ANALISIS KOMPARATIF EFISIENSI KODE DAN PERFORMA ANTARMUKA *SHUTTERSPORT.ID* MENGGUNAKAN *TAILWIND CSS* DAN *BOOTSRAP*

Rio Setiawan [1], An-Nisa Aina Nurwaida[2] , Helmi Abdul Aziz[3]

[1,2,3] Fakultas Komunikasi dan Informasi, Program Studi Rekayasa Perangkat Lunak, Universitas Garut
Email: [1]rio.setiawan@uniga.ac.id, [2]nisaaina@uniga.ac.id, [3]24073122022@fkominfo.uniga.ac.id

Abstrak: Pengembangan antarmuka web modern menuntut keseimbangan antara kecepatan pengembangan dan performa teknis aset digital. Penelitian ini dilatarbelakangi oleh tantangan optimasi pada platform Shuttersport.id, sebuah sistem manajemen aset visual yang memerlukan infrastruktur *front-end* ringan namun fleksibel. Terdapat perdebatan teknis mengenai efisiensi antara paradigma *component-based* yang diwakili oleh Bootstrap dan paradigma *utility-first* oleh Tailwind CSS. Tujuan penelitian ini adalah untuk menganalisis perbandingan efisiensi pengembangan melalui metrik ukuran berkas (*payload*), kompleksitas kode (*Lines of Code*), serta performa akses berdasarkan parameter *First Contentful Paint* (FCP) dan *Largest Contentful Paint* (LCP). Metode yang digunakan adalah eksperimen terkontrol dengan membangun dua prototipe antarmuka yang identik untuk platform Shuttersport.id menggunakan kedua *framework* tersebut. Pengukuran data dilakukan secara kuantitatif menggunakan instrumen Google Lighthouse untuk aspek performa dan analisis statis untuk kompleksitas kode. Hasil eksperimen menunjukkan bahwa Tailwind CSS menunjukkan performa lebih baik dari Bootstrap dengan reduksi ukuran *payload* hingga 80,1% dan memangkas kompleksitas kode sebesar 33,1% melalui pergeseran beban kompleksitas ke dalam markup. Dari sisi performa, Tailwind mempercepat waktu *Largest Contentful Paint* (LCP) sebesar 31,7% dan *First Contentful Paint* (FCP) sebesar 6,7%. Kesimpulannya, paradigma *utility-first* terbukti jauh lebih efisien untuk platform sarat aset visual seperti Shuttersport.id karena mampu meminimalisir beban *rendering*. Implikasi praktis penelitian ini merekomendasikan adopsi Tailwind CSS pada tahap produksi guna memaksimalkan alokasi bandwidth untuk aset gambar utama.

Kata kunci: *Bootstrap; Efisiensi Pengembangan; Front-end; Tailwind CSS; Web Performance*

Abstract: *Modern web interface development requires a balance between development speed and the technical performance of digital assets. This study is motivated by the optimization challenges of Shuttersport.id, a visual asset management platform that requires a lightweight yet flexible front-end infrastructure. A technical debate exists regarding the efficiency of the component-based paradigm represented by Bootstrap and the utility-first paradigm represented by Tailwind CSS. This study aims to analyze the comparative development efficiency of the two frameworks based on file size (payload), code complexity (Lines of Code), and access performance measured using First Contentful Paint (FCP) and Largest Contentful Paint (LCP). A controlled experimental method was employed by developing two identical interface prototypes for the Shuttersport.id platform using the respective frameworks. Data were quantitatively measured using Google Lighthouse for performance evaluation and static analysis for code complexity assessment. The experimental results show that Tailwind CSS outperformed Bootstrap, reducing the payload size by up to 80.1% and decreasing code complexity by 33.1% through shifting a portion of the complexity into the markup. In terms of performance, Tailwind CSS reduced Largest Contentful Paint (LCP) by 31.7% and First Contentful Paint (FCP) by 6.7%. In conclusion, the utility-first paradigm proved to be considerably more efficient for an asset-intensive visual platform such as Shuttersport.id because it minimizes rendering overhead. The practical implication of this study recommends adopting Tailwind CSS in the production stage to maximize bandwidth allocation for primary image assets.*

Keywords: *Bootstrap; Development Efficiency; Front-end; Tailwind CSS; Web Performance*

1. PENDAHULUAN

Dalam ekosistem digital kontemporer, pengembangan antarmuka sisi klien (front-end web development) telah bertransformasi dari sekadar tahap estetika menjadi pilar strategis dalam keberhasilan rekayasa perangkat lunak. Antarmuka pengguna (User Interface) kini berfungsi sebagai gerbang utama interaksi manusia dan komputer yang secara langsung merepresentasikan kualitas sistem, kredibilitas platform, dan profesionalisme sebuah organisasi [1]. Di tengah fragmentasi perangkat keras yang masif serta keberagaman resolusi layar, tantangan fundamental bagi seorang pengembang front-end adalah menjaga

keseimbangan dinamis antara desain yang adaptif, performa rendering yang cepat, serta struktur kode yang memiliki tingkat pemeliharaan (maintainability) dan skalabilitas yang tinggi [2].

