ANALISIS PERBANDINGAN STATE MANAGEMENT ANTARA RECOIL DAN REDUX DALAM REACTJS

Damareindra Ihya Ullummuddien, Aries Suharso, Garno
Program Studi Informatika, Fakultas Ilmu Komputer, Universitas Singaperbangsa Karawang
Jl. HS. Ronggo Waluyo, Telukjambe Timur, Karawang, Indonesia
2010631170061@student.unsika.ac.id

ABSTRAK

Penelitian ini menganalisis kinerja dua alat manajemen state populer, Recoil dan Redux, yang memiliki pendekatan sistem berbeda. React, sebagai *framework* website, didukung oleh pengelolaan state canggih seperti Recoil dan Redux. Dalam periode November 2022 – November 2023, Redux mendapat 444,581,918 unduhan, sedangkan Recoil 22,958,079 unduhan. Berdasarkan survei State of React 2023 yang melibatkan 1502 responden, debugging dan performa menjadi masalah utama dengan 829 dan 506 pengguna melaporkan kesulitan ini, yang mana penggunaan *state management* dapat mengatasi masalah *debugging* dan performa pada sebuah proyek react. Penelitian ini menggunakan metode performance testing, meliputi studi kasus, analisis kebutuhan tool, perancangan skenario pengujian, pelaksanaan, dan analisis. Recoil menunjukkan processing speed dan memory usage yang lebih rendah dibandingkan Redux, dengan nilai $p<0.05$, menunjukkan perbedaan signifikan. Perbandingan dilakukan menggunakan metode Mann-Whitney pada parameter 100, 500, dan 1000 film, di mana Recoil lebih cepat dan efisien dalam penggunaan memori. Kesimpulannya, Recoil lebih baik dibandingkan Redux dalam performa namun Redux lebih kecil dalam segi ukuran proyek.

Kata kunci : Analisis Perbandingan, React, Recoil, Redux, Website.

1. PENDAHULUAN

ReactJS adalah sebuah *library* JavaScript yang dibuat untuk tujuan membangun komponen UI yang dapat digunakan kembali. Komponen-komponen ini menerima beberapa masukan yang disebut *props*, yang akan menentukan bagaimana komponen tersebut direnderer. Dalam React, sebuah komponen dapat disematkan di dalam komponen lain hingga membentuk jaringan lengkap, yang membangun *tree component* yang disebut *virtual* DOM[1].

Untuk menerima data di komponen level rendah, data harus ditransfer sebagai props melalui banyak komponen level menengah secara tidak perlu, yang mengakibatkan penulisan banyak kode tambahan dan memberikan properti yang tidak digunakan pada komponen level menengah. Untuk mengatasi masalah ini, ada banyak pustaka manajemen state, Recoil dan Redux, yang menyediakan solusi state global yang dapat diakses oleh semua komponen di virtual DOM. Redux adalah sebuah library state management yang mengikuti pola arus data satu arah (*unidirectional data flow*), yang memungkinkan penyimpanan state aplikasi secara terpusat (*centralized*). Dalam arsitektur Redux, seluruh *state* aplikasi disimpan dalam objek tunggal yang disebut "*store*."

Konsep ini mendukung pengelolaan state yang efektif dan menjamin alur data yang dapat diprediksi dan konsisten dalam aplikasi. Di sisi lain, Recoil mengadopsi pendekatan yang berbeda dengan menyimpan state secara terdesentralisasi (decentralized). Recoil dirancang oleh Facebook sebagai perpustakaan manajemen state khusus untuk aplikasi React. Dalam struktur Recoil, keadaan aplikasi dapat disimpan dalam berbagai atom (unit state) yang terpisah, memberikan fleksibilitas dalam mengelola state aplikasi tanpa ketergantungan yang tinggi pada satu pusat penyimpanan. Ada perbedaan dalam pendekatan konsep pada kedua manajemen state tersebut, sehingga mungkin ada perbedaan dalam kinerja atau penggunaan sumber daya yang perlu dianalisis. Namun, belum ada penelitian yang dilakukan untuk membandingkan kinerja kedua *state management* ini dalam pengembangan aplikasi menggunakan ReactJS. Terlebih Recoil dan Redux merupakan library yang populer bagi para pengembang ReactJS, Dikutip dari website npm-stat.com 26 November 2023, sebagaimana website tersebut dapat melakukan pelacakan pada jumlah unduhan paket library npm, jumlah unduhan library state management Redux dan Recoil dalam kurun waktu November 2022 – November 2023 Redux mendapat total unduhan sebanyak 444,581,918 sedangkan Recoil mendapat total unduhan sebanyak 22,958,079.

