# Penerapan Server Side Rendering untuk Meningkatkan Performa Website Kejaksaan Negeri X

Alfian Rahman Hidayatullah[1], I Made Suartana[2]

[1,2] Program Studi S1 Teknik Informatika, Universitas Negeri Surabaya
[1]alfian.20049@mhs.unesa.ac.id
[2]madesuartana@unesa.ac.id

***Abstrak*— Performa website menjadi salah satu aspek krusial dalam pengembangan aplikasi web modern, terutama bagi institusi pemerintahan yang menyampaikan informasi publik secara daring. Penelitian ini bertujuan untuk meningkatkan performa website Kejaksaan Negeri X dengan menerapkan teknik *Server Side Rendering* (SSR) menggunakan *framework* Next.js. SSR merupakan metode *rendering* halaman yang dilakukan di sisi server, sehingga HTML yang dikirim ke klien telah berisi konten lengkap dan siap ditampilkan, tanpa perlu menunggu proses rendering di sisi klien. Pendekatan penelitian yang digunakan adalah kuantitatif eksperimental, dengan metode pengujian performa menggunakan *Lighthouse Chrome DevTools*. *Website* versi lama yang dikembangkan dengan PHP dibandingkan dengan versi baru hasil pengembangan menggunakan SSR pada Next.js. Parameter pengujian meliputi *First Contentful Paint* (FCP), *Largest Contentful Paint* (LCP), *Total Blocking Time* (TBT), *Cumulative Layout Shift* (CLS), dan *Speed Index*. Hasil pengujian menunjukkan bahwa rata-rata skor performa *website* meningkat dari 50 (versi PHP) menjadi 89,4 (versi SSR). Selain itu, terjadi penurunan signifikan pada nilai LCP dan TBT, serta peningkatan kecepatan dan stabilitas tampilan halaman. Temuan ini menunjukkan bahwa SSR secara efektif dapat memperbaiki pengalaman pengguna (*user experience*) dan mendukung pengindeksan mesin pencari (SEO). Oleh karena itu, SSR direkomendasikan sebagai pendekatan strategis dalam pengembangan aplikasi web publik yang membutuhkan performa tinggi dan visibilitas digital yang optimal.**

*Kata Kunci*— ***server side rendering*, next.js, performa *website*, SEO, *user experience***

## I. PENDAHULUAN

Perkembangan teknologi informasi telah mengalami kemajuan yang sangat pesat dalam beberapa dekade terakhir. Berdasarkan survei terbaru dari Asosiasi Penyelenggara Jasa Internet Indonesia (APJII), pengguna internet di Indonesia pada tahun 2024 telah mencapai angka 79,5%, mengalami peningkatan sebesar 1,4% dari tahun sebelumnya [1]. Perkembangan ini mengindikasikan bahwa kebutuhan akan akses informasi yang cepat dan efisien melalui internet terus meningkat seiring waktu.

Sejalan dengan peningkatan jumlah pengguna internet, tuntutan terhadap performa sebuah *website* pun semakin tinggi. Pengguna kini mengharapkan pengalaman yang cepat, responsif, dan efisien saat mengakses layanan berbasis web. *Website* yang lambat dalam memuat halaman atau tidak optimal dalam performanya dapat mengakibatkan buruknya pengalaman pengguna (*user experience*), bahkan berpotensi menyebabkan pengguna meninggalkan situs sebelum informasi tersampaikan secara utuh. Permasalahan tersebut sangat relevan pada website media informasi Kejaksaan Negeri X, yang berdasarkan hasil pengujian awal menunjukkan nilai benchmarking rendah pada beberapa parameter penting seperti *performance, accessibility, best practices*, dan SEO. Selain itu, kecepatan *loading* yang lambat serta waktu render halaman yang tinggi turut memengaruhi kenyamanan dan kepuasan pengguna ketika mengakses informasi melalui *website* tersebut.

Untuk mengatasi permasalahan performa ini, diperlukan solusi teknologi yang mampu meningkatkan efisiensi dalam proses pemuatan halaman serta memperbaiki responsivitas secara keseluruhan. Salah satu pendekatan yang banyak digunakan dalam pengembangan web modern adalah *Server Side Rendering* (SSR). SSR adalah teknik *rendering* halaman web yang dilakukan di sisi server, sehingga HTML yang sudah dirender sepenuhnya dikirimkan ke klien. Hal ini memungkinkan konten untuk ditampilkan lebih cepat di *browser* pengguna, bahkan pada perangkat dengan sumber daya terbatas [5]. Framework seperti Next.js yang berbasis React telah mengintegrasikan fitur SSR untuk mendukung pembuatan aplikasi web dengan performa tinggi dan dukungan SEO yang lebih baik.

Dengan menerapkan teknologi SSR menggunakan Next.js pada website Kejaksaan Negeri X, penelitian ini bertujuan untuk mengoptimalkan parameter performa seperti *First Contentful Paint* (FCP), *Largest Contentful Paint* (LCP), dan *Time to Interactive* (TTI), yang diukur melalui alat bantu *Lighthouse Chrome DevTools*. Diharapkan bahwa hasil penelitian ini tidak hanya mampu meningkatkan skor performa secara signifikan, tetapi juga memberikan wawasan tentang keunggulan SSR dibandingkan metode rendering tradisional seperti *Client Side Rendering* (CSR) maupun arsitektur berbasis PHP.

Tujuan dari penelitian ini adalah untuk melakukan optimalisasi performa website media informasi Kejaksaan Negeri X dengan menerapkan *Server Side Rendering* (SSR) menggunakan *framework* Next.js. Pendekatan ini dipilih karena kemampuan SSR dalam mempercepat waktu pemuatan halaman dan merender konten secara langsung di sisi server sebelum dikirim ke klien. Penelitian ini juga bertujuan untuk membandingkan performa antara *website* berbasis PHP yang merupakan versi awal, dengan *website* hasil pengembangan menggunakan SSR pada Next.js. Fokus perbandingan ini mengarah pada parameter performa inti seperti *First*

*Contentful Paint* (FCP), *Largest Contentful Paint* (LCP), dan *Time to Interactive* (TTI) yang semuanya diuji menggunakan Lighthouse Chrome DevTools [2] [3] [5].

