Perbandingan Teknik Pemuatan Awal Eager Loading dan Lazy Loading Intersection Observer Terhadap Performa Website

Titania Nur Larissa[1], I Made Suartana[2]

[1,2] Program Studi S1 Teknik Informatika, Universitas Negeri Surabaya
[1]titanianur.22022@mhs.unesa.ac.id
[2]madesuartana@unesa.ac.id

Abstrak— Perkembangan teknologi informasi mendorong kebutuhan website dengan pengalaman pengguna optimal, khususnya pada konten visual dominan. Penelitian ini membandingkan performa teknik Eager Loading dan Lazy Loading Intersection Observer pada website Pusat Informasi Keamanan Siber (CSIC) berbasis Laravel. Pengujian menggunakan Chrome DevTools pada enam skenario dengan metrik First Contentful Paint (FCP), Largest Contentful Paint (LCP), Total Blocking Time (TBT), Load Time, Total Request, dan Total Page Size. Hasil pengujian halaman utama menunjukkan Lazy Loading Intersection Observer unggul pada FCP (1,13 detik) dan load time (162 request, 238 MB), sedangkan Eager Loading unggul pada LCP (3,66 detik) berkat atribut fetchpriority dan preload. Pada halaman dashboard, Lazy Loading Intersection Observer mendominasi seluruh metrik dengan load time 0,68 detik, 16 request, dan ukuran halaman 1,7 MB, dibandingkan Eager Loading yang mencapai 157,33 detik, 261 request, dan 241 MB. Pengujian ukuran aset gambar menunjukkan perbedaan performa semakin signifikan pada gambar berukuran besar (>1 MB). Pada halaman berbasis teks (Berita), ketiga teknik menghasilkan performa yang relatif serupa dengan LCP identik 2,23 detik karena tidak bergantung pada konten gambar. Pengujian orientasi layar menunjukkan mode portrait menurunkan performa Lazy Loading Intersection Observer secara signifikan pada halaman utama, dengan TBT mencapai 253,33 ms dan LCP hingga 28,57 detik. Kesimpulan penelitian menunjukkan tidak terdapat satu teknik yang unggul secara mutlak. Lazy Loading Intersection Observer lebih optimal untuk halaman berkonten visual padat dan interaksi berbasis scroll atau tab, sedangkan Eager Loading lebih sesuai untuk halaman yang membutuhkan konsistensi tampilan menyeluruh.

*Kata Kunci— **Eager Loading, Lazy Loading, Intersection Observer, Laravel, Web Performance.***

I. PENDAHULUAN

Perkembangan teknologi informasi yang sangat pesat telah mendorong kebutuhan akan website yang tidak hanya berfungsi sebagai media informasi, tetapi juga mampu memberikan pengalaman pengguna yang optimal. Salah satu fenomena yang muncul adalah meningkatnya jumlah website dengan konten visual yang dominan, terutama berupa gambar beresolusi tinggi. Keberadaan elemen gambar dalam jumlah besar memang mampu memperkaya tampilan, namun di sisi lain dapat menimbulkan permasalahan performa. Website yang lambat dimuat berpotensi menurunkan tingkat kepuasan pengguna, mengurangi kredibilitas, bahkan menyebabkan pengguna meninggalkan laman sebelum konten utama dapat diakses. Hal ini menjadikan optimasi kecepatan muat halaman sebagai salah satu aspek penting dalam pengembangan *web modern.* Dalam praktiknya terdapat dua pendekatan utama dalam pemuatan gambar, yaitu *eager loading* dan *lazy loading. Eager loading* bekerja dengan cara memuat seluruh gambar sekaligus ketika halaman pertama kali diakses, sehingga pengguna dapat langsung melihat seluruh konten visual tanpa interaksi tambahan.

Eager loading memuat seluruh gambar sekaligus saat halaman diakses sehingga konten lengkap tersedia sejak awal, namun meningkatkan beban awal browser. Sebaliknya, lazy loading menunda pemuatan gambar hingga elemen mendekati viewport pengguna sehingga konten utama tampil lebih cepat, meskipun terkadang menimbulkan jeda saat pengguna menggulir ke bawah. Lazy loading melalui IntersectionObserver lebih efisien dalam mengelola konten gambar berukuran besar dibandingkan native lazy loading karena mampu mempercepat waktu pemuatan dan menghemat bandwidth, meskipun konsumsi memorinya lebih tinggi [1].

Pemilihan metode pemuatan harus disesuaikan dengan kebutuhan dan konteks aplikasi web, sebab setiap teknik memiliki keunggulan dan keterbatasan tersendiri [2]. IntersectionObserver lebih unggul dibandingkan native lazy loading, terutama dalam hal kecepatan pemuatan gambar dan efisiensi bandwidth pada website dengan konten visual berukuran besar [1]. Eager loading adalah strategi di mana seluruh data terkait dimuat bersamaan dengan data utama dalam satu query, sehingga mengurangi jumlah query berulang dan menghindari masalah N+1 yang sering muncul pada lazy loading [3], [4]. Pendekatan ini terbukti sangat efektif pada website atau aplikasi dengan konten yang tidak terlalu besar atau ketika seluruh data perlu diakses sekaligus, seperti portal katalog buku, sistem manajemen inventaris, atau

aplikasi e-learning, di mana pengguna membutuhkan akses cepat ke daftar lengkap data tanpa menunggu pemuatan bertahap [4].

Meskipun lazy loading tetap menawarkan keuntungan dalam mengurangi beban memori dan mempercepat waktu muat awal pada website dengan banyak konten visual besar [5], namun konteks penggunaan menentukan strategi terbaik dan dalam banyak kasus eager loading setara atau bahkan lebih unggul. Eager loading menginisialisasi objek dan struktur data sejak awal, termasuk relasi antar tabel dalam basis data [6], dengan tujuan meningkatkan kinerja melalui pengurangan jumlah query [7]. Lazy loading menunda inisialisasi atau pengambilan objek hingga saat dibutuhkan [6], sehingga proses pemanggilan data menjadi lebih efisien dan kinerja program lebih optimal, sekaligus berfungsi sebagai teknik optimasi website dengan memuat elemen seperti gambar atau video hanya saat diperlukan pengguna [1]. Implementasi modern lazy loading menggunakan Intersection Observer API yang memantau posisi elemen secara real-time dan mengeksekusi fungsi callback ketika elemen target memasuki atau keluar dari viewport sesuai ambang batas visibilitas yang ditentukan [8].

