

ANALISIS KOMPARATIF PERFORMA LANDING PAGE ANTARA ASTRO JS DAN REACT JS MENGGUNAKAN GOOGLE LIGHTHOUSE, PAGESPEED INSIGHTS, DAN JMETER

¹Farhan Harits Prakoso,

²Dwi Widiastuti., Skom., MMSI

¹Universitas Gunadarma, Jl. Margonda Raya No. 100, Depok 16424,

farhan.harits05@gmail.com

²Universitas Gunadarma, Jl. Margonda Raya No. 100, Depok 16424,

dwidiastuti@staff.gunadarma.ac.id

Jurusan Informatika, Fakultas Teknologi Industri,

Universitas Gunadarma

ABSTRAK

React JS banyak digunakan untuk membangun antarmuka web modern, tetapi pendekatan Client-Side Rendering (CSR) dapat menambah pekerjaan pemuatan di sisi browser pada landing page yang sebagian besar berisi konten statis. Penelitian ini membandingkan performa landing page yang dibangun menggunakan Astro JS dengan Static Site Generation (SSG) dan Islands Architecture serta React JS dengan CSR. Penelitian menggunakan pendekatan kuantitatif komparatif melalui dua prototipe dengan konten, struktur, dan aset media identik. Pengukuran dilakukan menggunakan Google Lighthouse dan PageSpeed Insights untuk metrik First Contentful Paint (FCP), Largest Contentful Paint (LCP), Total Blocking Time (TBT), Cumulative Layout Shift (CLS), dan Speed Index (SI), serta Apache JMeter untuk response time, throughput, dan error rate. Astro JS menghasilkan metrik rendering yang lebih baik pada sebagian besar pengujian. Pada Lighthouse, FCP dan SI Astro JS lebih cepat 22,6% pada mode mobile dan 28,6% pada mode desktop. Pada PageSpeed Insights, LCP Astro JS lebih cepat 47,8% pada mode mobile dan 39,7% pada mode desktop. Sebaliknya, JMeter mencatat response time React JS lebih rendah dengan selisih 6,2-11,8 ms. Kedua prototipe menghasilkan error rate 0,00% dalam skenario 100, 200, dan 300 pengguna virtual. Hasil menunjukkan bahwa pengukuran pemuatan di browser dan waktu respons permintaan jaringan merepresentasikan dimensi performa yang berbeda.

Kata Kunci: Astro JS, Google Lighthouse, JMeter, PageSpeed Insights, React JS.

ABSTRACT

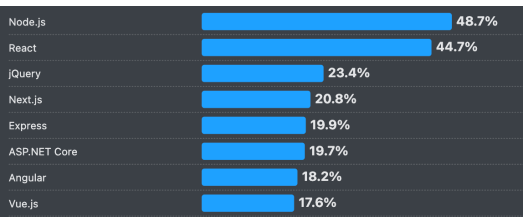
React JS is widely used to build modern web interfaces, but its Client-Side Rendering (CSR) approach may add browser-side work to landing pages that mainly contain static content. This study compares landing page performance built with Astro JS using Static Site Generation (SSG) and Islands Architecture and React JS using CSR. A quantitative comparative experiment was conducted using two prototypes with identical content,

structure, and media assets. Google Lighthouse and PageSpeed Insights measured First Contentful Paint (FCP), Largest Contentful Paint (LCP), Total Blocking Time (TBT), Cumulative Layout Shift (CLS), and Speed Index (SI), while Apache JMeter measured response time, throughput, and error rate. Astro JS achieved better rendering metrics in most tests. In Lighthouse, Astro JS FCP and SI were 22.6% faster in mobile mode and 28.6% faster in desktop mode. In PageSpeed Insights, its LCP was 47.8% faster in mobile mode and 39.7% faster in desktop mode. Conversely, JMeter recorded lower response time for React JS, with differences of 6.2-11.8 ms. Both prototypes maintained a 0.00% error rate in scenarios of 100, 200, and 300 virtual users. The findings indicate that browser rendering measurements and network request response times represent different performance dimensions.

