ANALISIS PERBANDIGAN PERFORMA *NETWORKING LIBRARY* HTTP DAN DIO PADA APLIKASI *MULTIPLATFORM* BERBASIS FLUTTER

PROPOSAL SKRIPSI

Disusun oleh:
Muhammad Abroor Milzam Radifan
225150707111058

PROGRAM STUDI TEKNOLOGI INFORMASI
DEPARTEMEN SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2025

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pengembangan aplikasi mobile *multiplatform* menjadi kebutuhan penting di era digital karena pengguna mengakses informasi melalui berbagai perangkat dan sistem operasi. Pendekatan *multiplatform* memungkinkan penulisan satu basis kode yang dapat berjalan di Android, iOS, dan web, sehingga mempercepat pengembangan dan menekan biaya (Mohammed & Ameen, 2022). Hal ini mendorong banyak perusahaan dan pengembang beralih pada solusi *multiplatform* untuk memenuhi tuntutan pasar yang semakin dinamis.

Di Indonesia, kebutuhan pengembangan aplikasi mobile semakin relevan dengan tingginya pengguna smartphone. Indonesia menempati peringkat keempat dunia dalam jumlah pengguna smartphone aktif, dengan estimasi 187,7 juta pengguna dari populasi sekitar 275,5 juta jiwa pada pertengahan 2025, atau penetrasi sekitar 68,1%. Selain itu, diprediksi jumlah pengguna smartphone di Indonesia akan terus meningkat hingga mencapai 258,77 juta pada tahun 2029 (Kompas.com, 2025; Radar Kudus, 2025). Fakta ini menandakan bahwa permintaan aplikasi mobile akan semakin besar sehingga solusi *multiplatform* menjadi penting untuk mempercepat produksi aplikasi lintas platform.

Flutter adalah framework open-source yang dikembangkan oleh Google untuk mendukung pengembangan aplikasi *multiplatform* menggunakan bahasa Dart. Flutter memiliki mesin rendering sendiri sehingga tampilan antarmuka konsisten dan responsif pada berbagai platform. Fitur hot reload juga mempermudah proses pengujian dan debugging, membuat iterasi pengembangan lebih cepat (Shukla et al., 2024). Popularitas Flutter terus meningkat karena fleksibilitasnya dalam membangun aplikasi *multiplatform* dengan performa tinggi.

Sebagai sarana pendukung dalam pengembangan aplikasi *multiplatform*, *networking library* memiliki peran penting dalam memastikan komunikasi aplikasi dan server berlangsung cepat dan efisien. Pemilihan library yang kurang tepat dapat menyebabkan aplikasi lambat, boros memori, dan tidak hemat jaringan. Di ekosistem Flutter, dua library populer adalah http, yang menawarkan API sederhana dengan overhead minimal (Dart Team, 2024), dan dio, yang lebih kaya fitur dengan dukungan interceptors, cancel tokens, multipart/form-data, dan parallel request (Flutter Chine, 2024). Fitur tambahan pada dio memberikan fleksibilitas tinggi tetapi berpotensi menambah overhead jika tidak dioptimalkan. Http cenderung lebih ringan dan cocok untuk permintaan sederhana, sedangkan dio menawarkan kontrol lebih baik untuk komunikasi data kompleks (Shukla et al., 2024).

Penelitian terdahulu telah banyak dilakukan untuk membandingkan performa *networking library*, namun sebagian besar berfokus pada ekosistem Android native. Penelitian yang dilakukan oleh Santoso et al. (2025), menguji

empat library (Retrofit, Ktor Client, OkHttp, dan Volley) dan menemukan bahwa setiap library memiliki keunggulan pada metrik yang berbeda; OkHttp unggul pada response time, sementara Retrofit lebih efisien dalam penggunaan memori. Serupa dengan itu, penelitian oleh Islam et al. (2025) yang membandingkan Retrofit dan Ktor juga menemukan bahwa Retrofit lebih unggul dalam efisiensi memori. Meskipun penelitian-penelitian tersebut memberikan landasan yang kuat, belum ada pengujian mendalam yang secara spesifik membandingkan *networking library* pada konteks pengembangan aplikasi *multiplatform* menggunakan Flutter, khususnya antara library http dan dio. Hal ini menimbulkan pertanyaan fundamental: apakah fitur-fitur canggih pada dio benar-benar memberikan keunggulan performa dibandingkan kesederhanaan http, atau justru menambah beban komputasi pada aplikasi *multiplatform*?