Berdasarkan uraian di atas, penelitian ini bertujuan untuk membandingkan teknik eager loading dan lazy loading IntersectionObserver dalam pengembangan website pusat informasi keamanan siber menggunakan Chrome DevTools [10]. Dengan hasil penelitian ini, diharapkan dapat memberikan rekomendasi praktis bagi pengembang web dalam memilih metode pemuatan gambar yang paling sesuai untuk mendukung optimalisasi performa website, dengan metrik FCP, LCP, TBT, Load Time, Total Request, dan Total Page Size [9], [11], [12], [13], guna memberikan rekomendasi praktis bagi pengembang web dalam memilih metode pemuatan yang optimal.

II. METODE PENELITIAN

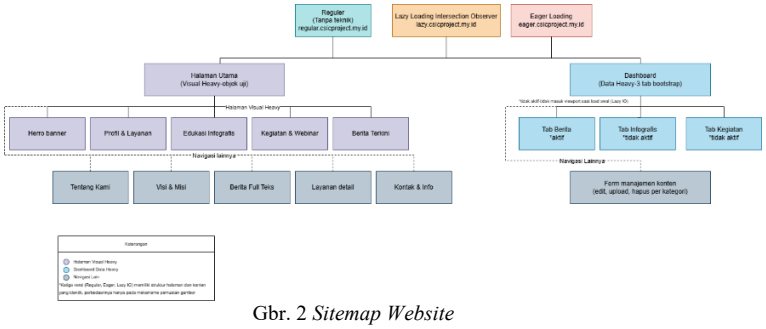
A. *Alur Metode Penelitian*

Penelitian ini dilakukan melalui beberapa tahapan terstruktur pada diagram alur penelitian bertujuan untuk melakukan perbandingan terhadap 2 teknik pemuatan awal yaitu eager loading dan lazy loading intersection observer. Studi literatur dalam penelitian ini dilakukan untuk memahami konsep dasar dari masing-masing teknik pemuatan awal dan penerapannya dalam pengembangan website sesuai dengan kebutuhan website pusat informasi keamanan siber. Selanjutnya ada analisis kebutuhan yang dilakukan untuk mengidentifikasi serta mengetahui apa saja yang diperlukan dalam melakukan sebuah penelitian untuk memenuhi kebutuhan dari masing-masing teknik pemuatan awal yang digunakan. Selain itu terdapat implementasi teknik yang dilakukan untuk mengembangkan website pusat informasi keamanan siber dengan menggunakan 2 teknik pemuatan awal untuk membandingkan performa dan pengalaman pengguna dalam menjalankan sebuah website. Pengujian sistem dilakukan untuk mengevaluasi terkait efisiensi serta kekurangan dan kelebihan dari setiap teknik pemuatan awal yang digunakan dalam pengembangan website, seperti yang diilustrasikan pada Gbr 1.

Gbr. 1 Alur Metode Penelitian

B. *Perancangan sitem*

Sitemap tersebut menunjukkan struktur navigasi website Pusat Informasi Keamanan Siber yang terdiri dari halaman utama dan dashboard. Pada halaman utama terdapat beberapa menu utama seperti Home Banner, Profil & Layanan, Edukasi Infografis, Kegiatan & Webinar, dan Berita Terkini beserta halaman turunannya. Sedangkan pada dashboard terdapat fitur pengelolaan konten berita, infografis, dan kegiatan. Tampilan sitemap website dapat dilihat pada Gbr 2.

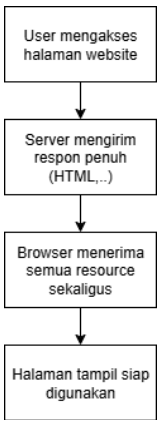


Gbr. 2 *Sitemap Website*

C. *Implementasi Eager Loading dan Lazy Loading Intersection Observer*

1. *Eager Loading*

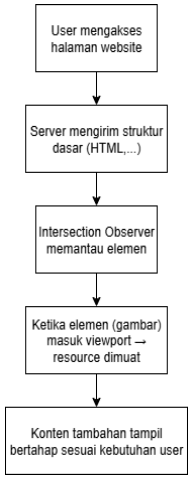
Eager Loading diterapkan dengan metode pemuatan seluruh elemen halaman termasuk gambar, teks, CSS, dan JavaScript secara bersamaan saat halaman pertama kali diakses. Pendekatan ini memungkinkan pengguna langsung melihat konten secara lengkap tanpa interaksi tambahan, sehingga interaksi antar elemen dapat berjalan lancar. Namun, waktu loading awal cenderung lebih lama akibat beban pemrosesan yang besar di sisi browser. Metode ini paling sesuai diterapkan pada website dengan jumlah konten yang terbatas atau ketika seluruh data dibutuhkan sekaligus oleh pengguna. Tampilan diagram implementasi eager loading dapat dilihat pada Gbr 3.



Gbr. 3 Diagram Implementasi *Eager Loading*

2. *Lazy Loading Intersection Observer*

Lazy Loading Intersection Observer diterapkan dengan metode pemuatan konten secara bertahap, di mana hanya elemen inti (HTML, CSS, dan konten utama) yang dimuat saat halaman dibuka. Elemen tambahan seperti gambar atau media baru dimuat ketika memasuki area tampilan (viewport) pengguna, dengan memanfaatkan Intersection Observer API untuk memantau posisi elemen secara otomatis. Pendekatan ini mempercepat waktu muat awal sekaligus menghemat bandwidth, meskipun dapat menimbulkan jeda singkat saat elemen baru muncul. Metode ini sangat cocok diterapkan pada website dengan konten visual yang padat, seperti galeri gambar, pusat informasi, atau portal berita. Tampilan diagram implementasi lazy loading intersecton observer dapat dilihat pada Gbr 4.



Gbr. 4 Diagram Implentasi *Lazy Loading Intersection Observer*

D. Skenario Pengujian

Penelitian ini menggunakan sejumlah skenario pengujian untuk mensimulasikan kondisi pemuatan yang berbeda pada kedua teknik. Setiap skenario dirancang untuk mengidentifikasi faktor-faktor yang berpotensi memengaruhi performa halaman web, sebagaimana ditunjukkan pada TABEL 1.

