Implementasi *Server Side Rendering* Pada Sistem Absensi Mahasiswa Berbasis Website

Richardo Sinulingga[1], I Made Suartana[2]

[1,2] Jurusan Teknik Informatika/Program Studi S1 Teknik Informatika, Universitas Negeri Surabaya
[1]richardosinulingga.20078@mhs.unesa.ac.id
[2]madesuartana@unesa.ac.id

Abstrak*— Di era digital yang terus berkembang, teknologi informasi telah mengalami percepatan yang luar biasa, terutama dalam bidang pengembangan website. Salah satu perkembangan website saat ini menjadi salah satu aspek penting dalam lingkungan akademik adalah sistem absensi. Server side Rendering merupakan suatu teknologi rendering dalam pengembangan suatu website dimana website tersebut dirender di sisi server.Tujuan penelitian ini berfokus untuk menjelaskan dan menunjukan implementasi Server Side Rendering (SSR) pada sistem absensi mahasiswa berbasis website, menjelaskan dan membandingkan performa website yang mengimplementasikan Server Side Rendering dengan website yang mengimplementasikan Client Side Rendering (CSR). Hasil penelitian ini menunjukan bahwa implementasi Server side Rendering berhasil digunakan dan diimplementasikan pada sistem absensi mahasiswa berbasis website. Website dapat berjalan dengan baik dan dapat dirender dengan baik di sisi server. Hasil pengujian menunjukan bahwa implementasi Server side Rendering memiliki perbandingan kecepatan index (SI) memiliki rasio perbandingan ± 0,15 detik, waktu muat pertama (FCP) sebesar ± 0,20 detik, dan muat konten terbesar (FCP) sebesar sebesar ± 0,20 detik berdasarkan pengambilan data menggunakan tool pengujian yang telah ditetapkan. Pengujian dan perbandingan setiap data yang dikumpulkan menunjukan bahwa website Server side Rendering memiliki keunggulan dalam performa dan waktu muat, namun memiliki kelemahan yang sangat penting, yaitu adalah biaya beban dan pemeliharaan server dibandingakan website yang menggunakan Client side Rendering. Namun keunggulan di sisi keamanan dan SEO yang menjadi poin penting pada saat mengembangkan website Server side Rendering.

***Kata Kunci*— *Server Side Rendering*, *Client Side Rendering*, Performa, Website, Absensi**

I. PENDAHULUAN

Di era digital yang terus berkembang, teknologi informasi telah mengalami kemajuan yang luar biasa khususnya dalam bidang pengembangan website. Awalnya, website adalah suatu penghubung antara dokumen satu dengan yang lain yang dikenal sebagai hypertext [1]. Hypertext sendiri akan berkembang menjadi hypertext link dan finalnya kita kenal sebagai HTML (Hypertext Markup Language) yang menjadi cikal bakal web statis [2].

Namun, seiring berjalannya waktu, website terus berkembang ke arah yang lebih kompleks dan interaktif dengan menawarkan dan memberikan pengalaman yang kaya dan dinamis kepada pengguna. Sehingga, perkembangan website terdiri atas adalah kumpulan halaman web yang memiliki keterkaitan satu dengan yang lain, baik dari segi topik sampai halaman yang ditempatkan pada suatu server yang dapat di akses melalui jaringan internet maupun jaringan interlokal (LAN) [3]. Peningkatan ini memungkinkan website dapat beradaptasi dengan berbagai perangkat dan ukuran layar, sehingga memberikan pengalaman yang konsisten kepada pengguna di berbagai perangkat.

Absensi merupakan suatu bentuk pendataan kehadiran atau kehadiran seseorang atau pegawai yang merupakan pelapor suatu organisasi yang berisi data - data status kehadiran yang disusun dan disusun dengan cermat dan mudah dicari, dan digunakan jika sewaktu-waktu diminta oleh pihak yang berkepentingan [4]. Sistem absensi memegang peranan yang sangat penting dalam mengatur kehadiran dan partisipasi mahasiswa dalam lingkungan akademik. Kehadiran mahasiswa tidak hanya mencerminkan keterlibatannya dalam kegiatan pembelajaran, tetapi juga memberikan data penting untuk mengevaluasi kinerja dan keberhasilan akademik. Oleh karena itu, penting untuk memiliki sistem waktu dan kehadiran yang efisien dan akurat. Sistem absensi yang baik harus mampu mengelola data kehadiran dengan baik, menyajikan laporan yang mudah diakses dan dipahami, serta mendukung keseluruhan proses pengelolaan akademik. Dalam beberapa tahun terakhir, terutama setelah pandemi COVID-19, terjadi perubahan signifikan dalam proses perkuliahan dengan banyaknya universitas yang menerapkan dan beralih ke model perkuliahan luring atau hybrid sehingga perubahan ini menimbulkan tantangan baru dalam proses pengambilan absensi mahasiswa.