Pada penelitian ini akan menganalisis penggunaan Recoil dan Redux sebagai manajemen keadaan arsip. dalam website ReactJS dengan standar pengukuran yang didasarkan pada parameter ukuran aplikasi dan performa (proses kecepatan dan penggunaan memori). Penelitian ini diharapkan dapat membantu pengembang ReactJS dalam memilih manajemen status terbaik untuk proyek mereka, sehingga mereka dapat meningkatkan kualitas dan performa website yang mereka buat.

2. TINJAUAN PUSTAKA

2.1. ReactJS

ReactJS adalah kerangka kerja sumber terbuka yang memanfaatkan library JavaScript untuk membuat antarmuka pengguna, dan biasanya digunakan dalam

pengembangan aplikasi satu halaman (single-page) dan aplikasi mobile [2].

Library ini berbasis komponen dan mengadopsi paradigma pemrograman deklaratif, yang memungkinkan penggunaan tampilan deklaratif untuk menciptakan antarmuka pengguna interaktif yang kompleks. React JS banyak digunakan untuk mengembangkan aplikasi web karena arsitektur berbasis komponennya, yang memungkinkan penggunaan kembali dan pemeliharaan kode dengan mudah. Ia dikenal karena optimasi kinerjanya dan DOM virtual, yang membantu dalam pembaruan antarmuka pengguna dengan efisien. React JS juga populer karena dukungan komunitas yang kuat, dokumentasi yang ekstensif, dan berbagai pustaka dan alat pihak ketiga yang meningkatkan kemampuannya.

2.2. Javascript

Javascript dikembangkan di Netscape pada tahun 1995 dan sekarang sudah menjadi salah satu bahasa pemrograman yang banyak dipakai terutama untuk melakukan pengembangan website. Hampir setiap browser web pada desktop, tablet, dan smartphone dilengkapi dengan penterjemah JavaScript, sehingga jumlah perangkat yang dapat menjalankan JavaScript sangat besar. JavaScript adalah bahasa yang dinamis, yang berarti bahwa variabel tidak ditentukan oleh tipe, melainkan tipe ditentukan oleh nilai variabel yang dapat berubah selama program berjalan. JavaScript, sebuah bahasa script dinamis, memungkinkan Anda membuat blok kode interaktif pada halaman HTML statis. JavaScript ada di hampir semua halaman web [3].

2.3. State Management

State management merupakan suatu metodologi yang digunakan untuk mengatur alur state dan komunikasi antara komponen dalam pengembangan perangkat lunak. Meskipun konsep dasarnya relatif mudah dipahami, pengelolaan state dapat menjadi rumit dan kacau ketika aplikasi mengalami perkembangan yang signifikan. Sebagai contoh, ketika sebuah komponen yang baru dibuat memerlukan data dari state komponen pada tingkat yang sama, state tersebut harus ditingkatkan (lifted) ke komponen induk terdekat yang umum untuk kedua komponen anak yang memerlukannya, dan kemudian disampaikan ke bawah [4].

Potensi munculnya state yang redundan atau duplikat menjadi penyebab umum terjadinya bug. Oleh karena itu, menjadi sangat penting untuk memanfaatkan alat-alat manajemen state yang telah tersedia, seperti Redux, Recoil, dan berbagai library lainnya. Penerapan library ini dapat membantu mengatasi kompleksitas manajemen state, meningkatkan keterbacaan kode, dan mengurangi risiko kesalahan pada aplikasi yang sedang dikembangkan. Pendekatan ini tidak hanya memfasilitasi pengembangan yang lebih efisien, tetapi juga meningkatkan kehandalan serta kualitas keseluruhan dari suatu perangkat lunak berbasis React.