TABEL 1
SKENARIO PENGUJIAN

No	Aspek yang Diuji	Deskripsi Pengujian	Halaman yang Diuji	Teknik yang Diterapkan
1.	Perbandingan teknik loading	Membandingkan performa antara Eager Loading dan Lazy Loading (Intersection Observer) dengan konten yang sama.	Halaman utama (visual) & dashboard (data)	Eager dan Lazy IO
2.	Uji silang (cross-test)	Menerapkan teknik yang tidak sesuai dengan karakter halaman: Lazy pada dashboard, Eager pada halaman visual.	Halaman utama & dashboard	Eager (dipaksa) dan Lazy IO (dipaksa)
3.	Orientasi layar (mode tampilan)	Menguji performa pada dua model tampilan layer, portrait dan landscape.	Halaman utama & dashboard	Eager dan Lazy IO
4.	Jenis Halaman	Menguji halaman penuh teks.	Halaman berita(penuh teks)	Eager dan Lazy IO
5.	Ukuran aset gambar	Menguji tiga kategori ukuran file gambar kecil (≤100KB), sedang (200–500KB), dan besar (≥1MB).	Halaman utama & dashboard	Eager dan Lazy IO
6.	Jumlah gambar dalam halaman	Menguji dua kondisi jumlah aset gambar, halaman ringan dan halaman padat.	Halaman utama	Eager dan Lazy IO

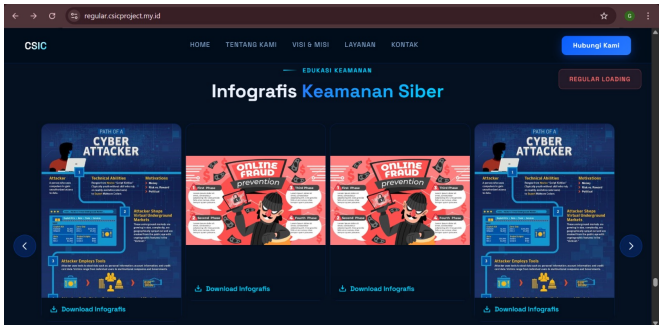
E. Pengujian Teknik

Proses pengujian dilakukan menggunakan Chrome DevTools sebagai alat utama untuk memperoleh data performa secara detail. Metrik yang digunakan dalam pengujian ini meliputi First Contentful Paint (FCP), Largest Contentful Paint (LCP), Total Blocking Time (TBT), Load Time, Total Request, dan Total Page Size.

Penelitian ini menggunakan dua jenis aplikasi web sebagai studi kasus untuk membandingkan efektivitas teknik Eager Loading dan Lazy Loading IntersectionObserver berdasarkan karakteristik konten yang berbeda. Pemilihan dua aplikasi ini bertujuan untuk melihat bagaimana setiap teknik bekerja pada konteks tampilan visual yang berat dan pada aplikasi yang lebih berorientasi data.

1. Aplikasi Website Halaman Utama

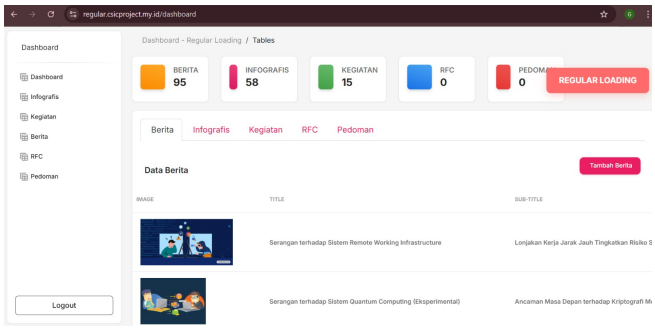
   Halaman Utama, halaman yang menampilkan konten infografis, berita, kegiatan, serta konten besar yang menjadi objek pengujian. Tampilan halaman dapat dilihat pada Gbr 4.



Gbr. 4 Website Halaman Utama

2. Aplikasi Website Dashboard

   Website dashboard terdiri atas halaman ringkasan data berbentuk *card* statistik, tabel daftar data yang menampilkan berbagai jenis konten, serta form upload dan manajemen konten untuk pengelolaan data pada sistem. Tampilan dashboard dapat dilihat pada Gbr 5.



Gbr. 5 Dashboard

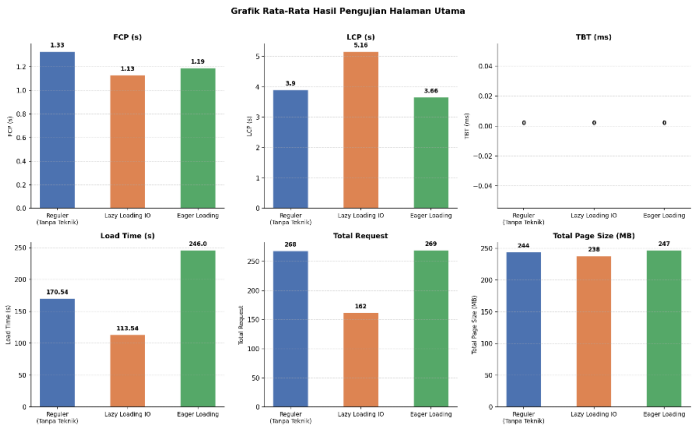
Alur pengujian dirancang secara sistematis untuk menghasilkan data yang valid. Pengujian dilakukan dengan membuka halaman pada browser Google Chrome, mengaktifkan Chrome DevTools pada tab Network, serta menonaktifkan cache melalui opsi *Disable Cache*. Setiap kombinasi teknik dan halaman diuji sebanyak tiga kali, baik pada halaman utama maupun dashboard, dengan mencatat keenam metrik performa yang telah ditentukan. Rata-rata setiap metrik kemudian dihitung untuk meminimalkan bias akibat fluktuasi jaringan atau sistem. Hasil pengujian selanjutnya dibandingkan antara *Eager Loading* dan *Lazy Loading Intersection Observer* guna mengetahui perbedaan efisiensi waktu pemuatan serta dampaknya terhadap performa website secara keseluruhan.

III. Hasil Penelitian dan Pembahasan

A. Hasil Pengujian Halaman Utama

Pada halaman utama, *Lazy Loading* berbasis *Intersection Observer* (IO) menghasilkan nilai FCP terbaik sebesar 1,13 detik, lebih cepat dibandingkan *eager loading* (1,19 detik) dan reguler (1,33 detik), menunjukkan bahwa penundaan pemuatan *resource* yang tidak terlihat mampu mempercepat kemunculan konten pertama. Sebaliknya, *eager loading* unggul pada metrik LCP dengan nilai 3,66 detik dibandingkan reguler (3,90 detik) dan *lazy loading* IO (5,16 detik), yang disebabkan oleh penggunaan atribut *fetchpriority="high"* dan *<link rel="preload">* untuk memastikan elemen *above-the-fold* dimuat lebih awal.

Dari sisi efisiensi jaringan, *lazy loading* IO unggul secara signifikan dengan 162 *request* dan ukuran halaman 238 MB, lebih rendah dibandingkan metode reguler (268 *request*, 244 MB) dan *eager loading* (269 *request*, 247 MB), menunjukkan bahwa *lazy loading* efektif dalam mengurangi jumlah *resource* yang dimuat, terutama untuk konten yang tidak langsung terlihat oleh pengguna [1]. Nilai TBT ketiga teknik tercatat 0 ms, menunjukkan tidak ada pemblokiran *main thread* pada halaman utama dengan orientasi *landscape*. Grafik hasil pengujian halaman utama dapat dilihat pada Gbr 6.