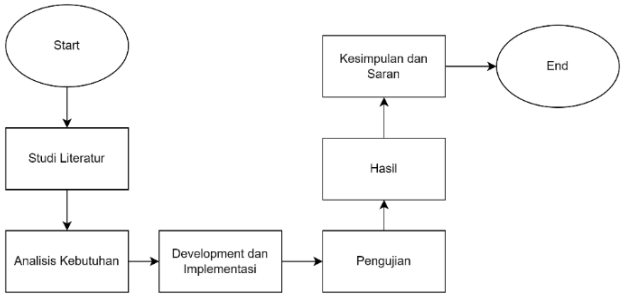
Dalam pengembangan website, konsep rendering berperan penting dalam memproses halaman website yang tidak hanya cepat tetapi juga responsif. Secara tradisional, teknologi rendering sangat bergantung pada sisi klien, di mana pembuatan dan pemrosesan halaman web terjadi langsung di browser pengguna. Namun, mesin browser tidak dapat membaca data mentah, ia harus bekerja dengan model objek dokumen (DOM). Untuk mengubah data mentah menjadi DOM, mesin browser pertama-tama mengubah data mentah menjadi karakter [5]. Meskipun pendekatan ini memberikan kontrol tingkat tinggi terhadap interaksi pengguna, pendekatan ini cenderung memiliki kelemahan, termasuk waktu muat halaman yang lebih lama. Untuk mengatasi keterbatasan ini, server-side rendering (SSR) diperkenalkan sebagai solusi. Ketika klien mengunjungi website yang menggunakan SSR, browser akan mengirimkan permintaan ke server. Server akan mengirimkan kembali berkas HTML yang di render sepenuhnya dan menampilkannya di sisi klien. Ini berarti bahwa semua berkas Javascript dan CSS akan di render dan tidak perlu menunggu browser untuk memuat file-file ini sebelum konten terlihat [5]. Pendekatan ini menawarkan banyak manfaat, termasuk waktu

pemuatan halaman yang lebih cepat, penggunaan sumber daya yang lebih efisien.

Tujuan utama penelitian ini bertujuan untuk mengeksplorasi bagaimana SSR mampu meningkatan kecepatan muat halaman. Kecepatan muat yang lebih cepat dapat mengurangi waktu tunggu pengguna dan meningkatkan kenyamanan dalam mengakses halaman demi halaman pada website. Oleh karena itu, penelitian ini tidak hanya berkontribusi terhadap perkembangan teknologi tetapi juga berkontribusi terhadap peningkatan kualitas pendidikan dan manajemen akademik.

II. METODE PENELITIAN

Jenis metode penelitian yang diadopsi adalah pendekataan kuantitatif untuk mengimplementasikan Server-side Rendering pada sistem absensi website yang bertujuan untuk meningkatkan performa dan response time saat merender halaman website. Berikut alur penelitian yang dijelaskan pada Gambar 1.



Gbr. 1 Alur Penelitian

A. *Studi Literatur*

Pada tahapan ini, akan dilakukan pengambilan setiap informasi yang dibutuhkan dalam penelitian ini melalui catatan jurnal dan buku yang dapat di akses via Google Schoolar, Scribd dan website jurnal seperti doi.org. Fokus utama yang akan dicari adalah literatur seputar teknologi mengenai Server-side Rendering pada framework bahasa pemrograman Javascript/Typescript, yaitu Next.j sebagai acuan dan pedoman dalam proses implementasi sistem absensi mahasiswa yang dipaperkan pada dokumentasi resminya.

B. *Analisis Kebutuhan*

Pada tahapan ini, akan dilakukan pengumpulan setiap data serta informasi yang telah di himpun dari beberapa jurnal dan buku sehingga peneliti dapat menentukan analisis kebutuhan, baik secara fungsional maupun non-fungsional. Adapun kebutuhan fungsional pada sistem absensi berbasis website tersebut sebagai berikut:

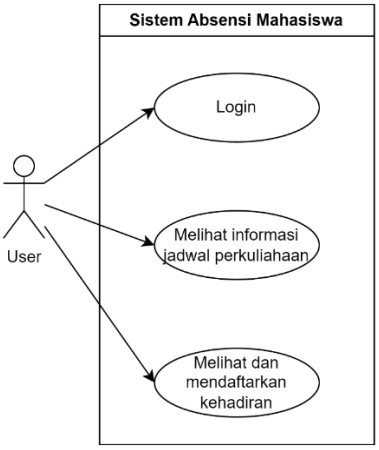
1. Sistem dapat digunakan untuk melihat informasi jadwal perkuliahan.
2. Sistem dapat menampilkan data diri mahasiswa.
3. Sistem dapat digunakan untuk mendaftarkan kehadiran pada jadwal perkuliahan.

Sedangkan kebutuhan non-fungsional pada sistem absensi berbasis website tersebut sebagai berikut:

1. Sistem mudah digunakan dengan antarmuka pengguna yang jelas, dan tentunya bersifat responsif.
2. Sistem mudah diakses di berbagai perangkat yang telah terkoneksi oleh internet.

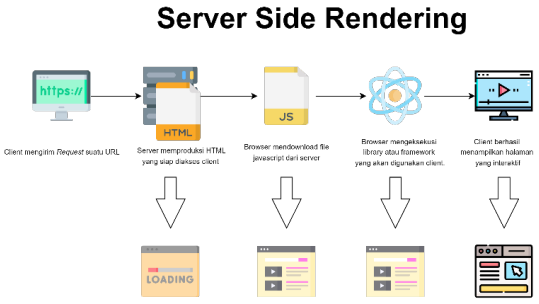
C. *Development dan Implementasi*

Pada tahapan ini akan di mulai dengan dilakukannya tahapan development dengan menentukan design User Inteface sederhana yang dirancang pada platform design seperti Figma berdasarkan usecase pada Gambar 2 dibawah ini.