Adapun manfaat yang diharapkan dari penelitian ini terbagi dalam dua aspek. Pertama, bagi pengembang (developer), penelitian ini dapat menjadi acuan dalam menerapkan SSR untuk meningkatkan kualitas aplikasi web, khususnya dari sisi performa dan SEO. Penelitian ini juga memberikan wawasan praktis tentang penerapan *framework* Next.js dan cara melakukan pengujian performa secara terukur menggunakan alat bantu modern seperti *Lighthouse*. Kedua, bagi pengguna (*user*), hasil penelitian ini diharapkan mampu memberikan pengalaman yang lebih baik dalam mengakses *website* yang cepat, responsif, dan stabil, khususnya dalam konteks layanan publik seperti media informasi kejaksaan yang memuat banyak konten dinamis dan informatif.

Dengan peningkatan performa melalui SSR, website Kejaksaan Negeri X diharapkan dapat memberikan layanan informasi hukum dan publik yang lebih efisien dan inklusif, sekaligus menjadi contoh implementasi teknologi web modern pada instansi pemerintah.

## II. METODE PENELITIAN

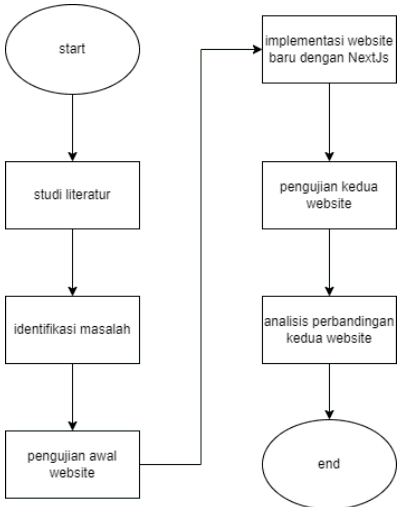
### A. Format Penulisan Jenis dan Pendekatan Penelitian

Penelitian ini menggunakan pendekatan kuantitatif dengan metode eksperimen untuk mengukur dan membandingkan performa dua versi *website* Kejaksaan Negeri X, yaitu versi lama berbasis PHP dan versi baru yang dikembangkan menggunakan Next.js dengan teknik *Server Side Rendering* (SSR). Pendekatan kuantitatif dipilih karena penelitian ini berfokus pada data numerik berupa hasil skor pengujian performa dari masing-masing *website*, yang dapat dianalisis secara objektif dan terukur.

Metode eksperimen digunakan untuk melakukan pengujian berulang terhadap kedua versi *website* dalam kondisi dan lingkungan yang dikendalikan, sehingga perbandingan hasil dapat diukur secara adil dan akurat. Eksperimen dilakukan dengan menjalankan pengujian performa menggunakan *Lighthouse Chrome DevTools*, sebuah alat audit otomatis dari Google yang digunakan untuk menilai kualitas halaman web berdasarkan parameter-parameter utama, seperti *First Contentful Paint* (FCP), *Largest Contentful Paint* (LCP), *Speed Index, Time to Interactive* (TTI), *Total Blocking Time* (TBT), dan *Cumulative Layout Shift* (CLS).

Penggunaan pendekatan ini memungkinkan peneliti untuk menilai dampak penerapan SSR terhadap peningkatan performa *website* secara langsung melalui metrik-metrik performa yang telah distandarkan oleh industri pengembangan web. Dengan membandingkan hasil pengujian dari dua pendekatan pengembangan web tersebut, penelitian ini bertujuan untuk memberikan bukti empiris mengenai efektivitas *Server Side Rendering* dalam meningkatkan kualitas teknis dan pengalaman pengguna pada aplikasi web modern.

Untuk menggambarkan urutan proses penelitian yang dilakukan, berikut ditampilkan diagram alur metode penelitian:




Gbr.1 Alur Metode Penelitian

Gbr 1 menjelaskan tahapan-tahapan utama dalam pelaksanaan penelitian ini, dimulai dari studi literatur, identifikasi masalah, hingga pengujian performa dan analisis perbandingan antara *website* lama dan *website* baru.

### B. Studi Literatur dan Identifikasi Masalah

Studi literatur dalam penelitian ini dilakukan untuk memperoleh pemahaman yang mendalam mengenai konsep-konsep yang berkaitan dengan performa *website*, teknik *rendering*, serta perbandingan antara metode *Server Side Rendering* (SSR) dan *Client Side Rendering* (CSR). Literatur yang digunakan meliputi jurnal, artikel ilmiah, skripsi terdahulu, serta dokumentasi teknis yang relevan dengan pengembangan aplikasi web berbasis JavaScript, khususnya *framework* Next.js.

Penelitian oleh Iskandar et al. (2020) membandingkan performa website yang dikembangkan dengan PHP dan JavaScript melalui audit Google, menunjukkan bahwa rendering di sisi server memberikan hasil yang lebih baik dalam metrik-metrik performa seperti *First Contentful Paint* dan *Speed Index* [3]. Penelitian lainnya oleh Geovanny (2022) menggunakan ReactJS di sisi klien dan Next.js di sisi server untuk merancang dua aplikasi web dan menemukan bahwa penerapan SSR secara signifikan meningkatkan kecepatan muat halaman [2]. Selain itu, penelitian oleh Sinulingga (2024) menunjukkan bahwa implementasi SSR pada sistem absensi mahasiswa mampu mempercepat waktu render halaman dan meningkatkan pengalaman pengguna dibandingkan pendekatan CSR [5].

Dari literatur tersebut dapat disimpulkan bahwa SSR berperan penting dalam mempercepat waktu respon website dan meningkatkan skor performa pada berbagai alat uji seperti *Lighthouse*. *Lighthouse* sendiri merupakan alat *open-source* yang mengukur kinerja *website* dari berbagai aspek, termasuk performa, aksesibilitas, SEO, dan *best practices* [4].

Berdasarkan studi literatur dan kondisi nyata yang dihadapi, peneliti mengidentifikasi bahwa website Kejaksaan Negeri X mengalami masalah serius dalam hal kecepatan muat halaman, stabilitas layout, serta rendahnya skor performa berdasarkan hasil audit *Lighthouse* awal. Halaman utama situs tersebut lambat dalam merender konten utama dan

memiliki skor *performance* yang tergolong rendah, sehingga mempengaruhi kenyamanan pengguna dalam mengakses informasi.