2.4. Redux

Redux adalah state container yang mudah digunakan untuk aplikasi JavaScript, yang merupakan library untuk membantuk melakukan state managing, yang dapat dijalankan dilingkungan yang berbeda (klien, server, dan native) [5]. Redux adalah library manajemen state JavaScript yang populer yang terinspirasi oleh arsitektur Flux. Redux menawarkan wadah state yang dapat diprediksi yang dapat mengelola dan memperbarui state aplikasi. Redux adalah library manajemen state JavaScript yang populer yang terinspirasi oleh arsitektur Flux. Redux menawarkan wadah state yang dapat diprediksi yang dapat mengelola dan memperbarui state aplikasi [6].

2.5. Recoil

Recoil dikembangkan oleh Meta, sebelumnya dikenal sebagai Facebook, pada tahun 2020 sebagai perpustakaan manajemen state sendiri. Dibandingkan dengan alat lain Recoil dibangun dengan baik berdasarkan hooks dan memperkenalkan dua konsep baru atom dan selector yang digunakan untuk menyimpan state dan menghitung data turunan, secara berturut-turut [7].

2.6. Speed Processing

Mengenai kinerja teknologi state management pada reactjs, kecepatan pemrosesan peristiwa adalah kriteria penting [1]. Penulis menggunakan browser Google Chrome sebagai alat untuk melakukan pengetesan pada tahap ini, dengan perhitungan, waktu proses aktual yang didapat dari total waktu proses dikurangi waktu tidak aktif untuk lebih jelas bisa dilihat pada gambar dibawah.

2.7. Memory Usage

Nilai General Memory meningkat menunjukkan bahwa lebih banyak node DOM dibuat pada pohon DOM [8]. Pengukuran dapat dilakukan dengan menggulir daftar data dengan meminta data dari API. Kinerja aplikasi ditentukan oleh jumlah memori yang digunakannya.

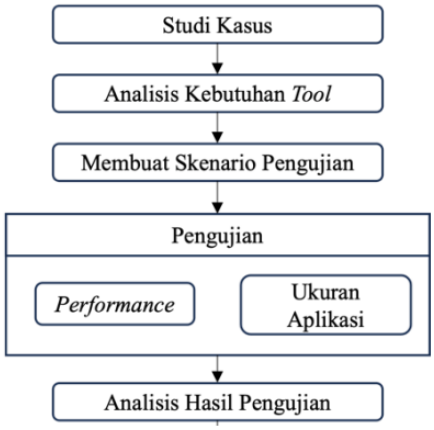
2.8. Performance Testing

Performance testing merupakan metodologi yang diterapkan untuk menguji kinerja perangkat lunak dalam suatu pengembangan. Pentingnya metode ini tidak dapat disangkal dalam ranah pengembangan perangkat lunak. Dalam metode ini, dilakukan evaluasi secara mendalam terhadap aspek kecepatan, efisiensi, dan fungsionalitas jaringan, program komputer, serta perangkat lunak dan keras yang terlibat [9].

3. METODE PENELITIAN

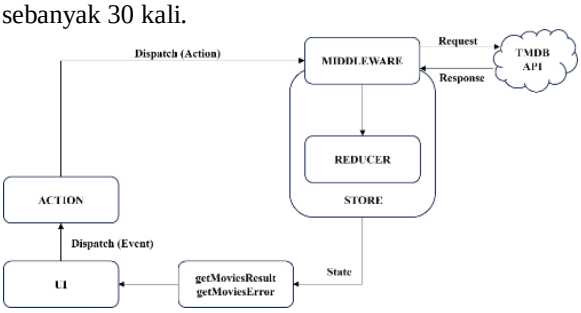
Penelitian ini menerapkan metode performance testing langkah-langkahnya mencakup studi kasus, analisis kebutuhan tool, merancang skenario

pengujian, pelaksanaan pengujian, dan analisis, berdasarkan hal tersebut, metode yang digunakan pada penelitian ini adalah performance testing [10].

