**LAPORAN EKSPERIMEN MENGENAI PERBANDING REDUX DENGAN ZUSTAND DALAM PENGELOLAAN STATE**

*Laporan ini disusun untuk memenuhi tugas 4 mata kuliah Pengembangan Web (Teori)*



Disusun oleh kelompok B4:

Asri Husnul Rosadi 221524035

Faris Abulkhoir 221524040

Mahardika Pratama 221524044

Muhamad Fahri Yuwan 221524047

Najib Alimudin Fajri 221524053

Sarah 221524059

Septyana Agustina 221524060

Dosen Pengampu:

Joe Lian Min, M.Eng.

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**PROGRAM STUDI D4 TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2024**

# **DAFTAR ISI**

[DAFTAR ISI i](#_Toc175471873)

[A. IDENTIFIKASI PROBLEM 1](#_Toc175471874)

[B. DESKRIPSI PROBLEM 1](#_Toc175471875)

[C. METODOLOGI EKSPERIMEN 1](#_Toc175471876)

[D. PELAKSANAAN EKSPERIMEN 1](#_Toc175471877)

[E. ANALISIS HASIL EKSPERIMEN 2](#_Toc175471878)

# **IDENTIFIKASI PROBLEM**

Manajemen state dalam aplikasi React adalah salah satu aspek yang paling penting namun seringkali kompleks, terutama dalam aplikasi skala besar. Dengan banyaknya library yang tersedia untuk manajemen state, seperti Redux dan Zustand, pengembang sering kali bingung memilih solusi terbaik yang sesuai dengan kebutuhan aplikasi mereka.

# **DESKRIPSI PROBLEM**

Redux adalah salah satu library manajemen state yang paling populer dan telah digunakan dalam berbagai aplikasi skala besar. Namun, banyak pengembang merasa bahwa Redux memiliki terlalu banyak boilerplate dan kompleksitas, terutama untuk aplikasi yang lebih kecil. Di sisi lain, Zustand adalah library manajemen state yang lebih ringan dan minimalis, menawarkan cara yang lebih sederhana untuk mengelola state tanpa mengorbankan fleksibilitas.

Masalah yang dihadapi adalah bagaimana memilih library manajemen state yang tepat berdasarkan kebutuhan aplikasi, dan untuk itu perlu dilakukan evaluasi yang sistematis terhadap performa, kemudahan implementasi, dan fleksibilitas dari kedua library ini.

# **METODOLOGI EKSPERIMEN**

Untuk membandingkan Redux dan Zustand, eksperimen ini dilakukan dengan tiga pengujian utama:

1. **Pengujian Waktu Implementasi:** Mengukur waktu yang diperlukan untuk mengimplementasikan fitur manajemen state dasar menggunakan Redux dan Zustand.
2. **Pengujian Performa:** Mengukur efisiensi dalam hal rendering dan penggunaan memori ketika melakukan operasi increment secara berulang.
3. **Pengujian Fleksibilitas dan Kemudahan Penggunaan:** Mengevaluasi seberapa mudah kedua library menangani operasi asynchronous dan skenario yang lebih kompleks.

# **PELAKSANAAN EKSPERIMEN**

1. Pengujian Waktu Implementasi

* **Redux:** Implementasi fitur increment/decrement sederhana dilakukan dengan mengatur store, reducer, dan action. Waktu implementasi dicatat sejak mulai membuat store hingga fitur dapat berjalan di UI.
* **Zustand:** Implementasi dilakukan dengan membuat store yang lebih sederhana tanpa banyak boilerplate. Waktu implementasi dicatat dengan cara yang sama seperti pada Redux.

1. Pengujian Performa

* **Setup:** Pada kedua implementasi, ditambahkan tombol *Mass Increment* yang menjalankan 100 kali operasi increment secara berurutan.
* **Pengukuran:** Menggunakan Chrome DevTools untuk memprofil waktu rendering dan penggunaan memori saat tombol ini diklik.

1. Pengujian Fleksibilitas dan Kemudahan Penggunaan

* **Redux:** Menambahkan middleware redux-thunk untuk menangani operasi asynchronous dan membuat action yang mengambil data dari API.
* **Zustand:** Menambahkan fungsi async langsung di dalam store Zustand tanpa middleware tambahan.
* **Pengukuran:** Mengevaluasi jumlah kode tambahan dan kompleksitas yang diperlukan untuk operasi asynchronous di kedua library.

# **ANALISIS HASIL EKSPERIMEN**

1. Waktu Implementasi

* **Redux:** Waktu implementasi lebih lama dibandingkan Zustand karena Redux membutuhkan lebih banyak boilerplate, termasuk konfigurasi store, reducer, dan action.
* **Zustand:** Waktu implementasi lebih cepat karena API yang lebih sederhana dan tidak memerlukan banyak setup.

1. Performa

* **Redux:** Performa dalam hal rendering dan penggunaan memori cukup baik, namun overhead yang dihasilkan lebih besar dibandingkan Zustand karena Redux lebih terstruktur.
* **Zustand:** Menunjukkan efisiensi yang lebih baik dalam rendering dan penggunaan memori. Karena desainnya yang minimalis, Zustand lebih ringan dan menghasilkan overhead yang lebih rendah.

1. Fleksibilitas dan Kemudahan Pengguna

* **Redux:** Menyediakan struktur yang ketat dan cocok untuk aplikasi besar, namun membutuhkan lebih banyak kode dan setup untuk menangani operasi async.
* **Zustand:** Lebih fleksibel dan mudah diimplementasikan, terutama dalam operasi async, karena tidak memerlukan middleware tambahan dan setup yang rumit.