BAB II TINJAUAN PUSTAKA

A. Penelitian Terkait

H. A. Jartarghar, G. R. Salanke, dan S. Dalali [9] yang melakukan penelitian tentang performa *Server-side Rendering Next.js* menyatakan. Aplikasi web dikembangkan menggunakan berbagai *framework* web yang berbeda, dan pengembang dapat melakukannya pilih dari berbagai *framework* web saat mengembangkan aplikasi web. *Library Next.js* memberikan fleksibilitas untuk membangun Komponen *User Interface* (UI) yang dapat digunakan kembali. ini menggunakan pendekatan dari *rendering* sisi klien, yang memuat konten HTML menggunakan *JavaScript*. Render sisi klien menyebabkan halaman dimuat dengan lambat dan klien berkomunikasi dengan server hanya untuk data *run-time*. *Next.js Framework* memecahkan masalah ini dengan menggunakan *rendering* sisi server. Saat browser meminta halaman web, server memproses halaman web dengan mengambil data spesifik *user* dan mengirimkannya kembali ke browser melalui internet. Setelah di simpulkan, *React App* dengan *rendering* sisi server dapat dibangun menggunakan *framework Next.js*, yang mengoptimalkan kinerja aplikasi. *Next.js* juga meningkatkan SEO untuk aplikasi web dan membuat crawling mudah diakses. Dua komponen bangunan terpenting yang mendukung keseluruhan pengalaman digital adalah *Next.js* dan React. Mereka memiliki kemampuan untuk mempercepat web melalui aplikasi dengan meningkatkan kinerja, mengurangi biaya pengembangan, dan memiliki tingkat produksi yang lebih besar.

Mengutip dari T. Fadhilah Iskandar, M. Lubis, T. Fabrianti Kusumasari, dan A. Ridho Lubis [10] yang melakukan penelitian tentang perbandingan antara *Server-side Rendering* dan *Client-side Rendering* Untuk Pengembangan *Website*. Server wajib untuk aplikasi universal yang dapat diakses oleh sejumlah *user* mungkin menjadi penghalang bagi korporasi dan berlebihan untuk aplikasi kecil meskipun dapat membawa keuntungan kompatibilitas. Mengetahui bahwa permintaan aplikasi web meningkat untuk memberikan kenyamanan dan kemudahan penggunaan kepada *User*, *rendering* sisi klien hadir untuk membuat perangkat lunak lebih cepat dan efisien. Hal ini dilakukan dengan mengarahkan permintaan ke file HTML kemudian server akan memberikan pesan tanpa konten apa pun atau layar pemuatan hingga perangkat mengambil semua *JavaScript* untuk memungkinkan browser mengkompilasi semuanya sebelum menampilkan konten. Setelah melakukan perbandingan antara metode sisi klien dan sisi server dengan aspek teknis dalam hal FCP, indeks kecepatan, waktu interaktif, keseluruhan konten dirender, CPU idle pertama dan perkiraan latensi input, yang menghadirkan kinerja yang lebih baik dengan 2.1s, 2.0s, 2.2s, 2.1s, 2.2s dan 20ms masing-masing di sisi server. Ini juga memberikan hasil yang lebih baik berdasarkan Google Audit dengan kinerja 100%, aksesibilitas 48%, praktik terbaik 93% dan 89% optimasi mesin pencari (SEO).

Johan Lutto, dan Oscar Rosquist [12] dalam Jurnalnya yang meneliti tentang perbedaan kinerja antara Server-Side dan *Client-side Rendering* dalam visualisasi data secara *realtime*. Visualisasi data *realtime* memiliki kemampuan untuk memvisualisasikan data dalam jumlah besar ke dalam grafik yang dapat dimengerti dan memungkinkan tindakan segera diambil pada tren yang muncul. Untuk melakukan perbandingan ini, dua aplikasi dibangun untuk mengukur kinerja kedua belah pihak. Hasilnya menunjukkan bahwa visualisasi data sisi klien lebih cepat daripada sisi server dengan *D3.js* saat menggunakan JSDOM di sisi server. Oleh karena itu mereka menyimpulkan bahwa menggunakan lingkungan yang sama, atau sangat mirip dengan aplikasi dalam penelitian ini, visualisasi data sisi klien menawarkan kinerja yang lebih baik. Render sisi server masih menawarkan kemungkinan untuk visualisasi data *realtime*. Namun, karena banyaknya faktor pengaruh yang berbeda, tidak dapat secara pasti mengkonfirmasi hasil untuk bertahan dalam situasi yang berbeda dengan lingkungan lain selain yang telah digunakan dalam penelitian ini.

Dalam jurnal Lyxeel [13]**,** Dalam beberapa tahun terakhir, teknologi pengembangan aplikasi web telah meningkat pesat dan *framework JavaScript* seperti *Next.js* dan *Angular* semakin populer. Kerangka ini menggunakan *rendering* sisi klien dan telah menyebabkan peningkatan popularitas metode *rendering* dibandingkan conquistador sebelumnya,*rendering* sisi server. Namun, *rendering* sisi server masih merupakan opsi yang layak dalam beberapa kasus. Pada penelitianya, dibangun dua aplikasi web identik yang berisi grid. Metode *rendering* menggunakan sisi server pada satu aplikasi dan metode *rendering* menggunakan sisi klien pada aplikasi lainnya. Metrik evaluasi dari metode *rendering* adalah waktu untuk merender grid dan seberapa bagus pengalaman *user* yang mereka berikan.

Hasilnya menunjukkan bahwa *Server-side Rendering* menawarkan waktu *rendering* yang lebih cepat dan pengalaman *user* yang lebih baik, terutama pada grid yang lebih besar. *Rendering* sisi server lebih cepat daripada *rendering* sisi klien pada semua ukuran grid dan mereka memiliki pengalaman *user* yang serupa. Secara keseluruhan, Lyxeel menyarankan penggunaan *Server-side Rendering*. Mengenai *rendering* sisi server, Lyxeel merekomendasikan untuk menggunakannya dengan hati-hati karena memerlukan implementasi dan pemeliharaan yang kompleks serta biaya yang lebih tinggi. Oleh karena itu, ini tergantung pada persyaratan aplikasi dan jika itu sepadan dengan implementasi, biaya, dan pemeliharaan yang rumit.

