

TUGAS AKHIR
ANALISIS PERBANDINGAN FRONT-END JAVASCRIPT
FRAMEWORK MENGGUNAKAN JS FRAMEWORK
BENCHMARK PADA PENGEMBANGAN WEBSITE



DEFRIANSYAH
203510130

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS ISLAM RIAU
PEKANBARU
2024

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Alhamdulillah, puji syukur kehadiran Allah SWT yang telah melimpahkan Rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi ini dengan judul **“Analisis Perbandingan Front-End Javascript Framework Menggunakan Js Framework Benchmark Pada Pengembangan Website”**. Penulisan tugas akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Teknik Program Studi Teknik Informatika Universitas Islam Riau. Dalam penyusunan skripsi ini, tentunya penulis menyadari bahwa banyak pihak yang telah membantu dan mendorong penulis untuk menyelesaikan tugas akhir ini serta memperoleh ilmu pengetahuan selama perkuliahan. Oleh karena itu dalam kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. M.Rizki Fadhillah, ST, M.Eng selaku dosen pembimbing, yang telah memberikan bimbingan, arahan, dan masukan yang berharga selama penulisan skripsi ini.
2. Ketua dan sekretaris prodi serta dosen-dosen yang sangat banyak membantu terkait perkuliahan, ilmu pengetahuan dan hal lain yang tidak dapat saya sebutkan satu per satu.
3. Orang tua dan keluarga tercinta, yang selalu memberikan doa, dukungan, serta motivasi tanpa henti selama proses penyusunan skripsi ini.

4. Shabat serta teman-teman seperjuangandan lainnya yang tidak bisa disebutkan satu-persatu, yang telah memberikan bantuan, seta semangan, dan dukungan selama masa perkuliahan dan penyusunan skripsi ini.

Doa saya, semoga Allah memberikan balasan atas segala kebaikan semua pilhakyang telah membantu. Penulis menyadari bahwa dalam penyusunan laporan skripsi ini masih jauh dari kata sempurna. Oleh karena itu dengan segala kerendahan hati penulis berharap kritik dan saran yang sifatnya membangun guna memperbaiki laporan skripsi ini. Akhir kata semoga laporan skripsi ini dapat menambah ilmu pengetahuan dan bermanfaat bagi semua pihak yang membacanya.

Pekanbaru, 27 November 2024

Penulis,



Defriansyah

203510130

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI.....	iii
DAFTAR GAMBAR	v
DAFTAR TABEL.....	vi
ABSTRAK.....	vii
ABSTRACT.....	viii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Identifikasi Masalah.....	2
1.3 Rumusan Masalah.....	2
1.4 Batas Masalah	3
1.5 Tujuan Penelitian	3
1.6 Manfaat Penelitian	4
BAB II LANDASAN TEORI.....	5
2.1 Tinjauan Pustaka.....	5
2.2 Dasar Teori.....	7
2.2.1 Website	7
2.2.2 Front End Development.....	7
2.2.3 Javascript.....	8
2.2.4 Framework	8
2.2.5 Document Object Model.....	8
2.2.6 Benchmarking	9
2.2.7 Node Js.....	9
2.2.8 Lighthouse.....	9
2.2.9 Js Framework Benchmark.....	10
2.2.10 State Of Javascript	10
2.2.11 React Js	10
2.2.12 Vue Js.....	10
2.2.13 Qwik Js.....	11
2.2.14 Svelte Js	11
2.2.15 Solid Js	12

2.2.16 Justifikasi Pemilihan Framework.....	12
2.3 Analisis Data.....	15
2.4 Kerangka Pemikiran.....	16
BAB III METODOLOGI PENELITIAN	17
3.1 Jenis Penelitian.....	17
3.1.1 Analisis Kebutuhan	17
3.1.2 Pengujian.....	19
3.1.3 Hasil Pengujian	21
3.1.4 Analisis Data	25
BAB IV HASIL DAN PEMBAHASAN	26
4.1 Hasil Penelitian	26
BAB V KESIMPULAN DAN SARAN	49
5.1 Kesimpulan	49
5.2 Saran	49
DAFTAR PUSTAKA	52

DAFTAR GAMBAR

Gambar 2.1 Kerangka Pemikiran	16
Gambar 3.1 Diagram Alur Penelitian	17
Gambar 3.2 Membuat Folder Benchmark	19
Gambar 3.3 Setup Project Kelima Framework	20
Gambar 3.4 Menyiapkan Backend	20
Gambar 3.5 Struktur Folder Benchmark	20
Gambar 3.6 Menjalankan Server Backend	23
Gambar 3.7 Menjalankan Framework	23
Gambar 3.8 Menjalankan Benchmark Pada Solid JS	24
Gambar 3.9 Hasil Menjalankan Pengujian Framework Solid JS	24
Gambar 4.1 Chart Hasil Pengujian Memori Framework React	29
Gambar 4.2 Chart Hasil Pengujian Memori Framework Vue	31
Gambar 4.3 Chart Hasil Pengujian Memori Framework Solid	32
Gambar 4.4 Chart Hasil Pengujian Memori Framework Svelte	34
Gambar 4.5 Chart Hasil Pengujian Memori Framework Qwik	35
Gambar 4.6 Chart Hasil Pengujian Memori	37
Gambar 4.7 Chart Hasil Pengujian Eksekusi Waktu Framework React	39
Gambar 4.8 Chart Hasil Pengujian Waktu Eksekusi Framework Vue	40
Gambar 4.9 Chart Hasil Pengujian Waktu Eksekusi Framework Solid	42
Gambar 4.10 Chart Hasil Pengujian Waktu Eksekusi Framework Svelte	43
Gambar 4.11 Chart Hasil Pengujian Waktu Eksekusi Framework Qwik	45
Gambar 4.12 Chart Hasil Pengujian Eksekusi Waktu	47
Gambar 4.13 Chart Hasil Perbandingan Ukuran Bundle	48

DAFTAR TABEL

Tabel 3.1 Skenario Pengujian Framework	22
Tabel 4.1 Hasil Pengujian Memori Framework React	28
Tabel 4.2 Hasil Pengujian Momori Framework Vue	30
Tabel 4.3 Hasil Pengujian Memori Framework Solid.....	31
Tabel 4.4 Hasil Pengujian Memori Framework Svelte	33
Tabel 4.5 Hasil Pengujian Memori Framework Qwik	34
Tabel 4.6 Hasil Pengujian Memori	36
Tabel 4.7 Hasil Pengujian Waktu Eksekusi Framework React.....	38
Tabel 4.8 Hasil Pengujian Eksekusi Waktu Framework Vue	39
Tabel 4.9 Hasil Pengujian Waktu Eksekusi Framework Solid	41
Tabel 4.10 Hasil Pengujian Waktu Eksekusi Framework Svelte.....	42
Tabel 4.11 Hasil Pengujian Waktu Eksekusi Framework Qwik	44
Tabel 4.12 Hasil Pengujian Waktu Eksekusi	45
Tabel 4.13 Hasil Pengujian Ukuran Bundle	47

ANALISIS PERBANDINGAN FRONT-END JAVASCRIPT FRAMEWORK MENGGUNAKAN JS FRAMEWORK BENCHMARK PADA PENGEMBANGAN WEBSITE

Defriansyah

Program Studi Teknik Informatika, Fakultas Teknik, Universitas Islam Riau

Email: defriansyah@student.uir.ac.id

ABSTRAK

Pemilihan framework front-end JavaScript yang tepat menjadi elemen krusial dalam pengembangan website modern, terutama dalam aspek kinerja, efisiensi penggunaan sumber daya, dan kecepatan waktu mulai (startup). Penelitian ini membandingkan lima framework JavaScript populer, yaitu React, Vue, Svelte, Solid, dan Qwik, melalui tiga parameter utama: performa, efisiensi memori, dan ukuran bundle. Pengujian dilakukan menggunakan **JS Framework Benchmark**, dengan skenario yang melibatkan peningkatan bertahap pada jumlah data untuk mengukur konsumsi memori, waktu eksekusi, dan efisiensi keseluruhan.

Hasil penelitian menunjukkan bahwa **Solid** dan **Svelte** unggul dalam efisiensi penggunaan memori dan memiliki ukuran bundle terkecil, membuat keduanya ideal untuk aplikasi ringan dan responsif. **React** dan **Vue**, meskipun memiliki ukuran bundle yang lebih besar, menawarkan fleksibilitas dan kestabilan yang baik untuk aplikasi berskala besar. **Qwik**, di sisi lain, menunjukkan performa yang konsisten tetapi dengan waktu eksekusi yang lebih tinggi dibandingkan framework lainnya.

Kata kunci: framework, benchmark, website, frontend, jsframeworkbenchmark

COMPARATIVE ANALYSIS OF FRONT-END JAVASCRIPT FRAMEWORKS USING JS FRAMEWORK BENCHMARK IN WEBSITE DEVELOPMENT

Defriansyah

Department of Informatics Engineering, Faculty of Engineering, Islamic
University of Riau

Email: defriansyah@student.uir.ac.id

ABSTRACT

The selection of the right front-end JavaScript framework is a crucial factor in modern website development, especially in terms of performance, resource efficiency, and startup speed. This study compares five popular JavaScript frameworks: React, Vue, Svelte, Solid, and Qwik, based on three main parameters: performance, memory efficiency, and bundle size. Testing was conducted using the JS Framework Benchmark, with scenarios involving gradual increases in data volume to measure memory consumption, execution time, and overall efficiency.

The results of the study show that Solid and Svelte excel in memory efficiency and have the smallest bundle sizes, making them ideal for lightweight and responsive applications. React and Vue, while having larger bundle sizes, offer good flexibility and stability for larger-scale applications. On the other hand, Qwik demonstrates consistent performance but with higher execution times compared to the other frameworks.

Keywords: framework, benchmark, website, front-end, JavaScript

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam dunia pengembangan *website* modern, pemilihan *framework front-end* yang tepat menjadi faktor krusial untuk memastikan efisiensi, kinerja, dan pengalaman pengguna yang optimal. Dengan banyaknya pilihan *framework* yang tersedia, pengembang sering kali dihadapkan pada dilema dalam menentukan teknologi yang paling sesuai untuk digunakan. Beberapa *framework* yang populer saat ini meliputi React, Vue, Svelte, Solid, dan Qwik. Masing-masing *framework* ini memiliki karakteristik, keunggulan, dan kelemahan yang berbeda.

React dikenal karena kemampuannya dalam mengelola komponen yang *reusable* dan komunitas yang besar. Vue unggul dalam hal kemudahan integrasi dan waktu *rendering* yang cepat. Svelte, sebagai *framework* yang lebih baru, menawarkan pendekatan unik dengan melakukan sebagian besar pekerjaan pada tahap kompilasi, sehingga menghasilkan aplikasi yang lebih ringan dan cepat. Solid, meskipun kurang dikenal, menawarkan kinerja yang sangat tinggi dengan pendekatan reaktif yang mirip dengan React. Sementara itu, Qwik, sebagai *framework* yang lebih lengkap, menawarkan solusi menyeluruh dengan ekosistem yang kuat namun sering dianggap lebih kompleks.

Dengan perbedaan karakteristik ini, penting untuk melakukan analisis perbandingan terhadap performa, efisiensi sumber daya, dan kecepatan startup dari kelima *framework* tersebut. Penelitian ini bertujuan untuk memberikan

panduan yang lebih jelas kepada pengembang dalam memilih *framework* yang sesuai dengan kebutuhan proyek mereka.

1.2 Identifikasi Masalah

Berdasarkan permasalahan yang telah diuraikan diatas, maka identifikasi masalah dalam penelitian ini sebagai :

1. Pengembang web menghadapi kesulitan dalam memilih framework front-end yang tepat di antara lima framework populer (React, Vue, Svelte, Solid, dan Qwik) berdasarkan kinerja dan efisiensi.
2. Setiap framework memiliki karakteristik unik, namun tidak ada pedoman yang jelas mengenai framework mana yang lebih unggul dalam aspek tertentu, seperti performa dan penggunaan sumber daya.
3. Diperlukan perbandingan yang komprehensif untuk memahami kelebihan dan kelemahan masing-masing framework dalam konteks pengembangan website modern.

1.3 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah ditemukan diatas maka dapat dirumuskan masalah sebagai berikut :

1. Bagaimana perbandingan performa dari React, Vue, Svelte, Solid, dan Qwik dalam pengembangan website?
2. Framework mana yang paling efisien dalam penggunaan sumber daya memori?

3. Framework mana yang lebih unggul secara keseluruhan dalam hal kecepatan dan efisiensi?

1.4 Batas Masalah

Batasan masalah yang akan dibahas dibawah ini meliputi beberapa hal pokok yaitu :

1. Penelitian ini hanya menganalisis lima framework front-end, yaitu React, Vue, Svelte, Solid, dan Qwik, dalam pengembangan aplikasi website.
2. Pengujian dilakukan dalam lingkungan pengembangan yang sama menggunakan perangkat keras dan perangkat lunak yang sudah ditentukan, tanpa mempertimbangkan integrasi framework dengan teknologi lain atau aspek keamanan.

