BAB 5
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi, pengujian, dan analisis perbandingan yang telah dilakukan pada bab-bab sebelumnya, dapat ditarik beberapa kesimpulan utama sebagai berikut:

1. *Modifiability*: Terdapat perbedaan yang jelas antara *state management* Provider dan BLoC. Provider unggul dalam hal kecepatan dan kemudahan pada implementasi awal, karena mereka membutuhkan lebih sedikit file dan boilerplate kode, sehingga cocok untuk pengembangan prototipe dan pengembangan yang membutuhkan waktu yang cepat. Sebaliknya, BLoC menawarkan struktur kode kode yang lebih formal, modular, dan dapat diprediksi, yang secara signifikan memudahkan pemeliharaan aplikasi dan skalabilitas dalam jangka pangjang, terutama untuk aplikasi yang kompleks.
2. *Testability*: BLoC unggul dalam hal ini. Untuk memvalidasi aliran state secara keseluruhan, library *bloc_test* menyediakan kerangka kerja pengujian yang jelas dan dapat diandalkan. Selain itu, bukti eksekusi menunjukkan bahwa rangkaian unit tes BLoC lebih cepat (total 1,1 detik) daripada rangkaian tes Provider (total 1,4 detik). Ini menunjukkan kemenangan kualitatif dan kuantitatif dalam hal keandalan eksekusi.
3. *Performance* (*Memory* dan *Frame Rendering*): Analisis performa menunjukkan bahwa BloC sangat menguntungkan, terutama pada perangkat yang memiliki sumber daya terbatas. Saat aplikasi berada di bawah tekanan, BLoC terbukti lebih efektif dalam penggunaan memori (RSS) dan secara signifikan mengurangi frekuensi pengumpulan sampah (GC). Keunggulan ini berhubungan langsung dengan kinerja rendering, di mana BLoC menghasilkan *frame jank* yang jauh lebih sedikit hingga 40% lebih banyak dalam skenario berat dibandingkan Provider. Hal ini menunjukkan bahwa BloC membuat aplikasi yang lebih cepat dan responsif.

Secara keseluruhan, meskipun Provider menawarkan keuntungan dalam hal kecepatan pengembangan di awal. BLoC terbukti menjadi pendekatan yang lebih baik dalam hal aplikasi UBP yang menerapkan Clean Architecture sebagai

arsitektur utamanya. Keunggulannya yang signifikan dalam stabilitas, efisiensi memori, dan kelancaran rendering menjadikannya pilihan yang lebih kuat dan dapat diandalkan untuk menghasilkan aplikasi berkualitas tinggi yang dapat dipelihara dalam jangka panjang.

5.2 Saran

Berdasarkan kesimpulan yang telah dibuat, berikut adalah beberapa saran yang dapat digunakan untuk penelitian dan pengembangan aplikasi serupa:

1. Pertimbangkan Skala Proyek dan Target Perangkat: Bagi pengembang aplikasi Flutter, pemilihan state management sebaiknya didasarkan pada skala proyek, kompleksitas fitur, dan target audiens (terutama spesifikasi perangkat). Untuk aplikasi yang menargetkan audiens luas (termasuk low-end) dan diharapkan berkembang, investasi awal untuk mengadopsi BLoC sangat direkomendasikan karena imbal hasil performa (memory dan rendering) dan maintainability-nya yang terbukti.
2. Perluas Perbandingan State Management: Penelitian selanjutnya dapat memperluas perbandingan dengan menyertakan solusi state management populer lainnya seperti Riverpod atau GetX dalam konteks Clean Architecture untuk memberikan gambaran yang lebih komprehensif.
3. Analisis Metrik Performa Tambahan: Metrik performa lain seperti konsumsi baterai, ukuran aplikasi final (APK/AAB), dan waktu startup aplikasi dapat dianalisis pada penelitian selanjutnya untuk memberikan evaluasi kinerja yang lebih holistik, melengkapi temuan mengenai memori dan rendering dari penelitian ini.
4. Terapkan pada Studi Kasus Berbeda: Metodologi perbandingan ini dapat diterapkan pada jenis aplikasi yang berbeda (misalnya, aplikasi e-commerce, media sosial, atau game) untuk menguji apakah temuan mengenai efisiensi memori dan rendering ini berlaku secara general atau spesifik pada tipe aplikasi tertentu.

Diharapkan saran-saran ini dapat memberikan kontribusi praktis bagi komunitas pengembang Flutter serta menjadi landasan bagi penelitian-penelitian berikutnya di bidang arsitektur perangkat lunak dan state management.