Untuk mengatasi kompleksitas tersebut dan mempercepat siklus Software Development Life Cycle (SDLC), industri perangkat lunak sangat mengandalkan penggunaan framework Cascading Style Sheets (CSS) sebagai standar abstraksi desain. Bootstrap, yang dipelopori oleh Twitter sejak tahun 2011, telah menjadi standar de facto industri dengan paradigma berbasis komponen (component-based). Paradigma ini memprioritaskan kecepatan implementasi melalui pustaka elemen siap pakai yang terstandarisasi, yang secara signifikan mampu memangkas waktu pengerjaan prototipe hingga 20-25% dibandingkan metode konvensional [3]. Namun, arsitektur Bootstrap yang mengandalkan kelas semantik kaku sering kali dianggap kurang fleksibel dalam menangani kebutuhan desain yang sangat spesifik tanpa menimbulkan redundansi kode atau CSS bloat yang dapat membebani kinerja browser.

Sebagai respon terhadap keterbatasan tersebut, muncul paradigma utility-first yang dipopulerkan oleh Tailwind CSS. Berbeda dengan pendekatan semantik Bootstrap, Tailwind CSS mengadopsi prinsip desain atomik (atomic design) yang memberikan kendali granular kepada pengembang untuk membangun antarmuka unik langsung pada struktur HTML [4]. Meskipun Tailwind CSS memerlukan kurva belajar yang lebih curam dan waktu pengembangan awal yang relatif lebih lama bagi tim pengembang, framework ini menawarkan efisiensi muatan aset digital yang lebih efisien. Melalui mekanisme optimasi seperti PurgeCSS, Tailwind mampu mereduksi baris kode CSS tambahan hingga 15%, memastikan bahwa hanya kode yang benar-benar digunakan yang akan dikirimkan ke browser pengguna, sehingga meminimalisir waktu muat secara signifikan [3].

Disparitas performa teknis antara kedua framework ini menjadi subjek perdebatan yang krusial dalam disiplin Rekayasa Perangkat Lunak (RPL). Berdasarkan pengujian metrik Core Web Vitals menggunakan Google Lighthouse, Bootstrap ditemukan memberikan stabilitas yang lebih baik pada aspek skor performa dan optimasi mesin pencari (Search Engine Optimization), yang sangat vital bagi platform berbasis konten [1]. Di sisi lain, Tailwind CSS terbukti lebih unggul dalam mencapai akurasi visual terhadap desain mockup asli serta mendukung standar aksesibilitas yang lebih inklusif bagi pengguna dengan kebutuhan khusus [2]. Selain itu, terdapat pertimbangan mengenai pengalaman pengembang (Developer Experience), di mana ketergantungan pada kelas utilitas yang masif pada Tailwind CSS sering dikritik karena menyebabkan struktur HTML menjadi sangat padat (verbose), yang berpotensi meningkatkan kompleksitas kode dalam jangka panjang jika tidak dikelola dengan standar dokumentasi yang ketat [5].

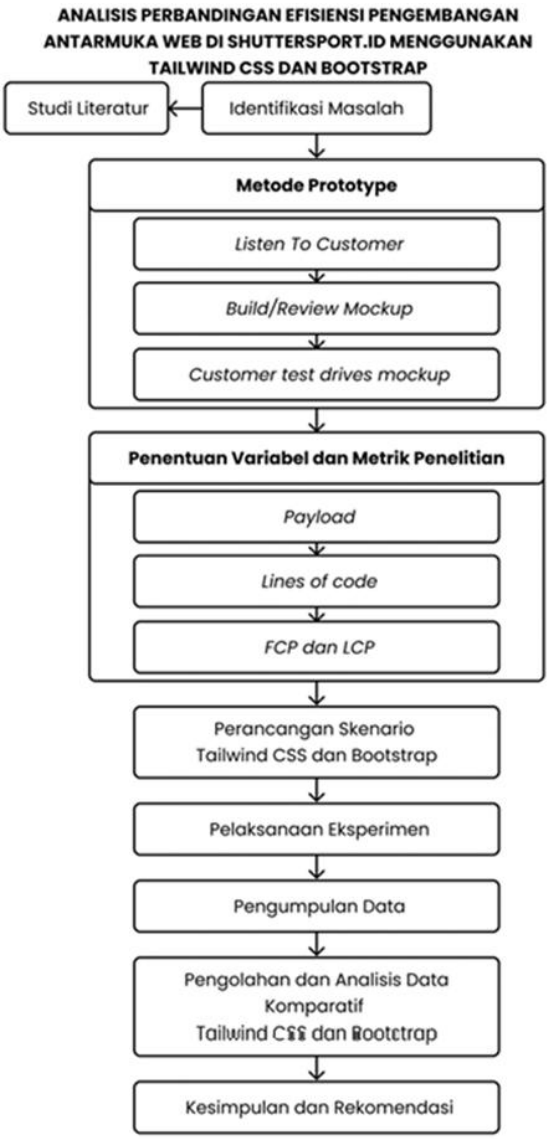
Sebagai entitas bisnis penyedia jasa fotografi olahraga yang sedang berkembang, operasional digital Shuttersport.id saat ini masih sepenuhnya bergantung pada ekosistem platform pihak ketiga, khususnya Instagram dan Linktree. Ketergantungan absolut ini memunculkan hambatan strategis yang krusial, algoritma kompresi media sosial secara agresif merusak detail visual dari aset foto beresolusi tinggi, sementara antarmuka yang kaku membatasi fleksibilitas manajemen portofolio dan menurunkan tingkat profesionalitas di mata klien berskala besar. Untuk mengatasi limitasi arsitektural tersebut, Shuttersport.id tengah berada pada fase transisi untuk membangun platform manajemen aset visual mandiri dari nol. Kondisi greenfield (pembangunan awal) inilah yang menuntut keputusan rekayasa perangkat lunak yang presisi. Ketiadaan infrastruktur legacy memberikan momentum ideal untuk melakukan eksperimen komparatif secara objektif guna menentukan framework front-end mana antara paradigma komponen atau utilitas yang mampu menghasilkan struktur paling efisien untuk menangani beban payload gambar masif sebelum sistem benar-benar diluncurkan ke tahap produksi.