Permasalahan ini diperparah dengan struktur website lama yang masih menggunakan arsitektur berbasis PHP, yang belum mengadopsi pendekatan rendering modern seperti Server Side Rendering (SSR). Arsitektur tradisional tersebut menyebabkan proses pemuatan halaman menjadi lambat, kurang efisien, dan tidak optimal dalam hal SEO maupun pengalaman pengguna secara keseluruhan. Hal ini menjadi sangat krusial terutama pada situs layanan publik yang memerlukan akses informasi yang cepat dan stabil. Oleh karena itu, penelitian ini dilakukan dengan pendekatan eksperimental yang bertujuan untuk menguji efektivitas implementasi SSR dalam meningkatkan performa teknis website. Dengan menggunakan framework Next.js sebagai teknologi utama, penelitian ini tidak hanya berfokus pada pengurangan waktu pemuatan halaman, tetapi juga pada peningkatan parameter performa utama seperti *First Contentful Paint* (FCP), *Largest Contentful Paint* (LCP), dan *Total Blocking Time* (TBT) [6] [7]. Diharapkan hasil dari penerapan SSR ini dapat menyelesaikan permasalahan teknis yang ada, serta memberikan pengalaman pengguna yang lebih responsif dan ramah mesin pencari. Temuan serupa juga telah dibuktikan dalam penelitian terdahulu yang menunjukkan bahwa penerapan SSR secara signifikan mampu meningkatkan efisiensi loading serta kualitas interaksi pengguna terhadap sistem berbasis web [8].

C. Spesifikasi Lingkungan Pengujian

Lingkungan pengujian dalam penelitian ini dirancang untuk mendukung proses pengembangan dan evaluasi performa kedua versi *website*, baik yang lama (berbasis PHP) maupun yang baru (berbasis Next.js dengan *Server Side Rendering*).

Spesifikasi perangkat keras (*hardware*) yang digunakan adalah sebagai berikut:

- Prosesor: Intel Core i7-11800H Generasi ke-11 @ 2.30 GHz, 8 Core, 16 Threads
- Memori: 16 GB DDR4 RAM
- Penyimpanan: SSD 500 GB
- Sistem Operasi: Windows 11 64-bit

Perangkat lunak (*software* dan *tools*) yang digunakan meliputi:

- *Text Editor*: Visual Studio Code
- *Browser*: Google Chrome versi terbaru
- Alat Uji: Lighthouse Chrome DevTools
- Node.js dan NPM: Untuk menjalankan *framework* Next.js
- *Framework*:
  - PHP (versi default XAMPP untuk *website* lama)
  - Next.js (latest stable version, berbasis React)

Lingkungan ini dipilih untuk mencerminkan kondisi pengembangan web modern dan memungkinkan dilakukannya evaluasi performa secara konsisten dan komparatif antar *platform*.

D. Prosedur Penelitian

Prosedur penelitian ini mengikuti alur eksperimental dengan langkah-langkah sistematis yang terdiri atas empat tahap utama sebagai berikut:

1) Pengujian Awal Website Lama Berbasis PHP

Pengujian dilakukan terhadap website Kejaksaan Negeri X versi lama yang dibangun menggunakan PHP. Evaluasi performa dilakukan menggunakan Lighthouse Chrome DevTools, dengan fokus pada metrik performa seperti FCP, LCP, TTI, dan Speed Index. Pengujian ini dilakukan sebanyak lima kali untuk memastikan konsistensi hasil.

2) Pengembangan Website Baru dengan Next.js dan Penerapan SSR

Tahap ini melibatkan proses rekonstruksi website menggunakan framework Next.js dengan penerapan *Server Side Rendering* (SSR). Pengembangan dilakukan dengan pendekatan modular sesuai arsitektur Next.js, dan konten utama website ditampilkan melalui mekanisme *getServerSideProps*() untuk merender halaman langsung di sisi server. Setelah pengembangan selesai, *website* dihosting secara lokal untuk pengujian. Pendekatan SSR dipilih karena terbukti mampu memberikan waktu muat halaman yang lebih cepat dibandingkan *Client Side Rendering*, serta menghasilkan performa yang lebih stabil dan ringan ketika diuji pada berbagai peramban modern [9]. Selain itu, penggunaan SSR juga direkomendasikan dalam pengembangan website dinamis yang memerlukan pembaruan konten secara real-time tanpa membebani sisi klien secara berlebihan.

3) Pengujian Performa *Website* Baru

*Website* baru yang telah dikembangkan kemudian diuji menggunakan alat yang sama (Lighthouse) dan dalam kondisi lingkungan yang identik. Sama seperti pengujian pertama, evaluasi dilakukan sebanyak lima kali untuk menjaga validitas hasil dan memperoleh rata-rata performa secara akurat.

4) Analisis dan Perbandingan Hasil Performa

Setelah kedua *website* diuji, hasil dari masing-masing metrik dibandingkan secara kuantitatif. Analisis ini dilakukan dengan menyajikan data skor dalam bentuk tabel dan grafik untuk melihat peningkatan atau penurunan performa secara spesifik pada setiap aspek. Fokus utama analisis adalah mengukur dampak implementasi SSR terhadap waktu muat halaman dan stabilitas tampilan.

Dengan prosedur ini, penelitian dapat menyajikan hasil empiris atas efektivitas *Server Side Rendering* dalam meningkatkan performa aplikasi web instansi publik.

E. Alat dan Teknik Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan dengan menggunakan Lighthouse Chrome DevTools, sebuah alat audit otomatis yang dikembangkan oleh Google untuk mengukur kualitas teknis dan performa halaman web. Lighthouse sangat cocok digunakan dalam penelitian ini karena dapat memberikan penilaian kuantitatif berdasarkan berbagai metrik yang telah distandarkan dalam pengembangan aplikasi web modern.

