BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Berdasarkan pada teori dan penelitian terdahulu, merupakan hal yang perlu dijadikan acuan dalam pembuatan penelitian analisis penggunaan cpu dan memori pada library image loader Glide, Picasso, dan Coil dalam implementasi studi kasus 'MovieApp' Android. Sebagai bahan pertimbangan dalam penelitian ini dicantumkan beberapa hasil penelitian terdahulu oleh beberapa peneliti yang pernah diketahui, diantaranya:

1. Analisis Performa Image Loading Library Glide dan Picasso pada Aplikasi MovieApp

   Bugi Pradana Nugroho, Muhammad Aminul Akbar, dan Aryo Prinandito (2023) melakukan penelitian dengan judul "Analisis Performa Image Loading Glide dan Picasso pada Aplikasi MovieApp". Penelitian tersebut melakukan perbandingan dua pustaka pemuatan gambar yaitu Glide dan Picasso dalam implementasi aplikasi berbasis Android. Pengujian dilakukan dengan menggunakan tiga parameter utama yaitu, memory usage, CPU utilization, dan rendering time. Metodologi penelitian yang digunakan terdiri dari tiga tahap utama yaitu pembuatan rancangan pengujian, pelaksanaan pengujian dan analisis, serta pengambilan kesimpulan. Penelitian tersebut menggunakan arsitektur MVVM (Model-View-ViewModel) dalam pengembangan aplikasinya dan melakukan 30 kali pengujian untuk setiap parameter. Hasil penelitian menunjukkan bahwa library Glide memiliki performa yang lebih baik dibandingkan Picasso dalam semua aspek pengujian yang dilakukan. Pada penggunaan memori, Glide mengkonsumsi rata-rata 73.63 MB sedangkan Picasso 130.22 MB. Untuk utilisasi CPU, Glide menggunakan 4.68% sementara Picasso 5.65%. Pada aspek rendering time, Glide membutuhkan waktu 6.30 ms sedangkan Picasso 7.06 ms.

2. Android Application Memory and Energy Performance: Systematic Literature Review

   (Degu, 2019) melakukan penelitian dengan judul "Android Application Memory and Energy Performance: Systematic Literature Review". Penelitian ini merupakan tinjauan literatur sistematis yang bertujuan untuk mengkaji studi-studi yang telah ada terkait peningkatan performa aplikasi Android dari segi penggunaan memori dan energi. Dalam penelitian ini, sebanyak 31 studi empiris dipetakan dan diklasifikasikan ke dalam lima sub-kategori utama, yaitu performance testing, resource leak, memory, energy, dan general performance. Penelitian ini menemukan bahwa

penggunaan sumber daya yang tidak efisien, khususnya memori, merupakan salah satu penyebab utama *bottleneck* dalam performa aplikasi Android. Selain itu, masalah resouce leak sering kali menyebabkan aplikasi mengalami crash atau penurunan performa, seperti lambatnya respons aplikasi dan konsumsi energi yang boros. Studi ini menyoroti pentingnya pengelolaan memori yang lebih efisien dalam pengembangan aplikasi, serta menyarankan perlunya teknik pengujian performa yang lebih baik untuk mendeteksi masalah-masalah ini lebih awal dalam siklus pengembangan.

Meskipun penelitian ini tidak berfokus secara spesifik pada topik utama yaitu analisis perbandingan pustaka image loader, kesimpulan dari penelitian ini sangat relevan untuk mendukung penelitian saya yang bertujuan untuk menganalisis performa pustaka Glide, Picasso, dan Coil dalam pemuatan gambar pada aplikasi Android. Penelitian ini memberikan landasan teoritis tentang pentingnya optimasi penggunaan memori dan sumber daya untuk meningkatkan efisiensi aplikasi, yang menajdi fokus utamma dalam pengujian performa pustaka image loader yang akan saya lakukan.

