

Analisis Tingkat Efisiensi Pengolahan Data Menggunakan Teknik Native dan Redux Toolkit pada Aplikasi Berbasis JavaScript

Rofiuddin^{1)*}, Trio Agung Purwanto²⁾

¹⁾²⁾Universitas Islam Madura, Indonesia

¹⁾rofiuddin@uim.ac.id, ²⁾trioagung@uim.ac.id



*Rofiuddin

Histori Artikel:

Submit: 2025-03-15

Diterima: 2025-03-28

Dipublikasikan: 2025-04-13

Kata Kunci:

Efisiensi Data, JavaScript, Redux Toolkit, Teknik Native

ABSTRAK

Pengolahan data yang efisien berperan penting dalam pengembangan aplikasi berbasis JavaScript. Teknik native menggunakan API bawaan JavaScript, sementara Redux Toolkit menawarkan solusi lebih terstruktur dengan fitur seperti slice dan middleware. Penelitian ini bertujuan membandingkan kedua teknik berdasarkan waktu eksekusi query API dan kompleksitas implementasi kode menggunakan dataset desa di Indonesia yang terdiri dari 83.809 entri. Hasil penelitian menunjukkan bahwa Redux Toolkit lebih efisien dalam menangani data kompleks dengan waktu eksekusi 5,50 milidetik, dibandingkan dengan teknik native yang memerlukan 189,80 milidetik, sehingga Redux Toolkit 34,5 kali lebih cepat. Selain itu, Redux Toolkit memberikan konsistensi state yang lebih baik dan mengurangi potensi bug yang sering muncul dalam pengelolaan state secara manual. Meskipun implementasinya lebih kompleks karena memerlukan pemahaman konsep seperti reducers, actions, dan store, hasil penelitian ini menunjukkan bahwa Redux Toolkit lebih unggul dalam pengolahan data dengan efisiensi waktu yang lebih baik dalam proses query API. Keunggulan Redux Toolkit juga terlihat dalam skalabilitas aplikasi, di mana pengelolaan state yang terpusat memudahkan pengembang dalam mengintegrasikan fitur baru. Temuan ini memberikan wawasan bagi pengembang dalam memilih teknik pengolahan data yang sesuai untuk meningkatkan performa dan kualitas aplikasi berbasis JavaScript, terutama pada proyek yang memerlukan penanganan data dalam skala besar. Dengan demikian, Redux Toolkit dapat menjadi pilihan utama untuk aplikasi yang memprioritaskan kecepatan dan konsistensi dalam pengolahan data.

Jurnal Pendidikan Sains dan Komputer is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0).

LATAR BELAKANG

Pengolahan data merupakan aspek yang sangat penting dalam pengembangan aplikasi berbasis JavaScript, terutama dengan pesatnya perkembangan teknologi web dan aplikasi mobile (Suherman et al., 2022). Pengolahan data yang efektif dapat meningkatkan pengalaman pengguna dan efisiensi aplikasi secara keseluruhan. Menurut Statista (2023), lebih dari 70% pengembang perangkat lunak menggunakan JavaScript sebagai bahasa pemrograman utama mereka, menunjukkan bahwa pemahaman yang mendalam tentang teknik pengolahan data yang efisien (Bismoputro et al., 2024).

Dua pendekatan utama dalam pengelolaan data pada aplikasi berbasis JavaScript adalah teknik native dan Redux Toolkit. Teknik native melibatkan penggunaan API bawaan JavaScript, seperti array dan objek, untuk mengelola data, yang telah lama menjadi pilihan bagi banyak pengembang. Pendekatan ini dikenal karena kesederhanaan dan kemudahan implementasinya. Namun, seiring dengan meningkatnya kompleksitas aplikasi, teknik native menghadapi tantangan dalam pengelolaan state dan konsistensi data (Ragkhitwetsagul et al., 2024).

Redux Toolkit, sebagai evolusi dari Redux, menawarkan solusi yang lebih terstruktur dan efisien dalam mengelola state. Redux Toolkit memungkinkan pengembang untuk menangani data dengan cara yang lebih

terorganisir dan dapat diprediksi (Patil, 2022).