Pengujian dilakukan langsung melalui Google Chrome, dengan langkah-langkah sebagai berikut:

1) Buka halaman utama *website* yang akan diuji (baik versi PHP maupun Next.js).
2) Buka Chrome DevTools dengan klik kanan → *Inspect* → tab *Lighthouse*.
3) Pilih kategori *Performance* dan target perangkat *Desktop*.
4) Klik tombol *Generate report*.
5) Ulangi pengujian sebanyak lima kali untuk masing-masing versi *website* guna memperoleh data yang stabil dan representatif.

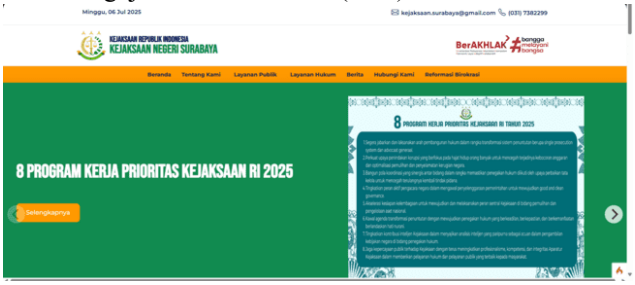
Metrik-metrik utama yang digunakan sebagai acuan dalam pengambilan data adalah sebagai berikut:

- *First Contentful Paint* (FCP): Mengukur waktu yang dibutuhkan untuk merender elemen visual pertama pada halaman.
- *Largest Contentful Paint* (LCP): Mengukur waktu pemuatan elemen terbesar yang terlihat di layar pengguna.
- *Time to Interactive* (TTI): Mengukur waktu hingga halaman benar-benar dapat digunakan dan merespons interaksi pengguna.
- *Total Blocking Time* (TBT): Menilai waktu tunda total akibat skrip yang memblokir interaksi pengguna.
- *Cumulative Layout Shift* (CLS): Mengukur stabilitas visual halaman dengan menghitung jumlah pergeseran layout yang tidak terduga.

Kelima metrik ini merupakan indikator utama dalam menilai pengalaman pengguna (*user experience*) dan efisiensi teknis halaman web. Data yang diperoleh dari pengujian Lighthouse kemudian digunakan dalam proses analisis perbandingan performa antara *website* lama dan baru, dengan tujuan mengidentifikasi sejauh mana penerapan SSR melalui Next.js dapat meningkatkan kualitas performa secara signifikan. Penggunaan metrik Lighthouse sebagai alat ukur performa juga sejalan dengan praktik terbaik dalam penelitian sebelumnya yang menilai efektivitas berbagai *framework* berbasis SSR secara menyeluruh, termasuk Next.js [10].

## III. Hasil dan Pembahasan

### A. Hasil Pengujian Website Lama (PHP)




Gbr 2. Landing Page Website Lama

Pengujian awal dilakukan terhadap website Kejaksaan Negeri X versi lama sesuai yang terlihat pada gbr 2. *Website* Kejaksaan Negeri X ini dibangun menggunakan bahasa pemrograman PHP. Tujuan dari pengujian ini adalah untuk memperoleh data performa dasar sebelum implementasi *Server Side Rendering* (SSR) dilakukan. Pengujian dilakukan pada halaman utama menggunakan *Lighthouse Chrome DevTools*, dengan fokus pada parameter performa.

Pengujian dilakukan sebanyak lima kali untuk menjaga akurasi data. Tabel 1 adalah ringkasan skor performa yang diperoleh:

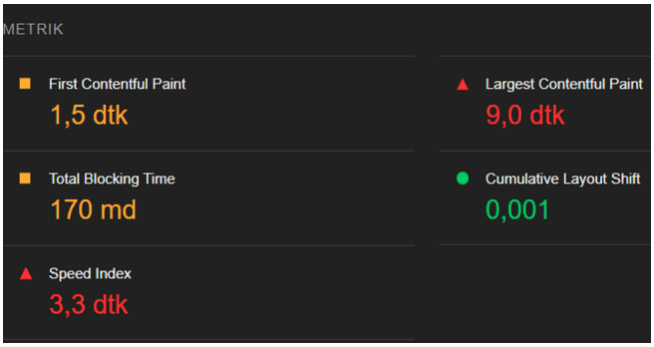
TABEL I
HASIL MATRIKS PERFOMANCE WEBSITE LAMA

| Skor | FCP | LCP | TBT | CLS | Speed Index |
|---|---|---|---|---|---|
| 47 | 1,9 dtk | 9,1 dtk | 320 md | 0,001 | 3,1 dtk |
| 50 | 1,6 dtk | 9,2 dtk | 320 md | 0,001 | 2,8 dtk |
| 58 | 1,5 dtk | 9,0 dtk | 170 md | 0,001 | 3,3 dtk |
| 49 | 1,7 dtk | 9,1 dtk | 330 md | 0,001 | 2,8 dtk |
| 46 | 1,4 dtk | 9,2 dtk | 430 md | 0,001 | 2,8 dtk |

Keterangan:

- FCP = *First Contentful Paint*
- LCP = *Largest Contentful Paint*
- TBT = *Total Blocking Time*
- CLS = *Cumulative Layout Shift*

Rata-rata skor performa dari kelima pengujian tersebut adalah 50, yang menandakan performa *website* berada pada kategori rendah. Nilai LCP yang sangat tinggi pada semua pengujian menunjukkan bahwa waktu tampil elemen terbesar di halaman berlangsung sangat lambat, memperburuk pengalaman pengguna. TBT juga tergolong tinggi, yang menandakan adanya banyak proses JavaScript yang memblokir interaksi awal pengguna dengan halaman.