3. Analisis Perbandingan Konsumsi Daya Library Image Loader pada Android Aplikasi Media Sosial

Muuhammad Abyan Safitra, Mahardeka Tri Ananta, dan Komang Candra Brata (2018) dalam jurnal "Analisis Perbandingan Konsumsi Daya Library Image Loader pada Android Aplikasi Media Sosial" melakukan penelitian terkair konsumsi daya dari tiga library pemuatan gambar yaitu Glide, Picasso, dan Universal Image Loader (UIL) pada aplikasi media sosial yang dibuat mirip dengan instagram. Penelitian ini menggunakan parameter konsumsi daya, penggunaan memori, dan CPU untuk mengukur performa masing-masing library. Hasil penelitian ini menunjukkan bahwa Glide memiliki konsumsi daya paling rendah dibandingkan dengan Picasso dan UIL, dengan rata-rata konsumsi daya sebesar 225,780 mW, sementara Picasso memiliki konsumsi daya tertinggi dengan rata-rata 480,316 mW. Selain itu, ukuran gambar yang diuji (100KB, 1MB, dan 3MB) juga mempengaruhi tingkat konsumsi daya pada Picasso dan UIL, tetapi tidak banyak mempengaruhi Glide.

2.2 Landasan Teori

2.2.1 Sistem Operasi

Sistem Operasi adalah lapisan pertama yang diproses oleh komputer ketika dinyalakan, dan berfungsi sebagai platorm bagi aplikasi-aplikasi lainnya. Pada saat komputer dinyalakan, Sistem Operasi dipasang pada memori utama sebagai software dasar yang mengelola sumber daya sistem secara efektif. Setelah sistem Operasi aktif, aplikasi-aplikasi lain dapat dijalankan karena meraka tidak perlu melakukan tugas-tugas inti seperti akses ke disk, manajemen memori, skeduling task, serta antarmuka pengguna sendiri. Sistem Operasi bertindak sebagai pengelola sumber daya dan bila diperlukan akan mengalokasikannya ke program-program tertentu (Wicaksana & Rachman, 2018).

Sistem Operasi menurut Rana (2023), "An Operating System (OS) is system software that acts as an interface or inherits the communication between end-user and system. An operating system is an essential part of a computer system that helps the user to run various other software."

Sistem operasi memiliki beberapa jenis diantaranya, yaitu Microsoft Windows, macOS, Linux, Unix, Chrome OS, dan Android. Pada penelitian ini berfokus pada sistem operasi Android, yang mana akan dijelaskan pada sub bab 2.2.2.

2.2.2 Android

Android merupakan sebuah sistem operasi mobile yang berbasiskan pada kernal Linux. Sistem operasi ini pertama kali dikembangkan oleh Android.Inc. Nama perusahaan yang kini akhirnya digunakan sebagai nama proyek pengembangan sistem operasi mobile. Android merupakan platform perangkat lunak yang dirancang khusus untuk perangkat mobile, seperti smartphone dan tablet, yang terdiri dari sistem operasi, middleware, dan berbagai aplikasi inti. Sistem operasi Android dikembangkan berbasis kernel Linux, yang awalnya menggunakan versi 2.6, dan dibuat untuk mendukung perangkat layar sentuh. Dengan arsitektur yang terbagi ke dalam beberapa lapisan utama, Android memastikan interaksi yang efisien antara perangkat keras dan perangkat lunak.

Pada tahun 2005 Google mengakuisisi Android.Inc yang menjadikkan sebagai anak perusahaan. Hal ini dilakukan sebagai strategi untuk memasuki pasar mobile sehingga Google dapat mengambil alih proses pengembangan serta tim pengembang Android. Google memutuskan Android menjadi sistem operasi Open Source dan gratis.

Kebanyakan kode Android dirilis di bawah lisensi Open Source Apache yang berarti setiap orang bebas untuk menggunakan dan mengunduh source code Android secara penuh (Open Handset Alliance, 2020).

Hal ini dapat dilihat dengan begitu banyaknya vendor yang menggunakan, bahkan melakukan modifikasi sistem operasi Android untuk disesuaikan pada perangkat buatan masing-masing pihak ketiga. Sehingga jika melihat pasar smartphone saat ini para pihak ketiga yang menggunakan sistem operasi Android memliki ciri khas masing-masing baik dari tampilan maupun fitur (Hadi Winata, Rivaldo 2022).

2.2.3 Arsitektur Android

Arsitektur Android terdiri dari beberapa lapisan utama, yaitu Linux Kernel, Hardware Abstraction Layer (HAL), Native C/C++ Libraries & Android Runtime, Application Framework, dan Aplikasi. Lapisan-lapisan ini bekerja sama untuk menjalankan sistem Android yang efisien dan aman.