Dengan meningkatnya adopsi Flutter, diperlukan penelitian yang menilai performa http dan dio secara langsung agar pengembang dapat memilih library yang tepat berdasarkan data, bukan sekadar popularitas. Hasil penelitian ini diharapkan menjadi acuan bagi developer untuk menghasilkan aplikasi *multiplatform* yang responsif, efisien, dan memberikan pengalaman pengguna yang optimal.

1.2 Rumusan Masalah

1. Apakah terdapat perbedaan yang signifikan pada waktu respons (response time) antara library http dan dio dalam aplikasi Flutter
2. Apakah terdapat perbedaan yang signifikan pada penggunaan memori (memory usage) antara library http dan dio dalam aplikasi Flutter?
3. Apakah terdapat perbedaan yang signifikan pada penggunaan CPU (CPU usage) antara library http dan dio dalam aplikasi Flutter?
4. Library manakah yang memiliki kinerja paling optimal secara keseluruhan berdasarkan hasil pengujian?

1.3 Tujuan

1. Menganalisis dan mengukur perbedaan waktu respons (response time) antara library http dan dio.
2. Menganalisis dan mengukur perbedaan penggunaan memori (memory usage) antara library http dan dio.
3. Menganalisis dan mengukur perbedaan penggunaan CPU (CPU usage) antara library http dan dio.
4. Menentukan *networking library* yang paling optimal berdasarkan perbandingan metrik performa yang diuji.

1.4 Manfaat

Melalui penelitian yang dilakukan, diharapkan dapat memberikan manfaat diantaranya sebagai berikut :

1.4.1 Manfaat Praktis (Bagi Pengembang)

Memberikan rekomendasi teknis berbasis bukti untuk pemilihan *networking library* di Flutter, sehingga dapat membantu pengembang dalam menentukan penggunaan *networking library* sesuai dengan kebutuhan dan meningkatkan efisiensi serta performa aplikasi yang dikembangkan.

1.4.2 Manfaat Teoritis (Bagi Akademisi)

Hasil penelitian ini diharapkan dapat menambah khazanah keilmuan di bidang rekayasa perangkat lunak, khususnya terkait analisis perbandingan performa pada teknologi pengembangan aplikasi *multiplatform*. Penelitian ini juga dapat menjadi referensi dan dasar bagi penelitian selanjutnya yang membahas topik serupa.

1.4.3 Manfaat Bagi Penulis

Memenuhi salah satu syarat kelulusan program sarjana, sebagai wadah untuk memperdalam pemahaman dan menggali informasi tentang manajemen komunikasi data pada aplikasi mobile, serta melatih kemampuan berpikir kritis dan analitis, kemampuan teknis dan kemampuan dalam memecahkan masalah

1.5 Batasan Masalah

Penelitian ini difokuskan pada perbandingan performa dua *networking library* http dan dio dengan batasan masalah sebagai berikut :

1. Pengujian dilakukan dalam lingkungan pengembangan Flutter pada versi tertentu
2. Metrik performa yang diukur terbatas pada response time, memory usage, dan CPU usage.
3. Aplikasi yang digunakan adalah aplikasi prototipe yang dibuat khusus untuk tujuan pengujian.
4. Pengujian tidak mencakup aspek keamanan, kemudahan penggunaan (usability) library, atau fitur-fitur spesifik di luar pengukuran performa.