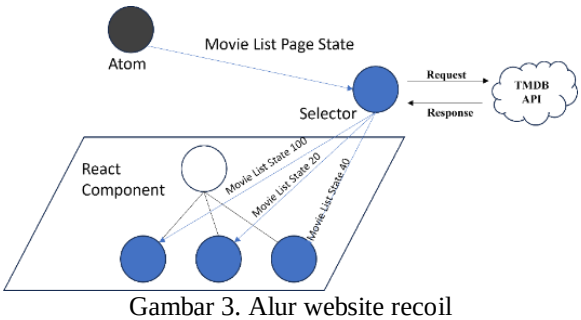


Gambar 1. Alur penelitian

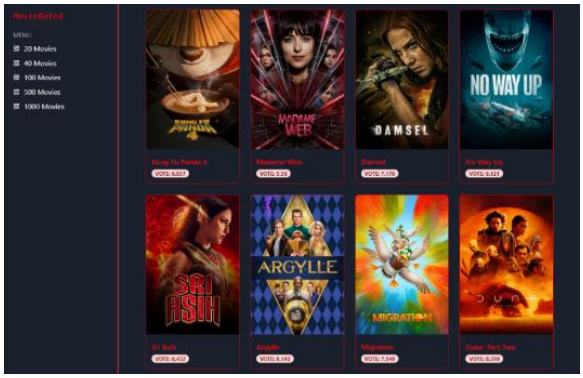
Tahap ini melakukan implementasi perangkat lunak. Website MovieRate yang sebelumnya sudah ada akan dilakukan pembaruan dengan mengimplementasikan state management yang berbeda. MovieRate menerapkan API (Application Programming Interface). API yang digunakan diambil dari laman api.themoviedb.org. API ini dipilih karena open source, dan berupa data JSON (JavaScript Object Notation). Website yang dibuat terdapat tampilan daftar yang dapat di-scroll dan dirancang secara sederhana untuk menitikberatkan pada State Management. Parameter pengujian yang akan diukur adalah ukuran website, processing speed, dan memory usage. Setiap perangkat lunak dilakukan pengujian untuk setiap kategori. Kategori dibedakan berdasarkan jumlah data yang digunakan yaitu 100, 500, dan 1000 yang nantinya akan diambil data dari hasil uji sebanyak 30 kali.



Gambar 2. Alur website redux



Gambar 3. Alur website recoil



Gambar 4. Tampilan website movierate

Setelah kedua aplikasi selesai dibuat, selanjutnya akan dilakukan pengujian untuk mengetahui efektivitas penggunaan state management Recoil dan Redux pada aplikasi ReactJS dengan memperhatikan parameter pengujian seperti ukuran aplikasi pada *disk* dan performa (speed processing dan konsumsi memory).

4. HASIL DAN PEMBAHASAN

Pengujian ukuran aplikasi dilakukan dengan membangun dengan perintah *"npm run build"*. Hasil pengujian ukuran aplikasi menunjukkan bahwa website ReactJS yang mengimplementasikan penggunaan state management Recoil dan Redux tidak memiliki perbedaan ukuran aplikasi yang signifikan, dengan hasil build website masing-masing menunjukkan ukuran 20.2 MB.

Sedangkan pengujian performa dilakukan dengan mengambil data dari pengujian menggunakan ChromeDevtools, *speed processing* dan *memory usage* ketika menggunakan aplikasi, adapun skenario pengujian aplikasi yang akan digunakan terdapat pada Tabel 1.

Tabel 1. Skenario pengujian *performance*

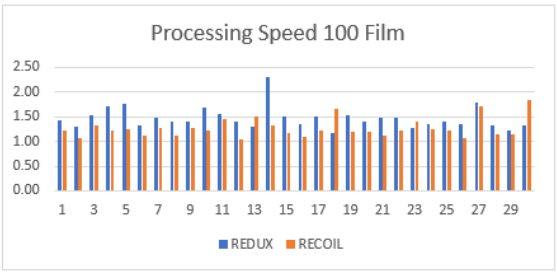
No	Skenario Pengujian
1	Klik menu pada navigasi untuk memanggil daftar film
2	Daftar film selesai *load* dan *render*
3	Menggulirkan daftar film

Pengujian ini menggunakan tools ChromeDevtools bawaan Google Chrome, serta untuk menyimulasikan penggunaan nyata dari website, kedua aplikasi dijalankan pada perangkat fisik Macbook Pro 2015.