Lapisan pertama adalah Linux Kernel, yang menjadi pondasi Android karena berbasis pada kernel linux. Kernel ini menyediakan driver, manajemen daya, pengelolaan memori, dan berbagai fungsi dasar lainnnya. Dengan menggunakan Linux Kernel, Android memanfaatkan fitur keamanan yang telah diterapkan pada kernal tersebut (Meng et al., 2018).

Pada lapisan kedua adalah Hardware Abstraction Layer (HAL) yang berfungsi sebagai penghubung antara perangkat keras dan perangkat lunak. HAL memungkinkan API Framework di lapisan aplikasi untuk mengakses komponen perangkat keras, seperti sensor, kamera, bluetooth, dan audio (Meng et al., 2018).

Lapisan ketiga adalah Native C/C++ Libraries & Android Runtime. Banyak komponen inti Android dibangun menggunakan kode asli, seperti sistem runtime dan HAL. Android Runtime (ART), yang menggantikan Dalvik Virtual Machine mulai dari Android versi 5.0, memungkinkan aplikasi berjalan pada prosesnya sendiri dan mengelola eksekusi bytecode dengan efisien melalui kompilasi ahead-of-tim dan just-in-time, yang memberikan peningkatan kinerja dan penghematan energi (Meng et al., 2018).

Lapisan keempat adalah Java API Framework, yang menyediakan berbagai fitur yang dapat digunakan oleh pengembang melalui API berbasis Java. Di sinilah pengelolaan tampilan dan antarmuka pengguna diatur, termasuk manajemen siklus

hidup aktivitas melalui Activity Manager. Aplikasi yang berjalan di Android berinteraksi langsung dengan framework ini untuk membuat antarmuka yang digunakan oleh pengguna (Meng et al., 2018).

Tampilan antarmuka pengguna dalam konteks pengembangan aplikasi saat ini menurut survei yang telah dijelaskan pada latar belakang penelitian ini, data masyarakat Indonesia lebih banyak mengakses aplikasi media sosial, yang mana aplikasi media sosial di dalamnya terdapat banyak gambar sebagai konten utamanya. Untuk optimasi pemuatan sumber daya, Android mendukung teknik pemuatan gambar yang efisien, menajamene konten melalui caching, kompresi sumber daya, serta pemuatan sumber daya secara dinamis. Hal ini memungkinkan aplikasi untuk mengelola sumber daya dengan lebih baik, mengurangi waktu pemuatan, dan meningkatkan efisiensi penggunaan memori. Oleh karena itu, setelah ini akan membahas lebih dalam mengenai pustaka pemuatan gambar atau library image loading pihak ketiga yang dalam konteks pengembangan aplikasi android.

2.2.4 Pustaka Pemuatan Gambar (Image Loading Libraries)

Pustaka pemuatan gambar merupakan komponen penting dalam pengembangan aplikasi, terutama aplikasi yang sering menampilkan gambar dari berbagai sumber. Pustaka pemuatan gambar dikembangkan untuk membantu pengembang mengatasi masalah ini, dengan menyediakan cara yang lebih mudah untuk diimplementasikan dan lebih efisien dalam memuat, menampilkan, dan mengelola gambar. Penggunaan pustaka pemuatan gambar dapat memberikan beberapa manfaat saat melakukan pengembangan aplikasi Android. (Mukherjee & Mondal, 2014) menekankan bahwa implementasi pustaka ini secara substansial mengurangi kompleksitas kode dan mempercepat proses pengembangan.