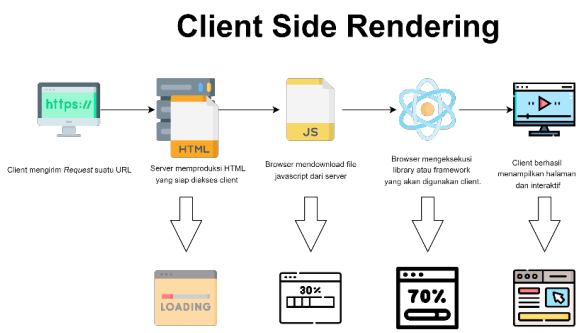


Gbr. 2 Usecase Diagram Sistem Absensi Mahasiswa

Untuk tahapan implementasi, Sistem absensi akan menggunakan metode Server-side Rendering sebagai tujuan utama penelitian ini untuk melihat dan mengukur perbandingan kecepatan load time respon saat merender halaman yang dikirimkan oleh server dan performa pertama kali akses website. Berikut mekanisme dan cara kerja dari Server-side Rendering dan Client-side Rendering yang tertera pada Gambar 3 dan Gambar 4.



Gbr. 3 Cara Kerja *Server-side Rendering*



Gbr. 4 Cara Kerja *Client-side Rendering*

Pada Server-side Rendering, server akan mengirimkan file HTML yang diproduksi pada server yang sangat berbeda dengan Client-side Rendering [6] dimana HTML kosong akan ditampilkan terlebih dahulu dan Javascript akan direnderer oleh browser klien secara bertahap bersamaan dengan CSS dan database sebagai respon dari server. Perbedaan inilah yang akan dikomparasikan berdasarkan parameter load time respon kedua metode tersebut dengan catatan pada satu teknologi pembuatan website yang sama yang telah terhosting kedalam domain website yang nantinya dapat di uji menggunakan tool pihak ketiga.

D. *Pengujian*

Pengujian akan dilakukan dan menggunakan tool pihak ketiga seperti Google Lighthouse, PageSpeed Insight dan WebPage Test untuk mengukur variabel yang dibutuhkan, skor atau nilai performa website, dan lain sebagainya. Berikut tahapan-tahapan yang dilakukan pada pengujian.

1. Website dipastikan terhosting dengan baik.
2. Memastikan setiap server dapat berjalan dengan baik.
3. Mengakses setiap tool agar nantinya akan di eksekusi, sehingga dapat menampilkan hasil analisis.
4. Mencatat parameter yang dibutuhkan.
5. Membandingkan beberapa tool dan mencari perbandingan serta menghitung rata-rata dari gabungan yang didapatkan dari data tool tersebut.

III. Hasil Dan Pembahasan

Paragraf harus teratur. Semua paragraf harus rata, yaitu sama-sama rata kiri dan dan rata kanan.

A. *Hasil*

Implementasi *server-side rendering* pada sistem absensi mahasiswa terbagi kedalam empat tahapan, seperti tahapan analisis kebutuhan (Analysis), tahap implementasi (Implementation), tahap pengujian dan analisis hasil.

1. Tahapan Analisis

Pada tahapan ini, peneliti mengumpulkan setiap data-data dan beberapa dokumentasi penggunaan server-side rendering pada Next.js melalui dokumentasi pada website resmi next.js, tulisan-tulisan blog developer, dan video-video tutorial lainnya . Peneliti mempelajari proses dan cara kerja dan beberapa penggunaan metode dan fungsi yang telah dipaparkan oleh Next.js sendiri. Peneliti juga diharuskan untuk membangun suatu database untuk mendukung server belakang (backend service) agar sistem absensi dapat berjalan sebagaimana mestinya.

Peneliti mengamati dan mempelajari bagaimana sistem kerja dari framework next.js sendiri dalam merender halaman website. Pada Next.js versi 13, sistem routing Next.js sendiri telah terbagi menjadi dua, yaitu Pages Router dan App Router. App Router pada Next.js mengatur setiap file folder yang didalamnya yang mengandung javascript/typescript akan merepresentasikan suatu route yang nantinya akan terbaca oleh sistem melalui file page.jsx/page.tsx. Sebelumnya di Pages Router, Next.js memiliki method yang bernama getServerSideProps yang akan membangun suatu permintaan di server. Namun pada App Router, setiap komponen file Next.js bersifat server komponen, sehingga konten-konten seperti HTML dan Javascript akan dibangun dan diproses di server dan akan mengalami proses hydration, sehingga nantinya proses caching akan berjalan baik di sisi server maupun di sisi klien.

Next.js sendiri sudah memulai mengintegrasikan setiap halaman dapat direnderer di server, peneliti bermaksud untuk membangun sisi belakang (Backend) menggunakan Object-Relational Mapping (ORM) sebagai alternatif untuk manajemen database dan proses quering nantinya untuk sistem absensi mahasiswa. ORM yang cukup sering digunakan oleh beberapa orang untuk manajemen database saat menggunakan Next.js ialah Prisma. Peneliti menggunakan PostgreSQL dan Supabase yang merupakan Backend-as-a-service (BaaS).