Studi oleh Pradana (2024) menunjukkan bahwa penggunaan Redux Toolkit dapat memungkinkan developer untuk membangun aplikasi dengan state management yang handal dan efisien, meningkatkan produktivitas dan meminimalisir risiko kesalahan dalam pengelolaan state, sehingga dapat mengurangi waktu pengembangan hingga 30% dibandingkan dengan teknik native, terutama dalam aplikasi yang memiliki interaksi data yang kompleks (Pradana, 2024). Temuan ini mengindikasikan bahwa pemilihan teknik pengolahan data yang tepat dapat memiliki dampak signifikan terhadap produktivitas pengembang dan kualitas aplikasi yang dihasilkan. Oleh karena itu, penting untuk menganalisis dan membandingkan efisiensi kedua teknik ini dalam konteks pengolahan data pada aplikasi berbasis JavaScript.

Dari studi diatas menunjukkan bahwa teknik native dapat memberikan kinerja yang baik dalam aplikasi sederhana. Namun, teknik sering kali menghadapi kesulitan saat menangani state yang kompleks. Di sisi lain, Redux Toolkit menawarkan fitur-fitur seperti slice dan middleware yang dapat meningkatkan efisiensi pengolahan data, meskipun pengembang yang belum familiar dengan konsep ini mungkin membutuhkan waktu lebih lama untuk mempelajarinya (Pradana, 2024). Penelitian ini bertujuan untuk menggali perbedaan antara kedua teknik ini secara lebih mendalam dan memberikan gambaran yang jelas mengenai kelebihan dan kekurangan masing-masing teknik.

Tujuan dari penelitian ini adalah untuk menganalisis dan membandingkan tingkat efisiensi pengolahan data menggunakan teknik native dan Redux Toolkit dalam aplikasi berbasis JavaScript. Dengan analisis ini, diharapkan dapat ditemukan informasi yang berguna bagi pengembang dalam memilih teknik yang paling sesuai untuk aplikasi mereka.

Secara spesifik, penelitian ini bertujuan untuk mengidentifikasi dan membandingkan metrik efisiensi, seperti waktu respon antara teknik native dan Redux Toolkit; menganalisis dampak masing-masing teknik terhadap performa aplikasi dalam konteks pengolahan data yang kompleks; serta memberikan rekomendasi kepada pengembang aplikasi mengenai teknik pengolahan data yang paling efisien berdasarkan temuan penelitian ini.

Dalam pendahuluan, peneliti diharapkan mampu menjelaskan fenomena yang ada atau latar belakang informasi seperti pekerjaan sebelumnya, hipotesis, masalah yang akan dibahas. Ini diikuti dengan pernyataan tujuan dari masalah atau masalah penelitian dan/atau serangkaian pertanyaan yang Anda coba jawab dalam penelitian Anda.

STUDI LITERATUR

a) Pengolahan Data dalam Aplikasi JavaScript

Pengolahan data adalah aspek yang sangat penting dalam pengembangan aplikasi berbasis JavaScript, terutama dalam menciptakan pengalaman pengguna yang efisien dan responsif (Suherman et al., 2022). Pengolahan data dalam aplikasi web melibatkan berbagai teknik untuk mengambil, menyimpan, dan memanipulasi data dari berbagai sumber, seperti database atau server, untuk disajikan kepada pengguna (Kasmawaru et al., 2021). Teknik pengolahan data yang efisien dapat meningkatkan kinerja aplikasi dan mempermudah pemeliharaannya.

Salah satu pendekatan pengolahan data yang umum digunakan dalam aplikasi JavaScript adalah menggunakan teknik native. Dalam pendekatan ini, pengembang menggunakan API bawaan JavaScript seperti array, objek, dan fungsi untuk mengelola data dan mengupdate state aplikasi. Meskipun pendekatan ini lebih mudah dipahami dan diimplementasikan, masalah dapat muncul ketika aplikasi berkembang menjadi lebih kompleks dengan banyak komponen yang saling berinteraksi (Ardiansyah & Fatwanto, 2022). Hal ini membuat pengelolaan state menjadi lebih sulit, terutama dalam hal konsistensi data dan pemeliharaan. Aplikasi yang menggunakan teknik native cenderung menghadapi tantangan dalam menjaga konsistensi state dan pengelolaan data, terutama pada aplikasi yang melibatkan banyak komponen dan membutuhkan pembaruan state secara dinamis (Fredrick, 2023). Oleh karena itu, untuk menangani masalah ini, banyak pengembang beralih menggunakan pustaka manajemen state yang lebih terstruktur, seperti Redux dan Redux Toolkit.