Keywords: Astro JS, Google Lighthouse, JMeter, PageSpeed Insights, React JS.

PENDAHULUAN

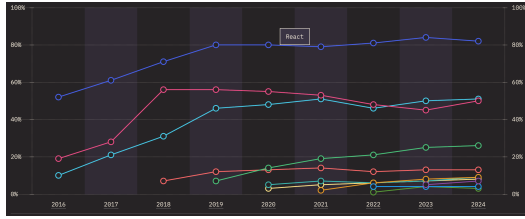
Landing page merupakan salah satu titik pertama interaksi pengguna dengan produk atau layanan digital. Kecepatan halaman menjadi aspek penting karena keterlambatan pemuatan dapat memengaruhi pengalaman pengguna dan hasil pemasaran; Google dan SOASTA (2017) melaporkan bahwa keterlambatan satu detik dapat menurunkan tingkat konversi hingga 20%. Dalam pengembangan frontend, React JS banyak digunakan. Stack Overflow Developer Survey 2025 mencatat penggunaan React oleh 44,7% developer profesional dalam kategori teknologi yang disurvei, sebagaimana ditunjukkan pada Gambar 1.



Gambar 1. Penggunaan Web Framework dan Teknologi Web (2025)

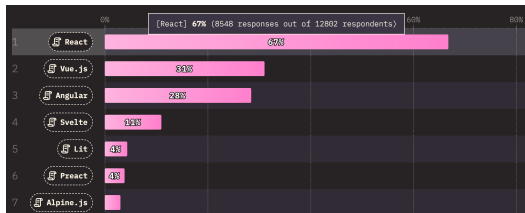
Sumber: Stack Overflow Developer Survey (2025).

Dominasi React juga terlihat dalam survei State of JavaScript. Gambar 2 memperlihatkan bahwa penggunaan React pada survei framework frontend berada pada tingkat lebih dari 80% sejak 2019 sampai 2024, sedangkan Gambar 3 menunjukkan React digunakan secara profesional oleh 67% responden atau 8.548 dari 12.802 responden pada 2024 (State of JS, 2024). Angka adopsi tersebut menunjukkan popularitas, tetapi tidak dengan sendirinya membuktikan bahwa arsitektur yang sama paling sesuai untuk setiap jenis halaman.



Gambar 2. Tren Penggunaan Front-End Framework (2016-2024)

Sumber: State of JavaScript (2024).



Gambar 3. Penggunaan Front-End Framework di Tempat Kerja (2024)

Sumber: State of JavaScript (2024).

React JS pada umumnya menerapkan Client-Side Rendering (CSR), sehingga browser menjalankan JavaScript untuk membentuk antarmuka. Pola ini mendukung antarmuka dinamis, tetapi dapat menambah pekerjaan awal pada halaman yang kontennya mayoritas statis. Sebagai alternatif, Astro JS mendukung Static Site Generation (SSG) dan Islands Architecture, yang memungkinkan konten statis dikirim sebagai HTML serta interaktivitas diaktifkan pada komponen terpilih (Astro, 2024). Perbedaan arsitektur tersebut perlu diuji pada implementasi yang setara, bukan hanya disimpulkan dari karakteristik teknisnya.

Penelitian terdahulu telah membandingkan framework frontend dan pendekatan rendering dari sisi performa (Lazuardy & Anggraini, 2022; Khasanah

& Suartana, 2024; Zaidan & Suyatno, 2025). Namun, penelitian ini memfokuskan perbandingan langsung antara Astro JS dengan SSG dan Islands Architecture dan React JS dengan CSR pada landing page dengan aset media berat, menggunakan pengukuran performa browser serta uji beban jaringan secara bersamaan. Permasalahan yang dibahas adalah apakah kedua pendekatan menghasilkan perbedaan pada metrik Web Vitals dan bagaimana respons keduanya ketika menerima simulasi beban pengguna bersamaan.