2. Tahap Implementasi

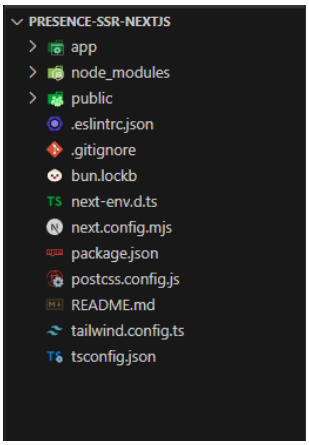
Peneliti mengimplementasikan server-side rendering kedalam website sistem absensi mahasiswa dan juga client-side rendering pada website sistem absensi mahasiswa sehingga dapat dijadikan sampel pengambilan data yang nantinya di proses oleh tool untuk mengukur pengujian performa kedua website.

Peneliti memulai menginisiasi proyek dari Next.js menggunakan App Router dan Typescript. Berikut inisiasi dari Next.js menggunakan Bun.js sebagai alternatif lain dari Node.js sebagai alat

pengelola paket pada ekosistem lingkungan Javascript seperti pada Gambar 5 dibawah ini.

Gbr. 5 Perintah untuk membuat proyek Next.js

Ketika proses inisiasi proyek sudah selesai, kita dapat melihat App Dir dari Next.js itu sendiri yang disajikan pada Gambar 6 sebagai berikut.



Gbr. 6 Daftar File dan Folder pada Next.js

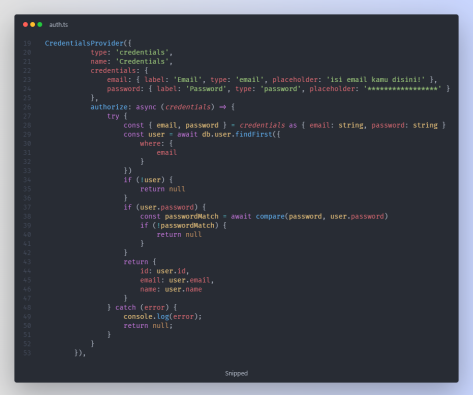
Kemudian peneliti mengintegrasikan Prisma yang merupakan ORM sebagai alternatif manipulasi data dan quering pada database, kemudian ditampung didalam suatu fungsi dan method REST API untuk dikonsumsi oleh sisi depan (Frontend). Metode yang perlu diinisiasi adalah sebagai berikut:

a. Mengirim data login berupa email dan password.
b. Mendapatkan data user dan masuk kedalam dashboard
c. Mendapatkan data kehadiran berdasarkan hari.
d. Merubah data kehadiran yang awalnya FALSE menjadi TRUE

Selanjutnya, peneliti membuat model yang diperlukan untuk diubah ke dalam tabel pada database. Berikut model yang inisiasikan pada Gambar 7 dibawah ini.

Gbr. 7 Model pada ORM Prisma

Setelah peneliti menginisiasikan proyek dari Next.js dan mengintegrasikan ORM sebagai manipulasi query pad database, peneliti akan memulai pengerjaan dan pembuatan website halaman demi halaman, mengerjakan otentikasi login yang terintegrasi dengan database menggunakan *library* next-auth pada Gambar 8.



Gbr. 8 Credential Provider sebagai otentikasi login pada website

Kemudian peneliti menginisiasikan layout pada proyek dari Next.js. Layout ini digunakan untuk mengatur layout pada halaman suatu website yang terintegrasi dengan Next.js. Pada body, children akan berada pada wrapper kustom ProviderComponent untuk menangkap sesi user yang sedang login pada website yang dipaparkan pada Gambar 9 dibawah ini.

Gbr. 9 Layout pada website Next.js

Peneliti juga membuat suatu pengecekan berupa halaman protected yang berfungsi untuk mengatur sesi login. Ketika sesi login tidak ditemukan, maka tidak akan diarahkan ke halaman dashboard dan langsung di redirect ke halaman utama pada Next.js yang telah diinisiasikan seperti pada Gambar 10 dibawah ini.

Gbr. 10 Halaman Protected

3. Tahap Pengujian dan Analisis Hasil

Peneliti melakukan pengujian website yang telah terhosting di VPS cloud dan platform Vercel. Berikut hasil pengujian yang dituangkan kedalam Tabel 1 s.d. Tabel 3 berdasarkan hasil data yang diambil melalui tool yang telah ditetapkan sebagai berikut.

TABLE I
HASIL PENGUJIAN DARI WEBSITE SERVER-SIDE RENDERING DAN CLIENT SIDE RENDERING PADA VPS CLOUD DAN VERCEL MENGGUNAKAN GOOGLE LIGHTHOUSE

No	Parameter	Skor				Penjelasan
		Server-VPS	Server - Vercel	Client - VPS	Client - Vercel	
1.	First Contentful Paint (FCP)	0,5	0,3	0,5	0,3	Waktu yang didapatkan untuk merender konten pertama yang dikumpulkan keseluruhan.
2.	Total Blocking Time (TBT)	0	0	0	0	Rentang waktu halaman mengalami pemblokiran yang berpotensi mengganggu pengalaman user saat mengakses halaman website.
3.	Speed Index (SI)	0,525	1	0,725	0,45	Waktu yang dibutuhkan agar konten pada halaman dapat dilihat oleh user .
4.	Largest Contentful Paint (LCP)	0,8	0,35	1,025	0,325	Waktu yang dibutuhkan oleh browser untuk merender konten terbesar pertama atau paling penting pada halaman web .
5.	Cumulative Layout Shift (CLS)	0,00925	0	0,00975	0	Banyaknya pergeseran tata letak konten pada halaman web saat dimuat.