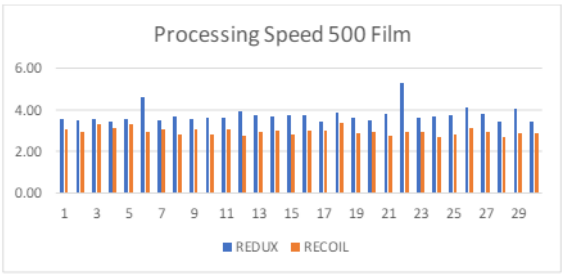
4.1. Pengujian Processing Speed

Data *processing speed* menggunakan satuan detik (s) yang diperoleh dari *ChromeDevtools* pada perangkat lunak Redux dan Recoil dengan menjalankan skenario yang sebelumnya sudah ditentukan, data diperoleh dengan menghitung waktu dari saat tombol profiler diklik, hingga aplikasi tidak lagi menampilkan poster film atau berarti seluruh

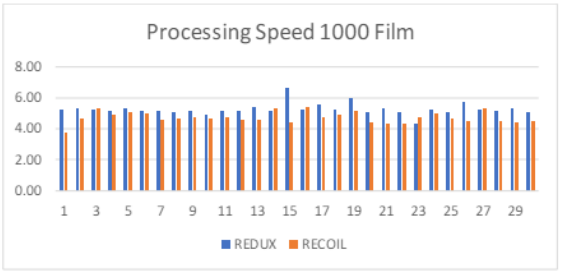
proses telah berhasil dan masing-masing kategori 100, 500 dan 1000 total film diuji sebanyak 30 kali.



Gambar 5. Processing speed 100 film



Gambar 6. Processing speed 500 film

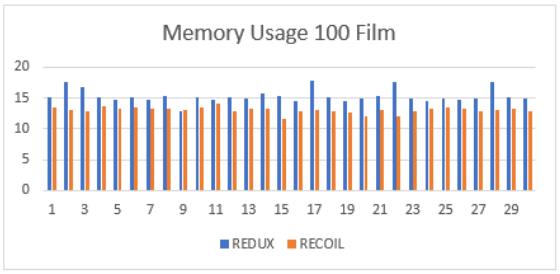


Gambar 7. Processing speed 1000 film

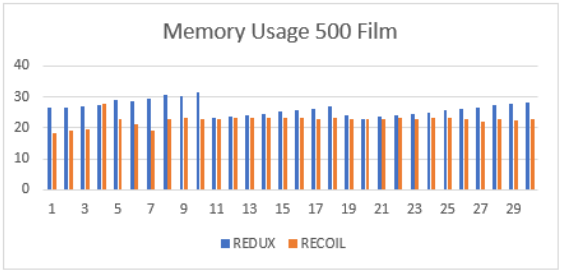
Berdasarkan hasil pengujian processing speed sebanyak 30 kali, Recoil menunjukkan kinerja yang lebih baik dibandingkan Redux dalam semua skenario yang diuji. Untuk 100 film, rata-rata processing speed Redux adalah 1.44 detik dengan nilai minimum 0.81 detik dan maksimum 2.03 detik, sedangkan Recoil memiliki rata-rata 1.22 detik, minimum 0.62 detik, dan maksimum 1.75 detik. Pada pengujian 500 film, Redux memiliki rata-rata 3.14 detik, minimum 2.18 detik, dan maksimum 4.96 detik, sementara Recoil menunjukkan rata-rata 2.72 detik, minimum 2.01 detik, dan maksimum 3.72 detik. Untuk 1000 film, rata-rata processing speed Redux adalah 4.65 detik dengan minimum 3.59 detik dan maksimum 6.27 detik, sedangkan Recoil memiliki rata-rata 3.85 detik, minimum 2.92 detik, dan maksimum 5.16 detik. Data ini menunjukkan bahwa Recoil lebih cepat dan efisien dibandingkan Redux dalam speed processing.