Tujuan penelitian adalah mengukur dan membandingkan performa kedua landing page melalui Google Lighthouse dan PageSpeed Insights, mengevaluasi respons serta stabilitasnya melalui Apache JMeter, dan merumuskan rekomendasi pemilihan teknologi berdasarkan hasil pengukuran. Agar perbandingan terkontrol, kedua prototipe memakai konten, struktur, dan aset media yang sama; perbedaan utama terletak pada framework serta mekanisme rendering.

METODE PENELITIAN

Rancangan Penelitian

Penelitian menggunakan metode kuantitatif komparatif melalui eksperimen terhadap dua prototipe landing page. Tahapan yang dilakukan meliputi studi

literatur, perancangan dan pembangunan prototipe, deployment, pengujian performa dengan Google Lighthouse dan PageSpeed Insights, pengujian beban dengan Apache JMeter, serta analisis komparatif data. Setiap instrumen dijalankan dengan pengaturan yang ditetapkan sebelum pengukuran, dan setiap skenario diulang lima kali untuk memperoleh nilai rata-rata sebagaimana dicatat dalam PI.

Objek dan Pengendalian Variabel

Objek penelitian adalah landing page layanan klinik hewan peliharaan fiktif “PawCare - Premium Pet Care & Clinic”. Prototipe Astro JS menggunakan Astro JS 5.1.8 dengan mode SSG dan Islands Architecture. Prototipe pembandingan menggunakan React JS 18.3.1 dengan CSR berbasis Vite 6.0.5. Keduanya ditempatkan pada Vercel dan memakai aset, teks, section, styling Tailwind CSS, animasi CSS Keyframes, Google Fonts, video MP4, gambar PNG, serta sematan Google Maps yang identik. Animasi tidak menggunakan pustaka pihak ketiga, untuk mengurangi variabel luar yang dapat memengaruhi hasil (Anggraeni et al., 2024; Hermanto & Engel, 2025).

Perangkat pengembangan dan pengujian yang digunakan meliputi MacBook Air dengan prosesor Apple M2,

memori 16 GB, penyimpanan SSD, dan macOS Tahoe 26.3. Perangkat lunak yang digunakan meliputi Visual Studio Code, Google Chrome, Node.js, npm, Git dan GitHub, Vercel, Google Lighthouse, PageSpeed Insights, serta Apache JMeter 5.6.3. Rincian hasil build tercantum pada Tabel 1.

Tabel 1. Spesifikasi Lingkungan Penelitian

Aspek	Spesifikasi
Perangkat keras	MacBook Air; Apple M2; RAM 16 GB; SSD 256 GB; macOS Tahoe 26.3
Framework Astro	Astro JS 5.1.8; SSG; Islands Architecture
Framework React	React JS 18.3.1; CSR; Vite 6.0.5
Styling	Tailwind CSS; animasi CSS Keyframes
Deployment	Vercel; dua URL deployment terpisah
Instrumen	Google Lighthouse; PageSpeed Insights; Apache JMeter 5.6.3

Untuk pengujian, prototipe Astro JS dipublikasikan pada <https://astro-landing-five.vercel.app> dan prototipe React JS pada <https://react-landing-jade.vercel.app>. Kedua URL tersebut digunakan sebagai target instrumen. Penggunaan hosting yang sama dimaksudkan untuk mengurangi perbedaan infrastruktur, meskipun lingkungan hosting bersama tetap dapat dipengaruhi kondisi jaringan dan caching.