1.5 Tujuan Penelitian

Tujuan dari penelitian ini adalah :

1. Membandingkan performa lima framework front-end (React, Vue, Svelte, Solid, dan Qwik) berdasarkan waktu rendering komponen dan pembaruan tampilan.
2. Mengevaluasi efisiensi penggunaan memori dari masing-masing framework dalam skenario pengembangan yang sama.
3. Memberikan rekomendasi bagi pengembang untuk memilih framework yang paling sesuai berdasarkan hasil analisis performa dan efisiensi.

1.6 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan panduan yang lebih mendalam dan komprehensif bagi pengembang dalam memilih framework front-end yang sesuai dengan kebutuhan proyek, terutama dalam hal performa, efisiensi penggunaan sumber daya, dan kecepatan startup. Selain itu, hasil penelitian ini juga akan memberikan kontribusi akademis yang bermanfaat sebagai referensi dalam studi perbandingan framework front-end. Bagi perusahaan dan organisasi, penelitian ini dapat membantu meningkatkan efisiensi pengembangan website, sehingga dapat memberikan pengalaman pengguna yang lebih baik dan responsif

BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Dari penelitian sebelumnya yang ditulis oleh chastro C, Darmawan E, dengan judul “Perbandingan Perkembangan Front-End Menggunakan Blade Template dan Vue Js” penelitian ini menggunakan metode audits panel pada chrome dev tools, hasil dari penelitian ini *blade template* memiliki performa yang lebih cepat daripada vue js. Namun, vue js memberikan lebih banyak kemudahan dalam mengembangkan aplikasi website yang kompleks. Selain itu, dalam pengujian kecepatan *scripting*, *blade template* juga menunjukkan kecepatan yang jauh lebih tinggi dibandingkan dengan vue js. Oleh karena itu, kesimpulannya adalah untuk membangun aplikasi sederhana sebaiknya menggunakan *blade template*, sementara untuk aplikasi yang kompleks, vue js dapat menjadi pilihan yang lebih baik.(Chastro et al., 2020)

Penelitian berikutnya yang ditulis oleh Khoirurriz, dengan judul “Analisis Perbandingan *Framework Front-End JavaScript SolidJS dan VueJS* pada pengembangan website interaktif” penelitian ini menggunakan metode adalah metode kuantitatif eksperimen untuk membandingkan performa dan efisiensi antara solid js dan vue js dalam pengembangan *front-end*. Langkah-langkah penelitian tergambar dalam gambar metodologi penelitian yang disajikan. Hasil dari penelitian ini uji mann-whitney menunjukkan perbedaan signifikan antara kedua kelompok data efisiensi. Solid js unggul dalam performa, efisiensi, dan waktu startup dibandingkan dengan vue js. Solid js menonjol dengan penggunaan memori yang efisien dan konsistensi hasil pengujian yang kuat.

Dengan demikian, solid js layak dipertimbangkan sebagai pesaing yang kuat dalam pengembangan *front-end JavaScript*.(Khoirurrizal et al., 2024)

Penelitian selanjutnya yang ditulis oleh Axza, dengan judul “Analisis Perbandingan Framework Front-End JavaScript ReactJS dan VueJS” penelitian ini menggunakan metode kuantitatif eksperimen untuk membandingkan performa dan efisiensi react js dan vue js dalam pengembangan *front-end*. Hasil dari penelitian ini react js menunjukkan performa yang lebih baik dalam hal efisiensi, waktu startup, dan responsivitas terhadap perubahan dibandingkan dengan vue js. Vue js memiliki tingkat stabilitas yang lebih tinggi dalam pengembangan *front-end*. Uji Mann-Whitney U digunakan karena distribusi data tidak normal, menunjukkan perbedaan signifikan antara kedua kelompok data. Dengan demikian, react js layak dipertimbangkan sebagai alternatif yang kompetitif dalam pengembangan aplikasi *front-end JavaScript*.(Axza et al., 2023)

Penelitian berikutnya yang ditulis oleh Ilievska, yang berjudul “Analisis dan Evaluasi Komparatif Teknologi Front-end untuk Pengembangan Aplikasi Web”. Dari penelitian ini dapatlah hasil pengujian perbandingan react cocok untuk aplikasi satu halaman sederhana. Kinerja Aplikasi: React unggul dalam kecepatan, penggunaan memori, dan waktu pemuatan. Kesimpulan react direkomendasikan sebagai pilihan terbaik untuk pengembangan aplikasi web *front-end* berdasarkan hasil perbandingan yang dilakukan.(Ilievska & Gramatikov, 2022)

Penelitian selanjutnya yang ditulis oleh Wenqing Xu, dengan judul “*Benchmark Comparison of JavaScript Frameworks React, Vue, Angular and Svelte*” penelitian ini menggunakan metode pemilihan kerangka kerja *front-end*. Memilih kerangka kerja *front-end* yang representatif berdasarkan permintaan pasar dan evaluasi pengguna. Hasil dari penelitian ini vue js dan react js memiliki kinerja yang baik dalam implementasi realworld application. Lighthouse menunjukkan bahwa svelte memiliki waktu tampilan konten pertama dan indeks kecepatan yang lebih baik daripada react js, vue js, dan angular.(Wenqing Xu, 2021)

2.2 Dasar Teori

2.2.1 Website

Website adalah kumpulan halaman web yang dijalankan menggunakan browser dan internet. *Website* berada dalam domain atau subdomain yang sering disebut dengan www atau *world wide web*. Sebuah *website* dibuat dengan bahasa pemrograman html (*Hyper Text Markup Language*) yang diakses melalui protokol di internet (Endra et al., 2021).

2.2.2 Front End Development

Front-End Development adalah disiplin dalam pengembangan *website* yang bertugas untuk merancang, mengembangkan, dan menjaga komponen-komponen tampilan yang berhubungan langsung dengan pengguna. Javascript.(Khoirurrizal et al., 2024b)

2.2.3 Javascript

JavaScript adalah sebuah bahasa pemrograman yang diterapkan pada pengembangan web untuk menciptakan situs web yang interaktif, dinamis, dan responsif. Bahasa ini berfungsi di sisi klien, berarti dijalankan di peramban web pengguna, memungkinkan setiap pengembang untuk dapat menambahkan berbagai fitur yang dapat meningkatkan pengalaman pengguna. (Sari & Hidayat, 2022)

2.2.4 Framework

Setiap bahasa pemrograman mempunyai macam-macam *framework* dengan kelebihan dan kekurangan masing-masing. *Framework* tidak bisa lepas dari developer karena sebagai *developer* pasti akan menggunakan *framework*. Karena dengan menggunakan *framework* merupakan salah satu cara untuk membuat *website* yang kita buat bisa menjadi lebih menarik dan dapat membuat pengunjung merasa nyaman berlama-lama pada *website* yang kita buat. (Devianty et al., 2021)

2.2.5 Document Object Model

Document Object Model (DOM) adalah antarmuka pemrograman untuk dokumen *html* dan *xml*. Ini mewakili halaman sehingga program dapat mengubah struktur, gaya, dan konten dokumen. DOM mewakili dokumen sebagai node dan objek. Dengan begitu, bahasa pemrograman dapat terhubung ke halaman. (Anhar et al., 2023)

2.2.6 Benchmarking

Benchmarking adalah proses evaluasi *performance* dari suatu sistem pada kondisi tertentu. *Benchmarking* dilakukan dengan menjalankan sebuah atau kumpulan program yang dinamakan *benchmarking tools*. Program tersebut akan menjalankan operasi-operasi tertentu untuk menguji kemampuan dari suatu sistem. *Benchmarking tools* dalam dunia komputasi secara umum menguji beberapa aspek dalam komputer seperti kemampuan CPU (HyperPi, Prime95, CPU-M), storage (CrystalDiskMark), graphics (3DMark, Unigine Heaven, Maxon Cinebench), maupun sistem secara keseluruhan (PCMark, SiSoftware Sandra, NovaBench, PassMark Performance Test). (Evalina, 2021)

2.2.7 Node Js

Node.js adalah *platform* yang *powerful* dan *versatile* untuk pengembangan aplikasi web dan *server-side*. Cocok untuk developer yang ingin membangun aplikasi *real-time*, *api*, *microservices*, dan *tools command-line* dengan *JavaScript*. (Putu Mahendra Putra et al., 2023)

2.2.8 Lighthouse

Google *lighthouse* adalah *automated tool* untuk meningkatkan kualitas dari laman website dengan cara menjalankan beberapa langkah audit terhadap laman, lalu membuat laporan terhadap seberapa baik laman tersebut. (Aripin & Somantri, 2021)

2.2.9 Js Framework Benchmark

Js framework *benchmark* adalah sebuah repository pengujian *front-end javascript* yang dibuat oleh krausest, repository ini mendapat 6.6K star digithub. Repository ini digunakan untuk melakukan pengujian *framework front-end javascript*.

2.2.10 State Of Javascript

State of javascript adalah sebuah *website* yang berisikan laporan tahunan yang menyediakan wawasan mendalam tentang ekosistem *javacsript*. Laporan ini mencakup berbagai aspek termasuk tren penggunaan, preferensi pengembang, dan alat-alat terkait *Javascript*. Tujuan utamanya adalah untuk memberikan pemahaman yang lebih baik tentang bagaimana *Javascript* digunakan diseluruh industri pengembangan perangkat lunak.

2.2.11 React Js

React js adalah *front-end library* yang dikembangkan oleh *facebook*. *React js* digunakan sebagai pendukung dari *web-framework*, *react js* memiliki beberapa keunggulan diantaranya memberikan kecepatan, *simplicity*, dan *scalability*. *React js* memungkinkan pengembang dapat membangun sebuah komponen ui yang lebih interaktif, stateful, & reusable.(Panjaitan & Pakpahan, 2021)

2.2.12 Vue Js

Vue.js adalah sebuah *framework JavaScript* yang ringan dan fleksibel, serta mudah dipelajari dan digunakan untuk membangun aplikasi web. Vue.js dapat dengan mudah diintegrasikan dengan proyek- proyek yang

sudah ada, dan memiliki fitur-fitur seperti *directive*, *component*, dan *template syntax* yang memudahkan pengembang untuk membangun aplikasi web dengan lebih cepat dan efisien.(Dwi Bima Sakti et al., 2024)

2.2.13 Qwik Js

Qwik adalah kerangka kerja *JavaScript* modern yang dirancang untuk menciptakan aplikasi web yang sangat cepat dengan meminimalkan kebutuhan unduhan dan eksekusi *JavaScript* di sisi klien. Fokus utama Qwik adalah "*resumability*," yaitu kemampuan untuk melanjutkan eksekusi aplikasi dari kondisi terakhir yang ditinggalkan di server, tanpa proses "*hydration*" yang biasanya membebani waktu muat aplikasi. Qwik juga memanfaatkan strategi pemuatan *JavaScript* secara bertahap berdasarkan interaksi pengguna, yang mengurangi ukuran awal *bundle JavaScript*. Selain itu, Qwik mendukung rendering sisi *server (SSR)* untuk memastikan performa dan SEO yang optimal. Pendekatan inovatif ini membuat Qwik sangat cocok untuk aplikasi yang membutuhkan waktu *respons* cepat, seperti aplikasi *e-commerce* dan *Progressive Web Apps (PWAs)*.(Ayu et al., 2021)

2.2.14 Svelte Js

Svelte adalah kerangka kerja *JavaScript* yang lebih mudah dan lebih efisien untuk membangun antarmuka pengguna yang cepat. Svelte berbeda dari kerangka kerja *frontend* konvensional seperti *react* atau *vue*, yang melakukan sebagian besar pekerjaannya di-browser. Svelte berfokus pada pemindahan sebagian besar tugas yang biasanya dilakukan di-browser selama runtime ke tahap kompilasi. Ketika Anda membangun aplikasi

dengan svelte, komponen svelte Anda dikompilasi menjadi *javaScript* yang sangat efektif yang secara langsung memperbarui *document object model (DOM)*.(Kurniawan & Santoso, 2023)

2.2.15 Solid Js

SolidJS adalah kerangka kerja *javaScript* baru yang dirancang untuk meningkatkan kinerja aplikasi web, solid js mampu memberikan kinerja yang lebih baik dibandingkan kerangka kerja *javaScript* lainnya seperti react, angular, vue, dan svelte. Penelitian tersebut menunjukkan bahwa aplikasi web yang dikembangkan menggunakan solid js memiliki skor lighthouse yang lebih tinggi dan waktu muat halaman yang lebih singkat. SolidJS menggunakan teknik optimalisasi waktu kompilasi dan model pemrograman reaktif untuk meminimalkan ukuran input, yang berkontribusi pada peningkatan kinerja rendering aplikasi. Kerangka kerja ini juga mendukung pengembangan aplikasi web dengan interaktivitas yang tinggi dan waktu respons yang cepat, sehingga cocok untuk digunakan dalam berbagai aplikasi web modern.(Mantik et al., 2021)

2.2.16 Justifikasi Pemilihan Framework

Berikut adalah justifikasi pemilihan *framework* yang dapat digunakan berdasarkan data dari *State of JavaScript 2023* dan tren 2024:

1. React

- a) Popularitas Tinggi: React tetap menjadi *framework* yang paling banyak digunakan oleh pengembang dengan 57% dari total pengguna *JavaScript* memilihnya. Hal ini mencerminkan

keandalannya dalam berbagai jenis proyek, mulai dari aplikasi sederhana hingga *aplikasi web kompleks*.