Fx. Guntur Putra Susanto, Nizar Izzuddin Yatim Fadlan, dan Prita Haryani, S.Pd., [14] meneliti tentang penggunaan *Next.js* dalam pengembangan Web Management Informasi Kemahasiswaan di kabupaten Kedal. Pengembangan sistem informasi menjadi sangat penting untuk mempermudah pengelolaan organisasi kemahasiswaan daerah dan meningkatkan partisipasi dan keterlibatan anggota. Penelitianya difokuskan pada tahap perancangan sistem informasi yang meliputi perancangan basis data dan perancangan interface dimana tujuannya adalah merancang sistem informasi organisasi kemahasiswaan di wilayah Kendal

menggunakan teknologi Node.js dan *framework Next.js* yang menggunakan System Development Life Cycle dengan model air terjun untuk metode penelitian. *Next.js* adalah kerangka kerja populer yang menyederhanakan pengembangan aplikasi web dengan *rendering* sisi server dan pembuatan situs statis, meningkatkan kinerja dan keandalan sistem.

Kesimpulannya, perancangan Sistem Informasi Manajemen Berbasis Web Organisasi Kemahasiswaan di Kabupaten Kendal menggunakan *framework Next.js* berhasil menggabungkan berbagai komponen sistem, antara lain diagram *Unified Modeling Language, Entity-Relationship Diagram*, arsitektur sistem, dan desain *userinterface*. Penelitian masa depan harus fokus pada pengembangan dan implementasi sistem yang diusulkan untuk perbaikan lebih lanjut manajemen organisasi mahasiswa di Kabupaten Kendal.

Yusuf [4] dalam jurnalnya menyebutkan bahwa Situs web merupakan sekumpulan dokumen *Hypertext Markup Language (HTML)* statis yang dibangun untuk memudahkan setiap orang berbagi informasi, selama terhubung ke dalam jaringan internet. Salah satu bagian dari sistem sebuah situs web adalah web template. Web template adalah komponen dasar dari sistem web template berguna untuk memudahkan pengembang web merancang ulang sebuah halaman web. Salah satu yang mempengaruhi kinerja halaman web yaitu *loading time*, dimana *loading time* adalah waktu yang dibutuhkan oleh browser agar dapat menampilkan halaman web secara menyeluruh oleh *user* ketika *user* melakukan *request*, selain itu *loading time* merupakan salah satu bagian penting darioptimasi situs web. Optimasi merupakan suatu proses dimana memodifikasi atau merubah sesuatu yang telah ada agar efektifitasnya meningkat. Dalam sebuah situs web, terdapat beberapa konsep dalam optimasi, yaitu First Paint, Time To Interactivity (TTI), First Meaningful Paint (FMP) dan Long Task. Oleh karena itu pada penelitianya, Yusuf melakukan teknik optimasi dengan menggunakan critical *rendering* path, above the fold, priority *resource*, bundle and minify, gzip, dan splitting code.

Hasil dari peforma web berdasarkan metriks *first meaningful paint* (FMP), *first contetful paint* (FCP), dan *time to interactivity* (TTI) mengalami peningkatan rata-rata kecepatan (persentase) yaitu FMP sebesar 73%, FCP sebesar 60%, TTI sebesar 50%, dan *loading time* sebesar 29%. Selanjutnya, pada *resource* file rata-rata ukuran file menurun sebesar 59% dan jumlah *request* file menurun sebesar 21

Tabel 1 Tabel penelitian terkait

No	Peneliti	Judul	Hasil Penelitian	Perbedaan Penelitian
(1)	(2)	(3)	(4)	(5)

No	Peneliti	Judul	Hasil Penelitian	Perbedaan Penelitian
1.	Harish A Jartarghar, Girish Rao Salanke, Ashok Kumar A.R, Sharvani G.S and Shivakumar Dalali [9]	*Next.js Apps with Server-side Rendering: Next.js*	*Rendering Next.js* dapat mengoptimalkan kinerja aplikasi dan juga meningkatkan SEO. *Next.js* memiliki kemampuan untuk mempercepat web melalui aplikasi yang meningkatkan kinerja, mengurangi biaya pengembangan, dan memiliki tingkat produksi yang lebih tinggi.	Penelitian terdahulu memiliki fokus utama pada penggunaan *Server-side Rendering* dengan *Next.js* saja tanpa menjalani studi kasus tertentu. Di sisi lain, penelitian ini berbeda dalam hal fokus dan konteks. peneliti memilih untuk melakukan analisis performa *rendering* dengan pendekatan *loading* halaman web berbasis *Next.js*, dan menggunakan aplikasi "App Filmku" sebagai studi kasus khusus.
2.	Muharman Lubis, Tien Kusumasar, Arif Ridho Lubis [10]	*Comparison between Client-side and Server-side Rendering*	Hasilnya menunjukkan bahwa sebagian besar Developer telah	pada penelitian terdahulu hanya berfokus pada dua metode *rendering*, untuk penelitian ini

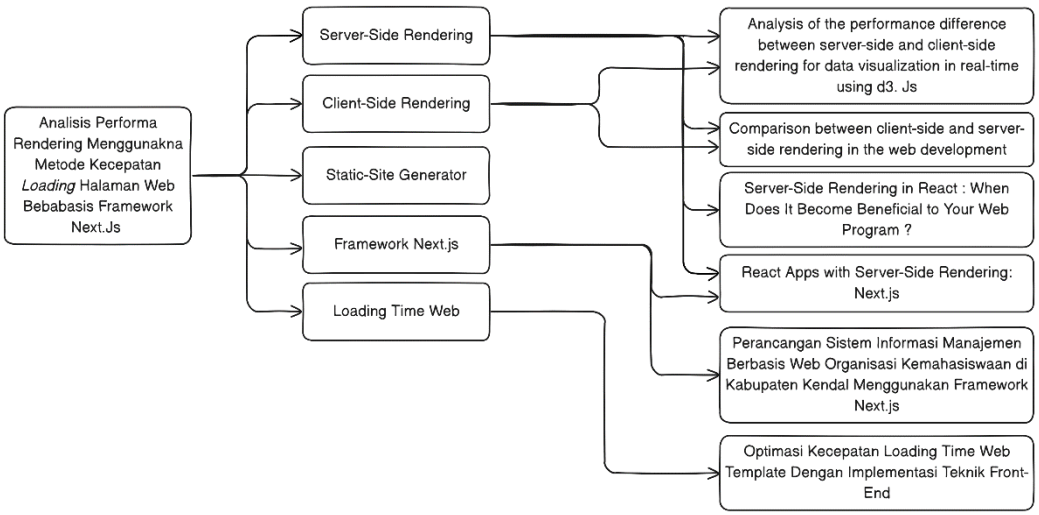
No	Peneliti	Judul	Hasil Penelitian	Perbedaan Penelitian
		in the web development	menggunakan perenderan sisi server dalam proyek mereka dan keuntungan dari perenderan sisi server mencakup peningkatan SEO dan pemuatan halaman awal yang lebih cepat.	berkembang menjadi tiga metode *rendering* pada *Next.js*.
3.	Johan Lotto, Oscar Rosquist [12]	*Analysis of the performance difference between server-side and Client-side Rendering for data visualization in realtime using d3. Js*	Saat memanfaatkan lingkungan yang dibuat dalam penelitian ini, *rendering* sisi klien lebih cepat daripada *rendering* sisi server. Render sisi server masih menawarkan kemungkinan untuk visualisasi data *realtime* pada banyak klien, tetapi tidak dengan set data dengan ribuan elemen.	Dalam hal study kasus tentu saja berbeda, pada penelitian terdahulu berfokus pada *realtime* data, sedangkan untuk penelitian kali ini berfokus pada performa *loading* halaman pada *Website*nya.