b) Teknik Native

Teknik native melibatkan penggunaan Fetch API bawaan JavaScript untuk mengelola data (Turcotte et al., 2022). Pendekatan ini sederhana dan ideal untuk aplikasi kecil atau dengan kebutuhan pengelolaan data yang minimal. Namun, seiring meningkatnya kompleksitas aplikasi, pengelolaan state menjadi lebih sulit. Aplikasi besar dengan banyak komponen interaktif sering kali menghadapi masalah konsistensi data dan pembaruan state secara dinamis (Sewell, 2023).

c) Redux Toolkit

Redux Toolkit adalah evolusi dari pustaka Redux, yang menawarkan cara yang lebih efisien dan terstruktur dalam mengelola state aplikasi. Redux Toolkit menyederhanakan pengelolaan state dan meminimalkan jumlah kode yang diperlukan, yang seringkali menjadi masalah dalam penggunaan Redux tradisional (Patil, 2022). Redux Toolkit menyertakan beberapa fitur penting, seperti slice, createAsyncThunk, dan middleware, yang memungkinkan pengembang untuk menangani state dan efek samping secara lebih mudah dan terorganisir (Pradana, 2024).

Salah satu fitur unggulan Redux Toolkit adalah penggunaan slice, yang memungkinkan pengembang untuk mendefinisikan state dan reducer dalam satu tempat, menjadikannya lebih mudah dipahami dan dikelola. Selain itu, fitur createAsyncThunk memungkinkan pengembang untuk menangani operasi asynchronous secara lebih efisien, yang sangat berguna ketika berurusan dengan pengolahan data yang melibatkan server atau API eksternal. Fitur-fitur ini secara signifikan mengurangi kode boilerplate yang biasa ditemukan dalam implementasi Redux tradisional (Ihya Ullumuddin et al., 2024).

Penggunaan Redux Toolkit dapat mempercepat proses pengembangan aplikasi, khususnya dalam aplikasi berskala besar yang melibatkan banyak tim pengembang (Veeranjaneyulu Veeri, 2024). Ini menunjukkan bahwa pengelolaan state yang efisien dengan Redux Toolkit tidak hanya meningkatkan kualitas aplikasi tetapi juga mempercepat siklus pengembangan. Aplikasi yang menggunakan Redux Toolkit dapat mengatasi masalah waktu respons yang lebih baik dan penggunaan memori yang lebih efisien dibandingkan dengan aplikasi yang menggunakan teknik native. Hal ini disebabkan oleh pengelolaan state yang lebih terstruktur dan penggunaan fitur-fitur canggih seperti slice dan middleware. Kemudahan dalam menggunakan Redux Toolkit dan struktur kode yang lebih jelas meningkatkan kepuasan yang lebih tinggi dalam hal pengelolaan kode dan pemeliharaan aplikasi, sehingga pengelolaan data lebih terorganisir, meningkatkan kecepatan respons aplikasi dan mengurangi kemungkinan terjadinya bug atau masalah yang berkaitan dengan pengelolaan state, yang menunjukkan bahwa teknik pengolahan data yang tepat dapat membawa dampak positif terhadap performa aplikasi secara keseluruhan. Hal ini juga memungkinkan pengembang untuk fokus lebih pada logika bisnis aplikasi ketimbang menangani masalah pengelolaan state secara manual.

Berdasarkan studi literatur yang disajikan, terdapat beberapa gap research yang dapat diidentifikasi terkait pengolahan data dalam aplikasi JavaScript, khususnya dalam perbandingan antara teknik native dan Redux Toolkit. Berikut adalah analisis gap research yang dapat dikembangkan lebih lanjut:

1) Perbandingan Kuantitatif yang Terbatas

Studi literatur yang ada lebih banyak membahas keunggulan Redux Toolkit secara konseptual, seperti pengurangan kode boilerplate dan kemudahan pengelolaan state. Namun, belum ada analisis kuantitatif yang mendalam mengenai perbandingan performa antara teknik native dan Redux Toolkit, terutama dalam hal waktu eksekusi, penggunaan memori, dan skalabilitas pada aplikasi berskala besar. Penelitian lebih lanjut diperlukan untuk mengukur secara empiris seberapa signifikan perbedaan performa antara kedua teknik tersebut.