- b) Ekosistem Matang: Dengan komunitas besar dan banyak alat pendukung, React menawarkan fleksibilitas dan kemudahan integrasi dengan teknologi lainnya, membuatnya ideal untuk proyek berskala besar.
- c) Keunggulan Teknis: Arsitektur berbasis komponen dan dukungan JSX memungkinkan pengembang untuk menulis kode yang modular dan dapat digunakan kembali, meningkatkan efisiensi pengembangan.

2. Vue

- a) Kemudahan Penggunaan: Vue dikenal dengan kurva pembelajarannya yang lebih rendah dibandingkan *framework* lainnya. Struktur sederhana dan fleksibel membuatnya cocok untuk pemula maupun profesional yang membutuhkan pengembangan cepat.
- b) Fleksibilitas: Vue memungkinkan pengguna untuk mengintegrasikan sebagian framework ke dalam proyek yang ada tanpa memerlukan adopsi penuh, menjadikannya ideal untuk proyek skala kecil hingga menengah.
- c) Tren Global: Meskipun memiliki popularitas tinggi di Asia, Vue terus menarik minat global dengan kemampuan penciptaan aplikasi yang ringan dan responsif.

3. Solid

- a) Performa Tinggi: Solid mengungguli banyak *framework* dalam hal waktu eksekusi dan efisiensi memori, menjadikannya salah satu *framework* terbaik untuk aplikasi dengan interaktivitas tinggi.
- b) Komunitas Berkembang: Walaupun komunitasnya tidak sebesar React atau Vue, Solid telah menarik perhatian pengembang dengan pendekatan reaktifnya yang mirip dengan React.

4. Svelte

- a) Pendekatan Inovatif: Svelte memindahkan sebagian besar pekerjaan dari *runtime* ke tahap kompilasi, menghasilkan aplikasi dengan performa lebih cepat dan *overhead* yang lebih rendah dibandingkan *framework* lain seperti React dan Vue
- b) Efisiensi Kinerja: Dengan ukuran bundle kecil dan kemampuan untuk menghasilkan aplikasi yang sangat cepat, Svelte sangat cocok untuk proyek yang mengutamakan performa dan waktu muat cepat
- c) Kekurangan: Dokumentasi dan komunitasnya relatif kecil dibandingkan *framework* lain, yang mungkin menjadi tantangan dalam pengembangan berskala besar.

5. Qwik

- a) Arsitektur Modern: Qwik dirancang untuk aplikasi yang mendukung pengiriman "streaming-first" atau *resumable apps*. Dengan strategi ini, Qwik hanya memuat dan menjalankan

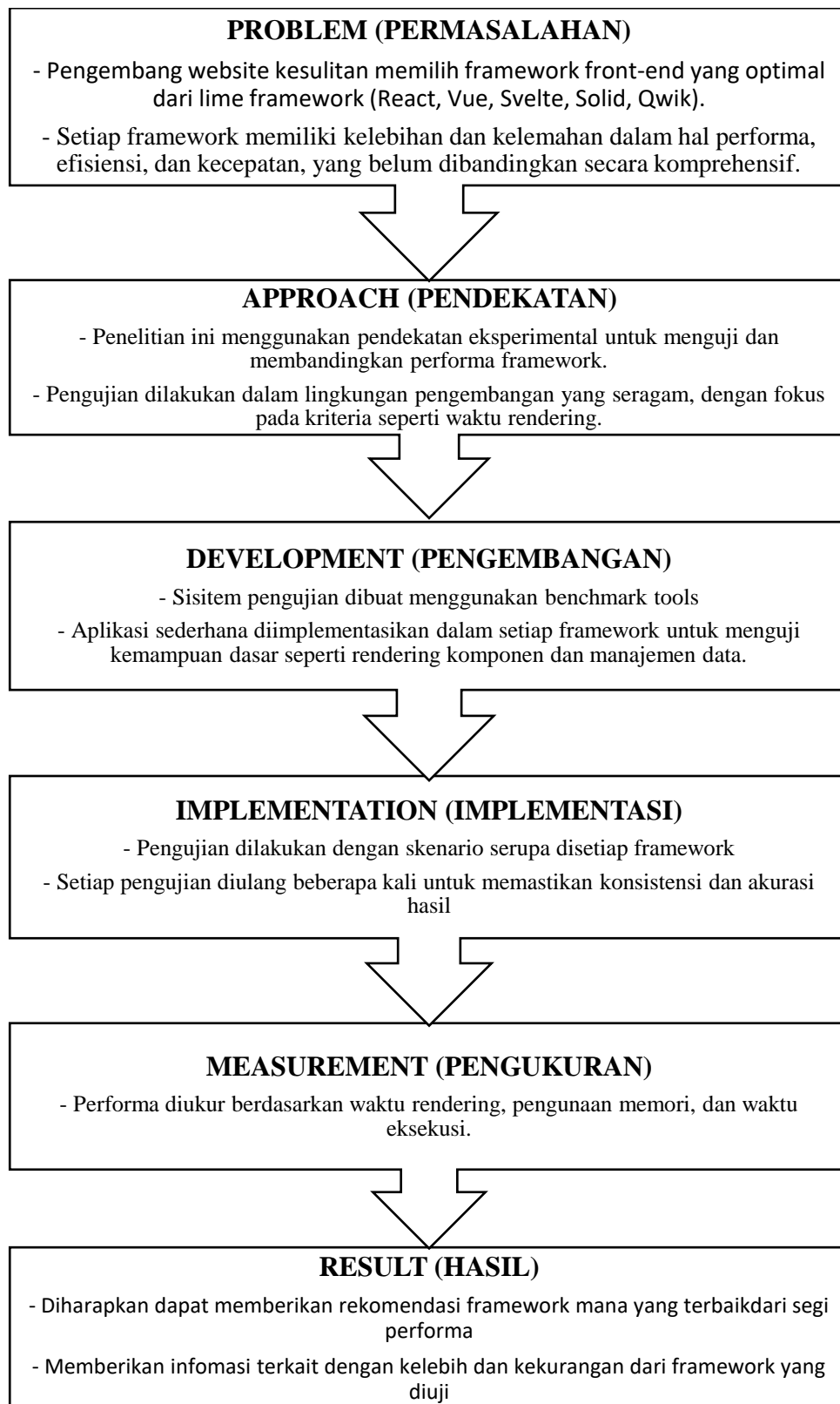
bagian UI yang diperlukan, sehingga meningkatkan performa pada koneksi lambat atau perangkat dengan sumber daya rendah.

- b) Pendekatan Baru: Menggunakan konsep rendering *progressive hydration*, Qwik dapat meningkatkan kecepatan startup pada aplikasi skala besar.
- c) Populer untuk *Edge Computing*: Karena kemampuannya menangani aplikasi yang tersebar di *cloud* atau *edge*, Qwik menarik perhatian untuk aplikasi web modern.
- d) Cocok Untuk: Proyek berskala besar yang membutuhkan pengoptimalan tinggi untuk aplikasi *web* yang sering digunakan pada perangkat dengan spesifikasi rendah atau lingkungan *cloud*.

2.3 Analisis Data

Data hasil pengujian akan dikelompokkan ke dalam tiga kategori utama: waktu rendering, penggunaan memori, dan ukuran bundle *framework*. Setiap kategori dianalisis untuk membandingkan performa *framework* React, Vue, Svelte, Solid, dan Qwik berdasarkan skenario pengujian seperti memuat, mengganti, dan memperbarui data dengan jumlah baris yang berbeda. Hasil analisis disajikan dalam bentuk grafik batang untuk menunjukkan rata-rata waktu rendering, grafik garis untuk menggambarkan pola konsumsi memori terhadap jumlah data, dan grafik lingkaran untuk membandingkan proporsi ukuran bundle setiap *framework*. Visualisasi ini membantu mengevaluasi keunggulan masing-masing *framework*, sehingga memudahkan dalam menentukan pilihan *framework* yang paling sesuai berdasarkan kebutuhan proyek.

2.4 Kerangka Pemikiran



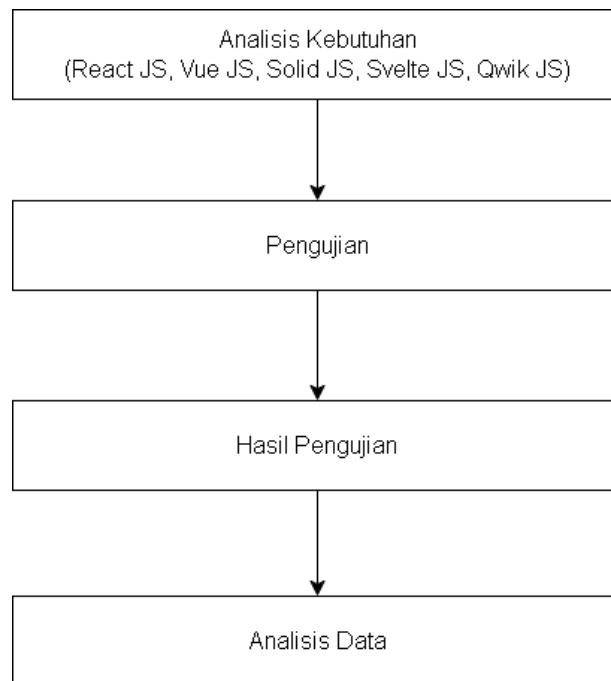
Gambar 2.1 Kerangka Pemikiran

BAB III

METODOLOGI PENELITIAN

3.1 Jenis Penelitian

Penelitian ini merupakan penelitian kuantitatif eksperimental yang bertujuan untuk membandingkan kinerja lima *framework JavaScript* dalam pengembangan *front-end website*. Penelitian ini fokus pada perbandingan kriteria seperti kecepatan *rendering*, efisiensi memori dan ukuran *bundle*. Berikut adalah diagram alur penelitian dapat dilihat pada gambar 3.1.



Gambar 3.1 Diagram Alur Penelitian

3.1.1 Analisis Kebutuhan

Pada tahap ini, dilakukan identifikasi kebutuhan perangkat keras dan perangkat lunak yang digunakan selama penelitian. Semua *framework* diuji dalam lingkungan yang sama dengan spesifikasi perangkat keras yang

konsisten. Hal ini dilakukan untuk menjaga objektivitas hasil dan memastikan bahwa perbedaan kinerja yang diamati adalah hasil dari *framework*, bukan perangkat yang digunakan.

Dibawah ini disebutkan beberapa kebutuhan yang dapat mendukung penelitian ini agar dapat berjalan sesuai tujuan, diantaranya:

a. Perangkat keras (Hardware)

Perangkat keras yang digunakan dalam penelitian guna mewujudkan tujuan penelitian yaitu perangkat laptop sebagai uji coba dengan spesifikasi berikut:

1. Prosesor: AMD Ryzen 3 3200U
2. RAM: 12 GB
3. Penyimpanan: SSD 512GB
4. Sistem Operasi: Windows 11 Home 64-bit

b. Kebutuhan perangkat lunak (Software)

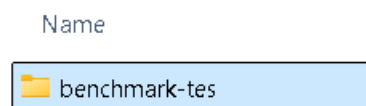
Perangkat lunak berfungsi untuk pengoperasian sistem pada penelitian ini. Pada penelitian ini digunakan adalah sebagai berikut:

1. Google Chrome: Versi 114.0.5735.199 (32-bit) untuk pengujian *browser*.
2. Visual Studio Code: Versi 1.79.2 sebagai *Integrated Development Environment (IDE)* untuk pengembangan dan pengujian kode.
3. Node.js: Versi 16.19.0 (LTS) untuk lingkungan pengembangan *JavaScript*.
4. NPM: Versi 8.19.3 untuk manajemen paket.

3.1.2 Pengujian

Lingkungan pengujian dirancang secara konsisten sesuai dengan kebutuhan penelitian. Setelah tahap persiapan lingkungan pengujian selesai, langkah berikutnya adalah melaksanakan implementasi dan pengujian untuk setiap kasus yang telah ditentukan. Pada tahap ini, penulis perlu mencatat durasi waktu yang dibutuhkan untuk setiap operasi dan menganalisis perbandingan performa antara React, Vue, Solid, Svelte dan Qwik.

Berikut adalah tahapan pembuatan alat benchmark. Ditahapan pertama membuat folder benchmar dimana didalam folder ini nantinya akan berisikan kelima framework yang akan diuji. Untuk lebih jelas dapat dilihat pada gambar 3.2.