4.2. Memory Usage

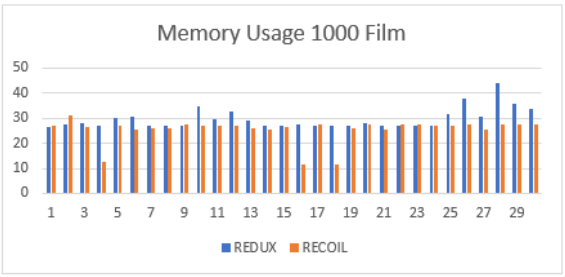
Memory usage dalam megabyte (MB), yang diperoleh dari ChromeDevtools yang mana menjalankan skenario yang sudah sebelumnya ditentukan, dengan menjalankan 3 kategori yaitu melakukan get data 100, 500, dan 1000 film sebanyak 30 kali lalu menggulirkan daftar film tersebut.



Gambar 10. *Memory usage* 100 film



Gambar 11. *Memory usage* 500 film



Gambar 12. *Memory usage* 1000 film

Berdasarkan hasil pengujian penggunaan memori untuk 100, 500, dan 1000 film, Recoil menunjukkan kinerja yang lebih efisien dibandingkan Redux. Untuk 100 film, rata-rata penggunaan memori Redux adalah 14.93 MB dengan nilai minimum 12.76 MB dan maksimum 18.42 MB, sementara Recoil memiliki rata-rata 12.45 MB, minimum 10.38 MB, dan maksimum 15.97 MB. Pada pengujian 500 film, rata-rata penggunaan memori Redux adalah 23.72 MB dengan nilai minimum 20.43 MB dan maksimum 31.84 MB, sedangkan Recoil menunjukkan rata-rata 19.36 MB, minimum 17.29 MB, dan maksimum 26.94 MB. Untuk 1000 film, rata-rata penggunaan memori Redux adalah 30.52 MB dengan nilai minimum 27.13 MB dan maksimum 43.76 MB, sementara Recoil memiliki rata-rata 24.87 MB, minimum 22.16 MB, dan maksimum 33.45 MB. Data ini menunjukkan bahwa Recoil lebih efisien dalam penggunaan memori dibandingkan Redux dalam semua skenario yang diuji.

4.3. Uji Normalitas

Data yang dikumpulkan akan diuji normalitas. Uji normalitas adalah prosedur statistik yang digunakan untuk menentukan apakah sampel data berasal dari populasi yang terdistribusi normal atau tidak yang nantinya akan menentukan metode uji

perbandingan. Uji normalitas yang akan digunakan dalam studi ini menggunakan metode Shapiro-Wilk dengan bantuan aplikasi IBM SPSS, pemilihan metode karena data dari tahap pengujian yang diuji kurang dari 50 data.

Tabel 2. Uji normalitas *speed processing*

Kategori	State Management	Shapiro-Wilk	Sig. (p-value)
TIME_100	Redux	0.842	< 0.001
	Recoil	0.822	< 0.001
TIME_500	Redux	0.685	< 0.001
	Recoil	0.947	0.142
TIME_1000	Redux	0.77	< 0.001
	Recoil	0.96	0.31

Tabel 3. Uji normalitas *memory usage*

Kategori	State Management	Shapiro-Wilk	Sig. (p-value)
MEM_100	Redux	0.783	< 0.001
	Recoil	0.912	0.017
MEM_500	Redux	0.965	0.421
	Recoil	0.708	< 0.001
MEM_1000	Redux	0.730	< 0.001
	Recoil	0.533	< 0.001

"Kategori" menunjukan jumlah film yang dirender pada saat pengujian, "State Management" menunjukan library state management yang diuji, "Sig." menunjukkan nilai signifikansi untuk uji Shapiro-Wilk. Nilai signifikansi yang lebih kecil dari 0,05 menunjukkan bahwa data tidak berdistribusi normal. "Statistic" menunjukkan nilai statistik Shapiro-Wilk. Nilai statistik Shapiro-Wilk berkisar antara 0 dan 1. Nilai yang lebih dekat ke 1 menunjukkan bahwa data lebih berdistribusi normal.