Pada Tabel I, website yang terhosting pada VPS cloud dan Vercel menunjukan angka FCP yang sama dengan nilai 0,5 detik dan 0,3 detik. Sedangkan pada metrik LCP, website *servers-side rendering* yang terhosting pada VPS menunjukan angka dengan selisih 0,225 detik lebih cepat dibandingkan dengan website *client-side rendering*. Sementara untuk metrik SI, perbandingan website *server-side rendering* dengan *client-side rendering* menunjukan selisih angka 0,2 detik lebih cepat di sisi website *server-side rendering*. Untuk hasil yang ditunjukan pada website yang terhosting pada Vercel memiliki perbandingan dengan nilai LCP sebesar 0,025 detik lebih lambat pada website *server-side rendering* dan nilai SI di angka 0,55 detik lebih cepat pada website *client-side rendering*.

TABLE II
HASIL PENGUJIAN DARI WEBSITE SERVER-SIDE RENDERING DAN CLIENT SIDE RENDERING PADA VPS CLOUD DAN VERCEL MENGGUNAKAN PAGESPEED INSIGHT

No	Parameter	Skor				Penjelasan
		Server-VPS	Server - Vercel	Client - VPS	Client - Vercel	
1.	First Contentful Paint (FCP)	0,3	0,5	0,5	0,3	Waktu yang didapatkan untuk merender konten pertama yang dikumpulkan keseluruhan.
2.	Total Blocking Time (TBT)	0	0	0	0	Rentang waktu halaman mengalami pemblokiran yang berpotensi mengganggu pengalaman user saat mengakses halaman website.
3.	Speed Index (SI)	0,4	0,9	0,6	0,9	Waktu yang dibutuhkan agar konten pada halaman dapat dilihat oleh user .
4.	Largest Contentful Paint (LCP)	0,5	0,7	0,7	0,4	Waktu yang dibutuhkan oleh browser untuk merender konten terbesar pertama atau paling penting pada halaman web .
5.	Cumulative Layout Shift (CLS)	0	0	0	0	Banyaknya pergeseran tata letak konten pada halaman web saat dimuat.

Pada Tabel II, website yang terhosting pada VPS menunjukan angka FCP dengan selisih 0,2 detik lebih cepat pada website *server-side rendering* dan 0,2 detik lebih lambat pada website *server-side rendering* yang terhosting pada Vercel. Sedangkan pada metrik LCP, website *servers-side rendering* yang terhosting pada VPS menunjukan angka dengan selisih 0,2 detik lebih cepat dibandingkan dengan website *client-side rendering*. Sementara untuk metrik SI, perbandingan website *server-side rendering* dengan *client-side rendering* menunjukan selisih angka 0,2 detik lebih cepat di sisi website *server-side rendering*. Untuk hasil yang ditunjukan pada website yang terhosting pada Vercel memiliki perbandingan dengan nilai LCP sebesar 0,3 detik lebih lamba t pada website *server-side rendering* dan nilai SI di angka 0,9 detik pada kedua website.

TABLE III
HASIL PENGUJIAN DARI WEBSITE SERVER-SIDE RENDERING DAN CLIENT SIDE RENDERING PADA VPS CLOUD DAN VERCEL MENGGUNAKAN WEBPAGE TEST

No	Parameter	Skor				Penjelasan
		Server-VPS	Server - Vercel	Client - VPS	Client - Vercel	
1.	First Contentful Paint (FCP)	0,780	1,146	1,489	0,786	Waktu yang didapatkan untuk merender konten pertama yang dikumpulkan keseluruhan.
2.	Total Blocking Time (TBT)	0	0	0	0	Rentang waktu halaman mengalami pemblokiran yang berpotensi mengganggu pengalaman user saat mengakses halaman website.
3.	Speed Index (SI)	0,832	1,196	1,548	0,848	Waktu yang dibutuhkan agar konten pada halaman dapat dilihat oleh user .
4.	Largest Contentful Paint (LCP)	0,879	1,685	1,780	1,132	Waktu yang dibutuhkan oleh browser untuk merender konten terbesar pertama atau paling penting pada halaman web .

5.	Cumulative Layout Shift (CLS)	0	0	0	0	Banyaknya pergeseran tata letak konten pada halaman web saat dimuat.

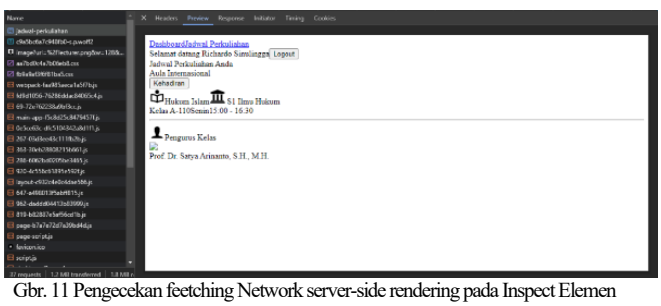
Pada Tabel III, website yang terhosting pada VPS cloud menunjukan angka FCP dengan selisih nilai sebesar 0,709 detik lebih cepat pada website *server-side rendering.* Sedangkan metrik LCP, website *servers-side rendering* yang terhosting pada VPS menunjukan angka dengan selisih 0,901 detik lebih cepat dibandingkan dengan website *client-side rendering*. Sementara untuk metrik SI, perbandingan website *server-side rendering* dengan *client-side rendering* menunjukan selisih angka 0,716 detik lebih cepat di sisi website *server-side rendering*. Untuk hasil yang ditunjukan pada website yang terhosting pada Vercel memiliki perbandingan dengan nilai FCP sebesar 0,360 detik, LCP sebesar 0,553 detik lebih lamba t pada website *server-side rendering* dan nilai SI di angka 0,348 detik lebih cepat pada website *client-side rendering*.