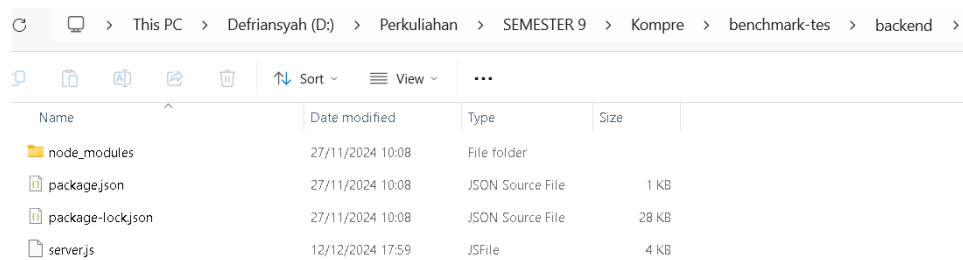
Gambar 3.2 Membuat Folder Benchmark

Tahapan kedua dilanjutkan dengan membuat project untuk kelima framework yang akan diuji, framework yang dipilih yaitu react, vue, solid, svelte dan qwik dimana framework yang di pasang menggunakan versi paling terbaru untuk saat ini. Untuk lebih jelas dapat dilihat pada gambar 3.3.

qwik-benchmark	11/12/2024 22:35	File folder
react-benchmark	27/11/2024 12:26	File folder
solid-benchmark	11/12/2024 22:22	File folder
svelte-benchmark	11/12/2024 22:25	File folder
vue-benchmark	27/11/2024 12:35	File folder

Gambar 3.3 Setup Project Kelima Framework

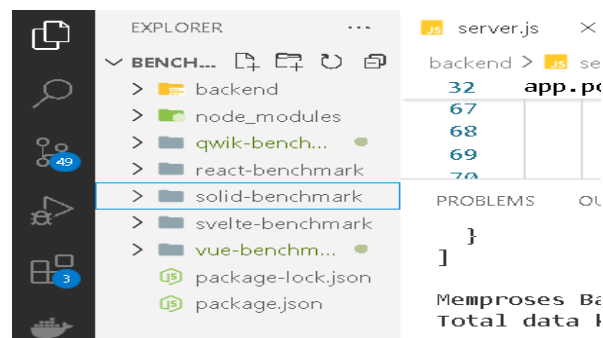
Tahapan ketiga menyiapkan backend untuk mengelola berbagai sampel data dummy yang digunakan untuk mengukur sebaik apa framework tersebut dan proses benchmark akan mengikuti skenario yang telah ditentukan sebelumnya. Dapat dilihat pada gambar 3.4.



Name	Date modified	Type	Size
node_modules	27/11/2024 10:08	File folder	
package.json	27/11/2024 10:08	JSON Source File	1 KB
package-lock.json	27/11/2024 10:08	JSON Source File	28 KB
server.js	12/12/2024 17:59	JSFile	4 KB

Gambar 3.4 Menyiapkan Backend

Untuk lebih lengkapnya struktur folder alat benchmark yang digunakan dapat dilihat pada gambar 3.5.



Gambar 3.5 Struktur Folder Benchmark

3.1.3 Hasil Pengujian

Data dikumpulkan melalui serangkaian uji performa yang meliputi *rendering time* waktu yang dibutuhkan oleh *framework* untuk Reaktivitas: Kemampuan *framework* untuk merespons perubahan UI secara cepat dan efisien.

Pengujian atau *benchmarking rendering time* dilakukan sebanyak 10 kali untuk memastikan konsistensi hasil yang diperoleh. Melakukan pengujian sebanyak 10 kali dapat mengurangi kemungkinan adanya anomali atau outlier yang dapat mempengaruhi hasil. Dalam setiap pengulangan, *framework* yang diuji dijalankan dengan parameter yang sama dan dalam kondisi yang serupa untuk memastikan bahwa setiap pengujian dilakukan dalam lingkungan yang sebanding.

Dalam pengujian berulang, beberapa nilai penting dihitung untuk memberikan informasi yang lebih *komprehensif* tentang data yang diperoleh. Nilai-nilai ini meliputi:

1. Nilai waktu esekusi (ms)
2. Nilai memori yang digunakan (MB)
3. Ukuran bundle setiap framework(MB)

Berikut adalah skenario benchmark yang akan digunakan untuk melakukan pengujian kelima framework tersebut dapat dilihat pada tabel

3.1.

Tabel 3.1 Skenario Pengujian Framework

No	Data Masuk	Jumlah Data Masuk (Per Batch)	Total Data yang Masuk
1	Batch ke-1	100	100
2	Batch ke-2	100	200
3	Batch ke-3	100	300
4	Batch ke-4	100	400
5	Batch ke-5	100	500
6	Batch ke-6	100	600
7	Batch ke-7	100	700
8	Batch ke-8	100	800
9	Batch ke-9	100	900
10	Batch ke-10	100	1000

Skenario di atas menggambarkan proses pemasukan data secara bertahap dalam sepuluh batch, di mana setiap batch berisi 100 data. Pada batch pertama, 100 data dimasukkan, sehingga total data menjadi 100. Kemudian, data terus ditambahkan dalam jumlah yang sama pada setiap batch berikutnya, sehingga total data yang terkumpul meningkat secara progresif. Pada batch ke-10, total data yang berhasil masuk mencapai 1.000. Proses ini menunjukkan pengelolaan data yang konsisten dan terorganisasi, dengan peningkatan bertahap yang seragam di setiap tahap.

Berikut adalah langkah-langkah menjalankan benchmark terhadap kelima framework yang akan diuji. Tahapan pertama menjalankan backend, masuk ke dalam direktori backend lalu jalankan server untuk lebih jelas dapat dilihat pada gambar 3.6.

```

PS D:\Perkuliahan\SEMESTER 9\Kompre\benchmark-tes> cd backend
PS D:\Perkuliahan\SEMESTER 9\Kompre\benchmark-tes\backend> node server.js
Server backend berjalan di http://localhost:5000

```

Gambar 3.6 Menjalankan Server Backend

Selanjutnya jalankan framework yang akan diuji disini penulis mencontohkan solid js. Dapat dilihat pada gambar 3.7.

```

PS D:\Perkuliahan\SEMESTER 9\Kompre\benchmark-tes\react-benchmark> cd ..
PS D:\Perkuliahan\SEMESTER 9\Kompre\benchmark-tes> cd solid-benchmark
PS D:\Perkuliahan\SEMESTER 9\Kompre\benchmark-tes\solid-benchmark> npm run dev

> solid-benchmark@0.0.0 dev

Re-optimizing dependencies because vite config has changed

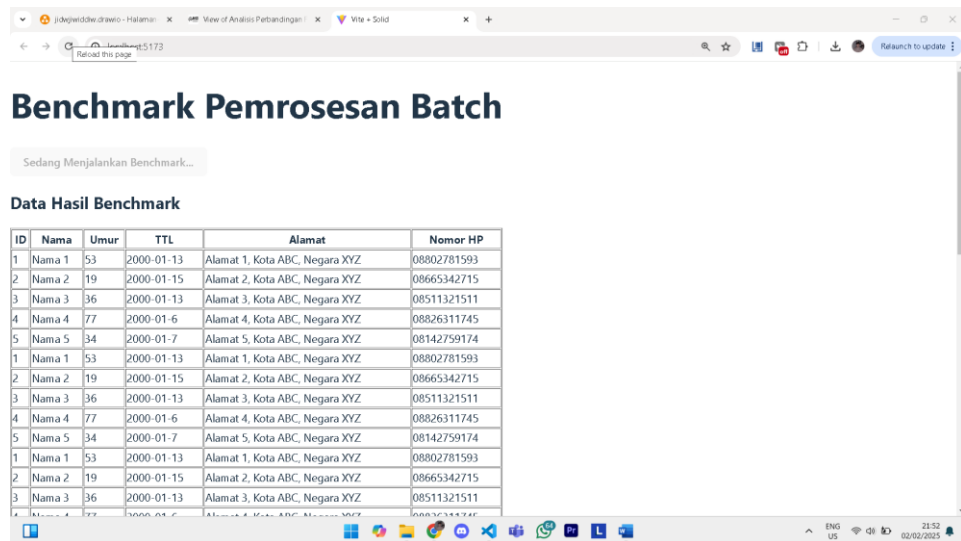
VITE v5.4.11 ready in 3187 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose

```

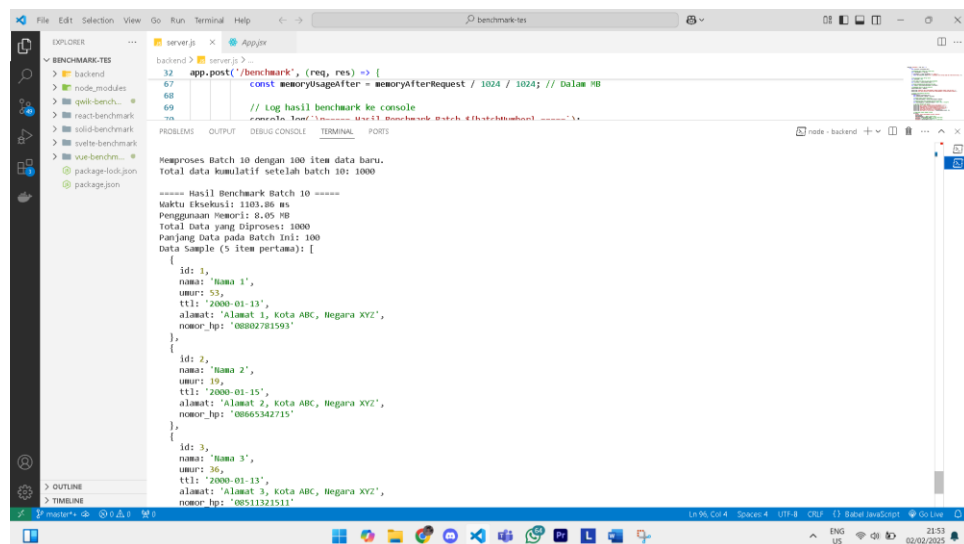
Gambar 3.7 Menjalankan Framework

Selanjutnya tampilan ui sederhana akan muncul dichrome lalu klik mulai benchmark untuk menjalankan benchmark pada framework solid js. Dapat dilihat pada gambar 3.8.



Gambar 3.8 Menjalankan Benchmark Pada Solid JS

Benchmark akan berjalan dilatar belakang secara otomatis menjalankan sesuai dengan skenario yang telah ditentukan secara bertahap dan langsung menampilkan hasil dengan 3 kategori yaitu waktu eksekusi, penggunaan memori dan ukuran bundle. Dapat dilihat pada gambar 3.9.



Gambar 3.9 Hasil Menjalankan Pengujian Framework Solid JS

3.1.4 Analisis Data

Seterlah data dikumpulkan, data akan di proses untuk memberikan kesimpulan mengenai framework mana yang terbaik berdasarkan waktu eksekusi, penggunaan memori dan ukuran bundle dengan demikian, analisis data akan menjadi landasan untuk memberikan rekomendasi mengenai framework mana yang lebih cocok untuk digunakan dalam pengembangan website

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil Penelitian

Data dalam penelitian ini diolah menggunakan metode pengujian bertahap sesuai skenario yang telah ditentukan. Setiap framework diuji dengan skenario penambahan data secara bertahap dari 100 hingga 1000 item, dengan pembagian ke dalam 10 batch (100 item per batch). Pengolahan data meliputi

Pengujian atau *benchmarking rendering time* dilakukan sebanyak 10 kali untuk memastikan konsistensi hasil yang diperoleh. Melakukan pengujian sebanyak 10 kali dapat mengurangi kemungkinan adanya anomali atau *outlier* yang dapat mempengaruhi hasil. Dalam setiap pengulangan, framework yang diuji dijalankan dengan *parameter* yang sama dan dalam kondisi yang serupa untuk memastikan bahwa setiap pengujian dilakukan dalam lingkungan yang sebanding.

Dalam pengujian berulang, beberapa nilai yang penting dihitung untuk memberikan informasi yang lebih komprehensif tentang data yang diperoleh. Nilai-nilai ini meliputi:

1. Nilai waktu esekusi (ms)

Dalam konteks pengujian nilai waktu eksekusi, pengukuran dilakukan berdasarkan waktu *rendering* yang diperlukan untuk menyelesaikan tugas atau operasi tertentu. Waktu *rendering* ini diukur dengan menggunakan satuan millisecond (ms). Semakin kecil nilai

waktu *rendering*, semakin baik performa dalam menyelesaikan tugas dengan cepat.

2. Nilai memori yang digunakan (MB)

Dalam konteks pengujian nilai memori yang digunakan, pengukuran dilakukan berdasarkan berapa memori yang digunakan dalam proses *benchmark* berjalan. Memori yang digunakan ini diukur dengan menggunakan satuan megabyte (mb). Semakin kecil nilai penggunaan memori, semakin baik performa dalam menyelesaikan tugas dengan cepat.

3. Ukuran bundle setiap framework(KB)

Dalam konteks pengujian ukuran bundle, pengukuran dilakukan berdasarkan total dari ukuran project dari *framework*. Total ukuran bundle ini diukur dengan menggunakan satuan megabyte (mb). Semakin kecil nilai ukuran bundle, semakin baik.

Dalam konteks pengujian tipe waktu eksekusi, pengukuran dilakukan berdasarkan waktu rendering yang diperlukan untuk menyelesaikan tugas atau operasi tertentu. Waktu rendering ini diukur dengan menggunakan satuan millisecond (ms). Semakin kecil nilai waktu rendering, semakin baik performa CPU dalam menyelesaikan tugas dengan cepat.

Sementara itu, dalam pengujian tipe memori, pengukuran dilakukan berdasarkan jumlah memori RAM yang dibutuhkan oleh sistem atau aplikasi saat

dieksekusi. Pengukuran ini menggunakan satuan megabyte (MB). Performa yang lebih baik dalam pengujian tipe memori akan ditunjukkan oleh kebutuhan memori yang lebih rendah.