Berdasarkan Tabel 2 dan Tabel 3, dapat disimpulkan bahwa data untuk semua kategori (TIME_100, TIME_500, TIME_1000, MEM_100, MEM 500, DAN MEM_1000) tidak berdistribusi normal. Hal ini ditunjukkan oleh nilai signifikansi yang lebih kecil dari 0,05 untuk semua variabel dan semua tingkat memori, serta nilai statistik Shapiro-Wilk yang kurang dari 0,9 untuk semua kategori oleh karena itu analisis data pengujian dilakukan dengan metode Mann-Whitney.

4.4. Analisis Data

Setelah uji normalitas selesai, maka sudah bisa ditentukan metode uji perbandingan yang akan digunakan, berdasarkan hasil dari uji normalitas, baik *processing speed* dan *memory usage* akan menggunakan uji perbandingan dengan metode *non-parametric* yaitu Mann-Whitney *test*, dan berikut merupakan hasil dari Mann-Whitney *test* pada Tabel 4 dan Tabel 5. Perlu dicatat bahwa untuk menentukan hipotesis apakah data berbeda, Ini sangat penting untuk proses pengambilan keputusan. Terdapat dua hipotesis, yaitu H0 dan Ha. Hipotesis H0 diterima jika nilai signifikansi t-test lebih dari 0.05, yang menunjukkan bahwa tidak ada perbedaan antara dua variabel. Hipotesis Ha diterima jika nilai signifikansi t-test kurang dari 0.05, yang menunjukkan bahwa ada perbedaan antara dua variabel. Detail hipotesis adalah:

a. H0 = Tidak adanya perbedaan yang signifikan pada processing speed / memory usage.
b. Ha = Adanya perbedaan yang signifikan pada processing speed / memory usage.

Tabel 4. Mann-Whitney *test processing speed*

Kategori	Mann-Whitney	Sig. (p-value)
MEM_100	24.500	< 0.001
MEM_500	40.500	< 0.001
MEM_1000	248.500	0.003

Pada interval waktu 100 film, nilai rata-rata data dari Redux dan Recoil masing-masing adalah 1.47 dan 1.27, dengan nilai $p < 0.001$, menunjukkan bahwa manajemen state Recoil lebih efisien dibandingkan Redux. Untuk interval waktu 500 film, nilai rata-rata Redux dan Recoil adalah 3.75 dan 2.96, dengan nilai $p < 0.001$, yang juga menunjukkan efisiensi lebih tinggi pada Recoil. Pada interval waktu 1000 film, nilai rata-rata Redux dan Recoil masing-masing adalah 5.27 dan 4.74, dengan nilai $p < 0.001$, yang kembali menunjukkan keunggulan efisiensi Recoil. Hasil uji Mann-Whitney menunjukkan nilai signifikan yang sangat rendah untuk semua interval waktu: 164.500 ($p < 0.001$) untuk TIME_100, 0.000 ($p < 0.001$) untuk TIME_500, dan 135.000 ($p < 0.001$) untuk TIME_1000, semakin memperkuat kesimpulan bahwa manajemen state Recoil lebih efisien dibandingkan Redux dalam semua kondisi yang diuji.

Tabel 5. Mann-Whitney *test memory usage*

Kategori	Mann-Whitney	Sig. (p-value)
TIME_100	164.500	< 0.001
TIME_500	0.000	< 0.001
TIME_1000	135.000	< 0.001

Berdasarkan hasil Mann-Whitney *test memory usage*, *library state management* antara Redux dan Recoil pada berbagai jumlah film menunjukkan bahwa Recoil lebih efisien. Pada 100 film, nilai p-value adalah <0.001, yang berarti Ha diterima, dengan rata-rata data Redux sebesar 15.35 dan Recoil sebesar 13.03. Pada 500 film, p-value juga <0.001 dan Ha diterima, dengan rata-rata Redux sebesar 26.42 dan Recoil sebesar 22.57. Terakhir, untuk 1000 film, p-value sebesar 0.003, yang juga <0.05 dan Ha diterima, dengan rata-rata Redux sebesar 29.63 dan Recoil sebesar 25.45. Dari hasil ini, Recoil terbukti lebih efisien dibandingkan Redux dalam *memory usage*.