Tabel 2. Perbandingan Build Output Prototipe

Metrik Build	Astro JS	React JS
Total ukuran HTML	85.52 KB	1.13 KB
Total ukuran JavaScript	211 KB	289.41 KB
Total ukuran CSS	23.86 KB	34.48 KB
Jumlah berkas JavaScript	8 berkas terpisah	1 berkas monolitik
Waktu build	1.62 detik	1.06 detik

Instrumen dan Skenario Pengujian

Google Lighthouse dijalankan melalui Chrome DevTools pada mode mobile dan desktop. Pengujian menggunakan kategori Performance, mode Navigation, konfigurasi throttling bawaan, dan lima iterasi untuk setiap prototipe serta mode. Metrik yang dicatat adalah First Contentful Paint (FCP), Largest Contentful Paint (LCP), Total Blocking Time (TBT), Cumulative Layout Shift (CLS), dan Speed Index (SI). Aplikasi lain ditutup dan jeda minimal 30 detik diberikan antariterasi untuk membantu menstabilkan kondisi browser.

PageSpeed Insights dijalankan melalui pagespeed.web.dev dengan URL yang sama, dua mode perangkat, lima iterasi, dan metrik yang sama. Berbeda dengan Lighthouse lokal, PageSpeed Insights berjalan pada infrastruktur Google dan memadukan lab data dengan data pengalaman pengguna ketika tersedia (Google, 2024). Perbedaan lingkungan

pengukuran perlu diperhatikan saat membandingkan hasil kedua instrumen.

Apache JMeter digunakan untuk mengukur average response time, throughput, dan error rate pada simulasi 100, 200, dan 300 pengguna virtual. Setiap skenario menggunakan Thread Group, loop count satu, metode GET, protokol HTTPS, lima iterasi, dan ramp-up proporsional. Hasil dibaca dari Aggregate Report, sejalan dengan dokumentasi alat dan praktik pengujian beban aplikasi (Apache Software Foundation, 2024; Permatasari et al., 2020). Sesuai batasan rancangan, pengukuran JMeter merepresentasikan waktu permintaan jaringan untuk mengambil berkas dari CDN, bukan waktu rendering penuh di browser, terutama karena HTML React hanya berupa shell.

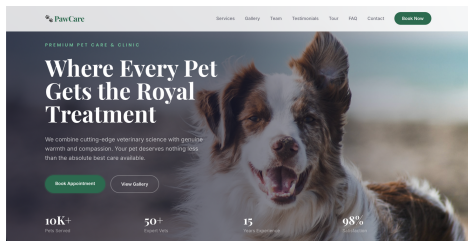
Tabel 3. Konfigurasi Pengujian Apache JMeter

Parameter	Skenario 1	Skenario 2	Skenario 3
Pengguna virtual	100	200	300
Ramp-up	10 detik	20 detik	30 detik
Loop count	1	1	1
Protokol / metode	HTTPS / GET	HTTPS / GET	HTTPS / GET
Jumlah iterasi	5 kali	5 kali	5 kali
Metrik	Response time, throughput, error rate	Response time, throughput, error rate	Response time, throughput, error rate

PEMBAHASAN

Gambaran Umum dan Hasil Prototipe

Kedua prototipe merupakan halaman tunggal yang berisi 11 section: Hero, Services, Gallery, Video, Team, Testimonials, Stats, Virtual Tour, Map & Contact, FAQ, dan Footer. Komposisi tersebut menghadirkan gambar, video, elemen interaktif, peta, serta tipografi eksternal dalam jumlah yang sama untuk kedua implementasi. Gambar 4 memperlihatkan salah satu hasil prototipe, yaitu Hero Section PawCare.



Gambar 4. Tampilan Hero Section Prototipe

Perbedaan implementasi utama bukan pada tampilan, melainkan pada cara HTML dihasilkan dan JavaScript diaktifkan. Astro JS merender konten statis ketika proses build dan menerapkan hidrasi selektif untuk komponen yang membutuhkan interaktivitas. Pada React JS, seluruh aplikasi dibangun melalui JavaScript di browser. Perbedaan ini tercermin pada build output: Astro menghasilkan HTML 85.52 KB dan CSS 23.86 KB, sedangkan React menghasilkan HTML shell 1.13 KB dan CSS 34.48 KB.

Total ukuran JavaScript yang tercatat masing-masing 211 KB dan 289.41 KB. Build React lebih cepat, 1.06 detik dibandingkan 1.62 detik pada Astro, tetapi waktu build bukan metrik utama performa yang dialami pengguna.