1.6 Sistematika Pembahasan

Pada penelitian yang dilakukan, terdapat sistematika pembahasan sebagai berikut :

1. **BAB 1 PENDAHULUAN**

   Bab 1 menjelaskan latar belakang masalah dari penelitian yang dilakukan, rumusan masalah, tujuan penelitian, dan manfaat. Pada pendahuluan juga terdapat batasan masalah dari penelitian.

2. **BAB 2 LANDASAN KEPUSTAKAAN**

   Bab ini menguraikan penelitian-penelitian terdahulu yang relevan sebagai acuan serta landasan teori yang mendukung penelitian, seperti

konsep pengembangan *multiplatform*, Flutter, komunikasi client-server, dan metrik performa.

3. **BAB 3 METODOLOGI**

   Bab ini menjelaskan langkah-langkah penelitian secara sistematis, mencakup jenis penelitian, perancangan skenario dan lingkungan pengujian, teknik pengumpulan data, serta teknik analisis data yang akan digunakan.

4. **BAB 4 HASIL DAN PEMBAHASAN**

   Bab ini menyajikan data hasil pengujian performa kedua library dalam bentuk tabel dan grafik, diikuti dengan analisis dan pembahasan mendalam terhadap hasil tersebut untuk menjawab rumusan masalah.

5. **BAB 5 KESIMPULAN DAN SARAN**

   Bab ini berisi rangkuman dari seluruh hasil penelitian dan pembahasan untuk menarik kesimpulan akhir, serta memberikan saran bagi pengembang maupun untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menguraikan landasan teori yang mendasari penelitian serta membangingkan penelitian terdahulu yang relevan. Pembahasan diawali dengan kajian pustaka, dilanjutkan dengan penjelasan tentang konsep pengembangan aplikasi *multiplatform*, teknologi Flutter dan Dart, komunikasi client-server, fungsi *networking library*, metrik pengukuran performa yang diterapkan serta metode statistik yang digunakan untuk mengolah dan menganalisis data dalam penelitian ini

2.1 Kajian Pustaka

Berikut adalah kajian pustaka yang berisi tinjauan terhadap penelitian-penelitian sebelumnya sebagai referensi dan pendukung dalam melaksanakan penelitian.

Penelitian pertama dilakukan oleh Santoso, et al. (2025) dengan judul "Analisis Performa *Networking Library* Retrofit, Ktor Client, Okhttp Dan Volley Pada Pengembangan Aplikasi Perangkat Bergerak Berbasis Android". Pada penelitian ini, dilakukan perbandingan performa terhadap empat *networking library* populer di lingkungan Android native. Pengujian dilakukan dengan membangun aplikasi prototipe untuk mengukur metrik performa seperti response time, memory usage, dan network usage pada skenario pengambilan data hingga 10.000 record. Hasil dari penelitian ini menunjukkan bahwa tidak ada satu library yang unggul di semua aspek; OkHttp menunjukkan response time tercepat, Retrofit paling efisien dalam penggunaan memori, dan Ktor Client unggul pada penggunaan data jaringan yang dikirim.

Penelitian kedua dilakukan oleh Islam, et al. (2025) dengan judul "Analisis Perbandingan Kinerja Komunikasi API Menggunakan Retrofit dan Ktor di Aplikasi Android (Studi Kasus: Aplikasi Katalog Film)". Penelitian ini secara spesifik membandingkan dua *networking library*, yaitu Retrofit dan Ktor. Metrik yang diukur meliputi response time, CPU usage, dan memory usage. Analisis data dilakukan secara statistik menggunakan Uji-t untuk menentukan signifikansi perbedaan performa. Kesimpulan dari penelitian ini adalah Retrofit terbukti secara signifikan lebih unggul dalam hal efisiensi penggunaan memori, sementara untuk response time dan CPU usage, tidak ditemukanf perbedaan yang signifikan secara statistik antara kedua library tersebut.