Meskipun literatur mengenai perbandingan antara framework berbasis komponen dan utilitas mulai berkembang, mayoritas penelitian yang ada masih bersifat deskriptif dan terbatas pada pengujian satu arah atau penggunaan templat standar yang bersifat generik [5]. Terdapat kesenjangan penelitian (research gap) yang signifikan dalam hal pengujian komparatif kuantitatif menggunakan metode eksperimen terkontrol pada objek desain yang sama. Penelitian sebelumnya sering kali mengabaikan variabel kontrol pada tata letak dan fungsionalitas, sehingga hasil performa yang didapatkan sering kali menjadi bias terhadap kemampuan individu pengembang, bukan efisiensi murni dari framework itu sendiri.

Dalam penelitian ini, efisiensi pengembangan tidak dimaknai sebagai kecepatan waktu pengerjaan, melainkan sebagai efisiensi luaran kode dan aset yang dihasilkan kedua framework. Melalui pembangunan dua prototipe eksperimental dengan desain yang identik, penelitian ini mengukur tiga hal: keringkasan struktur kode (Lines of Code), ukuran berkas CSS yang dikirim ke peramban (payload CSS), serta performa render berdasarkan standar Core Web Vitals (FCP dan LCP). Aset gambar beresolusi tinggi diperlakukan sebagai konteks sekaligus variabel yang dikontrol identik pada kedua prototipe, bukan sebagai objek ukur utama. Hasil akhir dari penelitian ini diharapkan dapat memberikan kontribusi nyata berupa rekomendasi teknis yang objektif bagi para pengembang perangkat lunak dalam menentukan strategi pengembangan front-end yang paling optimal, efisien, dan kompetitif

2. METODE PENELITIAN

Metodologi studi ini menggunakan metode eksperimen terkontrol dengan pendekatan *prototyping* untuk membandingkan efisiensi pengembangan antarmuka web pada platform Shuttersport.id menggunakan *framework* Tailwind CSS dan Bootstrap. Kerangka pemikiran penelitian diawali dengan identifikasi permasalahan terkait kebutuhan optimasi antarmuka web yang ringan dan efisien untuk pengelolaan aset visual, dilanjutkan dengan studi literatur mengenai paradigma *utility-first* dan *component-based* pada *framework* CSS. Selanjutnya dilakukan perancangan *mockup* antarmuka yang identik sebagai variabel kontrol, kemudian diimplementasikan ke dalam dua prototipe menggunakan Tailwind CSS dan Bootstrap pada lingkungan pengembangan yang sama. Data penelitian dikumpulkan melalui pengukuran kuantitatif terhadap ukuran berkas CSS (*payload*), kompleksitas kode menggunakan metrik *Lines of Code* (LOC), serta performa akses menggunakan parameter *First Contentful Paint* (FCP) dan *Largest Contentful Paint* (LCP) yang diperoleh melalui instrumen Google Lighthouse. Analisis data dilakukan secara komparatif dengan membandingkan hasil pengukuran dari kedua prototipe untuk mengidentifikasi tingkat efisiensi masing-masing *framework*. Hasil analisis kemudian digunakan sebagai dasar dalam menyusun rekomendasi teknis terkait pemilihan *framework* CSS yang paling optimal untuk mendukung pengembangan platform Shuttersport.id



Gambar 1. Metodologi Penelitian

2.1 Identifikasi Masalah

Tahap ini merupakan titik awal penelitian. Peneliti mengamati kondisi digital Shuttersport.id yang hingga kini masih bertumpu pada Instagram dan Linktree, sehingga detail foto menurun akibat kompresi dan pengelolaan portofolio menjadi terbatas. Karena platform mandiri belum dibangun, persoalan efisiensi *front-end* seperti besarnya *payload* dan potensi redundansi kode perlu diantisipasi sejak fase perancangan agar tidak terbawa ke tahap produksi. Kondisi inilah yang menjadi dasar diperlukannya optimalisasi sejak awal.

2.2 Studi Literatur

Peneliti melakukan penelusuran terhadap karya ilmiah terdahulu, teori paradigma *Utility-First* dan *Component-Based*, serta standar industri mengenai performa *web*. Tahap ini bertujuan untuk memperkuat landasan teori dan memastikan penelitian tidak bersifat redundan.

2.3 Metode Prototype

Berdasarkan permasalahan yang telah dirumuskan, peneliti memilih metode Eksperimen Terkontrol dengan pendekatan *Prototyping*. Metode ini dipilih agar terdapat kontrol penuh terhadap variabel-variabel desain, sehingga perbandingan antara Tailwind CSS dan Bootstrap dapat dilakukan secara objektif.

2.4 Penentuan Variable dan Matrix Penelitian

Pada tahap ini, peneliti mendefinisikan unit pengukuran atau "alat ukur" penelitian. Metrik yang ditetapkan mencakup aspek efisiensi kode (*Lines of Code*), efisiensi aset (*Payload*), serta performa akses (*First Contentful Paint* dan *Largest Contentful Paint*).