Hasil Google Lighthouse

Hasil lima iterasi Google Lighthouse menunjukkan bahwa Astro JS lebih cepat pada FCP, LCP, dan SI di mode mobile maupun desktop. Pada mode mobile, FCP dan SI Astro JS masing-masing 1.78 detik, sedangkan React JS 2.30 detik; LCP masing-masing 3.68 dan 4.00 detik. TBT kedua prototipe tercatat 0 ms, sementara CLS 0.013 pada Astro dan 0.016 pada React. Pada desktop, FCP dan SI Astro JS masing-masing 0.60 detik, sedangkan React JS 0.84 detik; LCP Astro 1.08 detik dan React 1.16 detik. Perbedaan CLS pada desktop sama-sama 0.

Tabel 4. Hasil Google Lighthouse Astro JS Mode Mobile

Iterasi	FCP (s)	LCP (s)	TBT (ms)	CLS	SI (s)
1	1.7	4.1	0	0.013	1.7
2	1.6	3.6	0	0.013	1.6
3	1.6	3.4	0	0.013	1.6
4	1.6	4.0	0	0.013	1.6
5	2.4	3.3	0	0.013	2.4
Rata-rata	1.78	3.68	0	0.013	1.78

Tabel 5. Hasil Google Lighthouse React JS Mode Mobile

Iterasi	FCP (s)	LCP (s)	TBT (ms)	CLS	SI (s)
1	2.5	3.7	0	0.043	2.5
2	2.6	4.1	0	0.013	2.6
3	2.4	4.1	0	0.013	2.4
4	2.4	3.5	0	0.013	2.4
5	1.6	4.6	0	0	1.6
Rata-rata	2.30	4.00	0	0.016	2.30

Tabel 6. Hasil Google Lighthouse Astro JS Mode Desktop

Iterasi	FCP (s)	LCP (s)	TBT (ms)	CLS	SI (s)
1	0.8	1.0	0	0	0.8
2	0.5	1.1	0	0	0.5
3	0.5	1.1	0	0	0.5
4	0.7	1.0	0	0	0.7
5	0.5	1.2	0	0	0.5
Rata-rata	0.60	1.08	0	0	0.60

Tabel 7. Hasil Google Lighthouse React JS Mode Desktop

Iterasi	FCP (s)	LCP (s)	TBT (ms)	CLS	SI (s)
1	1.0	1.1	0	0	1.0
2	0.9	1.1	0	0	0.9
3	0.8	1.2	0	0	0.8
4	0.9	1.2	0	0	0.9
5	0.6	1.2	0	0	0.6
Rata-rata	0.84	1.16	0	0	0.84

Hasil tersebut konsisten dengan karakteristik SSG, yaitu konten HTML tersedia tanpa harus menunggu seluruh aplikasi dirender di browser. Keunggulan relatif Astro lebih besar pada kondisi mobile yang memakai simulasi CPU dan jaringan lebih terbatas, sedangkan selisih pada desktop menjadi lebih kecil. Temuan

sejalan dengan pembahasan arsitektur frontend pada penelitian komparatif sebelumnya (Lazuardy & Anggraini, 2022; Khasanah & Suartana, 2024). Hasil ini terbatas pada konfigurasi prototipe dan lingkungan pengujian yang digunakan; tidak berarti satu framework selalu lebih cepat pada seluruh jenis aplikasi.

Hasil PageSpeed Insights

PageSpeed Insights Kembali menunjukkan keunggulan Astro JS pada sebagian besar metrik. Untuk mode mobile, rata-rata FCP Astro JS 2.74 detik dan React JS 2.90 detik; LCP masing-masing 3.80 dan 7.28 detik; TBT masing-masing 0 dan 10 ms; CLS 0.012 dan 0; serta SI 2.54 dan 3.88 detik. Pada desktop, rata-rata FCP Astro JS 0.70 detik dibandingkan 0.78 detik untuk React; LCP 0.88 dibandingkan 1.46 detik; TBT 0 dibandingkan 10 ms; CLS 0.001 dibandingkan 0; dan SI 0.72 dibandingkan 0.90 detik. Karena PageSpeed Insights memiliki lingkungan lab yang berbeda dari Lighthouse dan dapat menyertakan data lapangan, hasil kedua instrumen tidak harus identik (Google, 2024).