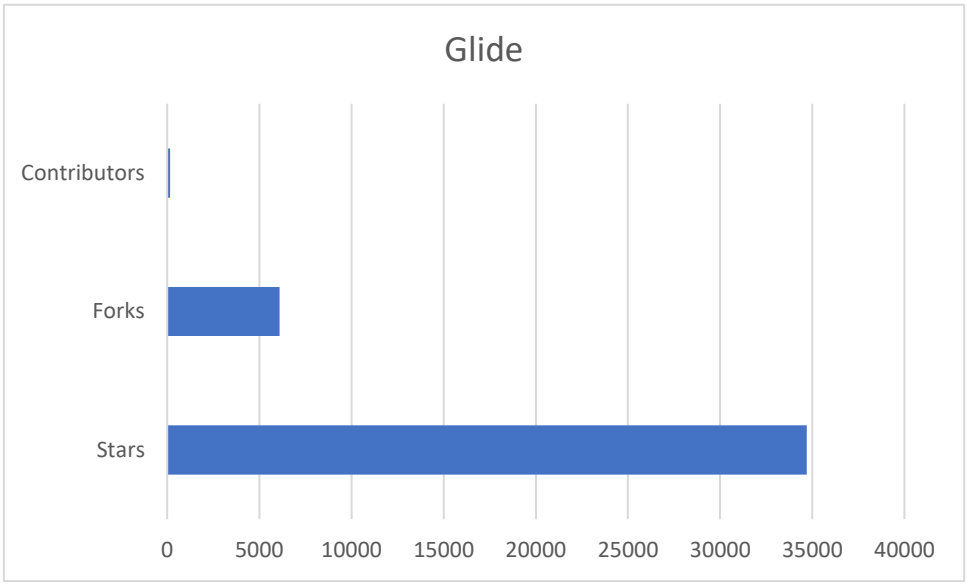
Fungsi utama dari pustaka ini antara lain, *caching* otomatis, di mana gambar yang telah dimuat disimpan di memori atau disk untuk mengurangi kebutuhan mengunduh gambar yang sama berulang kali, serta pemuatan asinkron yang dilakukan di luar thread utama untuk menjaga responsivitas antarmuka pengguna (UI) (Android Developers, n.d.).

2.2.5 Glide

Glide adalah salah satu pustaka pemuatan gambar yang populer di kalangan pengembang Android, dikembangkan oleh Bumptech. Glide pertama kali diperkenalkan untuk mengatasi masalah pemrosesan gambar yang lambat, terutama

dalam aplikasi dengan antarmuka yang memerlukan pemrosesan gambar secara dinamis dan cepat, seperti aplikasi media sosial. Glide dirancang untuk efisiensi memori dan kecepatan dalam pemrosesan gambar, dengan dukungan untuk pemuatan gambar dari berbagai sumber, termasuk internet, memori, atau file lokal.

Fitur utama Glide meliputi automatic caching, kompresi gambar yang cerdas, serta kemampuan untuk mengoptimalkan gambar berdasarkan resolusi perangkat. Selain itu, Glide mendukung GIF dan video stills, membuatnya lebih fleksibel dalam menangani berbagai format media. Sebuah studi oleh (Nugroho et al., 2023) menunjukkan bahwa Glide mengungguli pustaka lain seperti Picasso dalam hal kecepatan rendering dan penggunaan memori dalam pengujian aplikasi MovieApp, dengan Glide menggunakan memori rata-rata 73.63 MB dibandingkan dengan Picasso yang menggunakan 130.22 MB.



Gambar 2. 1 Data Glide

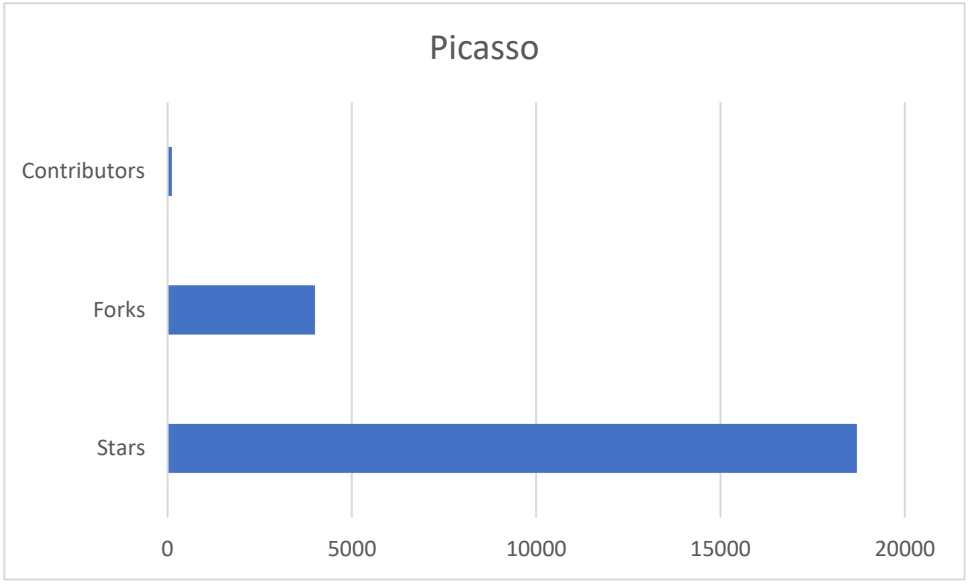
Gambar 2.1 merupakan data *stars, forks,* dan *contributors* berdasarkan laman GitHub resmi dari pustaka pemuatan gambar Glide. Yang mana memiliki jumlah bintang (34.7k stars), memiliki jumlah *fork* (6.1k forks), dan memiliki kontributor sebanyak 149 kontributor. Selanjutnya pustaka pemuatan gambar atau library image loading yang akan dibahas adalah Picasso.