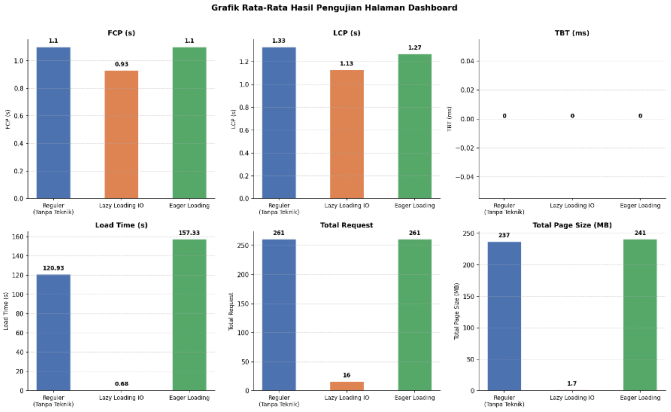
Gbr. 6 Hasil Pengujian Halaman Utama

B. Hasil Pengujian Dashboard

Pada halaman dashboard, *lazy loading* IO mendominasi hampir seluruh metrik pengujian. Nilai FCP *lazy loading*

sebesar 0,93 detik lebih unggul dibandingkan *eager loading* dan reguler yang sama-sama mencapai 1,10 detik, sedangkan LCP *lazy loading* sebesar 1,13 detik lebih cepat dari *eager loading* (1,27 detik) dan reguler (1,33 detik). Seluruh nilai tersebut telah memenuhi target parameter pengujian, yaitu FCP dan LCP di bawah 2,5 detik.

Perbedaan paling signifikan terlihat pada *load time*, di mana *lazy loading* hanya membutuhkan 0,68 detik dengan 16 *request* dan ukuran halaman 1,7 MB. Hal ini disebabkan oleh struktur tab Bootstrap pada dashboard yang membuat sebagian besar gambar belum masuk *viewport* saat halaman pertama dibuka, sehingga *Intersection Observer* belum memicu pengunduhan. Sebaliknya, metode reguler dan *eager loading* sama-sama memuat 261 *request* sekaligus karena tidak memiliki mekanisme penundaan. Hasil ini mempertegas bahwa *lazy loading* memberikan waktu muat awal yang lebih cepat secara signifikan dibandingkan *eager loading* pada konteks yang sesuai [5]. Grafik hasil pengujian dashboard dapat dilihat pada Gbr 7.

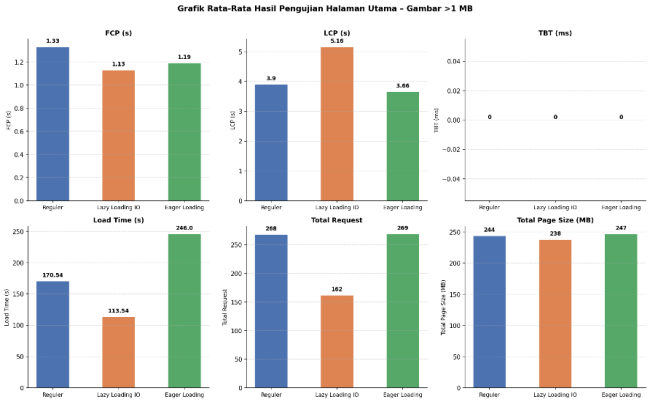


Gbr. 7 Hasil Pengujian Halaman Dashboard

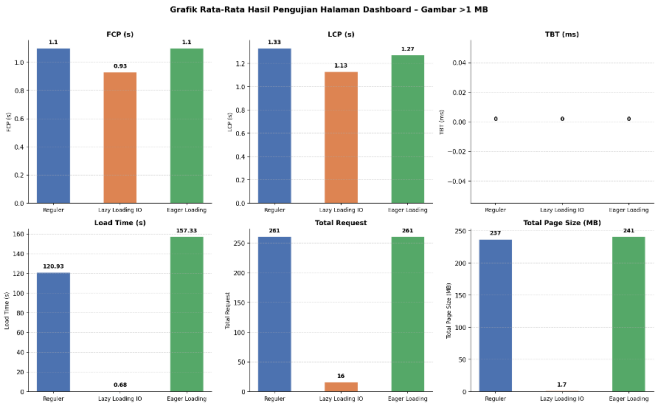
C. *Pengujian Ukuran Aset Gambar*

1. Gambar Berukuran Besar >1MB

Pada kondisi gambar berukuran besar (>1 MB), perbedaan antarteknik menjadi sangat mencolok, terutama pada metrik *load time* dan *total page size*. *Lazy loading* IO terbukti paling efisien dalam mengurangi beban bandwidth awal, baik pada halaman utama maupun dashboard, karena hanya mengunduh gambar yang benar-benar dijangkau pengguna. Sebaliknya, *eager loading* menghasilkan LCP terbaik pada halaman utama berkat mekanisme *preload* dan *fetchpriority*, namun dengan konsekuensi *total page size* tertinggi. Hasil ini selaras dengan temuan bahwa *Intersection Observer* lebih efisien untuk konten gambar berukuran besar karena mampu mempercepat waktu pemuatan sekaligus menghemat bandwidth [1]. Grafik hasil pengujian halaman utama dan dashboard pada gambar berukuran besar >1MB dapat dilihat pada Gbr 8 dan Gbr 9.



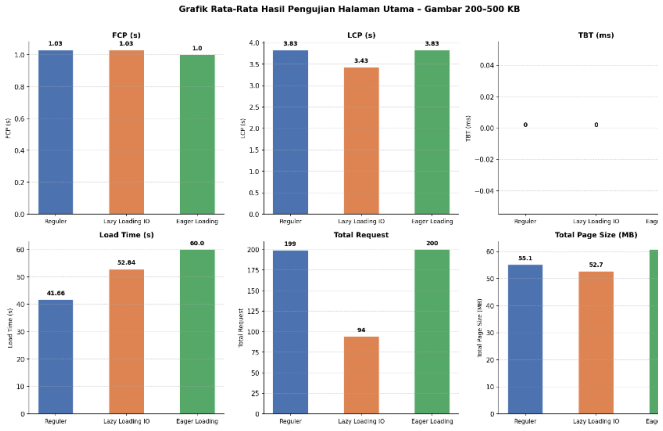
Gbr. 8 Halaman Utama Berukuran Besar >1MB