B. *Pembahasan*

Berdasarkan hasil penelitian yang dilakukan oleh peneliti dari pengembangan Implementasi Server Side Rendering Sistem Absensi Mahasiswa Berbasis Website, data yang didapat saat pengumpulan dapat menjawab dari rumusan masalah yang ada.

Peneliti menemukan perbandingan kode pada saat mengimplementasikan server-side rendering dan client-side rendering berdasarkan dokumentasi resmi dari Next.js yang peneliti temukan dan pelajari, serta beberapa pengujian kecil yang telah dilakukan dan telah diimplementasikan ke dalam website.

Menurut dokumentasi Next.js yang menggunakan App dir, Setiap komponent Next.js bersifat server-component , sehingga setiap kode yang dituliskan akan secara otomatis akan dirender di server (beberapa syarat tertentu). Pada sistem absensi mahasiswa yang menggunakan server-side rendering, setiap data yang berasal dari database akan mengalami proses fetching di server. Sehingga nantinya server tidak akan mengirimkan file HTML kosong, melaikan file HTML yang telah berisi data-data yang berasal dari database seperti pada Gambar 11 dibawah ini.



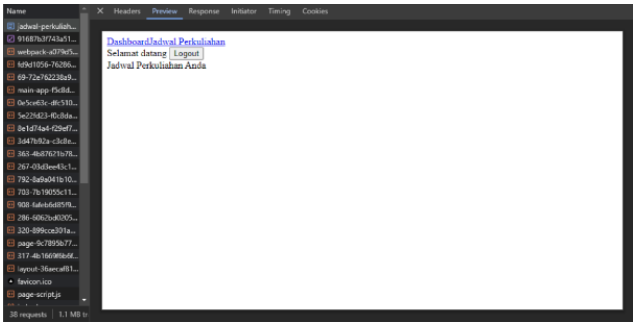
Gbr. 11 Pengecekan feetching Network server-side rendering pada Inspect Elemen Browser

Sehingga tidak lagi memerlukan Hooks untuk menampung data hasil fetching, karena akan ditampung ke dalam fungsi dan akan di proses di server dan dilakukan render berdasarkan data yang ada. Disamping itu, setiap respons dari API yang dilakukan di server tidak dapat dilihat secara kasat mata melalui Inspect pada browser karena telah dibangun di server dalam bentuk HTML.

Gbr. 12 Fungsi untuk melakukan *fetching data* pada *server component*

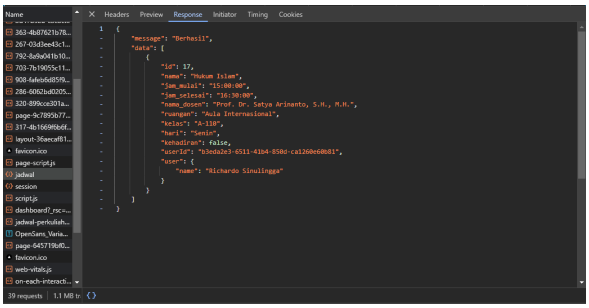
Pada Gambar 12 diatas, peneliti langsung membuat sutu fungsi biasa yang bersifat asinkronus untuk menangkap data yang berasal dari API yang telah dibuat. Nantinya akan di deklarasikan ke dalam variabel baru dan dilakukan proses mapping untuk menghasilkan data yang akan ditampilkan pada HTML.

Sedangkan menurut dokumentasi Next.js yang menggunakan App dir, ketika ingin menggunakan *client components,* setiap page harus mendeklarasikan “use client” pada bagian atas halaman sebelum mengimport paket library. Berbeda dengan halaman yang menggunakan *server-side rendering*, halaman web akan melakukan *fetching* data di sisi klien yang nantinya akan terintegrasi dengan HTML kosong yang telah dikirim oleh server secara asinkron seperti pada Gambar 13.



Gbr. 13 Kondisi file HTML yang dikirimkan oleh server

Setiap file HTML yang dikirim oleh server masih dalam keadaan kosong dan belum berisi data-data yang diambil dari API, yang nantinya akan dilakukan fetching data di sisi klien melalui browser yang ditunjukan pada Gambar 14 dibawah ini.



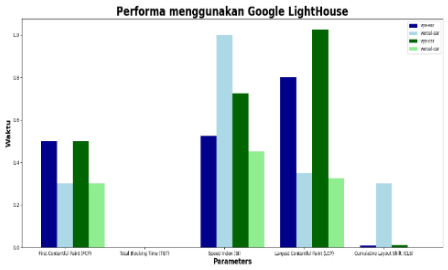
Gbr. 14 Proses *Fetching* data yang dilakukan pada browser

Setiap data yang akan melakukan proses fetching, setiap halaman harus mempersiapkan fungsi React Hook (useState, useEffect, useCallback) untuk menampung data yang telah di respon oleh API. Pada Gambar 15, proses fetching data dilakukan pada fungsi Hook useEffect dan nantinya akan ditampung pada Hook useState.