No	Peneliti	Judul	Hasil Penelitian	Perbedaan Penelitian
4.	Oskar Lyxell [13]	*Server-side Rendering in Next.js : When Does It Become Beneficial to Your Web Program ?*	kinerja SSR dan CSR ketika diterapkan pada program web dan lebih khusus lagi pada grid. Menunjukkan bahwa SSR memang lebih cepat karena mengalahkan CSR di setiap pengujian. Tren hasil menunjukkan bahwa SSR lebih baik pada halaman dengan banyak konten karena kesenjangan kecepatan. Itu sebelum SSR menunjukkan bahwa ia memiliki kekurangan akibat penurunan yang dilakukannya.	Pada penelitian sebelumnya hanya berfokus pada satu tehnik *rendering*, dan pada penelitian ini akan mengembangkan pada metode-metode *rendering* yang lain, serta diujikan pada beberapa kasus yang berbeda.

No	Peneliti	Judul	Hasil Penelitian	Perbedaan Penelitian
5.	Fx. Guntur Putra Susanto, Nizar Izzuddin Yatim Fadlan, dan Prita Haryani, S.Pd.[14]	Perancangan Sistem Informasi Manajemen Berbasis Web Organisasi Kemahasiswaan di Kabupaten Kendal Menggunakan *Framework Next.js*	Perancangan Sistem Informasi Manajemen Berbasis Web Organisasi Kemahasiswaan di Kabupaten Kendal menggunakan *framework Next.js* berhasil menggabungkan berbagai komponen sistem, antara lain diagram *Unified Modeling Language, Entity-Relationship Diagram, arsitektur sistem*, dan desain *interface user*.	Pada penelitian sebelumnya hanya memfokuskan penelitian pada penggunaan framwork *Next.js* saja, untuk penelitian ini akan memfokuskan pada sisi *rendering framework Next.js* dan dalam penggunaanya.
6.	Yusuf [4]	Optimasi Kecepatan *Loading Time* Web Template Dengan Implementasi Teknik Front-End	Hasil dari peforma web berdasarkan metriks *first meaningful paint (FMP), first contetful paint (FCP), dan time to interactivity (TTI)* mengalami	Pada penelitian terdahulu pengoptimasian *loading time* web terdapat pada front end nya, untuk penelitian ini peneliti akan melakukan pada

No	Peneliti	Judul	Hasil Penelitian	Perbedaan Penelitian
			peningkatan rata-rata kecepatan (perosentase) yaitu FMP sebesar 73%, FCP sebesar 60%, TTI sebesar 50%, dan *loading time* sebesar 29%. Selanjutnya, pada *resource* file rata-rata ukuran file menurun sebesar 59% dan jumlah *request* file menurun sebesar 2	tehnik *rendering* yang digunakan pada pembuatan *Website*nya.

Diagram *mind map* dapat dilihat pada gambar berikut ini

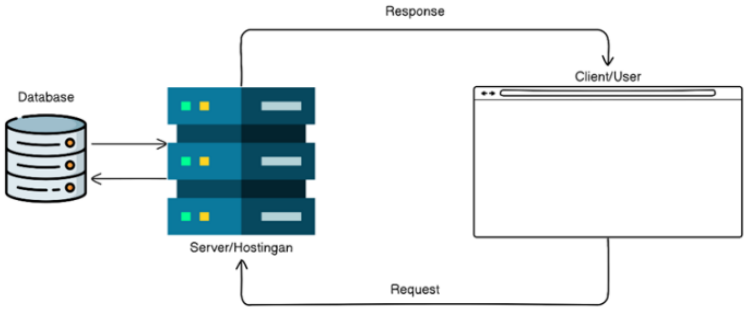


Gambar 1 Mind Map Penelitian Terkait

B. Landasan Teori

1. Konsep Dasar Rendering

   Pada dasarnya *rendering* adalah sebuah rangkaian kinerja pada saat menampilkan sebuah *Website* [9]. Hal ini mencakup sebuah pengolahan code pada file JS atau Html, pengambilan data dari *database* dan kemudian mengirimkan ke clien.



Gambar 2 Konsep dasar *Rendering*

   Pada dasarnya Next.*js* memiliki 5 tehnik *rendering* yaitu *Server-side Rendering*, *Client-side Rendering*, *Static Site Generation*, Automatic Static Optimazion, dan Edge and Node.js Run*time*. Tetapi yang sangat populer dan dapat efektif untuk *Rendering* halaman serta *fetching* data dari *database* adalah SSR, CSR, dan SSG. Oleh sebab itu penelitian ini hanya mencakup tiga metode *rendering* tersebut.

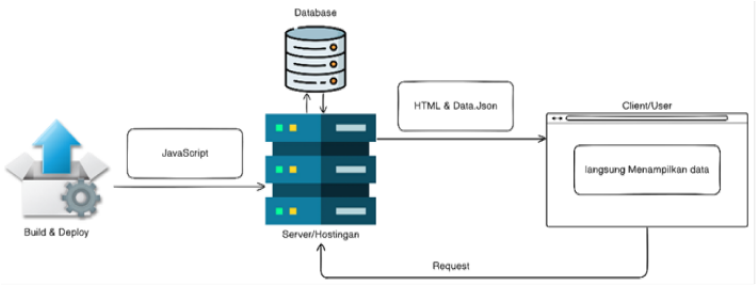
2. *Server-side Rendering* (SSR)

   Render sisi server (SSR) atau "render dinamis" adalah teknik *rendering* di mana kode HTML halaman dihasilkan pada setiap permintaan. Server-side rendering (SSR) merender halaman web di server dan mengirimkan HTML yang telah dirender sepenuhnya ke klien [15]. Jika suatu halaman menggunakan SSR, HTML untuk halaman tersebut dibuat berdasarkan setiap permintaan. Untuk menggunakan SSR pada suatu halaman, harus mengekspor fungsi asinkron bernama "getServerSideProps". Fungsi ini akan dipanggil oleh server pada setiap permintaan.