Tabel 8. Hasil PageSpeed Insights Astro JS Mode Mobile

Iterasi	FCP (s)	LCP (s)	TBT (ms)	CLS	SI (s)
1	2.7	3.6	0	0.012	2.7
2	2.6	3.6	0	0.012	1.6
3	2.7	4.1	0	0.012	2.7
4	2.7	4.0	0	0.012	2.7
5	3.0	3.7	0	0.012	3.0
Rata-rata	2.74	3.80	0	0.012	2.54

Tabel 9. Hasil PageSpeed Insights React JS Mode Mobile

Iterasi	FCP (s)	LCP (s)	TBT (ms)	CLS	SI (s)
1	2.9	4.0	0	0	3.2
2	2.9	9.7	0	0	4.8
3	2.9	4.0	10	0	3.8
4	2.9	9.0	20	0	2.9
5	2.9	9.7	20	0	4.7
Rata-rata	2.90	7.28	10	0	3.88

Tabel 10. Hasil PageSpeed Insights Astro JS Mode Desktop

Iterasi	FCP (s)	LCP (s)	TBT (ms)	CLS	SI (s)
1	0.7	0.9	0	0.001	0.7
2	0.7	0.9	0	0.001	0.7
3	0.7	0.9	0	0.001	0.8
4	0.7	0.9	0	0.001	0.7
5	0.7	0.8	0	0.001	0.7
Rata-rata	0.70	0.88	0	0.001	0.72

Tabel 11. Hasil PageSpeed Insights React JS Mode Desktop

Iterasi	FCP (s)	LCP (s)	TBT (ms)	CLS	SI (s)
1	0.8	1.8	20	0	1.0
2	0.7	1.0	0	0	0.7
3	0.8	1.8	10	0	0.9
4	0.8	0.9	0	0	0.8
5	0.8	1.8	20	0	1.1
Rata-rata	0.78	1.46	10	0	0.90

Perbedaan paling besar pada PSI terlihat pada LCP mobile, yaitu 3.80 detik untuk Astro JS dan 7.28 detik untuk React JS. Nilai tersebut bersesuaian dengan kesimpulan PI bahwa LCP Astro lebih cepat 47.8% di mode mobile dan 39.7% di mode desktop. Sebaliknya, metrik CLS bukan berarti semua prototipe bebas dari pergeseran tata letak secara universal; angka tersebut merupakan hasil pengukuran pada lima iterasi dan konfigurasi ini. Pengujian menguatkan temuan bahwa ketersediaan HTML awal dan pengurangan pekerjaan JavaScript pada pemuatan awal berhubungan dengan metrik pemuatan yang lebih baik, khususnya untuk halaman bermuatan media berat (Zaidan & Suyatno, 2025).

Hasil Pengujian Apache JMeter

Berbeda dengan pengujian browser, Apache JMeter mengukur respons permintaan pada tingkat jaringan/CDN. Pada 100, 200, dan 300 pengguna virtual, response time Astro JS berturut-turut 93, 94.6, dan 93 ms, sedangkan React JS 86.8, 82.8, dan 83 ms. Throughput keduanya 10.0 req/s dengan error rate 0.00% pada seluruh skenario. Response time React JS lebih rendah 6.2–11.8 ms, tetapi nilai ini tidak mewakili waktu rendering penuh karena JMeter menerima shell HTML kosong dari React.