Pada pengujian ukuran bundle, setiap framework akan dibangun ke production untuk melihat seberapa besar ukuran bundle dari setiap framework dan menentukan framework mana yang memiliki ukuran bundle yang sedikit

1. Hasil pengujian memori

Berikut adalah hasil dari pengujian tipe memori dari kelima framework yang diuji. Berikut adalah hasil pengujian memori dari framework react dapat dilihat pada gambar 4.1.

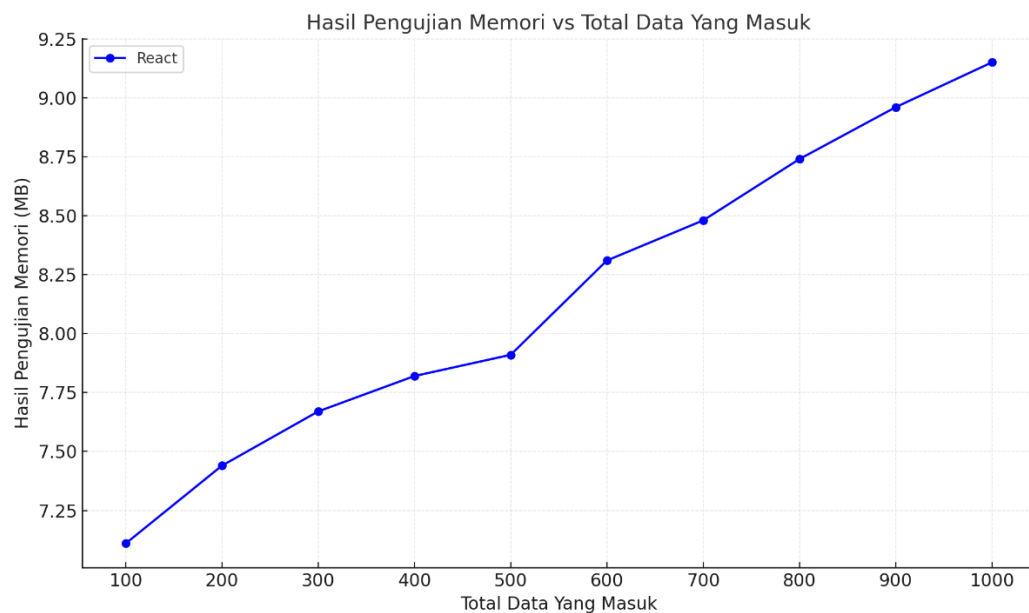
Tabel 4.1 Hasil Pengujian Memori Framework React

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Memori (MB)
			React JS
Batch ke-1	100	100	7.11
Batch ke-2	100	200	7.44
Batch ke-3	100	300	7.67
Batch ke-4	100	400	7.82
Batch ke-5	100	500	7.91
Batch ke-6	100	600	8.31
Batch ke-7	100	700	8.48
Batch ke-8	100	800	8.74

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Memori (MB)
			React JS
Batch ke-9	100	900	8.96
Batch ke-10	100	1000	9.15

Berikut adalah chart dari hasil pengujian memori untuk framework react.

Dapat dilihat pada gambar 4.1.



Gambar 4.1 Chart Hasil Pengujian Memori Framework React

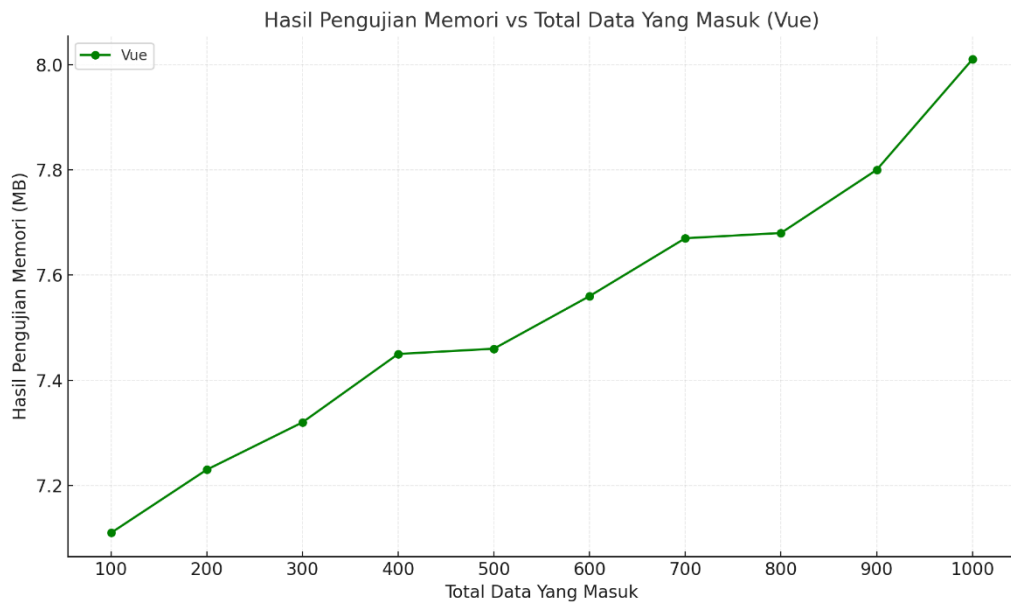
Selanjutnya hasil pengujian memori untuk framework vue, dapat dilihat pada tabel 4.2.

Tabel 4.2 Hasil Pengujian Momori Framework Vue

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Memori (MB)
			React JS
Batch ke-1	100	100	7.11
Batch ke-2	100	200	7.23
Batch ke-3	100	300	7.32
Batch ke-4	100	400	7.45
Batch ke-5	100	500	7.46
Batch ke-6	100	600	7.56
Batch ke-7	100	700	7.67
Batch ke-8	100	800	7.68
Batch ke-9	100	900	7.80
Batch ke-10	100	1000	8.01

Berikutnya adalah chart hasil pengujian memori pada framework vue.

Dapat dilihat pada gambar 4.2.



Gambar 4.2 Chart Hasil Pengujian Memori Framework Vue

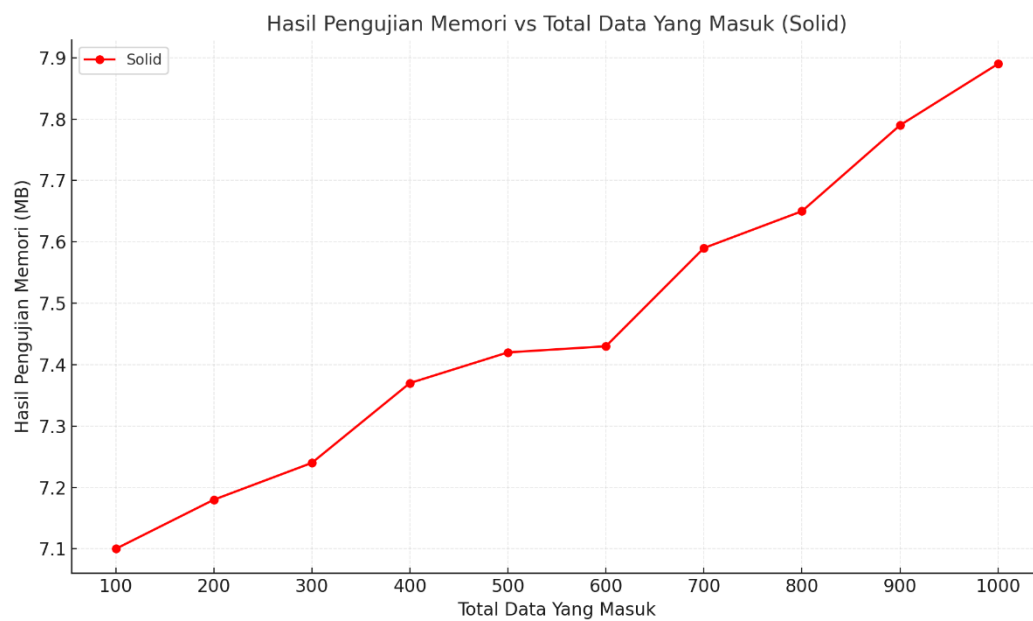
Selanjutnya hasil pengujian memori pada framework Solid. Dapat dilihat pada tabel 4.3.

Tabel 4.3 Hasil Pengujian Memori Framework Solid

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Memori (MB)
			Solid JS
Batch ke-1	100	100	7.10
Batch ke-2	100	200	7.18
Batch ke-3	100	300	7.24
Batch ke-4	100	400	7.37
Batch ke-5	100	500	7.42
Batch ke-6	100	600	7.43
Batch ke-7	100	700	7.59

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Memori (MB)
			Solid JS
Batch ke-8	100	800	7.65
Batch ke-9	100	900	7.79
Batch ke-10	100	1000	7.89

Berikut chart hasil pengujian memori pada framework solid. Dapat dilihat pada gambar 4.3.



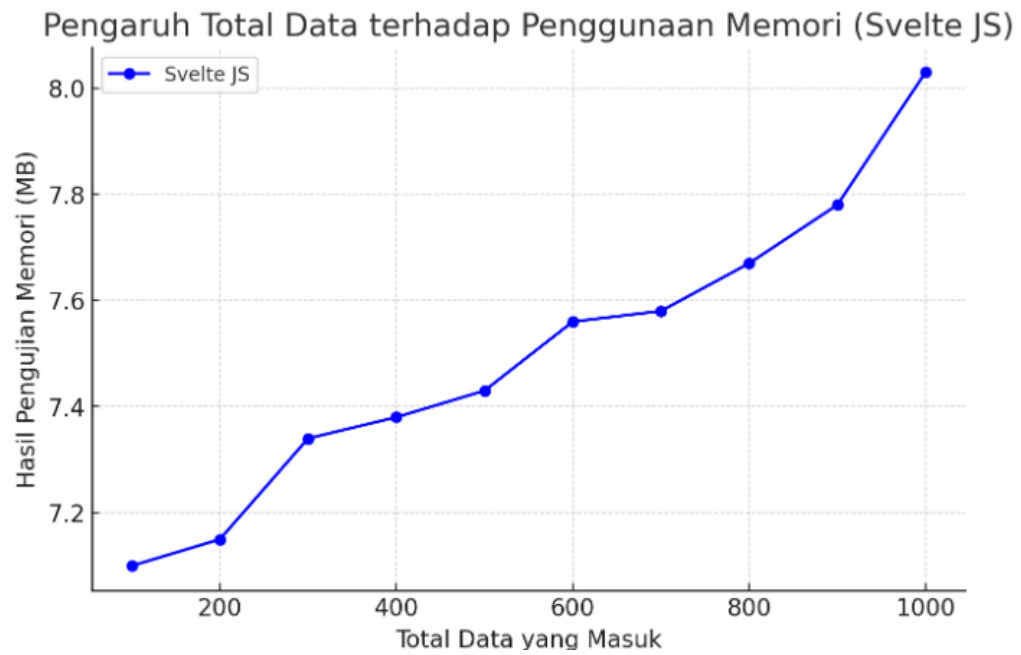
Gambar 4.3 Chart Hasil Pengujian Memori Framework Solid

Berikutnya hasil pengujian memori pada framework svelte. Dapat dilihat pada tabel 4.4.

Tabel 4.4 Hasil Pengujian Memori Framework Svelte

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Memori (MB)
			Svelte JS
Batch ke-1	100	100	7.10
Batch ke-2	100	200	7.15
Batch ke-3	100	300	7.34
Batch ke-4	100	400	7.38
Batch ke-5	100	500	7.43
Batch ke-6	100	600	7.56
Batch ke-7	100	700	7.58
Batch ke-8	100	800	7.67
Batch ke-9	100	900	7.78
Batch ke-10	100	1000	8.03

Berikut adalah chart hasil pengujian pada framework Svelte. Dapat dilihat pada gambar 4.4.



Gambar 4.4 Chart Hasil Pengujian Memori Framework Svelte

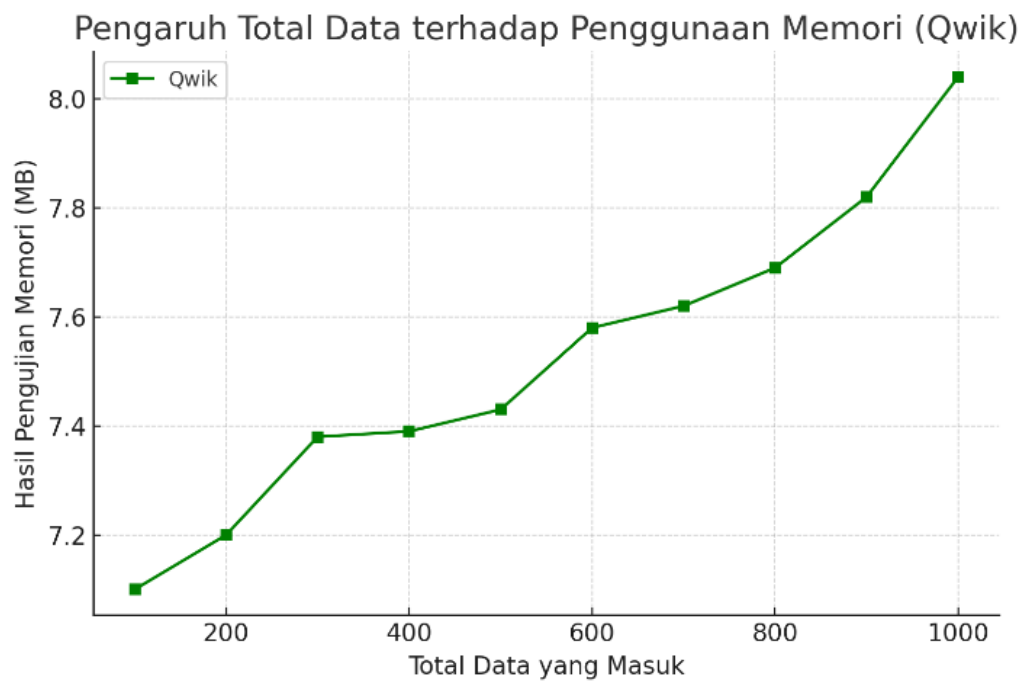
Selanjutnya hasil pengujian memori pada framework qwik. dapat dilihat pada tabel 4.5.