   Ketika *user* meminta halaman web, server menyiapkan halaman tersebut dengan mengambil data spesifik *user* dan mengirimkannya ke komputer *user*. Browser kemudian akan menafsirkan konten dan menampilkan halaman. Seluruh proses mengambil data dari *database*, membuat halaman HTML dan menampilkannya kepada *user* disebut SSR. *Next.js* menghasilkan halaman HTML pada waktu pembuatan dan menyajikan halaman yang telah dirender sebelumnya dari server ke browser dengan kode *JavaScript* minimal. Saat halaman dimuat oleh browser, kode *JavaScript*-nya dijalankan dan membuat halaman tersebut sepenuhnya interaktif (Proses ini disebut hidrasi).

   a. Cara kerja SSR

Pada tahapan *build* SSR tidak Membuat file Html, jadi yang di*deploy* ke Server adalah file JS, *User* meminta halaman web (melalui browser). Browser kemudian terhubung ke server, dan fungsi getServerSideProps dijalankan untuk *Fetching* data, setelah data didapat kemudian di buat File Htmlnya dan dikirim ke *Client*. Browser kemudian mengunduh Html serta Data yang telah dikirim, yang sudah tersedia di server. Halaman sekarang dimuat penuh dan dapat merespons interaksi yang dilakukan oleh *user*. Ketika *database* berubah data yang ada di halaman web tidak akan berubah sebelum *Client* melakukan *request* kembali.



Gambar 3 Cara kerja SSR

b. Kelebihan dan kekurangan SSR

Dibandingkan dengan CSR, SSR mendapat manfaat dari optimasi mesin pencari sehingga memungkinkan situs web yang menerapkan SSR mendapat peringkat lebih tinggi, seperti di pencarian Google. Memang benar, robot alat ini tidak kesulitan menemukan file HTML statis. Namun, crawler masih cenderung mengalami masalah dengan *JavaScript*. Karena SSR merender konten terlebih dahulu, kecepatan pemuatan halaman awal akan lebih cepat.

Namun, hal ini tidak selalu terjadi. Ini mungkin bergantung pada cache, pengunjung baru mungkin melihat waktu muat lebih cepat tetapi ketika pengunjung sebelumnya mencoba mengakses situs, hasilnya mungkin berbeda. SSR juga memiliki LCP yang lebih cepat. LCP adalah singkatan dari *"Latest Content Paint"* dan merupakan istilah yang digunakan untuk menggambarkan waktu yang diperlukan untuk merender konten terbesar di halaman web. Bisa berupa gambar atau blok teks, misalnya.

Jika konten yang lebih besar bersifat statis maka SSR mempunyai keuntungan karena merendernya di server daripada di browser. SSR biasanya memberikan CLS *(Cumulative Layout Changes)* yang lebih rendah. CLS digunakan untuk melacak jumlah perubahan konten dalam ukuran dan posisi. Ini memiliki konten SSR yang lebih rendah karena dilakukan di server. Jika hal ini harus dilakukan di browser maka browser harus melalui setiap langkah proses *rendering*, yang biasanya menghasilkan lebih banyak perubahan. Baik CLS dan LCP dikenal sebagai skor Core Web Vital, yang merupakan jenis skor yang diukur Google dalam algoritma peringkat pencariannya.

Namun, selain banyak kelebihan yang telah disebutkan, ada juga beberapa kelemahan yang disebutkan. Menerapkan SSR akan menambah kompleksitas program Anda. Itu belum tentu negatif, tapi itu sepadan. Seperti disebutkan sebelumnya, Google memiliki metrik dalam algoritma peringkat pencariannya yang disebut Core Web Vitals. Ini merupakan total tiga tindakan dan dua tindakan yang telah disebutkan sejauh ini.

Metrik ketiga adalah FID *(First Input Delay)*. FID adalah waktu sejak file HTML statis siap dirender hingga *JavaScript* yang tertanam dalam HTML dirender. Atau untuk menyederhanakannya, waktu yang berlalu sejak *user* membuka halaman hingga mereka dapat berinteraksi dengannya. Bisa juga disebut sebagai TTI seperti yang disebutkan sebelumnya. Jika suatu situs memiliki FID yang tinggi, beberapa trik UI dapat diterapkan agar tidak membingungkan *user*. Antarmuka *user* mungkin menerapkan beberapa indikator untuk memberi tahu *user* bahwa halaman tersebut belum siap untuk berinteraksi. Misalnya menonaktifkan tombol hingga siap ditekan.

SSR juga memiliki caching yang kurang efisien. Halaman dengan CSR dapat memanfaatkan cache browser secara maksimal karena halaman HTML awal sama untuk semua halaman di situs. Oleh karena itu, dapat dihosting dan dimuat menggunakan CDN (*Content Delivery Network*).

Tabel 2 Kelebihan dan kekurangan SSR

No.	Kelebihan	Kekurangan
1	Tidak ada tampilan *loading* pada halaman web	*User* menunggu data-*fetching* dan *build* dahulu, kemudian halaman web muncul
2	Bagus untuk SEO	Server bisa kecapeaan, karena setiap 1 *request* server perlu *fetch* dan *build*
3	Bisa untuk personalized conten	
4	Datanya *realtime*	

3. *Client-side Rendering* (CSR)

Client-side Rendering berarti semua proses dirender langsung di browser menggunakan JS. Itu termasuk semua logika, pengambilan data, *routing*, dan *templating* [13]. *Client* mengacu pada perangkat yang di gunakan, seperti *smartphone* atau komputer. Perangkat mengirimkan permintaan ke server dan menampilkan *interface* yang dapat kita gunakan untuk berinteraksi[7]. *React.js*, *Angular*, dan *Vue.js* menggunakan *Client-side Rendering* yang berjalan di browser. *Client-side Rendering* mengacu pada penggunaan *JavaScript* untuk menampilkan konten di browser. Oleh karena itu, hanya dokumen HTML minimum yang berisi file *JavaScript* yang diterima pada

pemuatan pertama. Alih-alih mendapatkan semua konten langsung dari dokumen HTML, halaman web lainnya dirender menggunakan browser.

