* dan jumlah *thread*, dengan semua nilai p < 0,001 yang lebih kecil dari ambang signifikansi 0,05. Hal ini menegaskan bahwa perbedaan *error rate* antar *framework* maupun antar tingkat beban bersifat signifikan secara statistik, sehingga kinerja *framework* dalam menangani error tidak terjadi secara acak, melainkan dipengaruhi langsung oleh karakteristik arsitektur masing-masing *framework*.

Berdasarkan hasil rata-rata pengujian yang tersaji pada gambar 9, Next**.**js menunjukkan *error rate* yang paling tinggi di semua skenario, yaitu 25,58% pada 250 *thread*, meningkat drastis menjadi 57,90% pada 500 *thread*, dan mencapai 78,10% pada 1000 *thread*. Nuxt.js mencatat performa jauh lebih baik dengan *error rate* yang rendah di 1,40% pada 250 *thread*, naik ke 11,03% pada 500 *thread*, dan 26,59% pada 1000 *thread*. Remix.js memiliki nilai error rate terendah pada 250 *thread* dengan hanya 0,78%**,** namun cenderung meningkat lebih tinggi dibandingkan Nuxt.js pada skenario 500 *thread* sebesar 14,58% dan 1000 *thread* sebesar 34,87%. Pola ini memperlihatkan bahwa Nuxt.js relatif lebih konsisten dan tangguh menghadapi beban tinggi, Remix.js unggul di skenario ringan, sedangkan Next.js paling rentan mengalami kegagalan *request* ketika jumlah pengguna bertambah.

Secara khusus, pada skenario 1000 *thread*, Next.js dapat dikatakan mengalami kegagalan fungsi (*catastrophic failure*) dikarenakan *error rate* yang sangat tinggi mencapai 78,10%. Angka ini menunjukkan bahwa mayoritas *request* tidak berhasil dilayani dengan baik, sehingga server tidak lagi mampu memberikan peforma yang andal di bawah beban berat. Fenomena ini sangat mungkin dipengaruhi oleh karakteristik *Server Side Rendering* tradisional pada Next.js, di mana server merender HTML lalu klien melakukan hidrasi untuk mengaktifkan interaktivitas. Mekanisme ganda ini menjadikan *bottleneck* ketika beban permintaan tinggi. Penelitian yang dilakukan oleh Chen menunjukkan bahwa tanpa optimasi seperti *partial hydration* atau *modular rendering*, *pipeline* SSR tradisional rentan memperlambat respons dan meningkatkan *error* (Chen, 2025). Hal ini sangat mungkin menjadi penyebab utama tingginya *error rate* Next.js pada 1000 *thread*.

Perbedaan *error rate* ini berdampak signifikan terhadap performa website berita. Tingginya *error rate* seperti pada Next.js berpotensi menyebabkan banyak pengguna gagal mengakses konten, sehingga menurunkan reliabilitas website dan memperburuk pengalaman pengguna, terutama pada situasi trafik tinggi seperti berita viral atau *breaking news*. Nuxt.js dan Remix.js *dengan error rate* yang rendah lebih mampu menjaga stabilitas layanan meskipun jumlah pengguna meningkat, sehingga dapat diandalkan untuk website berita dengan trafik besar.

Pada pengujian JMeter, Nuxt.js unggul pada metrik *response time*, *throughput* dan *error rate*. Keunggulan Nuxt.js pada metriks tersebut dapat dijelaskan karena *framework* ini dibangun di atas Vue.js**,** yang dikenal memiliki arsitektur reaktif dan efisien melalui penggunaan *Virtual DOM* serta sistem *reactivity* yang sederhana. Pernyataan tersebut diperkuat dengan penelitian yang dilakukan oleh Sinha & Sinha Jana yang menyatakan bahwa Vue.js menawarkan *layouts* yang lebih stabil dan interaktivitas yang lebih cepat dibandingkan React, terutama pada jaringan yang lebih lambat (Debalina Sinha Jana, 2024). Nuxt.js mengoptimalkan kekuatan tersebut dengan menerapkan *Server Side Rendering* bawaan, di mana konten dirender di sisi server sebelum dikirim ke klien, sehingga mengurangi waktu


TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

tunggu pengguna dan beban *rendering* di browser. Kombinasi efisiensi *Virtual DOM* milik Vue.js dan optimasi SSR bawaan Nuxt.js mampu menjaga *response time* tetap rendah serta mempertahankan *throughput* tinggi meskipun di bawah tekanan trafik yang besar, sekaligus menekan *error rate* karena distribusi beban lebih merata.