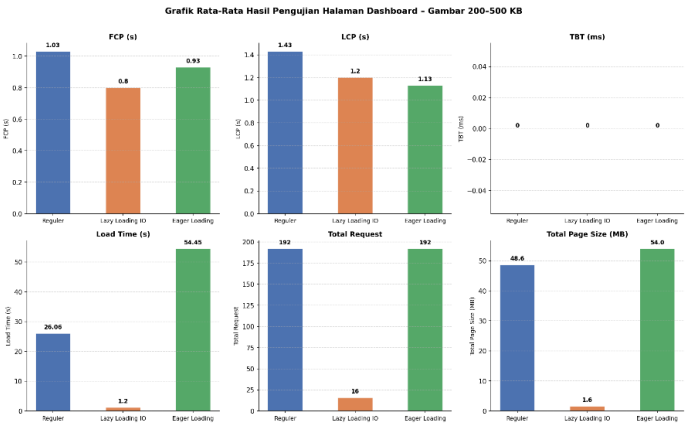
Gbr. 9 Dashboard Berukuran Besar >1MB

2. Gambar Berukuran Sedang 200KB-500KB

Pada kategori gambar berukuran sedang (200–500 KB), pola yang muncul berbeda dibandingkan gambar besar. Pada halaman utama, perbedaan LCP antarteknik menyempit, bahkan *lazy loading* IO menghasilkan LCP terbaik sebesar 3,43 detik karena ukuran gambar yang lebih kecil tidak menimbulkan penundaan berarti pada mekanisme pengunduhan bertahap. *Eager loading* unggul tipis pada FCP (1,00 detik), namun tetap memiliki *load time* tertinggi (60 detik) dan *total page size* terbesar (60,7 MB). Pada halaman dashboard, *lazy loading* IO tetap mendominasi dengan *load time* 1,20 detik dan *total page size* 1,6 MB, jauh lebih efisien dibandingkan metode reguler (26,06 detik) dan *eager loading* (54,45 detik). Grafik hasil pengujian halaman utama dan dashboard pada gambar berukuran sedang 200KB-500KB dapat dilihat pada Gbr 10 dan Gbr 11.



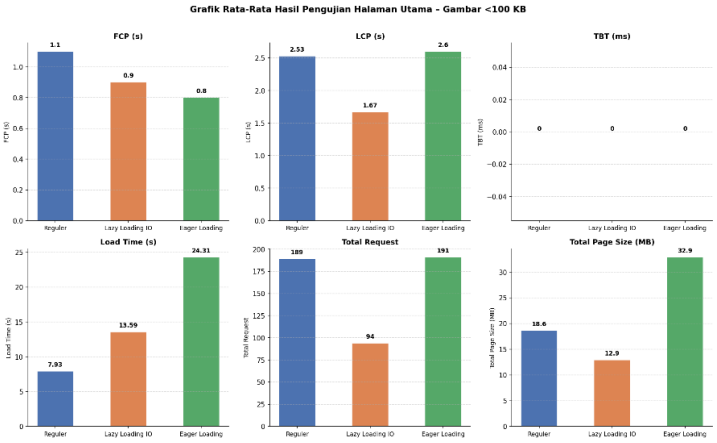
Gbr. 10 Halaman Utama Berukuran Sedang 200KB-500KB



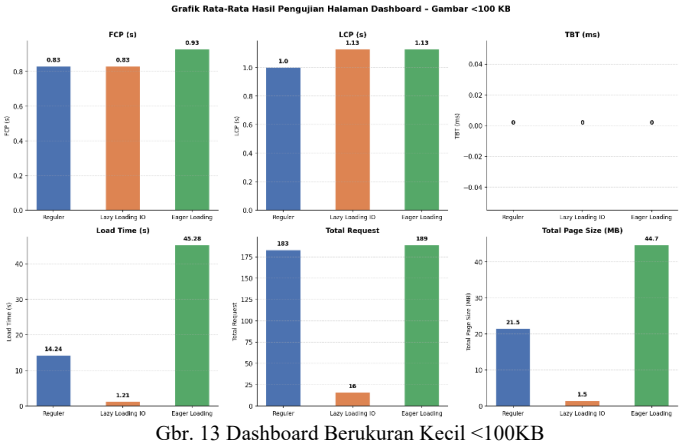
Gbr. 11 Dashboard Berukuran Sedang 200KB-500KB

3. Gambar Berukuran Kecil <100KB

Pada gambar berukuran kecil (<100 KB), *eager loading* menghasilkan FCP terbaik pada halaman utama sebesar 0,80 detik karena mekanisme *preload* dan *fetchpriority* bekerja optimal saat ukuran file kecil, sehingga browser dapat menyelesaikan unduhan gambar prioritas dengan sangat cepat. Meski demikian, *total page size eager loading* (32,9 MB) tetap jauh lebih besar dibandingkan *lazy loading* IO (12,9 MB) karena seluruh gambar tetap diunduh sekaligus. Pada halaman dashboard, *lazy loading* IO tetap unggul dengan *load time* 1,21 detik dan *total page size* 1,5 MB. Pada konteks ini, nilai FCP metode reguler dan *lazy loading* IO menjadi setara (0,83 detik), menunjukkan bahwa keunggulan *lazy loading* semakin terasa seiring bertambahnya ukuran gambar. Grafik hasil pengujian halaman utama dan dashboard pada gambar berukuran kecil <100KB dapat dilihat pada Gbr 12 dan Gbr 13.



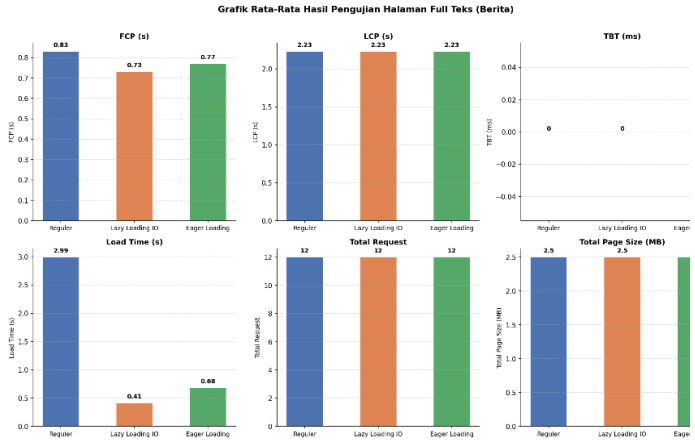
Gbr. 12 Halaman Utama Berukuran Kecil <100KB