2.2.6 Picasso

Picasso dikembangkan oleh Square dan dikenal sebagai pustaka pemuatan gambar yang mudah digunakan dan mendukung lazy loading, yang memungkinkan

gambar dimuat secara efisien tanpa membebani memori. Picasso mempermudah pengembang dengan menyederhanakan proses pemuatan gambar melalui sintaks yang intuitif dan pendekatan yang sederhana dalam penanganan caching serta pemrosesan gambar.

Picasso memiliki keunggulan dalam kemudahan integrasi dengan aplikasi Android, serta mendukung resize otomatis dan kompresi gambar, yang membantu pengembang dalam meminimalkan ukuran gambar untuk kebutuhan antarmuka pengguna yang spesifik. Studi oleh (Nugroho et al., 2023) dengan studi kasus sama seperti penelitian ini yaitu 'MovieApp' menemukan bawha rata-rata penggunaan memori Picasso mencapai 130,22MB, lebih tinggi daripada Glide yang hanya menggunakan 73,63 MB, sehingga Glide lebih optimal dalam penggunaan memori dan kecepatan rendering. Namun, Picasso masih tetap menjadi pilihan yang populer berkat dokumentasi yang kuat dan kemudahan penggunaannya.



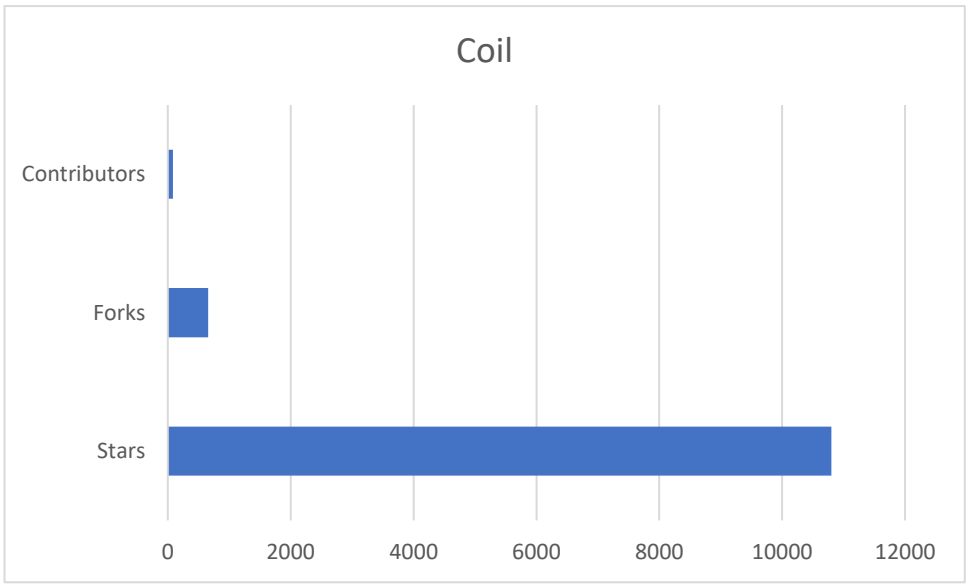
Gambar 2. 2 Data Picasso

Gambar 2.2 merupakan data *stars, forks,* dan *contributors* berdasarkan laman GitHub resmi dari pustaka pemuatan gambar Picasso. Yang mana memiliki jumlah bintang (18.7k stars), memiliki jumlah *fork* (4k forks), dan memiliki kontributor sebanyak 113 kontributor. Selanjutnya pustaka pemuatan gambar atau library image loading yang akan dibahas adalah Coil. Coil merupakan library baru yang menawarkan pendekatan modern dalam pemuatan gambar di Android.