2) Evaluasi Kompleksitas Implementasi



Meskipun Redux Toolkit diklaim lebih mudah diimplementasikan dibandingkan Redux tradisional, belum ada studi yang secara khusus menganalisis kompleksitas implementasi Redux Toolkit dibandingkan dengan teknik native. Penelitian lebih lanjut dapat mengukur tingkat kesulitan implementasi, waktu pengembangan, dan kurva pembelajaran yang diperlukan oleh pengembang untuk mengadopsi Redux Toolkit.

3) Dampak pada Pengalaman Pengembang (Developer Experience)

Studi literatur menyebutkan bahwa Redux Toolkit meningkatkan kepuasan pengembang dalam hal pengelolaan kode dan pemeliharaan aplikasi. Namun, belum ada penelitian yang secara khusus mengukur dampak Redux Toolkit terhadap pengalaman pengembang (developer experience), seperti produktivitas, kemudahan debugging, dan kepuasan secara keseluruhan. Penelitian lebih lanjut dapat menggunakan metode survei atau wawancara untuk mengumpulkan data kualitatif dari pengembang.

4) Analisis Skalabilitas pada Aplikasi Berskala Besar

Studi literatur menyoroti keunggulan Redux Toolkit dalam aplikasi berskala besar, tetapi belum ada analisis mendalam tentang bagaimana Redux Toolkit menangani skalabilitas pada aplikasi dengan jumlah komponen dan interaksi yang sangat kompleks. Penelitian lebih lanjut dapat menguji performa Redux Toolkit dalam skenario aplikasi dengan ribuan komponen dan pembaruan state yang sangat dinamis.

5) Dampak pada Pengalaman Pengguna (User Experience)

Meskipun Redux Toolkit diklaim meningkatkan kecepatan respons aplikasi, belum ada penelitian yang secara khusus mengukur dampaknya terhadap pengalaman pengguna (user experience), seperti waktu loading, responsivitas antarmuka, dan stabilitas aplikasi. Penelitian lebih lanjut dapat menguji dampak Redux Toolkit terhadap metrik UX seperti First Contentful Paint (FCP) dan Time to Interactive (TTI).

Dengan mengidentifikasi gap research ini, penelitian lebih lanjut dapat memberikan kontribusi yang signifikan dalam memahami keunggulan dan keterbatasan Redux Toolkit serta teknik native, sehingga membantu pengembang dalam memilih solusi pengolahan data yang optimal untuk aplikasi berbasis JavaScript.

METODE

a) Desain Penelitian

Penelitian ini menggunakan desain penelitian eksperimen komparatif yang bertujuan untuk menganalisis efisiensi waktu dalam proses query API menggunakan dua teknik pengolahan data dalam aplikasi berbasis JavaScript, yaitu Teknik Native dan Redux Toolkit (Sahara Munte et al., 2023). Penelitian ini dilakukan dalam dua tahap utama, sebagai berikut:

1) Implementasi dan Pengujian Skenario Kode

Membuat dua skenario pengujian: Satu menggunakan JavaScript Teknik Native dan satu lagi menggunakan Redux Toolkit untuk mengambil data melalui API. Menerapkan kode di lingkungan pengujian menggunakan Visual Studio Code dan menjalankan pengujian di browser (menggunakan Chrome DevTools).

2) Pengujian dan Analisis Kinerja

Mengukur waktu eksekusi query API menggunakan fungsi `measureQueryTime`. Membandingkan hasil antara kedua teknik berdasarkan kecepatan eksekusi query dan kompleksitas kode program.

b) Data yang Digunakan

Dalam penelitian ini, data yang digunakan adalah data desa seluruh Indonesia dengan total 83.809 data. Dataset ini memiliki beberapa atribut, diantaranya; 1) kode kecamatan, 2) kode desa, 3) meta data, 4) nama desa.