Gbr. 15 *Fetching* data pada React Hooks di sisi klien

Peneliti melakukan pengujian dan mendapatkan data melalui ketiga tool yang digunakan untuk mendapat beberapa parameter berupa data-data pembanding antara website yang menggunakan server-side rendering dan client-side rendering. Peneliti menguji keempat website dengan komposisi dua website server-side rendering dan dua website client-side rendering yang terhosting pada VPS Cloud dan platform Vercel yang menggunakan Supabase sebagai alternative database dan juga merupakan BaaS (Backend-as-a-services). Berikut grafik dalam berupa diagram batang yang dijabarkan kedalam tiga poin berikut.

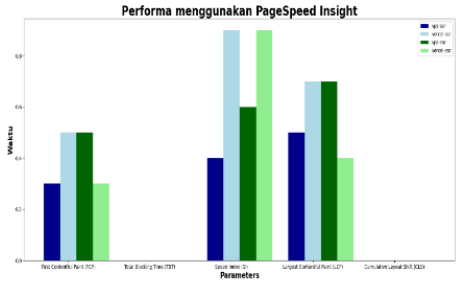
1. Perbandingan data website yang mengadopsi server-side rendering dan client-side rendering menggunakan Google LightHouse.



Gbr. 16 Diagram batang skor menggunakan tool Google LightHouse

Seperti pada Gambar 16, Website yang terimplementasi server-side rendering pada VPS Cloud (baca. vps-ssr) mendapatkan nilai waktu muat pertama (FCP) di angka yang sama dengan skor 0,5 detik, waktu muat konten terbesar (LCP) dengan rasio perbandingan sebesar 0,45 detik , sehingga menghasilkan nilai render konten visual (SI) jauh lebih baik dibandingkan pada website yang terimplementasi client-side rendering pada VPS Cloud dengan rasio perbandingan (baca. vps-csr) dengan rasio perbandingan sebesar 0,2 detik . Namun cukup berbanding terbalik ketika disandingkan dengan data-data yang berasal dari platform Vercel. Website yang mengimplementasikan server-side rendering, justru melambat dibandingkan client-side rendering dengan rasio perbandingan di angka ± 0,45 detik pada render konten visual (SI). Hal ini disebabkan oleh platform Vercel masih belum menyempurnakan bridge function pada serverless function yang merupakan fungsi yang digunakan untuk memparsing data yang di fetching di server.

2. Perbandingan data website yang mengadopsi server-side rendering dan client-side rendering menggunakan Google LightHouse.

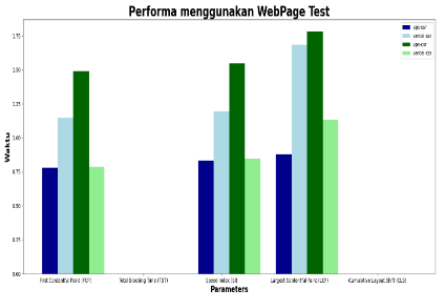


Gbr. 17 Diagram batang skor menggunakan tool PageSpeed Insight

Peneliti mendapatkan data melalui website PageSpeed dan kemudian menginput url dari setiap website yang telah terhosting, baik melalui VPS Cloud maupun platform Vercel. Berdasarkan data yang telah dikumpulkan oleh peneliti pada Gambar 17, website

yang menggunakan server-side rendering menunjukan kesamaan pola data seperti yang ditemukan peneliti menggunakan tool Google Lighthouse berupa skor dari FCP dan LCP dengan rasio perbandingan sebesar ± 0,20 detik dari website yang mengimplementasikaan server-side rendering lebih baik dibandingkan website yang menggunakan client-side rendering yang terhosting pada VPS Cloud.

3. Perbandingan data website yang mengadopsi server-side rendering dan client-side rendering menggunakan WebPage Test.



Gbr. 18 Diagram batang skor menggunakan tool WebPage Test

Tidak jauh berbeda dengan tool PageSpeed Insight, peneliti mengumpulkan data melalui website WebPage Test dan kemudian menginput url dari setiap website yang telah terhosting, baik melalui VPS Cloud maupun platform Vercel. Berdasarkan data yang telah terkumpul pada Gambar 4.22, website yang menggunakan server-side rendering menunjukan kesamaan pola data seperti yang ditemukan peneliti menggunakan tool Google Lighthouse maupun PageSpeed Insight berupa skor dari FCP dan LCP dengan rasio perbandingan sebesar ± 0,70 detik dari website yang mengimplementasikaan server-side rendering lebih baik dibandingkan website yang menggunakan client-side rendering.

IV. KESIMPULAN

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan peneliti, maka didapatkan suatu kesimpulan sebagai berikut:

Berdasrkan hasil pengujian yang dilakukan oleh peneliti, menunjukan bahwa waktu yang dibutuhkan untuk merender konten pertama (FCP) secara konsisten ditunjukan pada website server-side rendering yang dihosting pada VPS Cloud menunjukan angka yang lebih baik/cepat dibandingkan dengan website client-side rendering yang dihosting pada VPS Cloud dengan rasio perbandingan sebesar ± 0,20 detik (ditunjukan pada hasil dari diagram batang pada pengujian menggunakan PageSpeed Insight dan WebPage Test). Untuk website server-side rendering yang dihosting pada platform Vercel menunjukan hasil yang sebaliknya, yaitu lebih lambat dibandingkan pada website client-side rendering yang terhosting pada platform Vercel (terkecuali pada hasil Google LightHouse). Untuk waktu yang dibutuhkan website server-side rendering dalam meload konten terbesar (LCP) yang dihosting pada VPS Cloud menunjukan angka yang lebih baik dibandingkan website client-side rendering yang terhosting pada VPS Cloud dengan rasio perbandingan sebesar ± 0,20 detik (ditunjukan pada hasil dari diagram batang pada pengujian menggunakan ketiga tool pengujian). Sedangkan pada website server-side rendering yang dihosting pada platform Vercel menunjukan hasil yang sebaliknya, yaitu lebih lambat dibandingkan pada website client-side rendering yang terhosting pada platform Vercel (ditunjukan pada hasil ketiga tool pengujian). Dan yang terakhir ialah, waktu yang dibutuhkan untuk menampilkan konten secara visual (SI) pada website server-side rendering menunjukan angka yang jauh lebih baik dan cepat dibandingkan website client-side rendering yang dihosting pada VPS Cloud dengan rasio perbandingan sebesar ± 0,15 detik (ditunjukan secara konsisten pada ketiga tool pengujian). Namun hasil berbeda ditunjukan pada hasil pengujian pada website server-side rendering yang dihosting pada platform Vercel. Hasil menunjukan bahwa waktu untuk menampilkan konten visual (SI) lebih lambat dibandingkan pada website client-side rendering yang terhosting pada platform Vercel (terkecuali pada hasil PageSpeed Insight). Sementara pada hasil pengujian dari metrik atau parameter dari Total Blocking Time (TBT) menunjukan hasil yang konsisten yaitu 0 pada kedua website yang dilakukan pengujian pada ketiga tool pengujian dalam dua tempat hosting yang berbeda. Mengenai temuan peneliti atas hasil pengujian yang menunjukan skor performa rendering pada website server-side rendering pada platform Vercel [7]dan juga diperkuat oleh temuan yang didapatkan peneliti pada github discussion, beberapa keluhan yang ditulis oleh beberapa user Next.js itu sendiri pada saat menghosting website yang terimplementasi di server component dan server-side rendering yang disebabkan oleh penggunaan serverless function karena platform Vercel sendiri berfungsi sebagai handler setiap API Route [8] dan server-side rendered pada setiap halaman. Sehingga dampak nya adalah ketidakkonsitenan server-side rendering pada website yang terhosting pada platform vercel.

Hasil pembahasan menunjukan bahwa website server-side rendering akan melakukan fetching data pada server, sehingga server tidak lagi mengirimkan file HTML kosong [9], sehingga tidak lagi dua kali kerja pada sisi klien yang berbeda ditunjukan pada website client-side rendering, dimana server akan mengirimkan file HTML kosong dan secara asinkronus dilakukannya proses fetching data di sisi klien [10]. Website yang terimplementasi oleh server-side rendering akan memiliki waktu render yang jauh lebih cepat dibandingkan website yang menggunakan client-side rendering. Disamping itu, Next.js sebagai framework dari React.js itu sendiri merupakan suatu terobosan baru dalam merender setiap halaman secara default di dalam server [11], sehingga harapannya, website yang dibangun akan memiliki performa yang baik dan produk akhirnya adalah kepuasan dari pengguna website itu sendiri. Namun kelemahan dari website yang terimplementasi server-

side rendering ialah adalah penggunaan server yang mumpuni, sehingga dibutuhkan sumber daya dan pemeliharan yang besar.

REFERENSI

[1] Tim Berners-Lee, "The World Wide Web: A very short personal history," https://www.w3.org/People/Berners-Lee/ShortHistory.html.

[2] M. N. Francis, "The history of the Web," www.w3.org/community/webed/wiki/The_history_of_the_Web.

[3] Y. Susilowati, *Modul E-Commerce-Teaching Factory For Students*. Jl. Kenari No.35-c, Plosokerep, Sananwetan, Blitar, Jawa Timur, 66134: Mutiara, 2019.

[4] E. Simonna, "DATA KKP," https://simonna-erna.blogspot.com/2009/10/data-kkp.html.

[5] C. Nordström and A. Dixelius, "Comparisons of Server-side Rendering and Client-side Rendering for Web Pages," 2023. Accessed: Nov. 24, 2023. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1797261&dswid=7256

[6] Nextjs.org, "Using Client Components in Next.js," https://nextjs.org/docs/app/building-your-application/rendering/client-components#using-client-components-in-nextjs.

[7] R. Hanafi, A. Haq, and N. Agustin, "Comparison of Web Page Rendering Methods Based on Next.js Framework Using Page Loading Time Test," *Teknika*, vol. 13, no. 1, pp. 102–108, Mar. 2024, doi: 10.34148/teknika.v13i1.769.

[8] molink-huseong, "First SSR Page request is too slow."

[9] R. Hudiansyah, "Server Side Rendering & Client Side Rendering, Apa Bedanya?," https://blog.dot.co.id/server-side-rendering-dengan-next-js-44354c21031a.

[10] F. M. Carvalho and P. Fialho, "Enhancing SSR in Low-Thread Web Servers: A Comprehensive Approach for Progressive Server-Side Rendering with any Asynchronous API and Multiple Data Models," in *International Conference on Web Information Systems and Technologies, WEBIST - Proceedings*, Science and Technology Publications, Lda, 2023, pp. 37–48. doi: 10.5220/0012165300003584.

[11] O. Lyxell, "Server-Side Rendering in React: When Does It Become Beneficial to Your Web Program?," 2022, Accessed: Nov. 15, 2023. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1764528&dswid=-5000