5. KESIMPULAN DAN SARAN

Penelitian ini berhasil melakukan perbandingan perangkat lunak dengan menggunakan library Redux dan Recoil dengan metode performance testing. Teknologi dapat diuji kinerjanya dengan metode pengujian kinerja. Hasil pengujian ini dapat digunakan untuk membandingkan kinerja teknologi lain.

Teknologi yang dapat diukur kinerjanya salah satunya adalah library dari ReactJS yaitu library Redux dan Recoil.

Berdasarkan hasil analisis perbandingan performance (processing speed dan memory usage) dengan metode Mann-Whitney, dari ketiga parameter yang diuji, 100, 500, dan 1000 film, Recoil lebih cepat dalam processing speed dan lebih efisien dalam memory usage, jadi dapat disimpulkan bahwa Recoil lebih baik ketimbang Redux. Meskipun lebih cepat dan efisien dalam segi performa, ukuran folder proyek Recoil lebih besar dibandingkan dengan Redux, Sehingga hasilnya dapat dibandingkan dengan penelitian ini, disarankan untuk menggunakan API dan jumlah data yang berbeda. Selain itu, metode pengujian yang berbeda juga dapat digunakan, seperti melakukan komputasi berat terhadap state. Sebaiknya gunakan alat bantu profiling website yang lain untuk menguji performa.

DAFTAR PUSTAKA

[1] Le Thanh, "Comparison of State Management Solutions between Context API and Redux Hook in ReactJS," Metropolia University of Applied Science, Helsinki, 2021. Accessed: Dec. 03, 2023. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/494321/Le_Thanh.pdf?sequence=2

[2] F. F. Nursaid, A. Hendra Brata, and A. P. Kharisma, "Pengembangan Sistem Informasi Pengelolaan Persediaan Barang Dengan ReactJS Dan React Native Menggunakan Prototype (Studi Kasus : Toko Uda Fajri)," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 4, no. 1, pp. 46–55, 2020, Accessed: Dec. 16, 2023. [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/6859

[3] V. Siahaan and R. H. Sianipar, *Buku Pintar JavaScript*. Balige Publishing, 2020.

[4] K. Tran, "STATE MANAGEMENT IN REACT," 2022. Accessed: Dec. 03, 2023. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/798988/State%20management%20in%20React_kim%20tran.pdf?sequence=2&isAllowed=y

[5] "Getting Started with Redux." Accessed: Dec. 02, 2023. [Online]. Available: https://redux.js.org/introduction/getting-started

[6] D. Anh Le and V. Ammattikorkeakoulu, "E-COMMERCIAL FULL STACK WEB APPLICATION DEVELOPMENT with React, Redux, NodeJS, and MongoDB Technology and Communication 2023," 2023. Accessed: Dec. 01, 2023. [Online]. Available: https://urn.fi/URN:NBN:fi:amk-2023060621826

[7] "Recoil Core Concept." Accessed: Dec. 07, 2023. [Online]. Available: https://recoiljs.org/docs/introduction/core-concepts/

[8] K. Basques, "Fix Memory Problems." Accessed: Dec. 19, 2023. [Online]. Available: https://developer.chrome.com/docs/devtools/memory-problems/

[9] S. Pradeep and Y. K. Sharma, "A Pragmatic Evaluation of Stress and Performance Testing Technologies for Web Based Applications," in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 399–403. doi: 10.1109/AICAI.2019.8701327.

[10] Mgs. M. F. Abdillah, I. L. Sardi, and A. Hadikusuma, "Analisis Performa GetX dan BLoC State Management Library Pada Flutter untuk Perangkat Lunak Berbasis Android," *LOGIC: Jurnal Penelitian Informatika*, vol. 1, no. 1, p. 73, Sep. 2023, doi: 10.25124/logic.v1i1.6479