Data tersebut disimpan dalam database lokal, menggunakan PostgreSQL sebagai sistem pengelolaan basis data dan API digunakan untuk mengakses data dari database lokal.

c) Teknik dan Instrumen Pengumpulan Data

Data dikumpulkan melalui dua pendekatan utama:

- 1) Pengujian Kode: Dilakukan dengan menjalankan kode di Visual Studio Code dan memantau hasilnya melalui console log dan Chrome DevTools. Pengujian dilakukan untuk mengukur waktu eksekusi query API.
- 2) Analisis Kompleksitas Kode: Membandingkan tingkat kompleksitas kode antara JavaScript Teknik Native dan Redux Toolkit. Menilai kemudahan implementasi masing-masing teknik berdasarkan struktur kode.

d) Alat Analisis Data

Data hasil pengujian dianalisis menggunakan metode statistik deskriptif (Retnaningsih, 2024), yaitu analisis waktu eksekusi query API untuk kedua teknik. Perbandingan tingkat kompleksitas kode berdasarkan jumlah baris dan struktur kode. Perbedaan efisiensi antara JavaScript Teknik Native dan Redux Toolkit dianalisis dengan melihat langsung hasil pengujian di console VS Code dan Chrome DevTools.

e) Model Penelitian

Model penelitian yang digunakan adalah eksperimen komparatif dimana dua teknik pengolahan data dibandingkan berdasarkan efisiensi waktu query API, dan kompleksitas implementasi kode.

HASIL

a) Hasil Pengujian

Kode pengujian native menggunakan JavaScript Teknik Native dengan mengakses data melalui API. Fungsi *measureQueryTime* merupakan fungsi yang dibangun untuk menghitung lama proses query ke dalam database, fungsi *fetchData* digunakan untuk melakukan proses query ke dalam database menggunakan metode *fetch* bawaan JavaScript, selanjutnya, kedua fungsi tersebut dijalankan untuk mendapatkan hasil.



```
Console ×
Query executed in 189.80 ms
▼ (3) {success: true, message: "data desa"...}
  success: true
  message: "data desa"
  data: (83809) [{...}, {...}, {...}, {...}, {...}, ...]
  [[Prototype]]: {}
```

Gambar 1. Hasil Pengujian Native

Hasil uji menunjukkan bahwa query dengan Teknik Native memperoleh data dalam 189.80 mili detik untuk jumlah data sebanyak 83.809.

Perbedaan mendasar dalam kode pengujian menggunakan Redux Toolkit terletak pada fungsi query. *useGetVillagesQuery* digunakan untuk mengakses query data via fungsi Redux Toolkit. Hasil uji dengan Redux Toolkit menunjukkan hasil jauh lebih cepat, yaitu 5.50 mili detik untuk jumlah data yang sama sebanyak 83.809.



Gambar 2. Hasil Pengujian Redux Toolkit

PEMBAHASAN

a) Kode Pengujian

Kode pengujian Teknik Native menggunakan metode *Fetch* untuk mengakses data melalui API (Aditya Kappagantula, 2024). Fungsi *measureQueryTime* digunakan untuk menghitung waktu eksekusi query API. Fungsi *fetchData* digunakan untuk mengambil data dari API dengan metode *fetch* bawaan JavaScript.

```

//index.html < script.js >
1
2 function measureQueryTime(queryFunction) {
3   return async function (...args) {
4     const startTime = performance.now(); // Start time
5     const result = await queryFunction(...args);
6     const endTime = performance.now(); // End time
7
8     console.log('Query executed in: ' + (endTime - startTime).toFixed(2)) + ' ms';
9     return result;
10  };
11 }
12
13 const fetchData = async () => {
14   const respos = await fetch('http://127.0.0.1:8000/api/desa');
15   const dataDesa = await respos.json();
16   return dataDesa;
17 }
18
19 const timedFetchData = measureQueryTime(fetchData);
20
21 timedFetchData().then(data => console.log(data));