2.5 Perancangan Skenario Pengujian

Tahap rekayasa perangkat lunak di mana rancangan desain ditransformasikan menjadi dua prototipe fungsional secara paralel. Skenario A menggunakan Bootstrap dan Skenario B menggunakan Tailwind CSS, dengan tetap mempertahankan kesamaan visual sesuai *mockup*. Adapun perangkat lunak yang digunakan alat perancangan, lingkungan pengembangan, hingga instrumen audit performa sebagaimana tercantum pada tabel dibawah ini :

Tabel 1. Perangkat Lunak (*Software*)

No.	Kategori	Perangkat Lunak	Versi
1.	Sistem Operasi	Windows 11 *Home Single Language*	25H2
2.	*Web Browser*	Google Chrome	Chrome Versi 151.0.7922.77 (Build Resmi) (64 bit)
3.	Editor Teks	Visual Studio Code	Versi 1.127.0
4.	Alat Perancangan UI/UX	Figma	-
5.	*Runtime Environment*	Node.js (LTS) & NPM (*Node Package Manager*)	LTS v24.11.1 & NPM 11.7.0
6	*Framework & Library* Utama	Next.js (*App & Page Route*r) & React.js	next@16.2.9 & react@19.2.4
7.	*Framework* CSS	Bootstrap & Tailwind CSS	bootstrap@5.3.3 & tailwindcss@4.3.0
8.	Instrumen Audit	Google Lighthouse	-
9.	Alat Analisis Statis	VS Code Counter (perhitungan LOC)	3.7.2
10.	Alat Analisis Data	Microsoft Excel	Version 2607 Build 16.0.20228.20124) 64-bit

2.6 Pelaksanaan Experimen

Tahap rekayasa perangkat lunak di mana rancangan desain ditransformasikan menjadi dua prototipe fungsional secara paralel. Skenario A menggunakan Bootstrap dan Skenario B menggunakan Tailwind CSS, dengan tetap mempertahankan kesamaan visual sesuai *mockup*.

2.7 Pengumpulan Data

Tahap ini merupakan inti dari proses pengujian teknis. Peneliti melakukan pengukuran langsung (*running test*) menggunakan instrumen Google Lighthouse untuk menguji performa *rendering*, serta melakukan audit statis untuk mencatat besaran *payload* dan baris kode dari hasil build production kedua *framework*.

2.8 Pengolahan dan Analisis Data Komparatif

Data hasil pengujian yang telah dikumpulkan kemudian ditabulasi dan disandingkan untuk melihat disparitas performa secara kuantitatif. Peneliti melakukan interpretasi teknis guna menjelaskan korelasi antara arsitektur *framework* dengan hasil efisiensi yang didapatkan melalui proses pengujian sebelumnya.

3. HASIL DAN PEMBAHASAN

Tahap implementasi antarmuka (*user interface*) untuk platform Shuttersport.id dilakukan menggunakan dua pendekatan *framework* CSS yang berbeda, yakni Bootstrap dan Tailwind CSS. Tujuan dari implementasi ganda ini adalah untuk menghasilkan artefak perangkat lunak yang memiliki tampilan visual (*fidelity*) yang identik atau sangat mirip, sehingga dapat dijadikan objek pengujian yang valid untuk analisis komparatif efisiensi pengembangan.

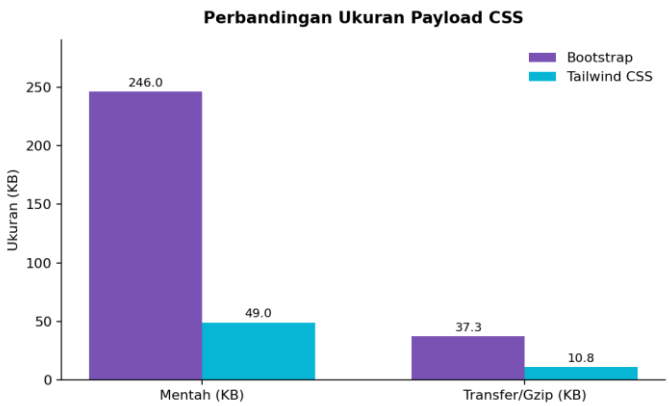
3.1 Perbandingan Ukuran Berkas Aset (*Payload*)

Pengukuran *payload* dilakukan dengan mengaudit seluruh berkas CSS yang dimuat oleh masing-masing prototipe pada kondisi *production build*, baik ukuran mentah (*uncompressed*) maupun ukuran transfer (gzip), melalui *Network tab* Google Chrome DevTools. Rangkuman hasil pengukuran disajikan pada Tabel 2.

Tabel 2. Perbandingan Ukuran *Payload* CSS

Prototipe	Ukuran Mentah (KB)	Ukuran Transfer/Gzip (KB)
Bootstrap	246	37,3
Tailwind CSS	49	10,8
Selisih	197	26,5
Efisiensi (reduksi Tailwind)	**80,1%**	**71,0%**

Berdasarkan Tabel 2, prototipe Bootstrap memuat total CSS sebesar 246 KB (mentah) atau 37,3 KB setelah kompresi gzip. Mayoritas bobot tersebut berasal dari berkas pustaka bootstrap.min.css yang berukuran 226 KB, karena Bootstrap memuat seluruh definisi komponennya terlepas dari komponen mana yang benar-benar digunakan. Sebaliknya, prototipe Tailwind CSS hanya memuat 49 KB (mentah) atau 10,8 KB (gzip), sebab mekanisme purge memastikan hanya kelas utilitas yang benar-benar dipakai yang dikompilasi ke berkas akhir. Selisihnya mencapai 197 KB (mentah) dan 26,5 KB (gzip), sehingga Tailwind CSS mereduksi ukuran *payload* sebesar 80,1% (mentah) dan 71,0% (gzip) dibandingkan Bootstrap untuk desain yang identik.



