BAB 1 PENDAHULUAN

1.1. Latar Belakang

Dalam era digital saat ini, performa perangkat lunak menjadi faktor penting dalam pengembangannya. Salah satu faktor yang mempengaruhi performa perangkat lunak adalah state management [1]. Flutter merupakan *framework* yang mendukung penerapan *state management* dalam pengembangan perangkat lunak *Android*. *Widget* dalam Flutter adalah komponen utama yang membentuk tampilan antarmuka pengguna UI (*user interface*). Setiap *widget* dapat menampilkan informasi atau merespons interaksi pengguna berdasarkan *state* yang dimilikinya. *State* adalah data yang dapat diakses secara *sinkron* saat *widget* dibuat dan dapat mengalami perubahan selama *widget* tersebut tetap aktif [2]. Setiap perubahan *state* mempengaruhi tampilan, sehingga *widget* harus diperbarui agar tetap sesuai dengan kondisi terkini [3]. Namun, Flutter secara *default* akan memperbarui semua *widget* dalam satu tampilan ketika metode *setState* dipanggil, meskipun hanya satu *widget* yang benar-benar memerlukan pembaruan [4]. Hal ini terjadi karena *setState* memicu pembaruan pada seluruh *widget* dalam tata letak tampilan, bukan hanya *widget* yang mengalami perubahan *state.*

Berdasarkan hal tersebut, diperlukan cara untuk mengelola *state (State Management)*. *State management* adalah mekanisme untuk mengelola *state* yang dapat berubah dan memastikan tampilan UI (*user interface)* diperbarui secara efisien sesuai dengan perubahan tersebut [5]. Dengan memanfaatkan *library state management,* hanya *widget* yang mengalami perubahan *state* yang akan diperbarui. Minimnya pembaruan pada *widget* membantu mengurangi penggunaan sumber daya secara keseluruhan [4]. Efisiensi kinerja perangkat lunak dapat diukur melalui berbagai aspek, salah satunya adalah pemanfaatan sumber daya. Pada perangkat lunak *mobile,* sumber daya tersebut mencakup *CPU,* memori, daya baterai, dan koneksi jaringan [6]. Penelitian sebelumnya [6] mengungkapkan bahwa pemilihan *state management* dapat mempengaruhi performa suatu perangkat lunak, sehingga setiap *state management* dapat memberikan hasil yang berbeda dalam performa.

Sehingga perlu adanya penelitian untuk membandingkan aspek performa dari *state management* perangkat lunak untuk mengetahui kelebihan dan kekurangan masing-masing *state managemen*t.

Flutter menyediakan berbagai metode manajemen *state*, seperti Provider, Riverpod, GetX, dan BLoC, yang memungkinkan pengembang mengelola perubahan data dalam perangkat lunak [7]. Provider menjadi salah satu metode yang paling banyak digunakan karena kesederhanaannya serta dukungan komunitas yang luas [8]. Namun, metode ini memiliki keterbatasan dalam mengelola *state* yang lebih kompleks, terutama karena bergantung pada *BuildContext*, yang berperan dalam menentukan lokasi *widget* dalam struktur perangkat lunak [9]. Ketergantungan ini dapat menyulitkan pengelolaan *state* di luar hierarki *widget* tertentu, sehingga dalam beberapa kasus dapat menyebabkan kode menjadi lebih sulit dipelihara dan diperluas .

Sebagai alternatif dari Provider, Riverpod dikembangkan untuk menghilangkan ketergantungan pada *BuildContext* dan memberikan fleksibilitas lebih besar dalam mengelola *state* [10]. Riverpod menerapkan sistem pengelolaan *state* yang lebih terpisah dari UI, memungkinkan *state* diakses secara lebih global tanpa bergantung pada konteks *widget* [11]. Selain itu, Riverpod mempermudah proses *refactoring code,* karena *state* tidak lagi terikat dengan struktur *widget*, sehingga lebih mudah diadaptasi dalam pengembangan perangkat lunak dengan kompleksitas tinggi.

Dalam pengelolaan *state,* banyaknya *widget* dan interaksi dalam perangkat lunak mempengaruhi efisiensi suatu metode. Semakin banyak *widget* yang harus diperbarui, semakin besar konsumsi sumber daya [4]. Faktor seperti *widget*, jumlah data, dan alur kerja mempengaruhi efisiensi. Meskipun Riverpod dikembangkan untuk mengatasi keterbatasan Provider, hingga saat ini belum ada penelitian yang secara langsung membandingkan performa keduanya. Untuk itu, perlu dilakukan pengujian dan analisis performansi Provider dan Riverpod.

1.2. Rumusan Masalah

Berdasarkan latar belakang penelitian ini, maka dapat dirumuskan masalah yang akan diselesaikan adalah :

1. Bagaimana menerapkan penggunaan *state management* Provider dan Riverpod terhadap perangkat lunak berbasis Flutter?
2. Bagaimana mengevaluasi performa perangkat lunak berbasis Flutter yang menggunakan Provider maupun Riverpod?

1.3. Tujuan dan Manfaat

Adapun tujuan dan manfaat penelitian ini adalah :

1. Menerapkan penggunaan state management Provider dan Riverpod dalam perangkat lunak berbasis Flutter.
2. Mengevaluasi performa perangkat lunak berbasis Flutter yang menggunakan Provider dan Riverpod, khususnya dalam aspek penggunaan CPU dan konsumsi memori.

1.4. Batasan Masalah

Batasan masalah yang di terapkan dalam penelitian ini adalah sebagai berikut.

1. Aspek performa yang diukur dalam penelitian ini terbatas pada konsumsi *CPU* dan memori tanpa menganalisis parameter lain seperti waktu respons atau penggunaan baterai.
2. Pengujian performa perangkat lunak hanya dilakukan pada platform *Android* dan tidak melibatkan platform lain seperti *IOS, web* atau *desktop*.
3. Jumlah data yang diuji terbatas pada 100, 1.000, 10.000, dan 20.000 entri, yang dipilih untuk mensimulasikan kondisi perangkat lunak dengan skala data kecil hingga besar. Rentang ini dipilih untuk menguji bagaimana Provider dan Riverpod menangani perbedaan jumlah data dalam pengelolaan *state.*

1.5. Metode Penelitian

Penelitian ini menggunakan metode kuantitatif data dengan tahapan sebagai berikut:

1. Studi literatur dilakukan dengan mengkaji dokumentasi resmi Flutter, penelitian sebelumnya, dan sumber-sumber ilmiah yang relevan terkait *state management*, khususnya Riverpod dan Provider, serta metode evaluasi performa perangkat lunak.
2. Perancangan perangkat lunak mencakup pengembangan perangkat lunak berbasis Flutter untuk *platform Android* dengan menerapkan dua pendekatan state management, yaitu Riverpod dan Provider.
3. Pengujian performa dilakukan untuk mengukur aspek seperti konsumsi memori, penggunaan CPU.
4. Analisis data dilakukan membandingkan efisiensi performa antara Riverpod dan Provider berdasarkan data hasil pengujian.
5. Kesimpulan dan evaluasi mencakup penyusunan kesimpulan berdasarkan analisis data dan rekomendasi terkait implementasi state management pada perangkat lunak berbasis Flutter.