# BAB V
# PENUTUP

## 5.1 Kesimpulan

Berdasarkan hasil implementasi, pengujian, dan analisis terhadap dua pendekatan arsitektur perangkat lunak, yaitu Clean Architecture dan Hexagonal Architecture, dalam pengembangan aplikasi Flutter berbasis AI (Google Gemini API), maka dapat disimpulkan beberapa hal sebagai berikut:

1. Keduanya berhasil diimplementasikan dengan baik pada aplikasi Flutter berbasis AI. Struktur kode yang dihasilkan dari kedua pendekatan mendukung prinsip *separation of concerns*, *modularitas*, dan *testability*, sehingga memudahkan proses pengembangan dan pengujian secara terpisah pada setiap komponen. Komponen utama seperti ChatMessage, SendMessageUseCase, CopyToClipboardUseCase, dan GeminiApiService berhasil diuji di dua platform (Linux dan Web) tanpa error atau kegagalan.
2. Dari aspek performa pengujian unit, Clean Architecture menunjukkan performa rata-rata waktu pengujian sebesar 35,45 ms, sementara Hexagonal Architecture sebesar 41,17 ms. Meskipun selisih waktu eksekusi tidak signifikan secara teknis, Clean Architecture sedikit lebih unggul dalam efisiensi waktu, khususnya pada lapisan *use case* dan layanan eksternal.
3. Perbedaan paling mencolok terlihat pada pengelolaan dan pemilihan dependensi eksternal. Clean Architecture memanfaatkan pemisahan antar lapisan tanpa kontrak eksplisit terhadap dependensi eksternal, mengandalkan mekanisme *dependency injection* secara langsung. Pendekatan ini relatif sederhana, cocok untuk tim kecil, dan cukup fleksibel untuk proyek berukuran kecil hingga menengah. Sebaliknya, Hexagonal Architecture secara eksplisit menggunakan pola *ports and adapters*, yang membungkus semua interaksi eksternal dalam abstraksi yang terpisah (port) dan menghubungkannya melalui adapter. Pendekatan

ini memberikan keuntungan dalam hal *testability*, fleksibilitas penggantian teknologi, dan *decoupling* yang lebih kuat. Namun, pendekatan ini lebih kompleks dan memerlukan kedisiplinan desain yang lebih tinggi, terutama dalam proyek besar dan tim berskala besar.

4. Menjawab rumusan masalah, yaitu, Bagaimana perbandingan Clean Architecture dan Hexagonal Architecture terhadap pengelolaan serta pemilihan dependensi eksternal dalam pengembangan aplikasi multiplatform?, Maka dapat disimpulkan bahwa:
    a. Clean Architecture menawarkan pendekatan yang lebih sederhana dan cepat dalam hal integrasi dependensi eksternal, tetapi dengan fleksibilitas yang terbatas terhadap perubahan teknologi eksternal.
    b. Hexagonal Architecture menyediakan fleksibilitas dan skalabilitas yang lebih tinggi karena dependensi eksternal dipisahkan dengan jelas melalui port dan adapter, sehingga lebih unggul dalam konteks sistem besar atau yang membutuhkan pengujian dan pergantian dependensi yang intensif.

Secara keseluruhan, pemilihan arsitektur sebaiknya disesuaikan dengan kebutuhan proyek, ukuran tim, serta skala dan kompleksitas sistem. Clean Architecture cocok untuk pengembangan cepat dan sederhana, sementara Hexagonal Architecture lebih cocok untuk sistem yang kompleks dan membutuhkan fleksibilitas jangka panjang.

## 5.2 Saran

Pada pemilihan arsitektur hendaknya disesuaikan dengan kebutuhan dan konteks proyek, termasuk skala aplikasi, tingkat kompleksitas, platform yang ditargetkan, serta pengalaman tim pengembang. Tidak ada satu arsitektur yang paling benar untuk semua jenis proyek, sehingga fleksibilitas dalam pengambilan keputusan sangat penting. Pada penelitian berikutnya, pengujian dapat dilakukan

pada dampak arsitektur terhadap performa UI/UX pada single code base di berbagai platform, seperti Android, iOS, Web, dan desktop, untuk menilai sejauh mana struktur arsitektur memengaruhi pengalaman pengguna secara nyata. Selain itu, dapat dilakukan pengujian aplikasi multiplatform di berbagai distribusi Linux non-GTK (seperti KDE berbasis Qt atau sistem ringan berbasis CLI), guna mengevaluasi portabilitas dan efisiensi aplikasi lintas sistem operasi dan lingkungan.