Gbr 3. Hasil Matriks ke 3 Website Lama

Gbr 3 merupakan hasil dari metrik pengujian ke 3 website lama dan tabel dibawah ini adalah data metrik utama dari salah satu pengujian representatif:

TABEL III
DATA MATRIK PENGUJIAN KE-3 WEBSITE LAMA

| Matrik | Nilai | Interpretasi |
|---|---|---|
| *First Contentful Paint* (FCP) | 1,5 detik | Cukup cepat namun masih bisa ditingkatkan dengan optimasi elemen awal. |
| *Largest Contentful Paint* (LCP) | 9,0 detik | Sangat lambat; elemen besar lambat ditampilkan, memperburuk UX. |
| *Total Blocking Time* (TBT) | 170 md | Masih cukup tinggi; perlu optimasi JavaScript untuk mengurangi waktu tunda. |
| *Cumulative* | 0,001 | Sangat baik; *layout* stabil, tidak ada |

| | | |
|---|---|---|
| *Layout Shift* (CLS) | | pergeseran elemen besar. |
| *Speed Index* | 3,3 detik | Relatif lambat; konten lambat muncul secara keseluruhan. |

Tabel 2 tersebut menunjukkan bahwa kelemahan utama website lama terletak pada LCP yang sangat tinggi dan *Speed Index* yang lambat. Artinya, pengguna harus menunggu cukup lama sebelum konten utama *website* muncul secara utuh. Hal ini secara langsung berdampak pada pengalaman pengguna, terlebih jika pengguna mengakses melalui koneksi internet lambat atau perangkat dengan spesifikasi rendah.

Faktor penyebab performa rendah antara lain:

- Tidak adanya teknik rendering sisi server; semua konten dimuat di klien secara bertahap.
- Kurangnya optimasi pada gambar dan elemen multimedia.
- Tidak adanya penggunaan *cache* atau teknik pemuatan asinkron.
- Struktur HTML dan CSS yang tidak efisien serta penggunaan JavaScript *blocking*.

Dengan kondisi tersebut, *website* tidak hanya lambat secara teknis tetapi juga berisiko ditinggalkan oleh pengguna, serta sulit bersaing dalam hasil pencarian mesin pencari (SEO).

B. Implementasi *Server Side Rendering* (SSR) dengan Next.js

Setelah melakukan pengujian awal pada *website* lama berbasis PHP, tahap berikutnya adalah melakukan pengembangan ulang website Kejaksaan Negeri X menggunakan *framework* Next.js dengan pendekatan *Server Side Rendering* (SSR). Pengembangan ini bertujuan untuk meningkatkan performa teknis, khususnya waktu muat halaman dan interaktivitas, serta memperbaiki skor performa pada pengujian Lighthouse.

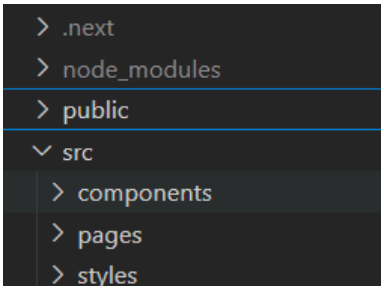
1) Proses Pengembangan *Website* Baru

Website dibangun kembali dari awal dengan menggunakan Next.js, yang merupakan *framework* berbasis React.js dan mendukung fitur SSR secara *native*. Pengembangan dilakukan secara modular dengan membagi setiap bagian halaman menjadi komponen, sehingga memudahkan pengelolaan struktur dan pemeliharaan kode.

Sumber data yang digunakan dalam halaman *website* disimulasikan sebagai data dinamis dan ditampilkan melalui mekanisme SSR, sehingga konten dapat ditampilkan secara lengkap pada saat halaman dimuat, tanpa perlu menunggu proses *rendering* di sisi klien.

2) Struktur Proyek Next.js

Struktur proyek Next.js yang digunakan mengikuti konvensi standar berikut:




Gbr 4. Struktur Web

Pada Gbr 4 dilihatkan bahwa folder *pages*/ digunakan sebagai representasi rute halaman, di mana setiap file secara otomatis dianggap sebagai sebuah *endpoint*. Komponen umum seperti *header, footer*, dan elemen konten dikelola dalam folder *components*/ untuk meningkatkan reusabilitas dan keteraturan kode.

3) Konfigurasi SSR menggunakan *getServerSideProps*()

SSR dalam Next.js diaktifkan melalui penggunaan fungsi khusus bernama *getServerSideProps*(). Fungsi ini dipanggil setiap kali halaman diminta oleh pengguna, dan proses pengambilan serta pengolahan data dilakukan di sisi server, bukan di *browser*.

Contoh implementasi SSR:

```
import { useState } from "react";
import Button from "../components/Button/Button";
import Header from "../components/Layout/Header";
import { Navbar } from "../components/Layout/Navbar";
import Footer from "../components/Layout/Footer";
import { Carousel } from "flowbite-react";
import Image from "next/image";
export async function getServerSideProps({ req, res }) {
  res.setHeader(
    "Cache-Control",
    "public, s-maxage=30, stale-while-revalidate=59"
  );

  try {
    const res = await fetch("https://kejari-surabaya.my.id/api/berita");

    if (!res.ok) {
      throw new Error(`HTTP error! status: ${res.status}`);
    }
          You, last week • Uncommitted changes
    const result = await res.json();
    return {
      props: { berita: result.data || [] }, // Passing `result.data` sebagai `berita`
    };
  } catch (error) {
    console.error("Error fetching data:", error.message);
    return {
      props: { berita: [], services: [] }, // Return array kosong jika ada error
    };
  }
}
```

Gbr 5. Implementasi SSR di Landing Page

Pada Gbr 5 dijelaskan bahwa implementasi *Server Side Rendering* (SSR) dilakukan dengan memanfaatkan fungsi khusus dari Next.js yaitu *getServerSideProps*(). Fungsi ini dijalankan secara otomatis di sisi server setiap kali pengguna mengakses halaman. Dengan demikian, proses pengambilan data dan rendering konten tidak lagi bergantung pada proses asinkron di sisi klien seperti *useEffect* + *fetch*. Sebagai hasilnya, HTML yang dikirim ke browser sudah lengkap berisi data aktual.

Berikut adalah penjelasan dari proses dan blok kode utama yang terlibat dalam implementasi SSR pada proyek ini:

1) *getServerSideProps*()

Merupakan fungsi utama dari Next.js untuk melakukan rendering halaman secara *server-side*. Fungsi ini

menggantikan kebutuhan untuk memuat data di sisi klien, sehingga HTML yang dikirim dari server telah berisi konten penuh yang siap ditampilkan.

2) *res.setHeader*(...)

Baris ini digunakan untuk mengatur header HTTP *cache-control*, dengan tujuan mengatur durasi cache konten pada *Content Delivery Network* (CDN). Dengan pengaturan ini, halaman tidak perlu selalu dirender ulang jika belum kedaluwarsa, yang tentunya berdampak pada peningkatan performa.