Gambar 2. Grafik perbandingan ukuran *payload* CSS

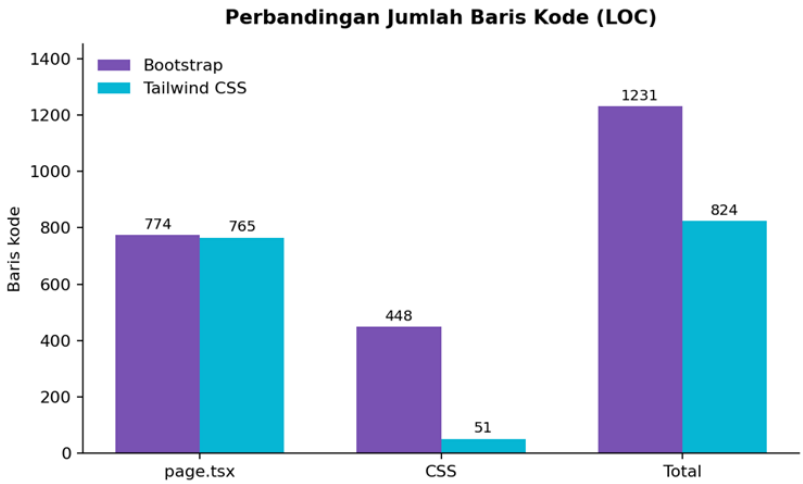
3.2 Perbandingan Kompleksitas Kode (*Lines of Code*)

Kompleksitas kode diukur menggunakan ekstensi VS Code Counter pada berkas penyusun masing-masing prototipe, mencakup berkas *markup* (page.tsx), berkas *layout*, dan berkas CSS. Jumlah baris kode efektif (*code*, tidak termasuk baris kosong dan komentar) beserta selisihnya disajikan pada Tabel 3

Tabel 3. Perbandingan Jumlah Baris Kode (*Lines of Code*)

Berkas	Bootstrap	Tailwind	Selisih
Markup page.tsx	774	765	9
Layout layout.tsx	9	8	1
CSS kustom	448	51	397
Total	1.231	824	407
Efisiensi (reduksi Tailwind)			**33,1%**

Tabel 3 menunjukkan bahwa jumlah baris markup kedua prototipe nyaris setara (selisih hanya 9 baris: 774 berbanding 765). Hal ini mengonfirmasi bahwa struktur HTML/JSX keduanya memang dibuat identik sebagai variabel kontrol, sehingga perbedaan yang muncul murni berasal dari pendekatan *framework*. Perbedaan paling mencolok terletak pada berkas CSS: Bootstrap memerlukan 448 baris CSS kustom untuk menyetarakan tampilan dengan desain Shuttersport.id, sedangkan Tailwind hanya 51 baris, selisih 397 baris atau 88,6%. Secara keseluruhan, total baris kode Tailwind 407 baris lebih sedikit (33,1% lebih ringkas). Temuan ini memperlihatkan karakteristik pergeseran kompleksitas (*complexity shift*), pada Tailwind kepadatan kelas utilitas terletak di dalam markup (*verbose* per baris) sementara berkas CSS sangat minim, sedangkan pada Bootstrap kompleksitas justru menumpuk pada berkas CSS kustom.



Gambar 3. Grafik perbandingan jumlah baris kode (LOC)

3.3 Perbandingan Skor Performa (FCP dan LCP)

Pengujian performa dilakukan menggunakan Google Lighthouse pada *mode Navigation Desktop* dalam kondisi *Incognito* dengan *network throttling* yang sama, diulang sebanyak sepuluh kali untuk tiap prototipe guna memperoleh nilai rata-rata yang stabil. Hasil tiap iterasi disajikan pada Tabel 4, sedangkan statistik diskriptif dan hasil pengujian FCP dan LCP ringkasan rata-rata beserta selisihnya disajikan pada Tabel 5.

Tabel 4. Hasil Pengujian FCP dan LCP Setiap Iterasi

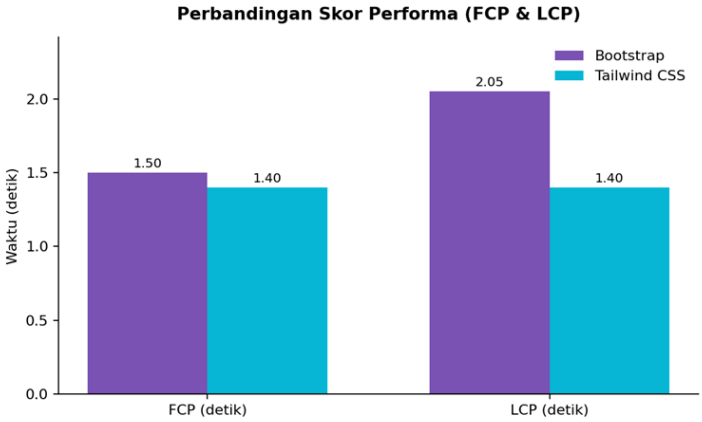
Iterasi	Bootstrap FCP (s)	Bootstrap LCP (s)	Tailwind FCP (s)	Tailwind LCP (s)
1	1,5	2,1	1,4	1,4
2	1,5	2,1	1,4	1,4
3	1,5	2,1	1,4	1,4
4	1,5	2,1	1,4	1,4
5	1,5	2,1	1,4	1,4
6	1,5	2,1	1,4	1,4
7	1,5	2,1	1,4	1,4
8	1,5	2,1	1,4	1,4
9	1,5	1,6	1,4	1,4
10	1,5	2,1	1,4	1,4
Rata-rata	**1,5**	**2,05**	**1,4**	**1,4**