Berdasarkan hasil penelitian menggunakan Google Lighthouse dan JMeter, dapat dilihat bahwa masing-masing *framework* memiliki keunggulan berbeda yang relevan dengan karakteristik website berita. Remix.js menunjukkan performa terbaik dalam metrik *rendering* seperti FCP, LCP, dan SI, yang sangat penting bagi website berita karena pengguna dapat segera melihat konten utama seperti *headline* atau gambar berita, sehingga memperkuat persepsi kecepatan dan kenyamanan akses. Di sisi lain, Nuxt.js lebih unggul dalam pengujian JMeter dengan *response time* paling rendah, *throughput* paling tinggi, dan *error rate* yang stabil dari tiap skenario beban, menjadikannya lebih andal untuk menghadapi lonjakan trafik saat terjadi *breaking news* atau berita viral. Dengan demikian, dalam penelitian ini Remix.js menunjukkan kinerja yang lebih optimal dalam menghadirkan pengalaman pengguna yang cepat dan responsif. Sedangkan Nuxt.js lebih unggul dalam ketahanan terhadap trafik tinggi, dengan skalabilitas dan reliabilitas yang lebih baik

4. KESIMPULAN

Penelitian ini menganalisis perbandingan peforma *Server Side Rendering* antara *framework* Next.js, Nuxt.js, dan Remix.js dalam konteks website berita. *Tools* yang dipakai pada penelitian ini adalah Google Lighthouse yang digunakan untuk mengukur peforma *rendering* dan JMeter untuk mengevaluasi ketahanan website. Tujuan penelitian ini adalah untuk mengidentifikasi perbedaan peforma *rendering* dan ketahanan terhadap beban pengguna pada tiap *framework*, serta memberikan rekomendasi berbasis data kepada pengembang web dan organisasi media tentang pilihan *framework* yang paling optimal untuk website berita. Berdasarkan hasil pengujian Google Lighthouse, Remix.js unggul pada metrik FCP, LCP, dan SI. Hal ini membuat konten utama seperti *headline* dan gambar berita tampil lebih cepat, sehingga meningkatkan persepsi kecepatan, kenyamanan akses, dan pengalaman pengguna. Dengan demikian, dalam aspek kecepatan akses dan *user experienc*e, Remix.js lebih unggul. Sementara itu, pengujian JMeter menunjukkan bahwa Nuxt.js memiliki *response time* paling rendah, *throughput* paling tinggi, serta *error rate* yang stabil dari setiap skenario beban. Artinya, Nuxt.js lebih andal dalam menghadapi lonjakan trafik besar, seperti saat *breaking news* atau berita viral, sehingga website tetap stabil, cepat diakses, dan mampu menangani skala pengguna yang meningkat drastis. Dengan demikian, dalam aspek ketahanan, skalabilitas, dan reliabilitas jangka panjang, Nuxt.js lebih unggul. Kombinasi karakteristik ini menegaskan bahwa pemilihan *framework* sebaiknya mempertimbangkan kebutuhan spesifik website berita, apakah lebih menekankan pada kecepatan persepsi pengguna atau pada kestabilan layanan di bawah beban tinggi. Untuk penelitian selanjutnya, perlu dilakukan evaluasi terhadap beberapa aspek non-fungsional yang mencakup konsumsi memori, efisiensi CPU, keamanan, kemudahan pengembangan, serta pengalaman *developer*. Hal ini akan memperkaya perspektif analisis sehingga pemilihan *framework* tidak hanya berdasarkan pada performa teknis, tetapi juga pada faktor keberlanjutan dan kenyamanan dalam pengembangan.

REFERENCES

Anggraeni, O. S. I., Sugiarto, L., & Agustin, T. (2024). Studi Komparatif Performa Framework Javascript Modern dalam Pengembangan Aplikasi Web. *Modem : Jurnal Informatika Dan Sains Teknologi.*, *2*(4), 162–177. https://doi.org/10.62951/modem.v2i4.239

Aryo Subarkah Eddyono. (2022). (PDF) Media Siber dan Search Engine Optimization (SEO): Melacak Motif, Adaptasi untuk Cari Untung, dan Upaya Menjaga Kualitas Jurnalisme. *ResearchGate*. https://doi.org/10.33021/exp.v5i2.4346

Bhanuartha, P. G. A., Pinandito, A., & Akbar, M. A. (2025). Analisis Perbandingan Server Side dan Client Side Data Fetching pada Framework Next.Js (Studi Kasus Aplikasi Online Course). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *9*(3). https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/14579

Cahyono, N., & Kamarudin. (2024). Perbandingan Gtmetrix, Lighthouse, Pingdom dan Pagespeed Insight dalam evaluasi Performa Website. *Jurnal Ilmiah Media Sisfo*, *18*(2), Article 2. https://doi.org/10.33998/mediasisfo.2024.18.2.1901