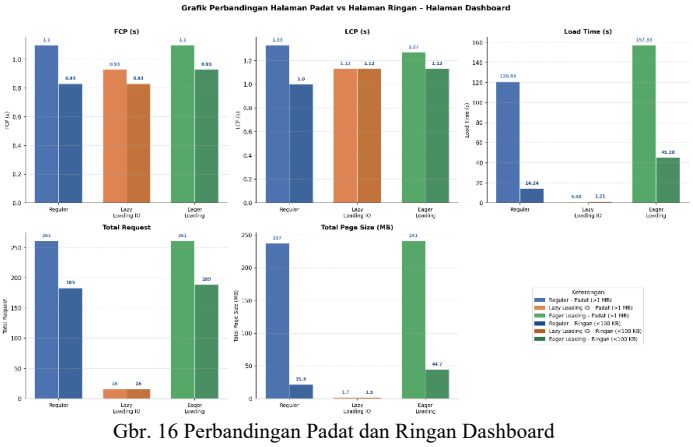
Gbr. 13 Dashboard Berukuran Kecil <100KB

D. Pengujian Jenis Konten (Halaman Full Teks)

Pada halaman *full text*, ketiga teknik menghasilkan performa yang hampir setara pada metrik FCP, LCP, *total request*, dan *total page size*. Nilai LCP ketiga teknik identik sebesar 2,23 detik karena elemen terbesar pada halaman berita adalah blok teks, bukan gambar, sehingga teknik pemuatan gambar tidak memberikan dampak berarti. Perbedaan yang masih terlihat hanya pada *load time*, di mana *lazy loading* IO (0,41 detik) dan *eager loading* (0,68 detik) jauh lebih cepat dibandingkan metode reguler (2,99 detik), karena event *window.load* pada kedua teknik tersebut selesai lebih cepat tanpa menunggu seluruh proses *parsing*. Temuan ini mengonfirmasi bahwa pada konten yang didominasi teks, pemilihan teknik pemuatan gambar tidak berpengaruh signifikan terhadap pengalaman visual pengguna. Grafik hasil pengujian halaman fuul teks dapat dilihat pada Gbr 14.



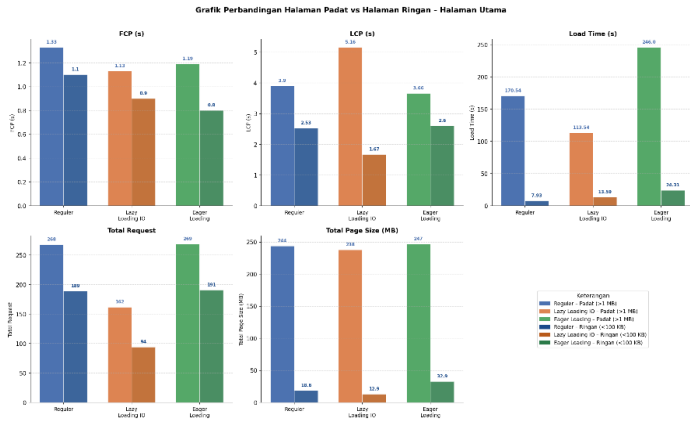
Gbr. 14 Halaman Full Teks



Gbr. 16 Perbandingan Padat dan Ringan Dashboard

E. Pengujian Jumlah Gambar Halaman Padat dan Ringan

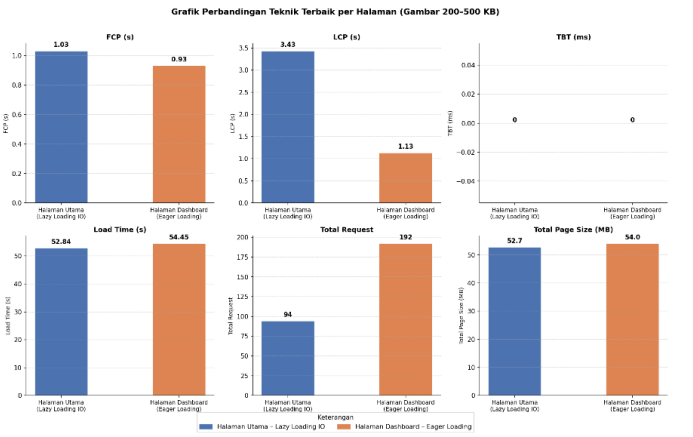
Perbandingan halaman padat dan halaman ringan mengungkap beberapa temuan penting. Pertama, keunggulan *lazy loading* IO dalam efisiensi bandwidth dan *load time* semakin besar seiring meningkatnya jumlah gambar, dengan selisih 106 *request* terhadap *eager loading* pada halaman padat dibandingkan 97 *request* pada halaman ringan. Kedua, *eager loading* mengalami penurunan performa lebih tajam pada halaman padat karena mengunduh seluruh gambar sekaligus, menghasilkan *load time* 246 detik pada halaman utama padat dibandingkan hanya 24,31 detik pada halaman ringan. Ketiga, pada halaman ringan, nilai FCP dan LCP ketiga teknik menjadi lebih berdekatan karena beban bandwidth yang rendah memperkecil dampak perbedaan strategi pemuatan terhadap kecepatan tampil konten awal. Hal ini memperkuat argumen bahwa kepadatan dan ukuran konten gambar merupakan faktor penentu utama dalam pemilihan teknik pemuatan yang optimal [5]. Grafik hasil pengujian halaman padat dan ringan dapat dilihat pada Gbr 15 dan Gbr 16.



Gbr. 15 Perbandingan Padat dan Ringan Halaman Utama

F. Pengujian Perbandingan Teknik Loading (Halaman Utama Lazy IO dan Dashboard Eager Loading)

Pemilihan teknik yang tepat sesuai karakteristik halaman terbukti menghasilkan performa yang kompetitif. *Lazy loading* IO pada halaman utama berhasil menekan *total request* hingga separuhnya (94 vs. 200 pada *eager loading*) dengan LCP lebih baik sebesar 3,43 detik dibandingkan 3,83 detik. Sementara itu, *eager loading* pada halaman dashboard menghasilkan FCP 0,93 detik dan LCP 1,13 detik yang telah memenuhi target parameter di bawah 2,5 detik. Meskipun *load time eager loading* pada dashboard mencapai 54,45 detik akibat pemuatan 192 *resource* sekaligus, hal ini justru memberikan keuntungan konsistensi tampilan tanpa jeda saat pengguna berpindah tab. Temuan ini sejalan dengan pernyataan bahwa pemilihan metode pemuatan harus disesuaikan dengan kebutuhan dan konteks aplikasi web, sebab setiap teknik memiliki keunggulan dan keterbatasan tersendiri [2]. Grafik hasil pengujian perbandingan teknik loading dapat dilihat pada Gbr 17.