```

Gambar 3. Kode Pengujian Native

Kode pengujian dengan Redux Toolkit, perbedaan utama dalam kode pengujian Redux Toolkit adalah penggunaan *useGetVillagesQuery* untuk melakukan *query* data melalui Redux Toolkit. Fungsi ini memungkinkan pengambilan data dari API dengan cara yang lebih terstruktur, serta struktur kode lebih kompleks dibandingkan metode Native.

```

1 import {useGetVillagesQuery} from '@store/api/zona/village/villageApiSlice';
2 //index.html < vite-redux
3
4 function measureQueryTime(queryFunction) {
5   return async function (...args) {
6     const startTime = performance.now(); // Start time
7     const result = await queryFunction(...args);
8     const endTime = performance.now(); // End time
9
10    console.log('Query executed in: ' + (endTime - startTime).toFixed(2)) + ' ms';
11    return result;
12  };
13 }
14
15 const fetchData = async () => {
16   const {data} = useGetVillagesQuery();
17   return {data};
18 }
19
20 const timedFetchData = measureQueryTime(fetchData);
21
22 timedFetchData().then(data => console.log(data));

```

Gambar 4. Kode Pengujian Redux Toolkit

Berdasarkan hasil pengujian, Redux Toolkit dapat menjadi pilihan yang lebih efisien untuk query data berbasis JavaScript. Hasil pengujian menunjukkan bahwa Redux Toolkit mampu menyelesaikan query data dalam 5.50 milidetik sedangkan Teknik Native membutuhkan waktu 189.80 milidetik. Dengan demikian, Redux Toolkit memiliki waktu eksekusi 34,5 kali lebih cepat dibandingkan JavaScript Teknik Native. Redux Toolkit hanya dapat digunakan pada aplikasi front-end berbasis JavaScript, struktur kode Redux Toolkit lebih kompleks dibandingkan Teknik Native, Namun, efisiensi waktu yang diberikan sangat signifikan untuk aplikasi yang membutuhkan query data cepat melalui API.

KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa penggunaan Redux Toolkit dalam pengolahan data berbasis JavaScript lebih efisien dibandingkan dengan metode JavaScript Teknik Native dalam hal kecepatan eksekusi query API. Hasil pengujian menunjukkan bahwa Redux Toolkit mampu menyelesaikan query data dalam 5,50 milidetik, sementara Teknik Native membutuhkan waktu 189,80 milidetik untuk jumlah data yang sama, yaitu 83.809 data desa. Dengan demikian, Redux Toolkit terbukti memiliki efisiensi waktu yang jauh lebih tinggi, yaitu sekitar 34,5 kali lebih cepat dibandingkan metode Native.

Meskipun demikian, penerapan Redux Toolkit memiliki tingkat kompleksitas kode yang lebih tinggi dibandingkan Teknik Native, sehingga pemilihan teknik perlu disesuaikan dengan kebutuhan dan skala proyek yang sedang dikembangkan. Penelitian ini hanya berfokus pada aspek efisiensi waktu tanpa mempertimbangkan penggunaan sumber daya seperti memori yang dapat menjadi faktor penting dalam implementasi skala besar. Sebagai rekomendasi, penelitian selanjutnya dapat memperluas cakupan analisis dengan menguji performa kedua metode dalam skenario yang lebih kompleks, seperti aplikasi dengan jumlah pengguna yang lebih besar atau penerapan dalam arsitektur berbasis cloud. Selain itu, penelitian lebih lanjut juga dapat mengeksplorasi teknik optimasi tambahan untuk meningkatkan efisiensi dalam pengolahan data menggunakan Redux Toolkit maupun pendekatan lain yang lebih ringan dan fleksibel.

UCAPAN TERIMA KASIH (PILIHAN)

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan dalam pelaksanaan penelitian ini. Terima kasih kepada rekan-rekan yang telah membantu dalam proses pengujian dan analisis data, serta semua pihak yang telah berkontribusi secara langsung maupun tidak langsung dalam penelitian ini. Semoga hasil penelitian ini dapat memberikan manfaat bagi pengembangan teknologi di bidang pengolahan data berbasis JavaScript dan menjadi referensi bagi penelitian selanjutnya.