Chen, K. (2025). *Improving Front-end Performance through Modular Rendering and Adaptive Hydration (MRAH) in React Applications* (No. arXiv:2504.03884). arXiv. https://doi.org/10.48550/arXiv.2504.03884

Debalina Sinha Jana, K. S. (2024). *(PDF) The Role of JavaScript Frameworks in Performance Optimization: A Comparative Study*. https://www.academia.edu/127691429/The_Role_of_JavaScript_Frameworks_in_Performance_Optimization_A_Comparative_Study

Fakhrunnisa, S., & Kurniawan, R. (2024). Perbandingan Kinerja Framework Front End Dalam Pengembangan Platform Talent Pool. *Technologia : Jurnal Ilmiah*, *15*(4), Article 4. https://doi.org/10.31602/tji.v15i4.16750

Hanafi, R., Haq, A., & Agustin, N. (2024). Comparison of Web Page Rendering Methods Based on Next.js Framework Using Page Loading Time Test. *Teknika*, *13*(1), 102–108. https://doi.org/10.34148/teknika.v13i1.769


TIN: Terapan Informatika Nusantara
Vol 6, No 5, October 2025, page 450-463
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i5.8315

Hidayat, P. S., & Nasution, M. I. P. (2024). Pengaruh Kinerja Situs Web Terhadap Kepuasan Dan Loyalitas Pelanggan di E Commerce. *Switch : Jurnal Sains Dan Teknologi Informasi*, *2*(4), 14–25. https://doi.org/10.62951/switch.v2i4.82

Ismail, A., Ananta, A. Y., Arief, S. N., & Hamdana, E. N. (2023). Performance Testing Sistem Ujian Online Menggunakan Jmeter Pada Lingkungan Virtual. *Jurnal Informatika Polinema*, *9*(2), Article 2. https://doi.org/10.33795/jip.v9i2.1190

Landgraf, A. (2023). *Full Stack Web Development with Remix: Enhance the user experience and build better React apps by utilizing the web platform*. Packt Publishing Ltd.

M Syahputra. (2025, May 23). *Analisis Perbandingan Bahasa Java dan JavaScript dalam Implementasi Algoritma Pemrograman | RIGGS: Journal of Artificial Intelligence and Digital Business*. https://journal.ilmudata.co.id/index.php/RIGGS/article/view/646

Maulana, I. D., & Susetyo, Y. A. (2025). Implementasi Fetch API dalam pengembangan Backend Website Daftar Film dengan Next.JS. *Kesatria : Jurnal Penerapan Sistem Informasi (Komputer Dan Manajemen)*, *6*(1), Article 1. https://doi.org/10.30645/kesatria.v6i1.560

Muhammad A Ayub. (2021). (PDF) Analisis Topik Ekonomi Dengan Algoritma K-Means Pada Media Online Era Pandemi Covid-19 Di Sulawesi Tenggara. *ResearchGate*. https://doi.org/10.33387/jiko.v4i2.3235

Mulyadi, E., & Suharman, T. (2024). Analisa Pemakaian Teknik Search Engine Optimization (Seo) Pada Media Berita Online Dalam Media Sustainability. *Journal Visioner : Journal of Television*, *6*(No, 1 Juli), Article No, 1 Juli. https://journal.atvi.ac.id/index.php/jurnal_visioner/article/view/53

Muna, S. S., Nurdin, N., & Taufiq, T. (2022). Tokopedia and Shopee Marketplace Performance Analysis Using Metrix Google Light-house. *International Journal of Engineering, Science and Information Technology*, *2*(3), Article 3. https://doi.org/10.52088/ijesty.v2i3.312

Nainggolan, Y., Divia, D., Hutapea, D. L., Sirait, W. F., Sirait, M., & Sianturi, R. (2025). Anava Satu Jalur (One Way – Anova). *Innovative: Journal Of Social Science Research*, *5*(1), 5670–5682. https://doi.org/10.31004/innovative.v5i1.17989

Ollila, R., Mäkitalo, N., & Mikkonen, T. (2022). Modern Web Frameworks: A Comparison of Rendering Performance. *Journal of Web Engineering*, *21*(3), 789–813. Journal of Web Engineering. https://doi.org/10.13052/jwe1540-9589.21311

Rao, N. S. (2025). Modern Server-Side Rendering: A Technical Deep Dive. *International Journal Of Research In Computer Applications And Information Technology*, *8*(1), 777–789. https://doi.org/10.34218/IJRCAIT_08_01_059

Siahaan, M., & Kenidy, R. (2023). Rendering performance comparison of react, vue, next, and nuxt. *Jurnal Mantik*, *7*(3), Article 3. https://doi.org/10.35335/mantik.v7i3.4242