Untuk mempermudah pemahaman posisi penelitian ini terhadap penelitian-penelitian sebelumnya, berikut disajikan tabel perbandingan.

Tabel 2.1 Landasan Kepustakaan

No	Persamaan	Perbedaan	
		Penelitian Terdahulu	Rencana Penelitian
1.	• Pengujian dilakukan pada aplikasi prototype yang dirancang khusus untuk penelitian • Metrik pengukuran performa serupa (response time dan memory usage)	• Dilakukan pada platform Android • Membandingkan empat *networking library* (Retrofit, Ktor, OkHttp, dan Volley)	• Dilakukan pada platform Android dan iOS (*multiplatform*) • Hanya membandingkan dua *networking library* (http dan dio)
2.	• Fokus pada perbandingan dua *networking library* • Metrik pengukuran performa sama (response time, memory usage, CPU usage) • Menggunakan metode statistik Uji-t untuk analisis data	• Dilakukan pada platform Android • Membandingkan dua *networking library* (Retrofit dan Ktor)	• Dilakukan pada platform Android dan iOS (*multiplatform*) • Membandingkan dua *networking library* (http dan dio)

2.2 Aplikasi Mobile *Multiplatform*

Pengembangan Aplikasi mobile *multiplatform* adalah sebuah pendekatan dalam rekayasa perangkat lunak yang memungkinkan pengembang untuk menulis satu basis kode (single codebase) dan menjalankannya di berbagai sistem operasi seperti Android dan iOS secara bersamaan. Pendekatan ini memberikan efisiensi tinggi dibandingkan pengembangan native, dimana pengembang harus membangun dan memelihara aplikasi secara terpisah untuk masing-masing platform (Jangassiyev et al., 2024).

2.3 Flutter

Flutter adalah sebuah UI toolkit bersifat open-source yang dikembangkan oleh Google untuk membangun aplikasi yang indah dan dikompilasi secara native dari satu basis kode. Flutter tidak menggunakan kompunen UI bawaan (native widget) dari sistem operasi, melainkan menggunakan mesin rendering grafisnya sendiri yang memiliki performa tinggi bernama Skia untuk menggambar setiap

piksel dilayar (Grönlund, 2023). Hal ini memastikan tampilan aplikasi akan selalu konsisten diberbagai versi sistem operasi dan perangkat.

2.4 Dart

Dart adalah bahasa pemrograman yang menjadi fondasi dari Flutter. Dikembangkan oleh Google, Dart dioptimalkan untuk membangun aplikasi client yang cepat diberbagai platform. Dart memiliki fitur-fitur modern seperti sound null safety untuk mengurari error dan sistem garbage collection yang efisien untuk manajemen memori. Salah satu keunggulan teknis utama yang disediakan oleh Dart adalah kemampuannya untuk dikompilasi dengan dua cara: pertama adalah Just In Time (JIT) Compilation yang memungkinkan fitur Hot Reload dengan mengkompilasi kode secara cepat saat dibutuhkan, kedua adalah Ahead Of Time (AOT) Compilation yang memungkinkan kode Dart dikompilasi langsung menjadi kode native (ARM atau x86). Proses AOT ini dapat memastikan aplikasi Flutter dapat berjalan dengan performa mendekati aplikasi native (Google, n.d.).

2.5 Komunikasi Client-Server

Komunikasi client-server adalah model arsitektur di mana aplikasi pengguna (client) meminta layanan atau data dari program pusat (server). Dalam konteks aplikasi mobile, aplikasi mengirimkan permintaan (request) melalui protokol HTTP ke server, yang kemudian memproses dan mengirimkan kembali respons (response). Model ini menjadi dasar bagi aplikasi modern untuk menampilkan konten dinamis.

2.6 *Application Programming Interface* (API)

Application Programming Interface (API) adalah perantara yang mendefinisikan aturan dan protokol komunikasi antara dua aplikasi. RESTful API adalah standar arsitektur populer yang menggunakan metode HTTP dan format data JSON untuk pertukaran informasi yang efisien antara client dan server (Ehsan et al., 2022).

