BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini telah berhasil menganalisis perbandingan performa teknis antara Kotlin Multiplatform dan Flutter dalam pengembangan aplikasi manajemen akademik mahasiswa berbasis *mobile* lintas *platform*. Melalui pendekatan *quasi-experimental design*, penelitian ini menghasilkan data empiris yang sistematis dan objektif dengan mengukur enam metrik performa utama, yaitu CPU *usage*, *memory consumption*, *battery consumption*, *application size*, *startup performance*, dan UI *performance* pada dua *platform* berbeda, yakni Android dan iOS. Berdasarkan hasil penelitian dan analisis yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut:

1. Hasil *benchmarking* menunjukkan bahwa Kotlin Multiplatform dan Flutter memiliki karakteristik performa yang berbeda secara signifikan pada masing-masing *platform*. Pada ekosistem Android, Kotlin Multiplatform mendominasi seluruh metrik pengujian. Framework ini mampu memberikan efisiensi CPU sebesar 46,1% lebih baik dibandingkan Flutter, konsumsi memori 158% lebih efisien, efisiensi baterai 88% lebih unggul, ukuran aplikasi 161% lebih kecil, waktu startup 54% lebih cepat, serta stabilitas UI yang lebih konsisten. Sementara itu, pada ekosistem iOS, pola yang muncul lebih bervariasi. Flutter memperlihatkan performa yang lebih baik pada empat metrik, termasuk ukuran aplikasi yang 42% lebih kecil dan waktu startup yang 76% lebih cepat, sedangkan Kotlin Multiplatform tetap mempertahankan keunggulannya dalam efisiensi memori dan konsumsi daya baterai. Temuan ini menegaskan bahwa performa *framework* lintas *platform* tidak bersifat seragam, melainkan sangat dipengaruhi oleh *platform* target yang digunakan.
2. Hasil *benchmarking* memperlihatkan adanya pola performa yang berbeda antara Kotlin Multiplatform dan Flutter pada masing-masing *platform*. Pada ekosistem Android, Kotlin Multiplatform mendominasi seluruh aspek pengujian dengan keunggulan konsisten di CPU, memori, baterai, ukuran

aplikasi, waktu startup, dan stabilitas UI. Dominasi ini disebabkan oleh sifat KMP yang secara *native* terintegrasi dengan Android Runtime sehingga lebih efisien dalam memanfaatkan sumber daya perangkat. Sementara itu, pada ekosistem iOS, pola yang muncul berbalik: Flutter unggul pada sebagian besar metrik terutama performa startup dan ukuran aplikasi, sedangkan KMP hanya unggul dalam efisiensi memori dan baterai. Dengan kata lain, *benchmarking* tidak hanya mengungkap siapa yang lebih unggul, tetapi juga menjelaskan bahwa masing-masing *framework* memiliki kekuatan spesifik yang dipengaruhi oleh arsitektur *platform*, mekanisme manajemen memori, serta strategi kompilasi yang digunakan. Temuan ini menegaskan bahwa evaluasi *framework* perlu mempertimbangkan kondisi ekosistem tempat aplikasi dijalankan, bukan hanya angka hasil uji semata.

3. Perbedaan pola performa pada hasil *benchmarking* memiliki implikasi penting bagi praktisi maupun peneliti dalam menentukan pilihan *framework* lintas *platform*. Untuk pengembangan aplikasi manajemen akademik mahasiswa dengan target pengguna dominan Android, Kotlin Multiplatform dapat menjadi pilihan yang lebih tepat karena memberikan efisiensi sumber daya, stabilitas performa, serta ukuran aplikasi yang lebih ringan. Namun, apabila tujuan utama adalah menghasilkan aplikasi lintas *platform* dengan konsistensi kinerja yang lebih seimbang di Android dan iOS, Flutter menawarkan keunggulan praktis berkat performa startup yang superior dan ekosistem dukungan yang lebih matang. Dengan demikian, hasil penelitian ini dapat dijadikan acuan empiris bagi pengembang, institusi pendidikan, maupun peneliti dalam mengevaluasi pilihan *framework* sesuai dengan kebutuhan spesifik dan konteks pengembangan aplikasi.