Tabel 4.5 Hasil Pengujian Memori Framework Qwik

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Memori (MB)
			Qwik JS
Batch ke-1	100	100	7.10
Batch ke-2	100	200	7.20
Batch ke-3	100	300	7.38
Batch ke-4	100	400	7.39
Batch ke-5	100	500	7.43
Batch ke-6	100	600	7.58

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Memori (MB)
			Qwik JS
Batch ke-7	100	700	7.62
Batch ke-8	100	800	7.69
Batch ke-9	100	900	7.82
Batch ke-10	100	1000	8.04

Berikutnyachart hasil pengujian memori pada framework Qwik. dapat dilihat pada gambar 4.5.



Gambar 4.5 Chart Hasil Pengujian Memori Framework Qwik

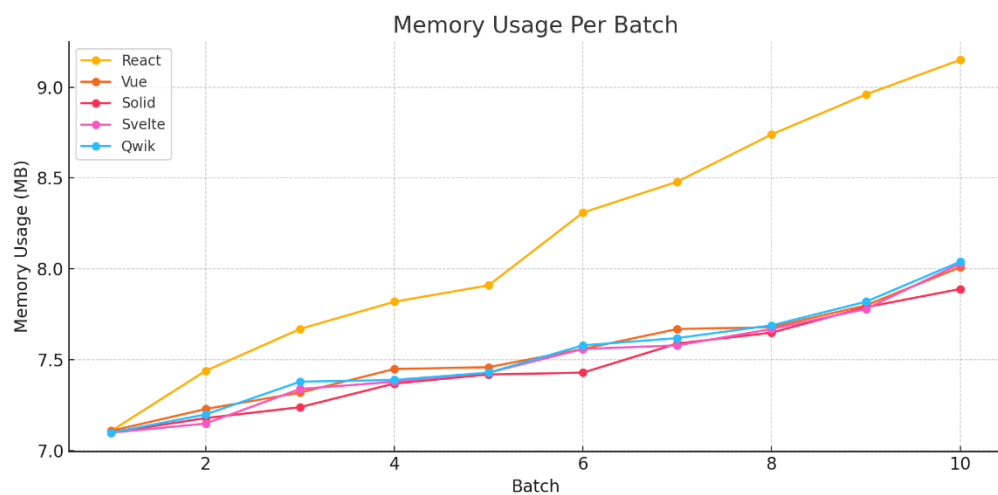
Dari semua tabel dan chart hasil pengujian kelima framework diatas maka dapat dilihat perbedaan framework mana yang memiliki penggunaan memori yang paling sedikit. Untuk lebih jelas dapat dilihat pada tabel 4.6.

Tabel 4.6 Hasil Pengujian Memori

Data Masuk	Jumlah Data Masuk (Per Batch)	Total Data yang Masuk	Hasil Pengujian Memori (MB)				
			React	Vue	Solid	Svelte	Qwik
Batch ke-1	100	100	7.11	7.11	7.10	7.10	7.10
Batch ke-2	100	200	7.44	7.23	7.18	7.15	7.20
Batch ke-3	100	300	7.67	7.32	7.24	7.34	7.38
Batch ke-4	100	400	7.82	7.45	7.37	7.38	7.39
Batch ke-5	100	500	7.91	7.46	7.42	7.43	7.43
Batch ke-6	100	600	8.31	7.56	7.43	7.56	7.58
Batch ke-7	100	700	8.48	7.67	7.59	7.58	7.62
Batch ke-8	100	800	8.74	7.68	7.65	7.67	7.69
Batch ke-9	100	900	8.96	7.80	7.79	7.78	7.82
Batch ke-10	100	1000	9.15	8.01	7.89	8.03	8.04

Pada tabel diatas hasil benchmark menunjukkan bahwa seiring dengan meningkatnya jumlah data yang diproses, semua *framework* mengalami kenaikan penggunaan memori, tetapi dengan kecepatan yang berbeda-beda. React mengalami lonjakan konsumsi memori paling besar, mencapai 9.15 MB di *batch* terakhir, sementara Vue juga mengalami kenaikan, tetapi lebih moderat, hingga 8.01 MB. Solid menunjukkan efisiensi memori terbaik, dengan penggunaan yang lebih rendah dan konsisten, berkisar antara 7.10 hingga 7.89 MB. Svelte dan Qwik

juga mengalami peningkatan, tetapi tetap berada di rentang yang relatif stabil. Secara keseluruhan, Solid tampil paling efisien dalam penggunaan memori, sementara React memerlukan lebih banyak memori, meskipun semua framework menunjukkan performa yang cukup baik dalam hal stabilitas. untuk lebih jelas dapat melihat chart pada gambar 4.1.



Gambar 4.6 Chart Hasil Pengujian Memori

2. Hasil Pengujian Waktu Eksekusi

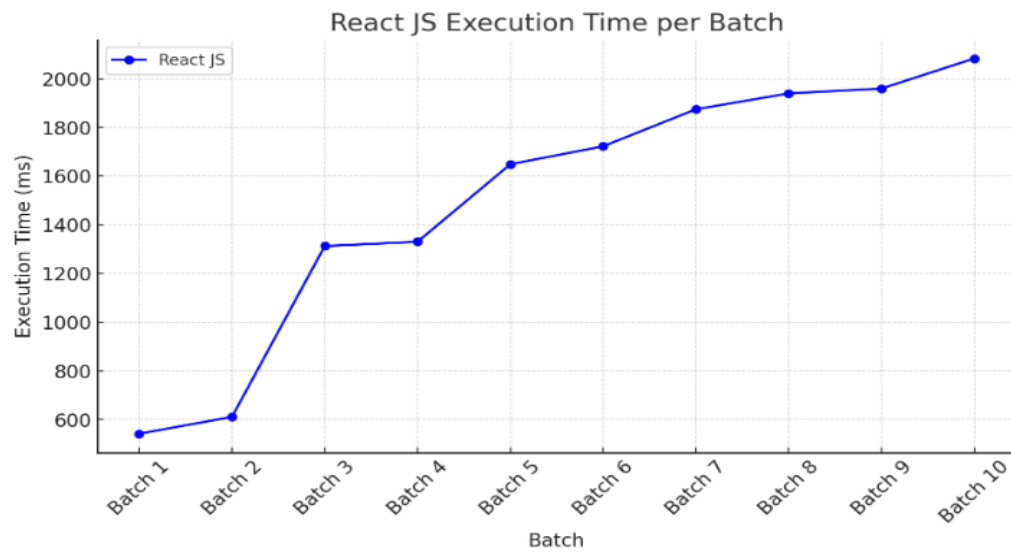
Berikut adalah hasil dari pengujian waktu eskekusi dari kelima framework.

Berikut adalah hasil pengujian waktu eksekusi pada framework react dapat dilihat pada tabel 4.7.

Tabel 4.7 Hasil Pengujian Waktu Eksekusi Framework React

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Waktu Eksekusi (MS)
			React JS
Batch ke-1	100	100	542.42
Batch ke-2	100	200	611.49
Batch ke-3	100	300	1313.26
Batch ke-4	100	400	1330.99
Batch ke-5	100	500	1649.31
Batch ke-6	100	600	1723.05
Batch ke-7	100	700	1875.04
Batch ke-8	100	800	1940.32
Batch ke-9	100	900	1960.16
Batch ke-10	100	1000	2083.76

Berikut chart hasil pengujian eksekusi waktu pada framework react. Dapat dilihat pada gambar 4.7.



Gambar 4.7 Chart Hasil Pengujian Eksekusi Waktu Framework React

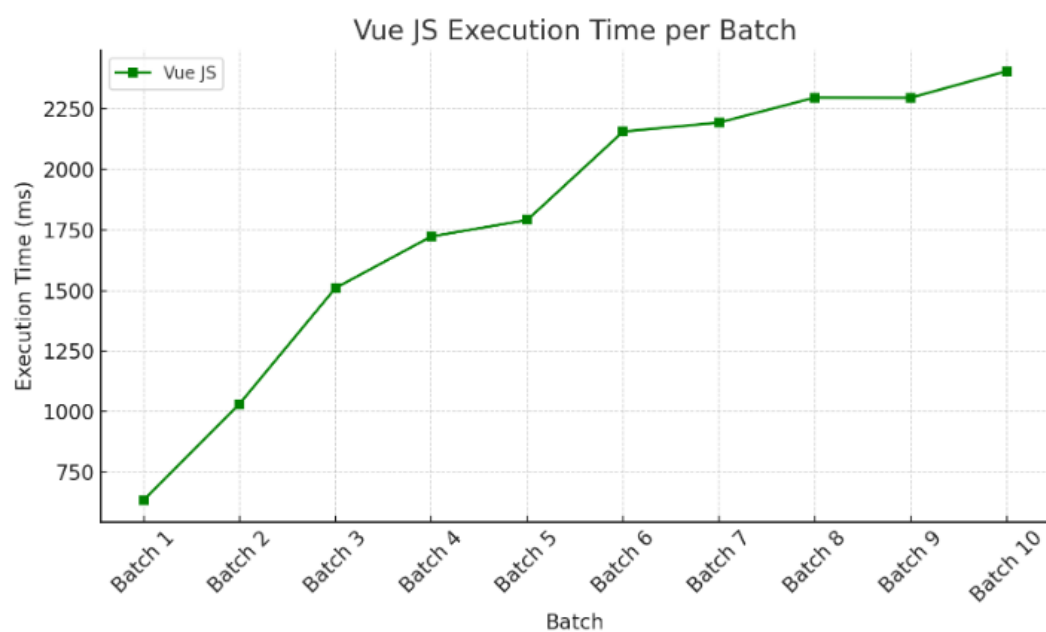
Selanjutnya hasil pengujian eksekusi waktu pada framework vue. Dapat dilihat pada tabel 4.8.

Tabel 4.8 Hasil Pengujian Eksekusi Waktu Framework Vue

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Waktu Eksekusi (MS)
			Vue JS
Batch ke-1	100	100	634.17
Batch ke-2	100	200	1031.75
Batch ke-3	100	300	1510.19
Batch ke-4	100	400	1723.47
Batch ke-5	100	500	1791.35
Batch ke-6	100	600	2156.83

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Waktu Eksekusi (MS)
			VueJS
Batch ke-7	100	700	2194.13
Batch ke-8	100	800	2297.44
Batch ke-9	100	900	2296.89
Batch ke-10	100	1000	2406.34

Berikutnya chart hasil pengujian waktu eksekusi framework vue. Dapat dilihat pada gambar 4.8.



Gambar 4.8 Chart Hasil Pengujian Waktu Eksekusi Framework Vue

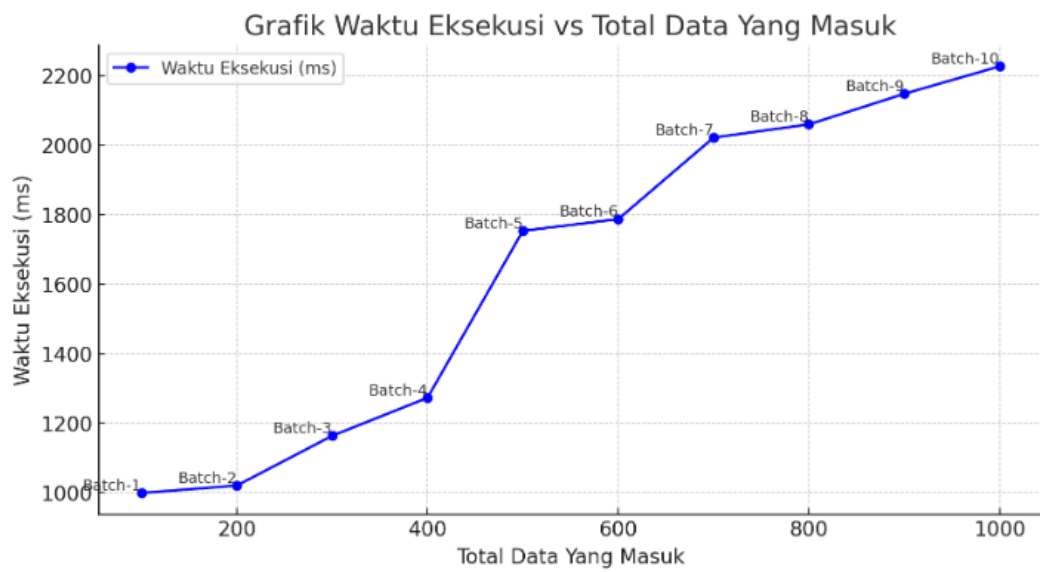
Selanjutnya hasil pengujian waktu eksekusi pada framework solid. Dapat dilihat pada tabel 4.9.