3) *fetch*(...) ke API eksternal

Fungsi ini digunakan untuk mengambil data dari endpoint berita yang tersedia di: https://kejari-X.my.id/api/berita. Proses ini dilakukan langsung di server, sehingga saat halaman dikirim ke klien, konten berita sudah terisi penuh. Pemeriksaan res.ok memastikan bahwa hanya respons dengan status 200–299 yang dianggap berhasil.

4) *Error Handling*

Untuk menjaga stabilitas halaman, blok *try–catch* diterapkan agar ketika terjadi kesalahan dalam proses pengambilan data, halaman tetap direnderer tanpa konten (*fallback* data kosong). Dengan cara ini, user tetap dapat mengakses halaman tanpa mengalami *blank screen*.

5) *Return props*

Setelah data berhasil diambil, hasil dari *await res.json*() akan dikonversi menjadi objek JavaScript dan dikirim ke komponen React melalui properti *props*. Data tersebut akan diakses sebagai berita oleh halaman yang bersangkutan, khususnya pada *landing page*.

6) Keunggulan SSR pada Implementasi Ini:

- SEO Friendly: Mesin pencari seperti Google dapat langsung membaca HTML yang telah ter-render lengkap, tanpa bergantung pada JavaScript.
- *Performance*: Konten dapat tampil lebih cepat karena tidak perlu menunggu proses *fetch* dan render di klien.
- *User Experience* Konsisten: Mekanisme *fallback* membuat halaman tetap stabil meskipun API mengalami gangguan, sehingga tidak terjadi layar kosong atau *error* tampilan.

Dengan konfigurasi ini, *website* Kejaksaan Negeri X mampu menyajikan informasi secara cepat, stabil, dan ramah mesin pencari — sebuah peningkatan signifikan dibandingkan arsitektur PHP sebelumnya.

4) Penjelasan Alur SSR dan Manfaat Teknis

Secara alur, proses SSR dapat digambarkan sebagai berikut:

- Pengguna mengakses halaman dari browser.
- Permintaan dikirim ke server Next.js.
- Server menjalankan fungsi *getServerSideProps*() untuk mengambil dan mengolah data.
- Server merender halaman HTML lengkap dengan kontennya.
- HTML dikirim ke browser dan langsung ditampilkan ke pengguna.

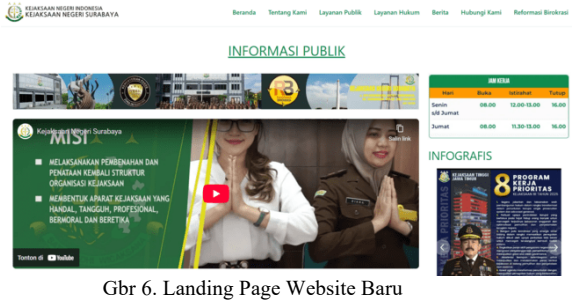
Manfaat teknis dari pendekatan SSR ini antara lain:

- Performa awal lebih cepat karena konten sudah tersedia saat halaman dimuat.
- SEO-friendly karena mesin pencari dapat mengindeks halaman lengkap tanpa perlu menjalankan JavaScript.
- Cocok untuk konten dinamis, seperti berita atau informasi hukum yang perlu diperbarui secara real time.
- Pengalaman pengguna lebih baik, karena tampilan halaman lebih stabil dan interaktif sejak awal.

Dengan arsitektur ini, *website* Kejaksaan Negeri X tidak hanya menjadi lebih cepat, tetapi juga lebih efisien dalam menyajikan informasi kepada masyarakat.

C. Hasil Pengujian *Website* Baru (Next.js dengan SSR)

Setelah proses pengembangan *website* baru dengan *framework* Next.js dan penerapan *Server Side Rendering* (SSR) selesai yang dapat dilihat pada gbr 6 dibawah ini, tahap selanjutnya adalah melakukan pengujian performa untuk menilai efektivitas implementasi tersebut. Pengujian dilakukan dengan alatyang sama yaitu *Lighthouse Chrome DevTools*, dalam kondisi lingkungan yang identik dengan pengujian sebelumnya. Tujuannya adalah memperoleh perbandingan yang adil dan objektif antara website versi lama (PHP) dan versi baru (Next.js + SSR).




Gbr 6. Landing Page Website Baru

1) Skor Pengujian Lighthouse (5 Kali Uji)

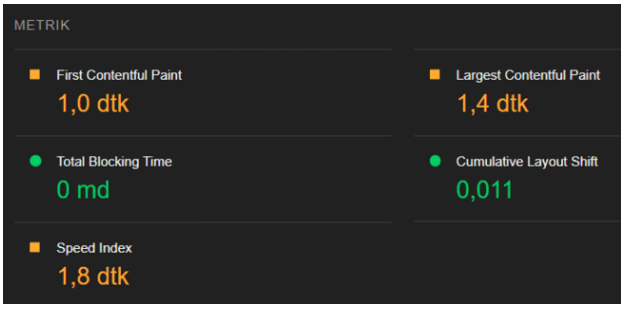
Pengujian dilakukan sebanyak lima kali pada halaman utama *website* hasil SSR. Hasil skor performa dari setiap pengujian disajikan dalam tabel berikut:

TABEL IIIII
HASIL MATRIKS PERFOMANCE WEBSITE BARU

| Skor | FCP | LCP | TBT | CLS | Speed Index |
|---|---|---|---|---|---|
| 92 | 1,0 dtk | 1,4 dtk | 0 md | 0,011 | 1,8 dtk |
| 87 | 0,8 dtk | 1,5 dtk | 20 md | 0,019 | 3,1 dtk |
| 88 | 0,9 dtk | 1,9 dtk | 40 md | 0,019 | 1,8 dtk |
| 93 | 1,0 dtk | 1,2 dtk | 0 md | 0,019 | 1,8 dtk |
| 87 | 1,0 dtk | 1,9 dtk | 30 md | 0,019 | 1,9 dtk |

Dari Tabel 3 dapat dilihat bahwa rata-rata skor performa dari pengujian ini adalah 90, yang menunjukkan peningkatan signifikan dibandingkan rata-rata skor *website* lama yang hanya 50. Berdasarkan skala warna dari Lighthouse, skor ini masuk dalam kategori baik (hijau: 90–100), yang mengindikasikan bahwa *website* telah memiliki performa optimal dari sisi kecepatan, efisiensi kode, dan pengalaman pengguna.