Berdasarkan Tabel 4, hasil pengujian selama 10 iterasi menunjukkan bahwa prototipe berbasis Tailwind CSS menghasilkan nilai FCP yang konsisten sebesar 1,4 detik pada seluruh iterasi, sedangkan prototipe berbasis Bootstrap menghasilkan nilai FCP sebesar 1,5 detik pada seluruh pengujian. Pada metrik LCP, Tailwind CSS juga menunjukkan hasil yang konsisten sebesar 1,4 detik pada seluruh iterasi, sementara Bootstrap menghasilkan nilai 2,1 detik pada sembilan iterasi dan 1,6 detik pada satu iterasi. Berdasarkan nilai rata-rata, Tailwind CSS menghasilkan FCP sebesar 1,4 detik dibandingkan 1,5 detik pada Bootstrap, sedangkan nilai rata-rata LCP masing-masing sebesar 1,4 detik dan 2,05 detik. Dengan demikian, pada lingkungan eksperimen yang digunakan, prototipe Tailwind CSS menunjukkan waktu rendering yang lebih rendah dibandingkan prototipe Bootstrap, dengan selisih rata-rata sebesar 0,1 detik pada FCP dan 0,65 detik pada LCP.

Tabel 5. Statistik Diskriptif dan Hasil Pengujian FCP dan LCP

Metrik	Framework	Mean (s)	SD (s)	Median (s)	95% CI
FCP	Bootstrap	1,50	0,00	1,50	1,50–1,50
FCP	Tailwind CSS	1,40	0,00	1,40	1,40–1,40
LCP	Bootstrap	2,05	0,16	2,10	1,94–2,16
LCP	Tailwind CSS	1,40	0,00	1,40	1,40–1,40

Berdasarkan Tabel 5, hasil analisis statistik deskriptif menunjukkan bahwa prototipe Bootstrap memiliki nilai FCP rata-rata sebesar 1,50 detik dengan standar deviasi 0,00 detik dan median 1,50 detik, sedangkan prototipe Tailwind CSS memiliki nilai FCP rata-rata sebesar 1,40 detik dengan standar deviasi 0,00 detik dan median 1,40 detik. Pada metrik LCP, Bootstrap menghasilkan nilai rata-rata sebesar 2,05 detik dengan standar deviasi 0,16 detik dan median 2,10 detik, sedangkan Tailwind CSS menghasilkan nilai rata-rata sebesar 1,40 detik dengan standar deviasi 0,00 detik dan median 1,40 detik. Interval kepercayaan 95% menunjukkan bahwa nilai FCP Bootstrap berada pada rentang 1,50–1,50 detik dan Tailwind CSS pada rentang 1,40–1,40 detik. Sementara itu, pada metrik LCP, interval kepercayaan 95% Bootstrap berada pada rentang 1,94–2,16 detik, sedangkan Tailwind CSS berada pada rentang 1,40–1,40 detik. Hasil tersebut menunjukkan bahwa Tailwind CSS memiliki nilai FCP dan LCP yang lebih rendah dibandingkan Bootstrap pada lingkungan eksperimen yang digunakan. Perbedaan rata-rata tersebut menunjukkan penurunan waktu sebesar 6,7% pada FCP dan 31,7% pada LCP, dengan variasi pengukuran yang relatif rendah, terutama pada prototipe Tailwind CSS



Gambar 4. Grafik perbandingan skor performa FCP dan LCP

3.4 Hubungan Paradigma *Framework* dengan Efisiensi Aset

Perbedaan ukuran *payload* yang sangat besar (Tabel 2) berakar pada perbedaan paradigma kedua *framework*. Bootstrap menganut pendekatan *component-based*, di mana seluruh definisi komponen (tombol, kartu, navbar, modal, sistem grid, dan sebagainya) dikirim sebagai satu pustaka utuh terlepas dari berapa banyak komponen yang benar-benar digunakan. Akibatnya, berkas bootstrap.min.css tetap berukuran 226 KB meskipun prototipe Shuttersport.id hanya memanfaatkan sebagian kecil komponennya. Sebaliknya, Tailwind CSS menganut paradigma *utility-first* yang dipadukan dengan mekanisme *purge/just-in-time*, sehingga hanya kelas utilitas yang benar-benar muncul pada *markup* yang dikompilasi ke berkas akhir, inilah yang membuat *payload* Tailwind hanya 49 KB (mentah). Efisiensi aset ini berdampak langsung pada performa: semakin kecil *payload* CSS yang harus diunduh dan diproses peramban, semakin cepat elemen konten dirender. Hal ini terbukti pada metrik LCP (Tabel 5), di mana Tailwind yang *payload*nya jauh lebih ringan mencatat LCP 31,7% lebih cepat. Dengan demikian, terdapat korelasi positif antara efisiensi ukuran aset dengan kecepatan akses antarmuka.