2.7 *Networking Library*

Networking library adalah kumpulan kode siap pakai yang menyederhanakan tugas komunikasi jaringan. Penggunaannya dapat mempercepat proses pengembangan, mengurangi potensi bug, dan menyediakan fitur-fitur canggih seperti manajemen error dan caching (Lachgar et al., 2018).

2.8 *Networking Library* http

Library http adalah pustaka jaringan resmi yang dikelola dan dikembangkan langsung oleh tim pengembang Dart, menjadikannya sebagai fondasi untuk komunikasi HTTP dalam ekosistem Dart dan Flutter. Library http berpusat pada kesederhanaan, control low-level, dan dependensi minimal. Samahalnya seperti *networking library* pada umumnya, http menyediakan fungsi-fungsi seperti get(), post(), put(), dan delete() yang bekerja secara asynchronous.

Keunggulan utama dari http adalah sifatnya yang ringan dan pengembang memiliki control penuh untuk setiap aspek request seperti penyusunan header hingga proses parsing data response secara manual. Dibalik sisi kesederhanaan library http, library ini tidak menyediakan fitur-fitur canggih secara bawaan seperti konfigurasai global (base URL, timeout), interceptors, atau pembatalan request (Dart Team, 2024).

2.9 *Networking Library* dio

Library Dio adalah klien HTTP populer di komunitas Flutter yang dirancang untuk menangani skenario jaringan kompleks dan meningkatkan produktivitas pengembang. Berbeda dari package http, Dio menawarkan fitur canggih seperti konfigurasi global untuk mengatur baseUrl, headers, dan connectTimeout pada satu instance untuk semua permintaan jaringan. Fitur interceptors memungkinkan pengembang mencegat, memodifikasi, atau membatalkan request dan response, sering digunakan untuk logging, otentikasi seperti token JWT, atau penanganan error terpusat. Dio juga menyediakan manajemen error terstruktur melalui objek DioError yang memberikan detail jelas tentang jenis error, request, dan response. Selain itu, Dio mendukung pembatalan request dengan CancelToken, pengunggahan berkas menggunakan FormData, serta pengunduhan berkas dengan pelacakan progres. Dengan kelengkapan fitur yang dibawakan, dio sangat cocok untuk aplikasi berskala menengah hingga besar yang membutuhkan arsitektur jaringan yang tangguh dan dapat dipelihara dengan mudah (Flutter China, 2024).

2.10 Metrik Pengukuran Performa

Metrik pengukuran performa digunakan untuk mengevaluasi performa *networking library* antara http dengan dio. Metrik seperti kecepatan waktu respon, pengunaan memori, dan penggunaan CPU menjadi indicator utama untuk mengukur tingkat performa yang dihasilkan dari masing-masing *networking library*. Berikut merupakan penjelasan mengenani masing-masing metrik secara singkat.

2.10.1 Kecepatan Waktu Response (Response Time)

Response time adalah durasi total sejak permintaan dikirim hingga respons diterima. Metrik ini secara langsung memengaruhi persepsi kecepatan aplikasi oleh pengguna dan menjadi faktor krusial bagi pengalaman pengguna (Zhou et al., 2016).

2.10.2 Penggunaan Memori (Memory Usage)

Penggunaan memori merujuk pada jumlah RAM yang dialokasikan aplikasi selama operasi jaringan. Efisiensi memori sangat penting untuk menjaga aplikasi tetap responsif, terutama pada perangkat dengan sumber daya terbatas (Ghag, 2024).

2.10.3 Penggunaan CPU (CPU Usage)

Penggunaan CPU mengukur beban kerja pada prosesor perangkat. Penggunaan CPU yang tinggi dapat menguras daya baterai lebih cepat dan menurunkan performa aplikasi secara keseluruhan (Altex & Ståhl, 2020).