Tabel 4.9 Hasil Pengujian Waktu Eksekusi Framework Solid

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Waktu Eksekusi (MS)
			Solid JS
Batch ke-1	100	100	998.61
Batch ke-2	100	200	1019.90
Batch ke-3	100	300	1163.30
Batch ke-4	100	400	1272.81
Batch ke-5	100	500	1753.80
Batch ke-6	100	600	1787.41
Batch ke-7	100	700	2022.35
Batch ke-8	100	800	2060.55
Batch ke-9	100	900	2148.49
Batch ke-10	100	1000	2227.73

Berikut chart hasil dai pengujian waktu eksekusi pada framework solid.

Dapat dilihat pada gambar 4.9.



Gambar 4.9 Chart Hasil Pengujian Waktu Eksekusi Framework Solid

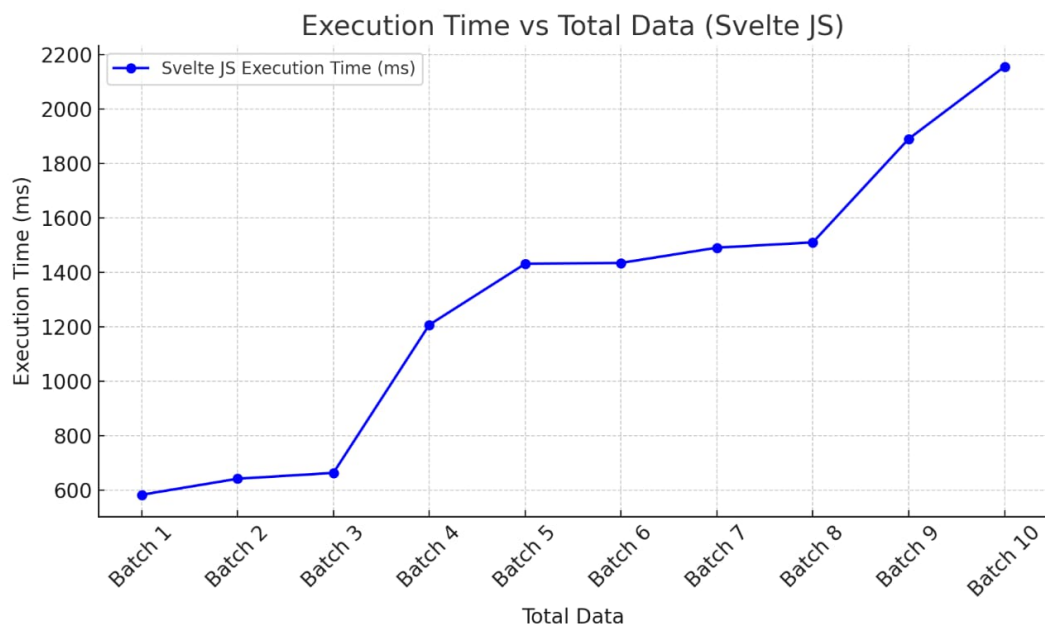
Selanjutnya hasil pengujian waktu eksekusi pada framework Svelte . dapat dilihat pada tabel 4.10.

Tabel 4.10 Hasil Pengujian Waktu Eksekusi Framework Svelte

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Waktu Eksekusi (MS)
			Svelte JS
Batch ke-1	100	100	582.91
Batch ke-2	100	200	641.74
Batch ke-3	100	300	663.13
Batch ke-4	100	400	1206.97
Batch ke-5	100	500	1431.00
Batch ke-6	100	600	1434.32

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Waktu Eksekusi (MS)
			SvelteJS
Batch ke-7	100	700	1490.27
Batch ke-8	100	800	1509.78
Batch ke-9	100	900	1890.01
Batch ke-10	100	1000	2155.34

Berikut chart hasil pengujian waktu eksekusi framework svelte. Dapat dilihat pada gambar 4.10.



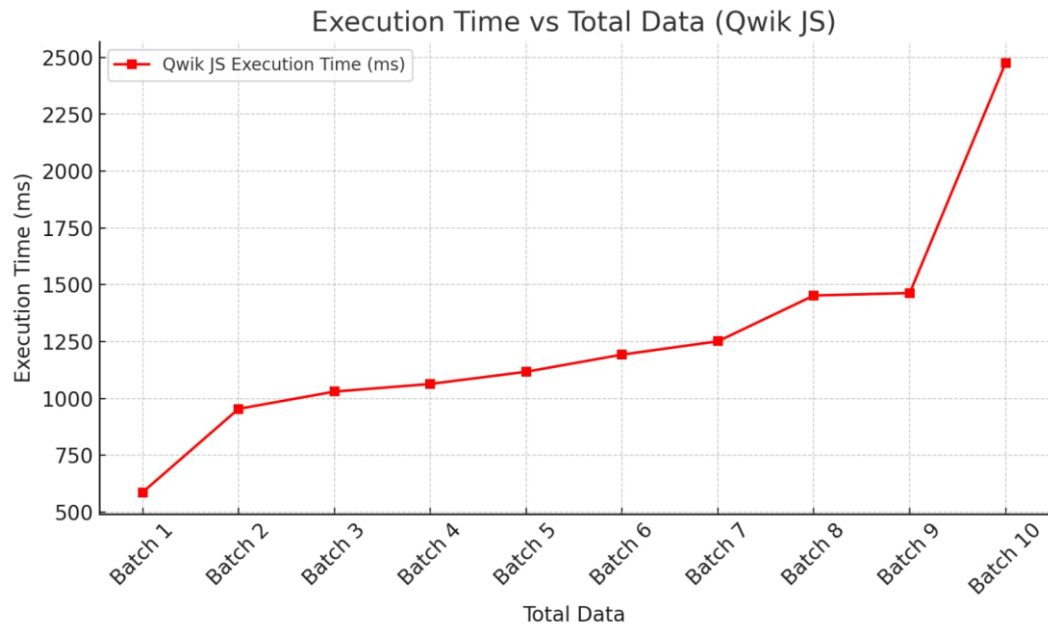
Gambar 4.10 Chart Hasil Pengujian Waktu Eksekusi Framework Svelte

Selanjutnya hasil pengujian waktu eksekusi pada framework qwik. dapat dilihat pada gambar 4.11.

Tabel 4.11 Hasil Pengujian Waktu Eksekusi Framework Qwik

Data Masuk	Jumlah Data (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Waktu Eksekusi (MS)
			Qwik JS
Batch ke-1	100	100	587.37
Batch ke-2	100	200	954.25
Batch ke-3	100	300	1030.27
Batch ke-4	100	400	1063.95
Batch ke-5	100	500	1117.55
Batch ke-6	100	600	1192.96
Batch ke-7	100	700	1251.54
Batch ke-8	100	800	1452.87
Batch ke-9	100	900	1463.80
Batch ke-10	100	1000	2476.46

Berikut chart hasil pengujian waktu eksekusi pada framework qwik. dapat dilihat pada gambar 4.11.



Gambar 4.11 Chart Hasil Pengujian Waktu Eksekusi Framework Qwik

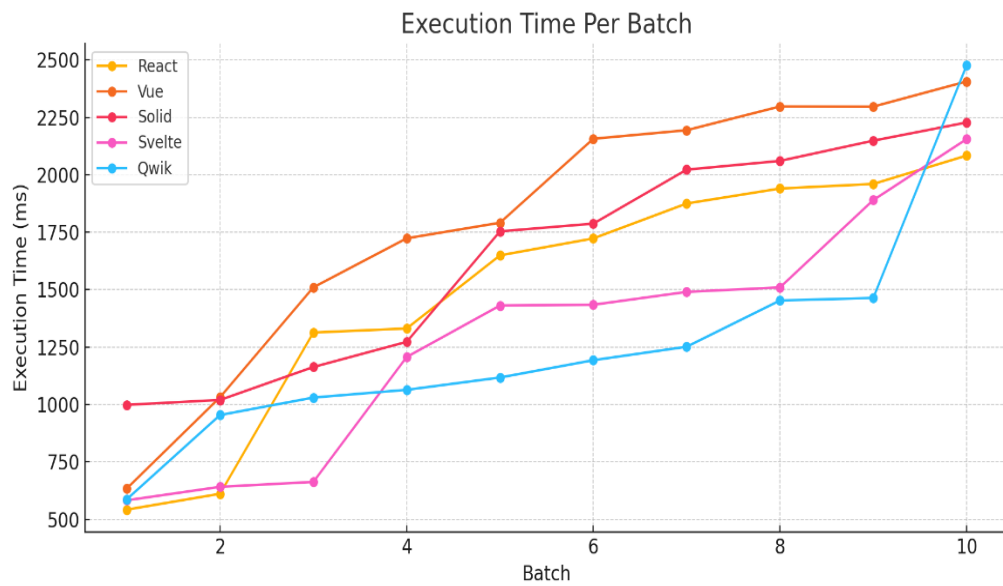
Dari keseluruhan hasil pengujian diatas dapat dibandingkan framework mana yang memiliki waktu eksekusi yang cepat dibandingkan dengan framework lainnya . untuk lebih jelas dapat dilihat pada tabel 4.12.

Tabel 4.12 Hasil Pengujian Waktu Eksekusi

Data Masuk	Jumlah Data Masuk (Per Batch)	Total Data yang Masuk	Hasil Pengujian Waktu Eksekusi (MS)				
			React	Vue	Solid	Svelte	Qwik
Batch ke-1	100	100	542.42	634.17	998.61	582.91	587.37
Batch ke-2	100	200	611.49	1031.75	1019.90	641.74	954.25
Batch ke-3	100	300	1313.26	1510.19	1163.30	663.13	1030.27
Batch ke-4	100	400	1330.99	1723.47	1272.81	1206.97	1063.95
Batch ke-5	100	500	1649.31	1791.35	1753.80	1431.00	1117.55
Batch ke-6	100	600	1723.05	2156.83	1787.41	1434.32	1192.96

Data Masuk	Jumlah Data Masuk (Per Batch)	Total Data Yang Masuk	Hasil Pengujian Waktu Eksekusi (MS)				
			React	Vue	Solid	Svelte	Qwik
Batch ke-7	100	700	1875.04	2194.13	2022.35	1490.27	1251.54
Batch ke-8	100	800	1940.32	2297.44	2060.55	1509.78	1452.87
Batch ke-9	100	900	1960.16	2296.89	2148.49	1890.01	1463.80
Batch ke-10	100	1000	2083.76	2406.34	2227.73	2155.34	2476.46

Pada tabel diatas hasil *benchmark* menunjukkan bahwa seiring dengan bertambahnya jumlah data yang diproses, waktu eksekusi untuk setiap *framework* meningkat, meskipun dengan kecepatan yang berbeda. React mengalami peningkatan waktu eksekusi yang cukup besar, mulai dari 542.42 ms pada *batch* pertama hingga 2083.76 ms pada *batch* terakhir. Vue juga menunjukkan peningkatan yang signifikan, mencapai 2406.34 ms di *batch* ke-10. Solid, meskipun mengalami kenaikan, tetap menunjukkan waktu eksekusi yang lebih rendah dibandingkan React dan Vue, dengan waktu eksekusi tertinggi 2227.73 ms. Svelte dan Qwik memiliki peningkatan waktu eksekusi yang lebih stabil, dengan Qwik cenderung lebih lambat di setiap *batch*, mencapai 2476.46 ms di *batch* terakhir. Secara keseluruhan, meskipun semua *framework* menunjukkan peningkatan waktu eksekusi, Solid relatif lebih efisien dibandingkan React dan Vue, sedangkan Qwik menunjukkan performa yang paling lambat dalam hal waktu eksekusi. Untuk lebih jelas dapat dilihat pada chart pada gambar 4.12.



Gambar 4.12 Chart Hasil Pengujian Eksekusi Waktu

3. Hasil pengujian ukuran bundle

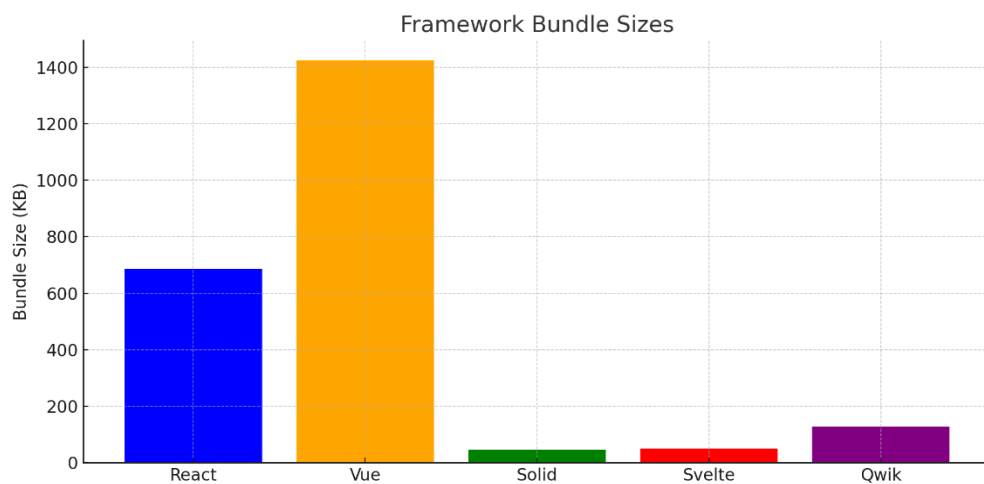
Berikut adalah hasil dari pengujian bundle dari kelima framework dapat dilihat pada tabel 4.3.