2) Penjabaran Metrik Utama




Gbr 7. Landing Page Website Baru

Gbr 7 merupakan salah satu hasil pengujian yang dianggap representatif ditampilkan dalam tabel 4 dibawah ini untuk menampilkan nilai-nilai metrik performa (pengujian ke-1) secara detail:

TABEL IVV
DATA MATRIKS KE 1 WEBSITE BARU

| Matrik | Nilai | Interpretasi |
|---|---|---|
| *First Contentful Paint* (FCP) | 1,0 detik | Sangat baik; konten pertama tampil sangat cepat. |
| *Largest Contentful Paint* (LCP) | 1,4 detik | Baik; elemen terbesar dimuat secara efisien dan tidak mengganggu UX. |
| *Total Blocking Time* (TBT) | 0 ms | Optimal; tidak ada pemblokiran *rendering* oleh JavaScript. |
| *Cumulative Layout Shift* (CLS) | 0,011 | Stabil; *layout* tidak mengalami pergeseran visual saat pemuatan. |
| *Speed Index* | 1,8 detik | Cepat; halaman secara visual segera terlihat oleh pengguna. |

3) Bukti Peningkatan Performa dan Stabilitas Halaman

Dibandingkan dengan skor *website* lama berbasis PHP (rata-rata 50), website baru dengan SSR memperlihatkan peningkatan drastis. Waktu tampil konten besar (LCP) turun dari 9 detik menjadi maksimal 1,9 detik. Total *Blocking Time* hampir nol, yang menunjukkan eksekusi JavaScript yang efisien dan non-blokade. Selain itu, *layout* stabil (CLS < 0,02) mencegah elemen bergeser saat halaman dimuat, yang sangat berpengaruh pada pengalaman pengguna.

Performa ini menandakan keberhasilan SSR dalam meningkatkan efisiensi pemuatan halaman, kecepatan akses informasi, dan responsivitas situs terhadap interaksi pengguna.

D. Analisis Perbandingan *Website* Lama dan Baru

Setelah dilakukan lima kali pengujian pada masing-masing versi *website* (lama berbasis PHP dan baru berbasis Next.js dengan SSR), dilakukan analisis perbandingan terhadap skor dan metrik performa untuk mengevaluasi efektivitas implementasi SSR.

1) Tabel Komparatif Skor Performa

TABEL V
KOMPARATIF SKOR PERFORMA WEBSITE

| Versi Website | Rata-rata Skor Performa | Kategori Lighthouse |
|---|---|---|
| PHP (lama) | 50 | Sedang (oranye) |
| Next.js + SSR (baru) | 89,4 | Baik (hijau) |

Dari Tabel 5 hasilnya menunjukkan peningkatan hampir dua kali lipat dalam skor performa rata-rata setelah penerapan SSR, dari skor 50 menjadi 89,4. Hal ini menandakan peningkatan kualitas secara keseluruhan dari segi kecepatan, stabilitas, dan kesiapan interaktif halaman.

2) Perbandingan Nilai Matrik Utama (Rata-Rata)

TABEL VI
PERBANDINGAN NILAI MATRIKS RATA-RATA

| Metrik | Website PHP (lama) | Website Next.js + SSR (baru) | Perbandingan |
|---|---|---|---|
| *First Contentful Paint* (FCP) | 1,62 detik | 0,94 detik | Lebih cepat 0,68 detik |
| *Largest Contentful Paint* (LCP) | 9,12 detik | 1,58 detik | Lebih cepat 7,54 detik (sangat signifikan) |
| *Total Blocking Time* (TBT) | 314 ms | 18 ms | Berkurang drastis (lebih responsif) |
| *Cumulative Layout Shift* (CLS) | 0,001 | 0,017 | Masih dalam kategori stabil (<0,1) |
| *Speed Index* | 2,96 detik | 2,08 detik | Pemuatan halaman visual lebih cepat |

Tabel 6 menampilkan ringkasan perbandingan antara website baru yang menggunakan teknologi Next.js dan SSR dengan *website* lama yang menggunakan teknologi bahasa pemrograman PHP.

3) Evaluasi Peningkatan *User Experience* dan Optimasi SEO

Dari data perbandingan di atas, dapat disimpulkan bahwa implementasi SSR memberikan peningkatan signifikan terhadap hampir seluruh metrik kinerja *website*:

- FCP dan LCP yang jauh lebih cepat menunjukkan bahwa pengguna kini dapat melihat konten utama lebih awal, yang meningkatkan kesan cepat dan profesional pada sistem.
- TBT yang menurun drastis menunjukkan bahwa website lebih responsif terhadap input pengguna karena proses JavaScript tidak lagi menghambat interaksi.
- CLS tetap rendah dan stabil, menunjukkan bahwa tata letak halaman tidak berubah-ubah saat dimuat, yang penting untuk kenyamanan visual.

- Speed Index yang lebih rendah mencerminkan peningkatan visual loading secara keseluruhan.

Dari sisi SEO (*Search Engine Optimization*), *website* baru menjadi lebih optimal karena:

- Konten sudah ter-render di server dan dikirim sebagai HTML penuh, yang langsung terbaca oleh crawler mesin pencari.
- Kecepatan *loading* meningkat, yang merupakan faktor penting dalam ranking Google.
- Struktur halaman lebih bersih dan efisien, mendukung indeksasi konten secara maksimal.

Dengan semua peningkatan tersebut, implementasi SSR pada Next.js terbukti memberikan dampak positif yang nyata terhadap kinerja teknis, pengalaman pengguna, dan kualitas SEO dari *website* Kejaksaan Negeri X.

E. Implikasi dan Rekomendasi

1) Manfaat Penerapan SSR bagi Sistem Informasi Publik

Hasil dari penelitian ini menunjukkan bahwa penerapan *Server Side Rendering* (SSR) melalui *framework* Next.js dapat memberikan peningkatan performa yang sangat signifikan untuk *website* institusi publik. Kecepatan pemuatan konten dan stabilitas tampilan halaman menjadi lebih baik, yang sangat penting untuk meningkatkan aksesibilitas informasi hukum dan pelayanan publik secara digital.