2.11 Uji-T

Uji-t Sampel Independen (Independent Samples t-Test) adalah sebuah metode statistik inferensial yang digunakan untuk menentukan apakah terdapat perbedaan yang signifikan secara statistik antara nilai rata-rata (mean) dari dua kelompok data yang tidak saling berhubungan (independen). Dalam konteks penelitian ini, "dua kelompok" tersebut adalah hasil pengukuran performa dari library http dan dio (Kim, 2015). Tujuan utama dari uji ini adalah untuk menguji hipotesis nol (H0) yang menyatakan bahwa tidak ada perbedaan nyata antara rata-rata kedua kelompok. Perhitungan statistik-t dilakukan menggunakan rumus berikut:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Keterangan:

- t = Nilai t-hitung
- $\bar{x}_1$ = Rata-rata kelompok 1
- $\bar{x}_2$ = Rata-rata kelompok 2
- s_1^2 = Varians kelompok 1
- s_2^2 = Varians kelompok 2
- n_1 = Jumlah sampel kelompok 1
- n_2 = Jumlah sampel kelompok 2

BAB 3 METODOLOGI

Gambar 3.1 Metodologi Penelitian

Bab ini menguraikan secara rinci kerangka kerja dan langkah-langkah sistematis yang akan dilaksanakan dalam penelitian. Metodologi ini dirancang untuk memastikan proses perbandingan performa *networking library* http dan dio pada aplikasi Flutter dilakukan secara objektif, terkontrol, dan dapat direplikasi. Alur penelitian secara garis besar digambarkan pada Gambar 3.1, yang dimulai dari identifikasi masalah hingga penarikan kesimpulan.

3.1 Identifikasi Permasalahan

Tahap awal penelitian ini adalah mengidentifikasi masalah utama, yaitu adanya kebutuhan bagi pengembang aplikasi Flutter untuk memahami perbedaan performa antara dua *networking library* yang paling populer, http dan dio. Masalah ini muncul karena belum ada studi komparatif berbasis data yang spesifik untuk lingkungan Flutter, sehingga pemilihan library sering kali didasarkan pada popularitas atau kelengkapan fitur tanpa mempertimbangkan dampaknya terhadap performa aplikasi.

3.2 Studi Literatur

Pada tahap ini, dilakukan pengumpulan dan pengkajian teori dari berbagai sumber literatur yang relevan. Studi ini mencakup buku, jurnal ilmiah, dokumentasi resmi, dan artikel teknis yang membahas konsep pengembangan aplikasi *multiplatform*, arsitektur Flutter, komunikasi client-server, serta penelitian terdahulu yang berkaitan dengan analisis performa aplikasi mobile, seperti penelitian oleh Santoso et al. (2025) dan Islam, et al. (2025).

3.3 Perancangan Skenario Pengujian

Tahap ini berfokus pada perancangan lingkungan dan skenario pengujian yang terstruktur untuk memastikan validitas dan reliabilitas data yang akan dikumpulkan. Perancangan skenario pengujian terbagi menjadi 2 tahapan utama :

3.3.1 Lingkungan Pengujian

Untuk menjaga konsistensi dan meminimalkan variabel eksternal, seluruh pengujian akan dilaksanakan dalam lingkungan yang terkontrol dengan spesifikasi sebagai berikut:

3.3.1.1 Perangkat Uji

Pengujian akan dilakukan pada perangkat virtual dengan spesifikasi sebagai berikut:

- Android : Emulator Android versi 15, dirilis pada tahun 2024 dan dijalankan melalui Android Studio
- iOS : Simulator iOS versi 18.0, dirilis pada tahun 2024 dan dijalankan melalui Xcode

3.3.1.2 Perangkat Lunak

perangkat lunak akan distandarisasi untuk menghindari inkonsistensi, meliputi Flutter SDK versi 3.24.0, Dart SDK versi 3.5.0, serta library http versi 1.2.2 dan dio versi 5.7.0.