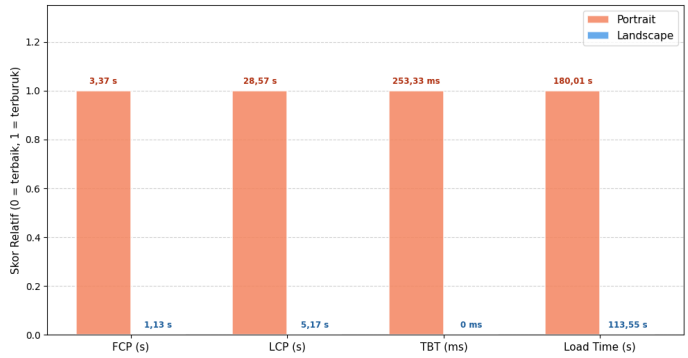
Gbr. 17 Perbandingan Teknik Halaman Utama Lazy IO dan Dashboard Eager Loading

G. Pengujian Orientasi Layar

1. Halaman Utama Lazy Loading Intersection Observer

Performa mode portrait jauh lebih buruk dibanding landscape pada semua metrik FCP portrait 3,37 detik vs 1,13 detik, LCP 28,57 detik vs 5,17 detik, dan TBT 253,33 ms vs 0

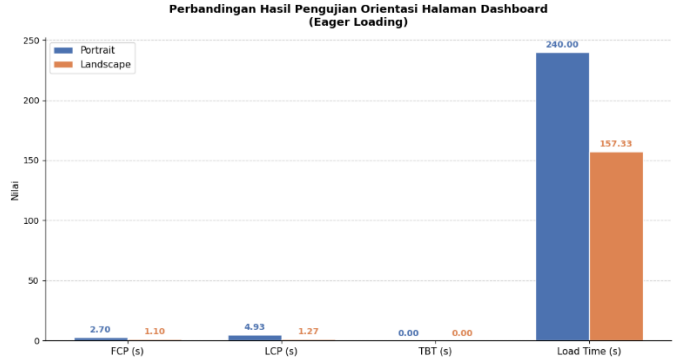
ms. Hal ini disebabkan viewport portrait yang lebih sempit memuat lebih banyak gambar sekaligus saat halaman dibuka, sehingga membebani main thread berbeda dengan landscape yang mendistribusikan gambar lebih melebar sehingga beban lebih ringan. Grafik hasil pengujian orientasi layar halaman utama lazy loading intersection observer dapat dilihat pada Gbr 18.



Gbr. 18 Perbandingan Potrait dan Landscape Halaman Utama Lazy Loading Intersection Observer

2. Dashboard Eager Loading

Performa dashboard mode portrait lebih buruk dibanding landscape FCP 2,70 vs 1,10 detik, LCP 4,93 vs 1,27 detik, dan load time 240,00 vs 157,33 detik. TBT keduanya tetap 0 ms, menunjukkan eager loading tidak memblokir main thread. Perbedaan FCP dan LCP disebabkan tata letak portrait yang memberikan lebih banyak ruang untuk header dan navigasi, sehingga konten utama tabel data lebih lama mencapai viewport. Grafik hasil pengujian orientasi layar dashboard eager loading dapat dilihat pada Gbr 19.



Gbr. 19 Perbandingan Potrait dan Landscape Dashboard Eager Loading

H. Analisis Uji Silang

1. Lazy Loading Intersection Observer pada Dashboard

Lazy loading Intersection Observer pada dashboard memberikan performa terbaik (FCP: 0,93 detik, LCP: 1,13 detik, load time: 0,68 detik) karena struktur tab secara alami cocok dengan mekanismenya gambar dari tab yang belum dikunjungi tidak perlu dimuat sejak awal. Konsekuensinya, pengguna akan mengalami jeda singkat saat pertama kali membuka tab baru.

2. Eager Loading pada Halaman Utama

Eager loading pada halaman utama menghasilkan FCP dan LCP yang baik (1,07 dan 3,60 detik) berkat fetchpriority="high", namun dengan biaya tertinggi 269 request, 247 MB, dan load time 44,03 detik. Seluruh gambar diunduh sekaligus termasuk yang tidak terlihat, sehingga berpotensi menjadi masalah pada koneksi lambat atau perangkat dengan bandwidth terbatas.

I. Pembahasan

Secara keseluruhan, tidak ada teknik pemuatan yang unggul di semua aspek setiap teknik memiliki keunggulan spesifik bergantung pada karakteristik halaman dan prioritas metrik. Eager loading terbukti unggul untuk halaman visual heavy dalam hal FCP dan LCP berkat fetchpriority dan <link rel="preload">, namun kurang efisien dari sisi bandwidth. Sebaliknya, lazy loading Intersection Observer ideal untuk halaman dengan konten tersembunyi di balik interaksi seperti sistem tab, menghasilkan load time, total request, dan ukuran halaman yang jauh lebih kecil, meski rentan terhadap TBT tinggi pada orientasi portrait. Temuan ini sejalan dengan [5] yang menyatakan lazy loading memberikan waktu muat awal rata-rata 40,8% lebih cepat pada konteks yang sesuai, sekaligus mengkonfirmasi [4] bahwa eager loading tetap unggul dalam konsistensi tampilan konten above-the-fold. Pemilihan teknik harus mempertimbangkan karakteristik halaman, jenis konten, dan kondisi jaringan pengguna.

IV. KESIMPULAN

Berdasarkan hasil penelitian perbandingan teknik pemuatan awal Eager Loading dan Lazy Loading berbasis Intersection Observer terhadap performa website, dapat disimpulkan bahwa tidak terdapat satu teknik yang unggul secara mutlak pada seluruh skenario pengujian. Setiap teknik memiliki keunggulan yang berbeda tergantung pada karakteristik halaman dan metrik performa yang menjadi prioritas. Oleh karena itu, pemilihan teknik pemuatan gambar yang sesuai terbukti memberikan pengaruh signifikan terhadap kinerja website.

Pada halaman utama dengan dominasi konten gambar, Lazy Loading Intersection Observer menghasilkan nilai First Contentful Paint (FCP) terbaik sebesar 1,13 detik serta efisiensi jaringan yang lebih tinggi dengan total 162 request dan ukuran halaman sebesar 238 MB. Di sisi lain, Eager Loading menunjukkan performa terbaik pada metrik Largest Contentful Paint (LCP) sebesar 3,66 detik melalui penerapan atribut fetchpriority="high" dan mekanisme preload yang memprioritaskan pemuatan gambar utama. Namun, teknik ini menghasilkan total page size tertinggi sebesar 247 MB dan load time terpanjang mencapai 246 detik.