3.5 Dampak Kompleksitas Kode Terhadap Pemeliharaan (*Maintainability*)

Dari sisi kompleksitas kode (Tabel 3), temuan menunjukkan adanya pergeseran kompleksitas (*complexity shift*) yang berimplikasi pada pemeliharaan. Pada Bootstrap, struktur *markup* memang ringkas karena satu kelas semantik (misalnya .card atau .btn) mewakili banyak aturan CSS bawaan; namun ketika desain menuntut tampilan khusus di luar gaya bawaan seperti palet warna brand, jarak antar-*section* yang besar, dan efek panel kaca pada Shuttersport.id—pengembang terpaksa menulis 448 baris CSS kustom beserta *override*. Model ini menuntut perpindahan konteks bolak-balik antara berkas *markup* dan berkas CSS serta berisiko menimbulkan masalah spesifisitas yang sulit dilacak. Sebaliknya, Tailwind memindahkan kompleksitas ke dalam *markup*, berkas CSS nyaris tidak ada (hanya 51 baris), tetapi setiap elemen memuat banyak kelas utilitas sehingga markup menjadi lebih padat (*verbose*). Keunggulannya, perubahan tampilan dapat dilakukan langsung pada elemen tanpa berpindah berkas dan tanpa risiko efek samping pada elemen lain, sehingga relatif lebih mudah dipelihara untuk antarmuka yang sering berubah. Perlu dicatat bahwa jumlah baris markup kedua prototipe hampir setara (selisih hanya 9 baris), perbedaan utama bukan terletak pada banyaknya baris, melainkan pada lokasi dan cara kompleksitas tersebut dikelola.

3.6 Implikasi Bagi Pengembangan Shuttersport.id

Sebagai platform manajemen aset visual yang menampilkan banyak foto beresolusi tinggi, Shuttersport.id sangat sensitif terhadap bobot halaman dan kecepatan *render*. Berdasarkan keseluruhan temuan, Tailwind CSS lebih sesuai untuk kebutuhan produksi Shuttersport.id karena menghasilkan *payload* CSS hingga 80% lebih kecil dan waktu *render* (LCP) yang lebih cepat dua faktor yang krusial untuk mempertahankan pengguna pada platform berbasis citra. Efisiensi aset ini juga menyisakan anggaran *bandwidth* yang lebih besar untuk aset gambar, yang merupakan inti bisnis Shuttersport.id. Meskipun demikian, Tailwind menuntut kurva belajar yang lebih tinggi dan menghasilkan *markup* yang lebih *verbose*, sehingga disiplin penataan komponen dan konvensi penamaan tetap diperlukan agar kode mudah dipelihara dalam jangka panjang. Sebaliknya, Bootstrap tetap menjadi pilihan rasional apabila prioritasnya adalah kecepatan pembuatan prototipe awal atau ketika tim belum terbiasa dengan paradigma *utility-first*. Untuk fase *greenfield* (pembangunan dari nol) seperti yang sedang dijalani Shuttersport.id, mengadopsi Tailwind CSS sejak awal dinilai paling strategis guna menjamin skalabilitas dan performa optimal di masa mendatang.

4. KESIMPULAN

Berdasarkan hasil eksperimen terkontrol pada antarmuka Shuttersport.id, penggunaan Tailwind CSS menunjukkan efisiensi yang lebih baik dibandingkan Bootstrap pada seluruh metrik yang dianalisis. Tailwind CSS menghasilkan reduksi ukuran *payload* sebesar 80,1%, pengurangan kompleksitas kode sebesar 33,1%, serta penurunan waktu *First Contentful Paint* (FCP) sebesar 6,7% dan *Largest Contentful Paint* (LCP) sebesar 31,7%. Hasil tersebut menunjukkan bahwa pada konfigurasi eksperimen yang digunakan, pendekatan *utility-first* Tailwind CSS mampu menghasilkan aset CSS yang lebih ringan dan waktu rendering yang lebih rendah dibandingkan pendekatan *component-based* Bootstrap. Temuan ini mendukung penggunaan Tailwind CSS sebagai salah satu alternatif *framework front-end* untuk pengembangan antarmuka Shuttersport.id yang berorientasi pada efisiensi aset dan performa akses.

Penelitian ini memiliki keterbatasan karena eksperimen hanya dilakukan pada satu objek antarmuka, yaitu Shuttersport.id, dengan rancangan prototipe dan lingkungan pengujian yang dikendalikan. Oleh karena itu, hasil penelitian tidak dapat digeneralisasikan bahwa Tailwind CSS selalu lebih baik dibandingkan Bootstrap pada seluruh jenis website atau kondisi pengembangan. Penelitian selanjutnya dapat memperluas objek pengujian pada berbagai karakteristik website, ukuran aset, kompleksitas antarmuka, perangkat, dan kondisi jaringan untuk memperoleh gambaran yang lebih luas mengenai efisiensi kedua framework.

DAFTAR PUSTAKA

[1] R. Pratama et al., "Perbandingan Kecepatan Performa Website E-Commerce Antara Bootstrap dan Tailwind CSS Menggunakan Lighthouse Untuk Optimasi," in Seminar Nasional Teknologi Informasi, Mekatronika dan Ilmu Komputer, vol. 5, pp. 27–34, 2026. [Online]. Available: http://prosiding.sentimeter.nusaputra.ac.id/index.php/prosiding/article/view/101.

[2] R. D. E. Putra, A. Lestari, and N. Kristianti, "Analisis Perbandingan CSS Framework Tailwind CSS dan Bootstrap dalam Pengembangan Landing Page Website POS Digitaliz," Journal of Information Technology and Computer Science, vol. 5, no. 1, pp. 63–75, 2025, doi: 10.47111/jointecoms.v5i1.

[3] M. F. Santoso, "Perbandingan Efektivitas Bootstrap dan Tailwind CSS dalam Pengembangan UI Web Responsif," Jurnal Teknologi dan Sistem Informasi Bisnis, vol. 7, no. 4, pp. 489–497, 2025, doi: 10.47233/jteksis.v7i4.2260.