3.3.1.3 Koneksi Jaringan

Pengujian akan memanfaatkan koneksi jaringan Wi-Fi Universitas Brawijaya yang dianggap memiliki latensi dan kecepatan yang lebih stabil dibandingkan koneksi internet publik.

3.3.1.4 Server API

Akan digunakan dummy API publik dari JSONPlaceholder yang menyediakan endpoint RESTful yang konsisten untuk berbagai operasi HTTP.

3.3.2 Skenario Pengujian

Skenario pengujian dirancang untuk mesimulasikan dua operasi HTTP yang umum digunakan pada aplikasi mobile:

3.3.2.1 Skenario Pengambilan Data

Melalukan permintaan HTTP GET untuk mengambil berbagai volume data, mulai dari data teks sederhana hingga data yang lebih besar.

3.3.2.2 Skenario Pengiriman Data

Melakukan permintaan HTTP POST untuk mengirimkan sebuah objek data JSON sederhana ke server.

3.3.2.3 Perulangan

Setiap skenario pengujian akan diulang sebanyak 20 kali untuk masing-masing library pada setiap platform (Android dan iOS).

3.4 Perancangan Aplikasi

Akan dirancang dan dibangun sebuah aplikasi prototipe sederhana menggunakan Flutter yang berfungsi murni sebagai alat uji. Arsitektur aplikasi akan dirancang secara modular dengan penekanan pada pemisahan logika jaringan (network layer). Secara spesifik, akan disiapkan struktur direktori yang berbeda untuk setiap implementasi networking library. Hal ini bertujuan agar kode untuk http dan dio sepenuhnya terisolasi dan tidak saling memengaruhi, sehingga memastikan perbandingan yang adil. Tampilan antarmuka aplikasi akan dibuat minimalis, hanya berisi elemen-elemen yang diperlukan untuk memicu skenario pengujian dan menampilkan hasil pengukuran.

3.5 Implementasi

Tahapan implementasi akan dibagi menjadi 3 tahapan utama :

3.5.1 Implementasi Penyediaan API

Tahapan ini digunakan untuk memastikan semua endpoint yang dibutuhkan dari JSONPlaceholder sudah siap untuk digunakan serta memahami struktur response yang diberikan untuk mempermudah proses parsing data.

3.5.2 Implementasi Aplikasi Uji

Mengembangkan aplikasi prototype Flutter, pembuatan logika dasar untuk pengujian serta tampilan aplikasi akan dilakukan pada tahapan ini.

3.5.3 Implementasi *Networking Library*

Mengintegrasikan library http dan dio ke dalam aplikasi. Untuk memastikan perbandingan yang adil, akan dibuat dua service class yang terisolasi, di mana masing-masing class bertanggung jawab penuh untuk menangani permintaan jaringan menggunakan salah satu library.

3.6 Pengambilan Data dan Pengujian

Proses pengujian dan pengambilan data akan dilakukan dengan menjalankan aplikasi prototipe pada emulator Android dan simulator iOS secara bergantian. Setiap skenario akan dieksekusi sebanyak 20 kali. Pengukuran metrik performa akan dilakukan menggunakan alat bantu sebagai berikut :

3.6.1 Response Time

Waktu respons akan diukur secara presisi menggunakan class Stopwatch dari Dart, yang dimulai tepat sebelum pemanggilan fungsi jaringan dan dihentikan segera setelah respons diterima.

3.6.2 Memory Usage

Penggunaan memori akan dipantau dan direkam dengan menggunakan Flutter DevTools yang merupakan alat profiling resmi yang telah disediakah oleh tim Flutter. Data penggunaan memori nantinya akan diambil dari tab “Memory” yang menampilkan alokasi memori real-time.