Pada halaman dashboard, Lazy Loading Intersection Observer menunjukkan performa yang jauh lebih baik dibandingkan Eager Loading. Teknik ini menghasilkan load time sebesar 0,68 detik dengan total 16 request dan ukuran halaman hanya 1,7 MB, sedangkan Eager Loading mencapai load time 157,33 detik dengan 261 request dan ukuran halaman 241 MB. Keunggulan tersebut diperoleh karena gambar yang berada pada tab yang belum terlihat tidak langsung dimuat hingga memasuki viewport pengguna.

Hasil pengujian berdasarkan ukuran aset gambar menunjukkan bahwa semakin besar ukuran file gambar, semakin besar pula perbedaan performa yang dihasilkan. Pada aset gambar berukuran lebih dari 1 MB, Lazy Loading Intersection Observer mampu menghemat bandwidth secara signifikan. Pada ukuran gambar sedang (200–500 KB), teknik ini juga menghasilkan nilai LCP terbaik pada halaman utama sebesar 3,43 detik sambil tetap mempertahankan efisiensi bandwidth dan waktu muat. Sebaliknya, pada gambar berukuran kecil di bawah 100 KB, perbedaan performa antar teknik menjadi lebih kecil dan Eager Loading menghasilkan nilai FCP terbaik sebesar 0,80 detik karena proses preload dapat bekerja secara optimal.

Pada halaman yang didominasi konten teks dengan jumlah gambar minimal, seperti halaman Berita, ketiga teknik menghasilkan performa yang relatif serupa. Nilai FCP berada pada rentang 0,73-0,83 detik dan LCP sekitar 2,23 detik dengan total request serta ukuran halaman yang hampir identik. Temuan ini menunjukkan bahwa pemilihan teknik pemuatan gambar tidak memberikan dampak yang signifikan pada halaman yang tidak bergantung pada konten visual.

Selain itu, pengujian orientasi layar menunjukkan bahwa mode portrait cenderung menghasilkan performa yang lebih rendah dibandingkan mode landscape. Pada halaman utama dengan Lazy Loading Intersection Observer, mode portrait menghasilkan FCP sebesar 3,37 detik dan LCP sebesar 28,57 detik, lebih lambat dibandingkan mode landscape yang masing-masing mencapai 1,13 detik dan 5,17 detik. Kondisi ini juga menyebabkan peningkatan Total Blocking Time (TBT) hingga 253,33 ms akibat lebih banyak gambar yang terpicu untuk dimuat secara bersamaan pada viewport yang lebih sempit. Sementara itu, pada halaman dashboard dengan Eager Loading, mode portrait menghasilkan FCP sebesar 2,70 detik dan LCP sebesar 4,93 detik, lebih tinggi dibandingkan mode landscape yang masing-masing sebesar 1,10 detik dan 1,27 detik.

Berdasarkan hasil uji silang, meskipun Lazy Loading Intersection Observer menghasilkan performa numerik terbaik pada halaman dashboard, teknik ini tetap memiliki konsekuensi berupa jeda singkat ketika pengguna pertama kali membuka tab yang belum dimuat sebelumnya. Sebaliknya, penerapan Eager Loading pada halaman utama memberikan konsistensi tampilan yang lebih baik karena seluruh gambar tersedia sejak awal, namun menghasilkan total request dan ukuran halaman yang sangat besar, yaitu 269 request dan 247 MB, sehingga berpotensi menurunkan performa pada jaringan dengan bandwidth terbatas. Dengan demikian, kombinasi penggunaan kedua teknik sesuai karakteristik halaman menjadi pendekatan yang paling efektif untuk memperoleh performa website yang optimal.

REFERENSI

[1] F. A. Setyowidodo, A. Pinandito, dan M. A. Akbar, "Studi Perbandingan Lazy Loading Native dan IntersectionObserver JavaScript Dalam Pengelolaan Konten Gambar Pada Pengembangan Website," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 9, no. 2, 2025.

[2] R. Ardiyanto dan E. Ardhianto, "Analisa Performasi Metode Client Side Rendering, Server Side Rendering, dan Incremental Static Regeneration dalam Proses Website Rendering," *Computer Science (CO-SCIENCE)*, vol. 4, no. 1, hal. 19–27, 2024.

[3] V. Pandey, "What is the difference between eager and lazy loading in Rails?" *Medium*, 16 April 2024. [Online]. Tersedia: https://imvishalpandey.medium.com/what-is-the-difference-between-eager-and-lazy-loading-in-rails-ba2cedc91907

[4] A. F. S. Harahap, "Analisis Perbandingan Algoritma Lazy Loading Dan Eager Loading Pada Framework Laravel," Skripsi, Universitas Islam Negeri Sultan Syarif Kasim Riau, 2022.

[5] P. Landbris, "Prestanda i Vue-komponenter: Ett experiment med lazy loading och eager loading mot Wagtail CMS," Higher Education Diploma Thesis, Blekinge Institute of Technology, Karlskrona, Sweden, 2025.

[6] B. Krisna, I. Saifudin, dan L. A. Muharom, "Analisis performa aplikasi portal portofolio virtual reality berbasis website dengan Apache Benchmark di PT Telekomunikasi Indonesia," *Journal of Digital Innovation & Information Technology*, vol. 1, no. 1, hal. 34–45, 2024.

[7] J. Yang, P. Subramaniam, S. Lu, C. Yan, dan A. Cheung, "How not to structure your database-backed web applications: a study of performance bugs in the wild," in *Proc. 40th Int. Conf. Software Engineering (ICSE)*, Gothenburg, Sweden, hal. 800–810, May 2018.

[8] MDN Web Docs, "Intersection Observer API," *Mozilla Developer Network*, Aug. 29, 2020. [Online]. Tersedia: https://developer.mozilla.org/en-US/docs/Web/API/Intersection_Observer_API

[9] DebugBear, "Speed Index Metric," *DebugBear Documentation*, 2024. [Online]. Tersedia: https://www.debugbear.com/docs/metrics/speed-index

[10] Google Developers, "Chrome DevTools Overview," 2023. [Online]. Tersedia: https://developer.chrome.com/docs/devtools

[11] Google, "Largest Contentful Paint (LCP)," *Web.dev*, 2024. [Online]. Tersedia: https://web.dev/articles/lcp?hl=id

[12] Google, "First Contentful Paint (FCP)," *Chrome for Developers*, 2024. [Online]. Tersedia: https://developer.chrome.com/docs/lighthouse/performance/first-contentful-paint/

[13] Google, "Total Blocking Time (TBT)," *Chrome for Developers*, 2024. [Online]. Tersedia: https://developer.chrome.com/docs/lighthouse/performance/lighthouse-total-blocking-time?hl=id