Terkadang pemuatan halaman awal sering kali agak lambat saat merender sisi klien, tetapi semua halaman dimuat dengan cepat setelahnya. Teknik ini berkomunikasi dengan server untuk mendapatkan data *runtime*. Selain itu, setelah setiap panggilan server, seluruh UI tidak perlu dimuat ulang. Dengan merender elemen DOM tertentu, kerangka sisi klien dapat menyegarkan interface *user* dengan data yang diperbarui [9].

Dalam *Client-side Rendering* (CSR) menggunakan *Next.js*, browser mengunduh halaman HTML minimal dan *JavaScript* yang diperlukan untuk halaman tersebut. *JavaScript* kemudian digunakan untuk memperbarui DOM dan merender halaman. Ketika aplikasi pertama kali dimuat, *user* mungkin melihat sedikit penundaan sebelum mereka dapat melihat halaman penuh, ini karena halaman tidak sepenuhnya dirender sampai semua *JavaScript* diunduh, diurai, dan dijalankan.

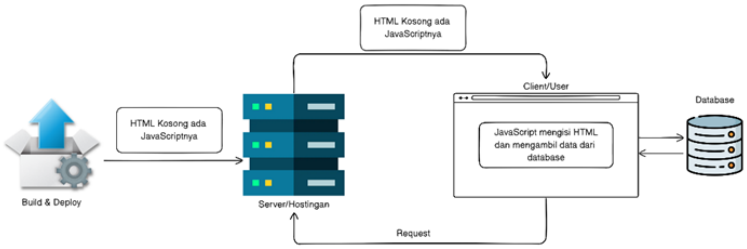
Setelah halaman dimuat untuk pertama kalinya, menavigasi ke halaman lain di situs web yang sama biasanya lebih cepat, karena hanya data yang diperlukan yang perlu diambil, dan *JavaScript* dapat merender ulang bagian halaman tanpa memerlukan penyegaran halaman penuh.

CSR sendiri memiliki pengaruh yang besar pada SEO. Beberapa mesin *Search* mungkin tidak menjalankan *JavaScript* sehingga hanya melihat status kosong atau status pemuatan awal aplikasi. Hal ini juga dapat menyebabkan masalah kinerja bagi *user* dengan koneksi internet atau perangkat yang lebih lambat karena mereka harus menunggu semua *JavaScript* dimuat dan dijalankan sebelum mereka dapat melihat keseluruhan halaman. *Next.js* mempromosikan pendekatan hibrid yang memungkinkan menggunakan kombinasi *rendering* sisi server, pembuatan web statis, dan *rendering* sisi klien untuk memenuhi kebutuhan setiap halaman aplikasi. Di *router* aplikasi, *Next.js* juga dapat menggunakan *interface Loading* UI dengan Tension untuk menampilkan indikator pemuatan saat halaman dirender.

a. Cara kerja CSR

Pada saat *build* CSR akan membuat file Html dan di *deploy* ke server, *User* meminta halaman web (melalui browser). Kemudian Server / CDN *(Content Delivery Network)* menanggapi permintaan dengan halaman HTML yang berisi tautan ke file *JavaScript* penting. Halaman tidak terlihat oleh *user* dan hanya halaman dengan tanda pemuatan yang dapat dilihat, Browser mengunduh *JavaScript* dan *database* melalui tautan dalam HTML. *JavaScript* kemudian dieksekusi melalui *framework*. *User* hanya dapat melihat *placeholder* pada tahap ini. Saat pengunduhan selesai *User* dapat melihat dan berinteraksi dengan halaman. Saat *database* berubah maka data yang ada di halaman web langsung berubah secara *realtime* dengan trigger.an *build* SSR tidak Membuat file Html, jadi yang di*deploy* ke Server adalah file JS, *User*

meminta halaman web (melalui browser). Browser kemudian terhubung ke server, dan fungsi getServerSideProps dijalankan untuk *Fetching*.



Gambar 4 Cara kerja CSR

b. Kelebihan dan kekurangan CSR

Tabel 3 Kelebihan dan kekurangan CSR

No.	Kelebihan	Kekurangan
1	Server tidak terbebani untuk *Fetch* dan *build*	*User* menerima halaman dengan tampilan *loading* terlebih dahulu setelah *fetching* data (di browser) selesai barulah muncul kontenya
2	*User* bisa cepat menerima halaman webnya (sebagai tampilan *loading*)	Tidak bagus untuk SEO
3	Bisa untuk personalized content	
4	Datanya *realtime*	

4. Static Site Generation (SSG)

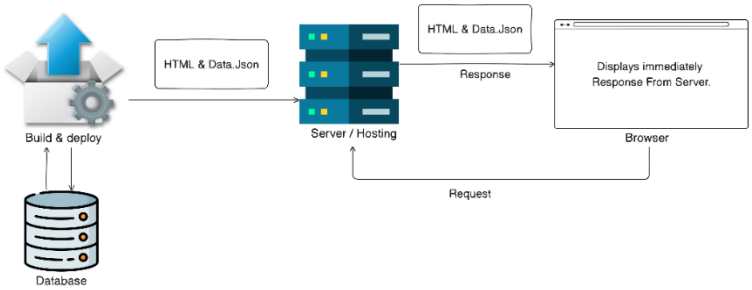
Static *Site Generation* (SSG) adalah alat yang mengubah halaman web yang disematkan menjadi file HTML, CSS, dan *JavaScript* statis untuk terus digunakan oleh *user*. Pada dasarnya SSG adalah istilah umum untuk alat atau perangkat lunak yang mampu menghasilkan halaman statis [11]. *Static Site Generation* (SSG) hadir sebagai cara efektif untuk mengelola *Website*. Dengan SSG atau Jamstack membantu developer tidak perlu lagi menulis banyak kode HTML untuk mengupdate konten *Website*. SSG menyediakan pengelolaan konten situs web yang sederhana, sehingga *user* tidak terlalu memerlukan *database* karena situs statis tidak memerlukan pembaruan rutin [11].

a. Cara kerja SSG

Pada SSG, pengambilan data dilakukan selama proses *build*, kemudian file HTML statis dan JSON dibentuk dan disebarkan ke server. Ketika klien melakukan permintaan, halaman statis langsung diberikan, tanpa proses

pembuatan dan pengambilan data di server atau klien. Yang perlu diketahui adalah jika terjadi perubahan pada data backend maka halaman tidak akan terupdate karena tidak ada trigger untuk generate dan mengambil data tersebut kembali.