Tabel 4.13 Hasil Pengujian Ukuran Bundle

Nama Framework	Ukuran Bundle (KB)
React	685
Vue	1.424
Solid	45,9
Svelte	50,5
Qwik	127

Pada tabel 4.3 Dari perbandingan ukuran *bundle* yang tertera, Solid dan Svelte menunjukkan ukuran yang paling kecil, masing-masing dengan 45,9 KB dan 50,5 KB, yang berarti keduanya lebih ringan dibandingkan dengan *framework* lainnya. React, meskipun cukup populer, memiliki ukuran *bundle* yang lebih besar,

yakni 685 KB, yang dapat mempengaruhi waktu muat aplikasi. Vue juga memiliki ukuran yang cukup besar, yaitu 1.424 KB, menjadikannya yang paling berat di antara semua *framework* yang diuji. Sementara itu, Qwik, dengan ukuran 127 KB, berada di tengah-tengah, lebih ringan daripada Vue tetapi masih lebih berat dibandingkan Solid dan Svelte. Secara keseluruhan, Solid dan Svelte unggul dalam hal efisiensi ukuran *bundle*, sementara Vue dan React cenderung lebih berat dalam hal konsumsi ukuran file. Untuk lebih jelas dapat dilihat pada chart digambar 4.3.



Gambar 4.13 Chart Hasil Perbandingan Ukuran Bundle

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian dan pengolahan data, dapat disimpulkan

1. Framework dengan ukuran bundle kecil, seperti **Solid** dan **Svelte**, menunjukkan performa yang unggul dalam efisiensi memori dan waktu eksekusi.
2. **React** dan **Vue**, meskipun memiliki ukuran bundle yang lebih besar, tetap menjadi pilihan yang stabil untuk aplikasi berskala besar.
3. Data pengujian memberikan gambaran menyeluruh tentang kelebihan dan kelemahan setiap framework, membantu pengembang dalam memilih framework yang sesuai untuk proyek mereka.

5.2 Saran

1. Pilih Framework Berdasarkan Keutuhan Proyek
 - a. Untuk aplikasi yang membutuhkan efisiensi tinggi dalam memori dan performa cepat, **Solid** atau **Svelte** menjadi pilihan unggulan, terutama untuk proyek yang mengutamakan kecepatan dan responsivitas antarmuka.
 - b. **React** atau **Vue** cocok digunakan pada proyek berskala besar dengan kebutuhan ekosistem yang matang, karena keduanya memiliki komunitas besar dan banyak alat pendukung.

- c. **Qwik** dapat dipertimbangkan untuk aplikasi yang membutuhkan performa stabil dalam skala besar, terutama untuk memanfaatkan arsitektur modernnya yang berorientasi pada skalabilitas.

2. Pertimbangan Ukuran Bundle

Framework dengan ukuran *bundle* kecil seperti Solid dan Svelte sangat ideal untuk aplikasi yang menargetkan pengguna dengan perangkat atau koneksi internet terbatas. Ukuran yang kecil membantu mempercepat waktu muat dan memberikan pengalaman pengguna yang lebih baik.

3. Penerapan Praktis

Sebelum memilih *framework*, pengembang perlu mempertimbangkan jenis aplikasi yang akan dibangun. Misalnya, untuk aplikasi interaktif ringan, **Solid** atau **Svelte** lebih efisien. Namun, untuk aplikasi dengan banyak modul atau komponen kompleks, **React** atau **Vue** mungkin lebih sesuai.

4. Pengembangan Studi Lanjut

- a. Penelitian ini dapat diperluas dengan menguji skenario yang lebih kompleks, seperti integrasi dengan server-side rendering (SSR) atau penggunaan dalam aplikasi mobile.
- b. Selain itu, uji coba pada perangkat keras dengan spesifikasi rendah atau dalam kondisi koneksi lambat akan memberikan wawasan tambahan tentang performa masing-masing *framework* dalam situasi nyata.

- c. Melibatkan pengguna akhir untuk mengevaluasi pengalaman mereka saat menggunakan aplikasi yang dibangun dengan berbagai *framework* juga dapat memberikan sudut pandang praktis yang lebih kaya.

5. Peningkatan Performa Framework

Hasil penelitian ini dapat menjadi bahan masukan bagi pengembang *framework* untuk mengoptimalkan konsumsi memori, waktu eksekusi, dan ukuran *bundle*. Dengan terus berinovasi, *framework* dapat memenuhi kebutuhan yang semakin kompleks di industri pengembangan web.

Dengan pendekatan yang lebih terencana dan mempertimbangkan berbagai faktor seperti performa, efisiensi memori, serta dukungan komunitas, pengembang dapat memilih *framework* yang tidak hanya relevan secara teknis tetapi juga memberikan manfaat jangka panjang untuk proyek mereka.

DAFTAR PUSTAKA

- Ahadi, G. D., & Zain, N. N. L. E. (2023). Pemeriksaan Uji Kenormalan dengan Kolmogorov-Smirnov, Anderson-Darling dan Shapiro-Wilk. *EIGEN MATHEMATICS JOURNAL*, 11–19. <https://doi.org/10.29303/emj.v6i1.131>
- Anhar, A., Firdaus, M., Pangestu, D. R. A., Salpiana, S., & Putri, J. A. (2023). Analisis dan Pengembangan Quality of Experience Website E-Commerce Menggunakan GTMetrix. *Jurnal Ilmiah Informatika*, 8(1), 65–73. <https://doi.org/10.35316/jimi.v8i1.65-73>
- Aripin, S., & Somantri, S. (2021). Implementasi Progressive Web Apps (PWA) pada Repository E-Portofolio Mahasiswa. *Jurnal Eksplora Informatika*, 10(2), 148–158. <https://doi.org/10.30864/eksplora.v10i2.486>
- Axza, F., Sofi'ie, F., & Qoiriah, A. (2023). Analisis Perbandingan Framework Front-End Javascript React dan Vue Pada Pengembangan Website. *Journal of Informatics and Computer Science*, 05.
- Ayu, S., Darmawan, D., Yuniar, E., Indah, D., Mumpuni, D., Ppkia, S., Paramita, P., Laksda, J., Sucipto, A., & Timur, J. (2021). IMPLEMENTASI FRAMEWORK ANGULAR 10 UNTUK SISTEM INFORMASI AKADEMIK (SIAKAD) STMIK PPKIA PRADNYA PARAMITA MALANG. In *Seminar Nasional Teknologi Informasi dan Komunikasi (SeNTIK) STMIK Jakarta STI&K*.
- Chastro, C., Darmawan, E., Kom, S., & #2, M. T. (2020). *Perbandingan Pengembangan Front End Menggunakan Blade Template dan Vue Js* (Vol. 2).
- Devianty, D., Nur Ibrahim, R., Wahyudi, H., & Mardira Indonesia, S. (2021). PERANCANGAN SISTEM E-ARSIIP MENGGUNAKAN SUBJECT FILING SYSTEM BERBASIS FRAMEWORK CODEIGNITER (STUDI KASUS STMIK MARDIRA INDONESIA). *Jurnal Computech & Bisnis*, 15(2), 100–107.
- Dwi Bima Sakti, R., Lestanti, S., Nur Budiman, S., Balitar Jl Majapahit No, I., Sananwetan, K., & Blitar, K. (2024). PERANCANGAN DASHBOARD MONITORING PENJUALAN PADA WEBSITE PATERON.ID MENGGUNAKAN FRAMEWORK LARAVEL DAN VUE JS. In *Jurnal Mahasiswa Teknik Informatika* (Vol. 8, Issue 2).
- Endra, R. Y., Aprilinda, Y., Dharmawan, Y. Y., & Ramadhan, W. (2021). Analisis Perbandingan Bahasa Pemrograman PHP Laravel dengan PHP Native pada Pengembangan Website. *EXPERT: Jurnal Manajemen Sistem Informasi Dan Teknologi*, 11(1), 48. <https://doi.org/10.36448/expert.v11i1.2012>

- Evalina, N. (2021). Analisis Perbandingan Kualitas Jaringan 4G LTE Operator X Dan Y Di Wilayah Kampus Utama UMSU. *Teknologi Rekayasa Jaringan Telekomunikasi (TRekRiTel)*, 1(1), 13–20. <https://doi.org/10.51510/trekritel.v1i1.396>
- Ilievska, F., & Gramatikov, S. (2022). *Analysis and comparative evaluation of front-end technologies for web application development*.
- Indrasetyaningsih, A., Haryanto, I. A., & Divaio, P. A. (2024). Analisis Kruskal-Wallis untuk Mengetahui Kemampuan Literasi Siswa SMP Miftahurrohman Gresik Berdasarkan Asesmen Kompetensi Minimum. *Indonesian Journal of Multidisciplinary on Social and Technology*, 2(1), 32–36. <https://doi.org/10.31004/ijmst.v2i1.286>
- Khoirurrizal, M. F., Hidayat, C. R., & Ruuhwan, R. (2024a). ANALISIS PERBANDINGAN FRAMEWORK FRONT-END JAVASCRIPT SOLIDJS DAN VUEJS PADA PENGEMBANGAN WEBSITE INTERAKTIF. *Jurnal Informatika Dan Teknik Elektro Terapan*, 12(2). <https://doi.org/10.23960/jitet.v12i2.4106>
- Khoirurrizal, M. F., Hidayat, C. R., & Ruuhwan, R. (2024b). ANALISIS PERBANDINGAN FRAMEWORK FRONT-END JAVASCRIPT SOLIDJS DAN VUEJS PADA PENGEMBANGAN WEBSITE INTERAKTIF. *Jurnal Informatika Dan Teknik Elektro Terapan*, 12(2). <https://doi.org/10.23960/jitet.v12i2.4106>
- Kurniawan, A., & Santoso, N. (2023). *Pengembangan Sistem Ujian Kompetensi Online berbasis Web menggunakan Metode Waterfall* (Vol. 7, Issue 5). <http://j-ptiik.ub.ac.id>
- Mantik, J., Siahaan, M., Octarian Vianto, V., Vianto, V. O., Ladi, B.-S., Gajah Mada, J., Indah, T., Sekupang, K., Batam, K., & Riau, K. (2021). Comparative Analysis Study of Front-End JavaScript Frameworks Performance Using Lighthouse Tool. In *Jurnal Mantik* (Vol. 6, Issue 3). Online.
- Panjaitan, J., & Pakpahan, A. F. (2021). Perancangan Sistem E-Reporting Menggunakan ReactJS dan Firebase. *Jurnal Teknik Informatika Dan Sistem Informasi*, 7(1). <https://doi.org/10.28932/jutisi.v7i1.3098>
- Putu Mahendra Putra, G., Tenriawaru, A., Studi Ilmu Komputer, P., Matematika dan Ilmu Pengetahuan Alam, F., & Halu Oleo, U. (2023). *Rancang Bangun Virtual Assistant Chatbot Menggunakan Node.Js pada Layanan Sistem Informasi Akademik* (Vol. 1, Issue 1).
- Rojali, I., Kurniasih, D., Farizi Rachman, dan, Studi Teknik Keselamatan dan Kesehatan Kerja, P., Teknik Permesinan Kapal, J., Perkapalan Negeri Surabaya, P., Studi Magister Teknik Keselamatan dan Resiko, P., & Studi Teknik Desain dan Manufaktur, P. (2023). *7 th CONFERENCE ON SAFETY ENGINEERING AND IT'S APPLICATION Analisis Perbedaan Perilaku*

Tindakan Tidak Aman antar Shift Kerja Menggunakan Metode One Way Anova.

- Sari, A. S., & Hidayat, R. (2022). Designing website vaccine booking system using golang programming language and framework react JS. *Journal of Information System, Informatics and Computing Issue Period*, 6(1), 22–39. <https://doi.org/10.52362/jisicom.v6i1.760>
- Tabelessy, W., Batkunde, A. A., Manajemen, J., Ekonomi, F., Bisnis, D., Pattimura, U., Program,), Akuntansi, S., Kabupaten, K., & Daya, M. B. (2022). PELATIHAN PENGGUNAAN APLIKASI IBM SPSS UNTUK PENGUJIAN HIPOTESIS. *Communnity Development Journal*, 3(3), 1647–1651.
- Wenqing Xu. (2021). Benchmark Comparison of JavaScript Frameworks React, Vue, Angular and Svelte. *A Research Paper Submitted to the University of Dublin, in Partial Fulfilment of the Requirements for the Degree of Master of Science Interactive Digital Media* .