Tabel 12. Hasil JMeter Skenario 1 Astro JS (100 Pengguna)

Iterasi	Response Time (ms)	Throughput (req/s)	Error Rate (%)
1	92	10.0	0.00
2	94	10.0	0.00
3	93	10.0	0.00
4	92	10.0	0.00
5	94	10.0	0.00
Rata-rata	93	10.0	0.00

Tabel 13. Hasil JMeter Skenario 1 React JS (100 Pengguna)

Iterasi	Response Time (ms)	Throughput (req/s)	Error Rate (%)
1	88	10.0	0.00
2	86	10.0	0.00
3	84	10.0	0.00
4	84	10.0	0.00
5	92	10.0	0.00
Rata-rata	86.8	10.0	0.00

Tabel 14. Hasil JMeter Skenario 2 Astro JS (200 Pengguna)

Iterasi	Response Time (ms)	Throughput (req/s)	Error Rate (%)
1	98	10.0	0.00
2	94	10.0	0.00
3	94	10.0	0.00
4	95	10.0	0.00
5	92	10.0	0.00
Rata-rata	94.6	10.0	0.00

Tabel 15. Hasil JMeter Skenario 2 React JS (200 Pengguna)

Iterasi	Response Time (ms)	Throughput (req/s)	Error Rate (%)
1	81	10.0	0.00
2	83	10.0	0.00
3	81	10.0	0.00
4	83	10.0	0.00
5	86	10.0	0.00
Rata-rata	82.8	10.0	0.00

Tabel 16. Hasil JMeter Skenario 3 Astro JS (300 Pengguna)

Iterasi	Response Time (ms)	Throughput (req/s)	Error Rate (%)
1	94	10.0	0.00
2	95	10.0	0.00
3	92	10.0	0.00
4	91	10.0	0.00
5	93	10.0	0.00
Rata-rata	93	10.0	0.00

Tabel 17. Hasil JMeter Skenario 3 React JS (300 Pengguna)

Iterasi	Response Time (ms)	Throughput (req/s)	Error Rate (%)
1	82	10.0	0.00
2	82	10.0	0.00
3	83	10.0	0.00
4	86	10.0	0.00
5	82	10.0	0.00
Rata-rata	83	10.0	0.00

Tabel 18. Rekapitulasi Hasil Pengujian Apache JMeter

Skenario	Metrik	Astro JS	React JS
100 pengguna	Response time (ms)	93	86.8
	Throughput (req/s)	10.0	10.0
	Error rate (%)	0.00	0.00
200 pengguna	Response time (ms)	94.6	82.8
	Throughput (req/s)	10.0	10.0
	Error rate (%)	0.00	0.00
300 pengguna	Response time (ms)	93	83
	Throughput (req/s)	10.0	10.0
	Error rate (%)	0.00	0.00

Kestabilan kedua prototipe terlihat dari error rate 0.00% pada seluruh skenario dan

iterasi. Namun, uji ini tidak mengisolasi sepenuhnya variasi jaringan maupun sifat shared infrastructure Vercel. Oleh karena itu, hasil perlu ditafsirkan dalam ruang lingkup skenario yang diuji, bukan sebagai bukti bahwa kedua teknologi mampu mempertahankan hasil yang sama pada seluruh skala trafik atau seluruh kondisi hosting. Perbedaan arah hasil JMeter dan pengujian Web Vitals bukan kontradiksi; instrumen tersebut mengukur tahap yang berbeda: JMeter mengukur waktu respons permintaan terhadap berkas, sedangkan Lighthouse dan PageSpeed Insights merekam pemuatan/rendering dari sudut pandang browser.

PENUTUP

Kesimpulan

Berdasarkan hasil pengujian, Astro JS dengan SSG dan Islands Architecture unggul pada sebagian besar metrik pemuatan browser dibandingkan React JS dengan CSR. Pada Lighthouse, FCP dan SI Astro JS lebih cepat 22.6% dalam mode mobile dan 28.6% dalam mode desktop. Pada PageSpeed Insights, LCP Astro JS lebih cepat 47.8% pada mobile dan 39.7% pada desktop. Build output juga memperlihatkan HTML yang lebih lengkap pada Astro JS (85.52 KB) dibandingkan React JS (1.13 KB),

sementara total JavaScript Astro JS sebesar 211 KB dan React JS 289.41 KB.