3.6.3 CPU Usage

Penggunaan CPU juga akan dipantau dengan menggunakan Flutter DevTools. Data penggunaan CPU akan didapatkan dari tab “Performance” yang merekam beban kerja pada setiap thread selama eksekusi

3.7 Pengolahan dan Analisis Data

Setelah data terkumpul, tahap selanjutnya adalah pengolahan dan analisis. Data mentah dari 30 kali pengulangan akan ditabulasikan dalam perangkat lunak spreadsheet (seperti Microsoft Excel atau Google Sheets). Analisis akan mencakup statistik deskriptif untuk mendapatkan gambaran umum (rata-rata, median, standar deviasi) dan dilanjutkan dengan analisis statistik inferensial menggunakan metode Uji-t (t-test). Uji ini akan digunakan untuk menentukan apakah perbedaan performa rata-rata antara http dan dio signifikan secara statistik.

3.8 Kesimpulan

Tahap akhir dari metodologi ini adalah penarikan kesimpulan berdasarkan temuan dari analisis data. Kesimpulan akan secara langsung menjawab rumusan masalah, menyajikan perbandingan performa yang didukung oleh data empiris, dan memberikan rekomendasi praktis bagi pengembang Flutter dalam memilih *networking library* yang paling optimal sesuai dengan kebutuhan performa aplikasi.

DAFTAR REFERENSI

Mohammed, D. Y. & Ameen, S. Y. 2022. Developing Cross-Platform Library Using Flutter. European Journal of Engineering and Technology Research, 7(2), pp. 18-21.

Shukla, P., Tyagi, N., Deepanshu, Agarwal, M. & Jain, S. 2024. Development of Apps Industry using Flutter: A Review. Quest Journals Journal of Software Engineering and Simulation, 10(6), pp. 11-19.

Santoso, D. A., Pramono, D. & Pinandito, A. 2025. Analisis Performa Networking Library Retrofit, Ktor Client, Okhttp Dan Volley Pada Pengembangan Aplikasi Perangkat Bergerak Berbasis Android. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 9(5).

Islam, M. S., Kharisma, A. P. & Adikara, P. P. 2025. Analisis Perbandingan Kinerja Komunikasi API Menggunakan Retrofit dan Ktor di Aplikasi Android (Studi Kasus: Aplikasi Katalog Film). Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 9(9).

Dart Team. 2024. *http package*. pub.dev.

Flutter China. 2024. dio package. pub.dev.

Kompas.com. 2025. Indonesia Masuk 5 Besar Pengguna Smartphone Terbanyak di Dunia.

Radar Kudus. 2025. Pasar Smartphone Indonesia Melonjak, Pengguna Diprediksi Capai 258 Juta pada 2029.

Jangassiyev, R., Umarova, Z., Ussenova, A., Makhanova, Z., Zhumatayev, N., Amirov, M. & Koishibekova, G. 2024. Comparative analysis of cross-platform development methodologies: a comprehensive study. TELKOMNIKA (Telecommunication Computing Electronics and Control), 22(1), pp. 235-244.

Grönlund, P. 2023. A comparison study between mobile cross platform frameworks Flutter and React Native. Tesis, KTH Royal Institute of Technology.

Google. n.d.. Dart overview. Dart documentation.

Ehsan, A., Abuhaliqa, M. A. M. E., Catal, C. & Mishra, D. 2022. RESTful API testing methodologies: Rationale, challenges, and solution directions. Applied Sciences, 12(9), 4369.

Lachgar, M., Benouda, H. & Elfirdoussi, S. 2018. Android Rest Apis: Volley vs retrofit. In 2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), pp. 1-6.

Zhou, R., Shao, S., Li, W. & Zhou, L. 2016. How to define the user's tolerance of response time in using mobile applications. In 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 281-285.

Ghag, O. M. 2024. Optimizing app memory usage in Android smartphones. International Journal of Science and Research (IJSR), 13(1), pp. 464-466.

Altex, N. & Ståhl, A. 2020. Cross-platform Framework Comparison: A case study on Flutter and React Native. Tesis, Blekinge Institute of Technology.

Kim, T. K. 2015. T test as a parametric statistic. *Korean Journal of Anesthesiology*, 68(6), pp. 540–546.