Saat proses *build* SSG membuat file Html dan data.json kemudian di *Deploy* ke Server, *User* melakukan *request* kemudian Server mengirimkan file html dan data.json ke *Client*, Browser kemudian menampilkan data yang telah dikirim oleh server. Ketika *database* berubah, data yang ada di halaman dan di server tidak akan berubah sebelum proses *build* dan *deploy* dilakukan kembali



Gambar 5 Cara kerja SSG

b. Kelebihan dan kekurangan SSG

Penggunaan halaman statis akan mempercepat proses *loading* dibandingkan halaman dinamis yang memerlukan pembacaan script dan *database*. Untuk mempublikasikan situs statis tidak memerlukan *hosting* yang mahal karena tidak memerlukan *hosting* yang dilengkapi dengan *database*. Bahkan banyak yang menawarkan hosting gratis untuk SSG, di antara yang paling populer adalah GitHub, GitLab atau Bitbucket, *Developer* kemudian dapat menghubungkan repository GitHub *Developer* ke platform seperti Netlify, GitHub Pages atau Vercel Karena tidak terlalu banyak kode yang berhubungan dengan data atau backend, otomatis kemungkinan adanya celah keamanan akan semakin sempit.

Kelemahan yang paling terlihat adalah proses pengambilan data dari SSG, dikarenakan pengambilan data saat proses *build* dan *Deploy* maka ketika data dalam *database* berubah, maka data di dalam server tidak akan berubah sebelum di buil dan di *Deploy* ulang. Bagi orang awam yang belum familiar dengan terminal atau command line/command prompt pasti akan kesulitan. Apalagi jika malas membaca dokumen yang disediakan. Untuk menambahkan sumber daya eksternal, seperti menambahkan meta tag, plugin atau modul yang dibutuhkan, dilakukan secara manual. Dan lagi-lagi kita harus membiasakan diri dengan command prompt/terminal. Tentu saja sangat berbeda dengan *Website* dinamis seperti CMS karena sangat sederhana [11].

Tabel 4 Kelebihan dan kekurangan SSG

No.	Kelebihan	Kekurangan
1	Server tidak terbebani dengan *Fetch* dan *build*	Tidak bisa untuk personalized content
2	Bagus untuk SEO	Datanya tidak akan update kecuali *build* ulang
3	Tidak ada tampilan *loading*	
4	*User* dengan cepat menerima halaman webnya	

5. Perbandingan SSR CSR dan SSG

Perbedaan utama antara SSR, CSR, dan SSG adalah untuk SSR, server mengirimkan halaman web HTML yang siap dirender ke browser sebagai respons, tetapi untuk CSR, browser hanya menerima dokumen tipis dengan tautan ke *JavaScript*. Artinya, daripada harus menunggu semua *JavaScript* diunduh dan dijalankan, browser akan segera mulai merender HTML dari server [9]. Sedangkan untuk *Static Site Generation* (SSG) memberikan kemampuan pada aplikasi web dalam menghasilkan konten informasi secara dinamis ke dalam tampilan halaman web yang bersifat statis [11]. *Next.js* perlu diunduh dalam kasus tersebut dan akan melakukan langkah yang sama untuk membuat DOM virtual dan menambahkan event untuk membuat halaman menjadi interaktif. Namun, dengan SSR, *user* dapat mulai melihat halaman saat semua proses sedang berlangsung. Agar suatu halaman dapat diakses, CSR mengharuskan semua peristiwa yang disebutkan di atas terjadi sebelum DOM virtual dikirim ke DOM browser.

Tabel 5 Perbandingan SSR, CSR, dan SSG

No.	SSR	CSR	SSG
1	Ideal untuk *Website* static	Ideal untuk web *app*	Untuk halaman web yang tidak berubah tergantung pada *user*
2	Akses halaman awal yang cepat dan stabil	*Rendering* situs cepat setelah pemuatan awal	Akses halaman lebih cepat

No.	SSR	CSR	SSG
3	Tidak ada ketergantungan JS	Interaksi situs yang kaya	Aman, tidak ada binari yang akan diretas
4	Situs yang mudah dijelajahi untuk SEO yang lebih baik	*Negative SEO For Incorrect rendering & API response delay*	Bagus untuk SEO.
5	*Full page reloads*	Dapat memperbarui bagian konten tanpa merender ulang halaman penuh.	*Full page reloads*
6	Lebih membutuhkan waktu untuk berganti halaman	Waktu ganti halaman lebih cepat	Waktu ganti halaman cepat
7	Pengambilan data saat di server	Pengambilan data saat di *Client*	Pengambilan data saat *build* dan *deploy*
8	Saat *request* dari *Client* banyak, server akan sangat terbebani	Saat reques dari *Client* banyak, server akan tetap stabil	Saat *request* dari *Client* banyak, kinerja akan tetap stabil

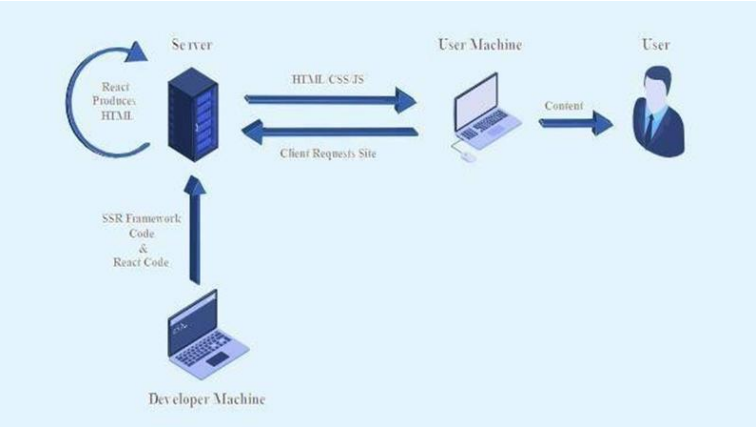
6. *Framework Next.js*

Gambar 6 Logo *Next.js* versi 13

Pada dasarnya *Next.js* adalah perpanjangan dari *React.js*, menggabungkan solusi siap pakai, fitur siap pakai, dan beberapa fungsi tambahan. Dengan kata lain, *Next.js* dibangun di atas *React.js*, untuk memperluas kemampuannya [7]. *Next.js* adalah *Library JavaScript* yang deklaratif, efisien, dan fleksibel untuk *User Interface* (UI) yang interaktif [6]. *Next.js* ini dibuat dan dikembangkan oleh Vercel dengan tujuan untuk mempermudah pembuatan UI dari sebuah aplikasi. *Next.js* JS didukung oleh Facebook, Instagram, komunitas pengembang, dan bisnis. Dalam penggunaanya, *Next.js* mengupayakan kecepatan, kesederhanaan, dan skalabilitas. seiring dengan kemajuan zaman, teknologi juga semakin maju. JQuery mengatasi keterbatasan *vanilla JavaScript*, dan kemudian *React.js* muncul untuk mengatasi kekurangan dan celah jQuery. Namun, *React.js* pun memiliki tantangannya sendiri, yang kini telah diatasi oleh alat lain yang disebut *Next.js* [7]. Keunggulannya terletak pada kemudahan dan produktivitas yang diberikan *Next.js.* Dengan memanfaatkan N*ext.js*, *developer* dapat memanfaatkan fitur-fitur *React* yang ada tanpa memerlukan pengaturan dan konfigurasi yang ekstensif [7].