[4] S. Azhariyah and M. Mukhlis, "Framework CSS: Tailwind CSS Untuk Front-End Website Store PT. XYZ," Jurnal Informatika, vol. 3, no. 1, pp. 30–36, 2023. [Online]. Available: https://jurnal.uniraya.ac.id/index.php/JI/article/view/1601/1130

[5] B. Harahap, A. Rambe, M. R. Ramadhan, and N. Kurniawan, "Analisis Framework, Library Front-End Populer: Bootstrap, Tailwind CSS, React, dan Vue Pada Mata Kuliah Perancangan Web Design," Riau Jurnal Teknik Informatika, vol. 4, no. 2, pp. 329–334, 2025, doi: 10.30606/rjti.v4i2.3496.

[6] Z. Abidin and M. Ridwan, "Systematic Literature Review: Metode Perancangan UI/UX Pada Toko Online," Jurnal Manajemen Informatika, Sistem Informasi dan Teknologi Komputer, vol. 3, no. 2, pp. 224–237, 2024, doi: 10.70247/jumistik.v3i2.101.

[7] R. Adelard, R. Muthicahya, E. S. Wicaksono, L. Kenneth V., S. Narulita, and Sekarlangit, "Implementasi Metode Pengembangan Sistem Prototype pada Rancang Bangun Aplikasi Marketplace Lensa Buana," Jurnal Publikasi Sistem Informasi dan Telekomunikasi, vol. 3, no. 2, pp. 69–81, 2025, doi: 10.62951/bridge.v3i2.424.

[8] Y. Anis, Purwatiningtyas, Retnowati, and E. A. N. Fajrina, "Penerapan Framework Bootstrap Dalam Sistem Informasi Rekam Medis Data Posyandu dengan Metode Waterfall," Jurnal Sistem Komputer dan Informatika, vol. 4, no. 2, pp. 310–318, 2022, doi: 10.30865/json.v4i2.4833.

[9] D. Chandra and Hermansyah, "Transformasi Digital Kedai Kopi Sudut Kota Berbasis Website Dengan Javascript," Jurnal Rekayasa Sistem Informasi dan Teknologi, vol. 2, no. 2, pp. 729–743, 2024, doi: 10.70248/jrsit.v2i2.1438.

[10] C. Christian and A. Voutama, "Implementasi Aplikasi Antrian Pencucian Mobil Berbasis Web Menggunakan PHP, Javascript, HTML, CSS, dan UML," Jurnal Mahasiswa Teknik Informatika, vol. 8, no. 2, pp. 2243–2248, 2024, doi: 10.36040/jati.v8i2.9460.

[11] R. Y. Endra, Y. Aprilinda, Y. Y. Dharmawan, and W. Ramadhan, "Analisis Perbandingan Bahasa Pemrograman PHP Laravel dengan PHP Native pada Pengembangan Website," Jurnal Manajemen Sistem Informasi dan Teknologi, vol. 11, no. 1, pp. 48–55, 2022, doi: 10.36448/expert.v11i1.2012.

[12] I. M. Faried and N. Lediwara, "Pengembangan Lanjutan Aplikasi Monitoring Logistik Simtelog Dengan Optimalisasi Visualisasi Data Gudang Dinas Informasi dan Pengolahan Data," Journal of Information System, Informatics and Computing, vol. 9, no. 2, pp. 397–417, 2025, doi: 10.52362/jisicom.v9i2.2190.

[13] D. Fithriyaningrum, S. S. Kusumawardhani, and S. Wibirama, "Analisis Aksesibilitas Website berdasarkan Web Content Accessibility Guidelines (WCAG): Ulasan Literatur Sistematis," Jurnal Ilmu Pengetahuan dan Teknologi Komunikasi, vol. 23, no. 1, pp. 79–92, 2021, doi: 10.17933/iptekkom.23.1.2021.79-92.

[14] T. Hidayatulloh and K. N. Isnaini, "Implementasi Metode Agile Extreme Programming Dalam Pengembangan Website Game Berbasis Javascript: Save Me," Jurnal Mahasiswa Teknik Informatika, vol. 8, no. 2, pp. 1404–1411, 2024, doi: 10.36040/jati.v8i2.8915.

[15] S. Khasanah and T. Sutabri, "Faktor-Faktor Tampilan UI/UX yang Mempengaruhi Psikologis Manusia," Jurnal Sain dan Teknik, vol. 5, no. 2, pp. 28–33, 2023, doi: 10.37577/sainteks.v5i1.5.

[16] M. F. Khoirurrizal, C. R. Hidayat, and R. Ruuhwan, "Analisis Perbandingan Framework Front-End Javascript SolidJS dan VueJS Pada Pengembangan Website Interaktif," Jurnal Informatika dan Teknik Elektro Terapan, vol. 12, no. 2, pp. 1026–1035, 2024, doi: 10.23960/jitet.v12i2.4106.

[17] A. Mardiansyah et al., "Pengembangan Dasar HTML dan CSS: Langkah Pertama Dalam Pengembangan Web," Jurnal Pengabdian Kepada Masyarakat, vol. 2, no. 3, pp. 281–286, 2024. [Online]. Available: https://jurnalmahasiswa.com/index.php/appa/article/view/2024

[18] M. B. Pramadipta, "Rancang Bangun Frontend Untuk Pemungutan Suara Dengan Menggunakan React.js," Jurnal Informatika dan Teknik Elektro Terapan, vol. 12, no. 4, pp. 1131–1140, 2024, doi: 10.23960/jitet.v12i4.4173.