REFERENSI

- Fredrick, A. (2023, June 16). (14) *A Comparison of Fetch and Axios: Advantages, Disadvantages, Use Cases, and Optimization* | *LinkedIn*. <https://www.linkedin.com/pulse/comparison-fetch-axios-advantages-disadvantages-use-cases-fredrick/>
- Aditya Kappagantula. (2024). The Evolution and Complexities of Asynchronous Programming in JavaScript: A Comprehensive Analysis. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 10(6), 1768–1778. <https://doi.org/10.32628/CSEIT241061221>
- Ardiansyah, H., & Fatwanto, A. (2022). Comparison of Memory usage between REST API in Javascript and Golang. *MATRIK: Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*, 22(1), 229–240. <https://doi.org/10.30812/matrik.v22i1.1325>
- Bismoputro, I., Huda, F. Al, & Brata, A. H. (2024). PENGEMBANGAN SINGLE PAGE APPLICATION BERBASIS REACTJS UNTUK USAHA PERCETAKAN ONLINE (STUDI KASUS: GLOBAL GRAFIKA). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 1(1), 1–10. <http://j-ptiik.ub.ac.id>
- Ihya Ullummuddin, D., Suharso, A., & Garno, G. (2024). ANALISIS PERBANDINGAN STATE MANAGEMENT ANTARA RECOIL DAN REDUX DALAM REACTJS. *JATI (Jurnal Mahasiswa Teknik Informatika)*, 8(3),

4241–4246. <https://doi.org/10.36040/jati.v8i3.9869>

- Kasmawaru, Herlinda, & Ahyuna. (2021). Sistem Pengolahan Data Penduduk Pada Kantor Desa Bowong Cindea Kecamatan Bungoro Kabupaten Pangkajene Dan Kepulauan Berbasis Web. *PROSIDING SEMINAR ILMIAH SISTEM INFORMASI DAN TEKNOLOGI INFORMASI*, 10(1), 88–97.
- Patil, K. (2022). Redux State Management System - A Comprehensive Review. *International Journal of Trend in Scientific Research and Development (IJTSRD)*, 6(7), 1021–1027. www.ijtsrd.com
- Pradana, A. A. (2024). *STUDI INDEPENDEN: RANCANG BANGUN WEBSITE PEMESANAN TIKET PESAWAT MENGGUNAKAN FRAMEWORK REACT LAPORAN KERJA PRAKTIK*.
- Ragkhitwetsagul, C., Kongwongsupak, K., Maneesawas, T., Puttiwarodom, N., Rojpaisarnkit, R., Choetkiertikul, M., Kula, R. G., & Sunetnanta, T. (2024). *jscefr: A Framework to Evaluate the Code Proficiency for JavaScript*.
- Retnaningsih, E. (2024). Capaian Kinerja Laboratorium Sebagai Baseline Penyusunan Renstra Laboratorium dengan Metode Statistik Deskriptif. *MARAS: Jurnal Penelitian Multidisiplin*, 2(3), 1512–1521. <https://doi.org/10.60126/maras.v2i3.426>
- Sahara Munte, R., Risnita, R., Jailani, M. S., & Siregar, I. (2023). *Jenis Penelitian Eksperimen dan Noneksperimen (Design Klausal Komparatif dan Design Korelasional)*. <https://doi.org/10.31004/jptam.v7i3.11128>
- Sewell, S. (2023, January 16). *Safe Data Fetching in Modern JavaScript*. [Www.Builder.Io. https://www.builder.io/blog/safe-data-fetching](https://www.builder.io/blog/safe-data-fetching)
- Suherman, Y., Azandra, E. N., Fajri, A. F., Hamdi, K., & Zevimarino, A. (2022). Mobile Application Sistem Monitoring Kegiatan Dan Keuangan Masjid. *BRAHMANA: Jurnal Penerapan Kecerdasan Buatan*, 4(1), 61–71.
- Turcotte, A., Arteca, E., Mishra, A., Alimadadi, S., & Tip, F. (2022). Stubbifier: debloating dynamic server-side JavaScript applications. *Empirical Software Engineering*, 27(7), 161. <https://doi.org/10.1007/s10664-022-10195-6>
- Veeranjaneyulu Veeri. (2024). State Management in React: Redux vs Zustand - A Comprehensive Guide. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 10(6), 211–218. <https://doi.org/10.32628/CSEIT24106172>