Pada Apache JMeter, React JS mencatat response time lebih rendah dengan selisih 6.2-11.8 ms, sedangkan throughput sebesar 10.0 req/s dan error rate 0.00% sama pada kedua prototipe dalam skenario 100, 200, dan 300 pengguna virtual. Hasil tersebut merepresentasikan pengambilan berkas pada tingkat jaringan/CDN dan bukan waktu rendering penuh di browser. Dengan demikian, Astro JS lebih sesuai untuk landing page yang didominasi konten statis dan aset media berat dalam kondisi penelitian ini, sedangkan React JS tetap relevan untuk kebutuhan antarmuka dinamis dan interaktivitas yang lebih tinggi. Pilihan akhir tetap perlu disesuaikan dengan kebutuhan proyek.

Saran

Penelitian lanjutan sebaiknya menguji lebih banyak landing page dengan karakter konten berbeda, menyertakan kondisi jaringan 3G/4G atau throttling tertentu, dan menggunakan infrastruktur hosting yang lebih terkontrol untuk mengurangi variabel eksternal. Bagi pengembang, pemilihan framework sebaiknya mempertimbangkan pola konten, kebutuhan interaktivitas, serta metrik performa yang paling relevan bagi

pengguna; response time jaringan tidak dapat diperlakukan sebagai satu-satunya ukuran performa.

DAFTAR PUSTAKA

- Anggraeni, O. S. I., Sugiarto, L., dan 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>
- Apache Software Foundation. 2024. *Apache JMeter - User's Manual*.
<https://jmeter.apache.org/usermanual/index.html>
- Astro. 2024. *Why Astro? Astro Docs*.
<https://docs.astro.build/en/concepts/why-astro/>
- Google. 2024. *About PageSpeed Insights*. Google for Developers.
<https://developers.google.com/speed/docs/insights/v5/about>
- Google dan SOASTA. 2017. *Business growth through mobile site speed*. Think with Google.
<https://business.google.com/uk/think/marketing-strategies/mobile-site-speed-importance/>
- Hermanto, R. R., dan Engel, M. M. 2025. “Analisis komparatif kinerja Next.js, Nuxt.js, dan Remix.js dalam implementasi server side rendering website.” *TIN: Terapan Informatika Nusantara*, 6(5), 450-463.
<https://doi.org/10.47065/tin.v6i5.8315>
- Khasanah, M. U., dan Suartana, I. M. 2024. “Perbandingan performa client side rendering pada implementasi static website (studi kasus Barokah RentCar).” *JINACS (Journal of Informatics and Computer Science)*, 6(2), 541-550.
- Lazuardy, M. F. S., dan Anggraini, D. 2022. “Modern front end web architectures with React.js and Next.js.” *International Research Journal of Advanced Engineering and Science*, 7(1).
- Permatasari, D. I., Ardani, M., Ma’ulfa, A. Y., Ilhami, N., Pratama, S. G., Astuti, S. R. D., dan Naufalita, N. W. 2020. “Pengujian aplikasi menggunakan metode load testing dengan Apache JMeter pada sistem informasi pertanian.” *JUSTIN (Jurnal Sistem dan Teknologi Informasi)*, 8(1).
- Stack Overflow. 2025. *Stack Overflow Developer Survey 2025: Technology*.
<https://survey.stackoverflow.co/2025/technology>
- State of JS. 2024. *Front-end frameworks*. State of JavaScript 2024. <https://2024.stateofjs.com/en-US/libraries/front-end-frameworks/>
- Zaidan, A. J., dan Suyatno, D. F. 2025. “Rendering performance analysis of Astro JS, Next JS, Nuxt JS, and SvelteKit frameworks using Google Lighthouse, PageSpeed Insight, and JMeter.” *Journal of Emerging Information System and Business Intelligence*, 6(1), 1-13.