Next.js digunakan untuk mengembangkan aplikasi statis dan aplikasi yang dirender di server, *Next.js* hadir dengan konfigurasi minimal dan fleksibel yang dapat diperluas tergantung pada tujuan aplikasi [6]. *Next.js* merupakan *framework full-stack*. Artinya, *Next.js* merupakan *framework front-end* untuk membangun tampilan *Website* sekaligus *framework back-end* untuk menangani proses *rendering* dan pengelolaan *database*. *Next.js* menggunakan direktori folder sebagai metode routing untuk halaman web. Dengan menggunakan direktori halaman, *Next.js* menyediakan halaman dengan *automatic routing*, sedangkan sisi server merender pengambilan dan data untuk setiap permintaan.

a. Cara kerja *Next.js*,js



Gambar 7 Cara Kerja *Next.js*

Pada dasarnya *Next.js* dibuat untuk memecahkan masalah *rendering* sisi klien *Next.js*. *Website* yang dibuat dengan *Next.js* "terasa ringan" karena tampilan *Website* sangat interaktif. Ketika data berubah, *Next.js* akan secara efisien memperbarui bagian halaman web yang benar-benar perlu diperbarui, tanpa memuat ulang seluruh halaman. Untuk mencapai semua ini, klien harus memuat semua file *JavaScript* sebelum konten halaman ditampilkan. Jika file JS cukup besar maka waktu download awal juga akan lebih lama. Masalah di atas dapat diselesaikan dengan menggunakan teknik pra-render. Artinya, halaman HTML dan file *JavaScript* dibuat sebelum dikirim ke klien. Ada dua bentuk pra-*rendering*, yaitu *Server-side Rendering* (SSR) dan *Static Site Generation* (SSG).

b. Kelebihan dan kekurangan *Next.js*

Tabel 6 Kelebihan dan kekurangan *Next.js*

No	Kelebihan	Keterangan
1	Dukungan dari komunitas yang besar	Next.js didukung oleh komunitas yang cukup besar dan dinamis, memberikan bantuan yang luas dan sumber daya yang dibutuhkan pengguna untuk mengembangkan aplikasi web. Selain itu, komunitas Next.js terus berkembang dan mengembangkan berbagai plugin dan pustaka yang dapat

No	Kelebihan	Keterangan
		mempercepat pengembangan aplikasi web.
2	Setup projek mudah	Memulai projek dengan *Next.js* sangat mudah bahkan untuk seorang pemula.
3	Deploy projek mudah	Next.js mendukung *deploy* menggunakan berbagai layanan cloud hosting. Hal ini bertujuan untuk memudahkan pengguna menyebarkan aplikasi web mereka dengan mudah.
4	Performa yang lebih cepat	Next.js menawarkan fitur bawaan seperti *Server-side Rendering*, *Static Site Generation*, dan pemisahan kode otomatis, yang mengoptimalkan kinerja aplikasi dengan memungkinkan pemuatan halaman awal lebih cepat, meningkatkan SEO, dan meningkatkan pengalaman pengguna [7].
5	Search engine Optimizion (SEO)	Dengan SSG atau SSR, server mengirimkan file HTML lengkap dan kode JavaScript minimal untuk merender hanya konten yang memerlukan interaksi sisi klien. Hal ini memungkinkan crawler mesin pencari mengakses dengan mudah dan mengindeks setiap halaman website Next.js secara akurat [7].

Namun, daftar fitur yang disediakan oleh Next.js melampaui apa yang telah disebutkan sejauh ini, *Next* menawarkan berbagai kemampuan, termasuk perutean berbasis file yang mulus, pemisahan kode yang efisien, gambar dan optimasi font, HMR (*Hot Module Replacement*), Rute API (*backend*), dukungan bawaan untuk Sass, modul CSS, pilihan pengambilan data (SSG, SSR, ISR), penanganan kesalahan, Metadata API (Untuk SEO), internasionalisasi (dukungan untuk bahasa lisan apa pun), dll [7].

7. *Loading Time Website*

Loading Time *Website* terdiri dari dua suku kata, yaitu *Loading* Time dan *Website*. *Loading Time* adalah waktu yang diperlukan browser untuk menampilkan sepenuhnya informasi situs web yang diminta *user* [16]. *Website* sendiri merupakan media informasi yang bersifat global karena dapat diakses dari mana saja asalkan terhubung dengan internet. Informasi dapat diakses dengan mudah melalui berbagai *device* [4]. Oleh karena itu, jangkauannya bisa lebih luas dibandingkan media tradisional seperti surat kabar lokal, majalah, radio atau televisi [16].

Website pada umumnya di tulis dalam format HTML. Informasi lainya disajikan dalam bentuk grafis (dalam format GIF, JPG, PNG, dll), suara (dalam format AU,WAV,dll), dan objek multimedia lainya (seperti MIDI, *Shockwave Quicktime Movie, 3D World,*dll) [17]. *Website* merupakan fasilitas internet yang menghubungkan dokumen dalam lingkup lokal maupun jarak jauh. Dokumen pada *Website* disebut dengan webpage dan link dalam *Website* memungkinkan *user* bisa berpindah dari satu page ke page lain (*hyper text*), baik diantara page yang disimpan dalam server yang sama maupun server diseluruh dunia. Pages diakses dan dibaca melalui browser seperti Netscape Navigator atau Internet Exploler berbagai aplikasi browser lainnya [17].

Kualitas suatu web server dikatakan baik jika mampu dengan cepat merespon setiap permintaan URL (*Uniform Resource Locator*) *user* dan mampu meminimalisir terjadinya kesalahan. Faktor kestabilan web server sangat penting untuk mempercepat proses *loading* web.

a. Faktor-faktor Pengaruh *Load time* pada *Website*

Ada begitu banyak faktor yang mempengaruhi *Load time* pada web dan sebagian besar berada di luar kendali perancang situs web. Waktu *download* situs web akan ditentukan oleh desain halaman web, server web, perangkat keras klien, konfigurasi perangkat lunak, dan karakteristik *router* internet yang menghubungkan *user* dan situs web. Salah satu temuan penelitian menyebutkan bahwa *Website* yang memiliki waktu *load* lambat kurang menarik dibandingkan *Website* dengan waktu *load* lebih cepat.