Dalam konteks Kejaksaan Negeri X, manfaat yang diperoleh mencakup:

- Penyampaian informasi lebih cepat kepada masyarakat, termasuk berita, pengumuman, dan informasi hukum.
- Tampilan halaman lebih stabil dan ramah pengguna, terutama untuk pengguna yang mengakses melalui perangkat dengan spesifikasi rendah atau jaringan lambat.
- Kinerja *website* lebih siap untuk diakses secara bersamaan oleh banyak pengguna, tanpa penurunan performa yang signifikan.

2) Rekomendasi untuk Institusi Pemerintah Lainnya

Melihat keberhasilan penerapan SSR pada studi kasus ini, peneliti merekomendasikan agar institusi pemerintah lain yang mengelola layanan atau informasi publik mulai mempertimbangkan penggunaan SSR dalam pengembangan situs web mereka. Saran implementatif yang dapat diambil antara lain:

- Migrasi bertahap dari platform lama (misalnya PHP murni atau CMS tradisional) ke *framework* modern seperti Next.js dengan SSR.
- Optimasi performa dan SEO sejak tahap awal pengembangan, bukan hanya setelah website online.
- Penggunaan pendekatan SSR untuk halaman-halaman yang bersifat publik, statis, atau membutuhkan visibilitas tinggi di mesin pencari, seperti halaman berita, informasi layanan, dan kebijakan.

Langkah-langkah ini tidak hanya akan meningkatkan efisiensi teknis, tetapi juga membantu pemerintah dalam mewujudkan layanan digital yang inklusif dan cepat diakses oleh seluruh masyarakat.

3) Catatan Batasan dan Peluang Penelitian Lanjutan

Penelitian ini memiliki beberapa batasan, antara lain:

- Skala pengujian terbatas pada satu studi kasus, yaitu *website* Kejaksaan Negeri X.
- Pengujian hanya mencakup metrik performa teknis, tanpa disertai uji langsung terhadap persepsi pengguna (*user testing*).
- Konten yang diuji bersifat umum (*public-facing*), belum mencakup modul interaktif seperti *login*, *dashboard*, atau integrasi dengan basis data kompleks.

Untuk penelitian selanjutnya, beberapa peluang yang dapat dikembangkan antara lain:

- Menguji implementasi SSR pada sistem informasi yang lebih kompleks seperti *e-learning, e-government*, atau *dashboard* internal.
- Menggabungkan SSR dengan teknik *caching* dinamis atau *static generation* (SSG) untuk kombinasi performa optimal.
- Melakukan studi komparatif dengan *framework* modern lainnya, seperti Nuxt.js (untuk Vue) atau Remix.

Dengan perluasan skenario uji dan kompleksitas sistem, penelitian di masa depan dapat memberikan gambaran yang lebih menyeluruh mengenai efektivitas SSR dalam berbagai konteks layanan digital publik.

## IV. KESIMPULAN

Penerapan Server Side Rendering (SSR) menggunakan framework Next.js terbukti secara signifikan meningkatkan performa website Kejaksaan Negeri X, yang ditunjukkan melalui peningkatan skor pengujian Lighthouse dari rata-rata 50 pada versi PHP menjadi 89,4 pada versi SSR. Peningkatan ini terjadi pada seluruh metrik utama seperti First Contentful Paint (FCP), Largest Contentful Paint (LCP), Total Blocking Time (TBT), dan Speed Index, yang berdampak langsung pada kecepatan pemuatan halaman, stabilitas tampilan, serta responsivitas terhadap interaksi pengguna. Selain itu, hasil render HTML di sisi server menjadikan website lebih SEO-friendly karena konten dapat langsung dibaca oleh mesin pencari. Oleh karena itu, pendekatan SSR direkomendasikan untuk diadopsi dalam pengembangan aplikasi web publik dan informatif, terutama bagi instansi pemerintah dan lembaga yang bertujuan menyampaikan informasi secara cepat, stabil, dan dapat diakses secara luas oleh masyarakat.

## REFERENSI

[1] APJII. (2024). Laporan Survei Penetrasi & Perilaku Pengguna Internet Indonesia 2024. Asosiasi Penyelenggara Jasa Internet Indonesia. https://apjii.or.id

[2] Geovanny, A. (2022). Analisis Rendering Performa Antara Server Side dan Client Side pada Web Application (Skripsi). Universitas Kristen Satya Wacana.

[3] Iskandar, T. F., Lubis, M., Kusumasari, T. F., & Lubis, A. R. (2020). Comparison between client-side and server-side rendering in the web development. IOP Conference Series: Materials Science and

Engineering, 1007(1), 012029. https://doi.org/10.1088/1757-899X/1007/1/012029

[4] Muna, S. S. (2022). Pengujian Performa Web Menggunakan Lighthouse Chrome DevTools (Artikel Ilmiah).

[5] Sinulingga, R. (2024). Implementasi Server Side Rendering pada Sistem Absensi Mahasiswa Berbasis Website. Universitas Sumatera Utara.

[6] P. G. A. Bhanuartha, A. Pinandito, dan M. A. Akbar, "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*, vol. 9, no. 3, Mar. 2025.

[7] I. D. Maulana dan Y. A. Susetyo, "Implementasi Fetch API dalam pengembangan Backend Website Daftar Film dengan Next.JS," *KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)*, vol. 6, no. 1, pp. 187–196, Jan. 2025.

[8] H. P. Putra dan A. P. Sari, "Implementasi Server Side Rendering pada Sistem Travel Berbasis Website," *Seminar Nasional Informatika Bela Negara (SANTIKA)*, vol. 4, 2024.

[9] R. Ardiyanto dan E. Ardhianto, "Analisa Peformasi Metode Rendering Website: Client Side, Server Side, dan Incremental Static Regeneration," *Computer Science (CO-SCIENCE)*, vol. 4, no. 1, pp. 19–27, Jan. 2024.

[10] A. J. Zaidan dan D. F. Suyatno, "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*, vol. 6, no. 1, pp. 1–13, 2025.