Berdasarkan temuan penelitian ini, dapat disimpulkan bahwa pemilihan *framework* lintas *platform* tidak dapat dilepaskan dari konteks *platform* target serta prioritas kebutuhan pengembangan. Kotlin Multiplatform menawarkan performa yang lebih optimal pada perangkat Android, sedangkan Flutter lebih unggul pada perangkat iOS. Oleh karena itu, keputusan pemilihan *framework* sebaiknya mempertimbangkan faktor *platform* dominan, karakteristik pengguna, serta tujuan jangka panjang dari aplikasi yang dikembangkan. Penelitian ini memberikan

kontribusi empiris dalam literatur evaluasi performa *framework* lintas *platform*, khususnya dalam konteks pengembangan aplikasi manajemen akademik mahasiswa, sekaligus membuka peluang untuk penelitian lanjutan dengan ruang lingkup perangkat dan skenario penggunaan yang lebih luas.

5.2 Saran

Berdasarkan temuan dan keterbatasan penelitian ini, terdapat beberapa area yang dapat dikembangkan untuk penelitian selanjutnya serta rekomendasi praktis untuk industri pengembangan aplikasi *mobile*. Penelitian ini telah memberikan baseline comparison yang valuable, namun masih terbuka ruang untuk eksplorasi yang lebih mendalam dalam berbagai aspek pengembangan aplikasi lintas *platform*:

1. Eksplorasi Domain Aplikasi yang Beragam: Penelitian selanjutnya disarankan untuk mengeksplorasi perbandingan performa pada domain aplikasi yang berbeda seperti aplikasi gaming dengan grafis intensif, aplikasi pemrosesan multimedia, *platform* e-commerce dengan transaksi kompleks, dan aplikasi komunikasi real-time. Setiap domain memiliki karakteristik komputasi dan pola penggunaan resource yang unik, sehingga dapat mengungkapkan trade-off performa yang berbeda dan memberikan wawasan tentang kesesuaian *framework* untuk use case yang spesifik.
2. Pengujian Multi-Device dan Segmentasi Hardware: Perluasan cakupan pengujian untuk mencakup spektrum perangkat yang lebih luas mulai dari perangkat entry-level hingga smartphone flagship, tablet, dan perangkat lipat. Penelitian khusus pada segmen entry-level dengan RAM terbatas dan chipset yang kurang powerful dapat memberikan insights kritis tentang aksesibilitas dan inklusivitas teknologi untuk pasar yang beragam, khususnya di emerging markets dengan penetrasi smartphone yang tinggi tetapi spesifikasi *hardware* yang terbatas.
3. Analisis Performa pada Kondisi Jaringan yang Beragam: Investigasi mendalam terhadap perilaku aplikasi dalam berbagai kondisi jaringan termasuk koneksi bandwidth rendah, jaringan latensi tinggi, konektivitas intermittent, dan skenario offline. Analisis tentang efisiensi sinkronisasi data, mekanisme caching, dan network resilience dapat memberikan

pemahaman yang lebih komprehensif tentang kemampuan *framework* dalam kondisi deployment dunia nyata yang tidak selalu optimal.

4. Studi Longitudinal Performa dan Degradasi: Penelitian jangka panjang untuk memantau performa aplikasi selama periode penggunaan yang diperpanjang dengan fokus pada deteksi memory leak, pola degradasi performa, sustainability optimisasi baterai, dan analisis stabilitas. Investigasi tentang overhead maintenance dan dampak performa *update* juga penting untuk memahami total biaya kepemilikan dalam siklus hidup aplikasi yang lengkap.
5. Analisis Kompleksitas Pengembangan dan Learning Curve: Penelitian kuantitatif dan kualitatif tentang produktivitas developer, kompleksitas learning curve, efisiensi debugging, dan beban maintenance untuk masing-masing *framework*. Analisis time-to-market, implikasi biaya pengembangan, dan kebutuhan transfer skill dapat memberikan perspektif bisnis yang valuable untuk keputusan adopsi teknologi dalam konteks organisasi.

Rekomendasi ini diharapkan dapat memberikan roadmap yang jelas untuk pengembangan penelitian selanjutnya dan praktik pengembangan aplikasi *mobile* yang lebih informed. Implementasi saran-saran tersebut akan berkontribusi pada pemahaman yang lebih mendalam tentang karakteristik *framework* lintas *platform* dan pada akhirnya mendukung pengambilan keputusan teknologi yang lebih baik dalam industri pengembangan aplikasi *mobile*.