Tabel 7 Faktor yang mempengaruhi *Loading* Time *Website*

No	Pengaruh	Keterangan
1	Tehnik Pemrograman	dalam hal ini bahasa pemrograman yang akan dibahas adalah *Vanilla JavaScript*, dasar dari *Fremework Next.js*
2	Penggunaan *JavaScript*	Gunakan *JavaScript*, bahasa pemrograman yang paling populer untuk web. Situs pengembangan *JavaScript* membuat beberapa *Library* baru, khususnya *Jquery*, *Next.js*, dan *Bootstrap Framework*.
3	Penggunaan CSS	untuk "mempercantik" tampilan atau tata letak desain interface *user* halaman *Website*.
4	Tehnik *Rendering*	Tehnik *rendering* yang digunakan dalam berbagai konten, ini mempengaruhi *Load Time* suatu *Website* karena cara pengambilan dan penampilan data pada *Website* biasanya diatur oleh Tehnik *rendering* yang digunakan. Pada *Next.js* sendiri terdapat banyak metode *rendering* yang ditawarkan seperti, *Server-side Rendering*, *Client-side Rendering*, *server-side generation*, dan lain sebaginya.
5	Aspek teknis	berkaitan dengan penggunaan Internet (kualitas sinyal atau jenis jaringan) dan perbedaan penggunaan perangkat keras di Internet seperti

No	Pengaruh	Keterangan
		jenis modem atau *router* tidak akan dibahas dalam artikel penelitian ini.

b. Pengaruh *Loading Time* pada *User*

Meningkatkan pengalaman *user* dapat membantu menarik lebih banyak pengunjung dan membaca konten di situs web atau mendorong pengunjung untuk mengambil tindakan. Jika *Website* lambat maka akan menguji kesabaran pengunjung dan menyebabkan mereka meninggalkan situs sebelum melihat konten yang ditawarkan.

8. Pengujian

Tujuan utama pengujian aplikasi web adalah menjalankan aplikasi menggunakan kombinasi input dan status untuk mendeteksi kegagalan. Kegagalan adalah ketidakmampuan suatu sistem atau komponen untuk melakukan fungsi yang diperlukan sesuai dengan persyaratan kinerja yang ditentukan [18]. Kegagalan tersebut mungkin disebabkan oleh kesalahan selama penerapan aplikasi.

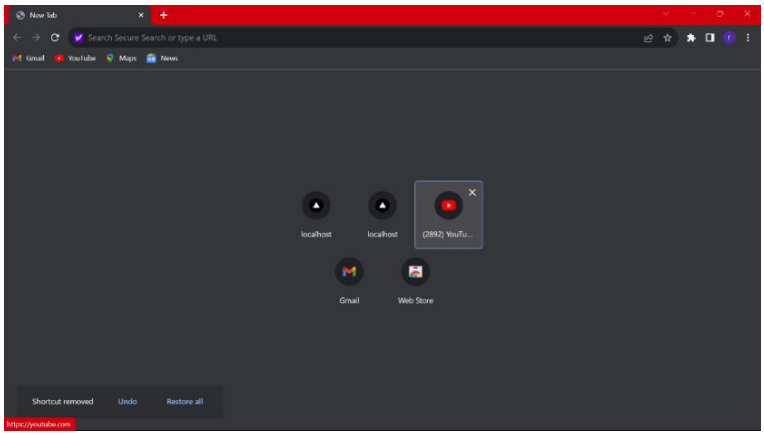
Secara umum, kegagalan akan terjadi terutama karena kesalahan dalam aplikasi itu sendiri, dan kesalahan akan terjadi terutama dalam lingkungan eksekusi atau pada interface antara aplikasi dan lingkungan di mana aplikasi tersebut dijalankan. Karena aplikasi web sangat erat kaitannya dengan lingkungan *runtime*, tidak mungkin mengujinya secara terpisah untuk mengetahui secara pasti komponen mana yang bertanggung jawab atas kesalahan. Oleh karena itu, berbagai jenis pengujian harus dilakukan untuk mendeteksi berbagai jenis kesalahan ini [18].

Lingkungan *runtime* terutama mempengaruhi persyaratan non-fungsional aplikasi web (misalnya kinerja, stabilitas, kompatibilitas), sedangkan aplikasi bertanggung jawab atas daya persyaratan fungsional. Jadi, pengujian aplikasi web harus dipertimbangkan dari dua sudut berbeda. Perspektif pertama mengidentifikasi berbagai jenis pengujian yang harus dilakukan untuk memverifikasi kesesuaian aplikasi web dengan persyaratan *non-fungsional* yang ditentukan. Perspektif lain melihat masalah pengujian persyaratan fungsional suatu aplikasi. Penting bagi suatu aplikasi untuk diuji dari kedua perspektif, karena keduanya saling melengkapi dan tidak saling eksklusif.

Load Testing adalah bentuk pengujian kinerja sederhana dalam bentuk pengujian beban yang dilakukan untuk memahami perilaku sistem pada beban tertentu yang diharapkan. Pengujian akan dilakukan dengan memberikan *feedback* dan data seluruh transaksi yang dijalankan sesuai skenario, sehingga dapat dengan mudah menemukan hambatan atau hambatan yang mempengaruhi kinerja sistem yang diuji [1].

Load Testing digunakan untuk menguji stabilitas dan keandalan sistem. Pengujian ini dapat mengetahui ketahanan sistem dan penanganan kesalahan pada kondisi beban yang sangat berat. Jumlah *user* pada waktu tertentu dapat berbeda-beda, dan pengujian beban menguji bagaimana respons situs web ketika dikunjungi oleh sejumlah *user* tertentu dan dalam jangka waktu tertentu [1].

a. Google Chrome



Gambar 8 *Home page* Google Chrome

Google Chrome adalah browser web lintas platform yang dikembangkan oleh Google. Ini pertama kali dirilis pada tahun 2008 untuk Microsoft Windows, dibuat dengan komponen perangkat lunak gratis dari Apple WebKit dan Mozilla Firefox [19]. Chrome DevTools adalah alat yang digunakan untuk *debugging* di browser Google Chrome. DevTools memberikan akses bagi pengembang situs web untuk melihat kode sumber aplikasi web di browser. DevTools digunakan dalam men-debug tata letak, mengelola *JavaScript*, dan untuk melihat kode dari halaman web.