2.2.7 Coil

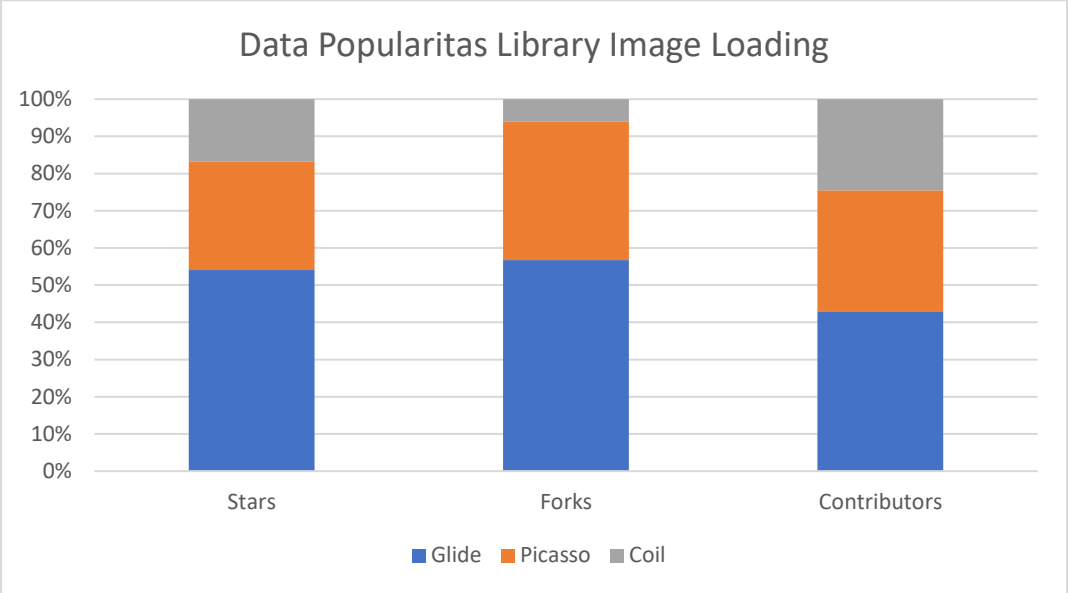
Coil adalah pustaka pemuatan gambar yang ditulis dengan Kotlin, pustaka pemuatan gambar ini dikenal karena performanya yang cepat dalam optimasi caching memori dan disk serta dukungan terhadap pemuatan gambar asinkron. Coil menawarkan API yang sederhana dan mudah digunakan , memanfaatkan fitur bahasa Kotlin untuk mengurangi *boilerplate code* (Coil Documentation, n.d.)

Salah satu fitur utama Coil adalah kemampuannya untuk mengintegrasikan gambar secara efisien dengan lifecycle-aware components, yang memungkinkan pemuatan gambar dihentikan ketika komponen antarmuka tidak lagi aktif, sehingga menghemat penggunaan memori. Selain itu, Coil mendukung SVG decoding dan image transformations dengan cara yang lebih sederhana dibandingkan pustaka sebelumnya.



Gambar 2. 3 Data Coil

Gambar 2.3 merupakan data *stars, forks,* dan *contributors* berdasarkan laman GitHub resmi dari pustaka pemuatan gambar Coil. Yang mana memiliki jumlah bintang (10.8k stars), memiliki jumlah *fork* (659 forks), dan memiliki kontributor sebanyak 85 kontributor.



Gambar 2. 4 Popularitas Library Image Loading

Gambar 2.4 merupakan data keseluruhan popularitas pustaka pemuatan gambar Glide, Picasso, dan Coil berdasarkan lama GitHub resmi dari masing-masing pustaka pemuatan gambar. Untuk laman GitHub Glide bisa diakses melalui (https://github.com/bumptech/glide), untuk laman GitHub Picasso bisa diakes melalui (https://github.com/square/picasso) , dan untuk laman GitHub Coil bisa diakses melalui (https://github.com/coil-kt/coil).

2.2.8 Penggunaan Sumber Daya pada Aplikasi Android

Dalam pengembangan aplikasi Android, pengelolaan sumber daya seperti CPU dan memori merupakan aspek penting yang dapat memengaruhi performa aplikasi dan pengalaman pengguna. Aplikasi yang mengandalkan pemuatan gambar sering kali menghadapi tantangan terkair penggunaan sumber daya yang efisien. Library seperti Glide, Picasso, dan Coil dikembangkan untuk mengatasi tantangan dengan mengoptimalkan pemrosesan gambar, manajemen memori, serta kecepatan pemuatan.

2.2.9 Penggunaan CPU

Dalam aplikasi Android, CPU digunakan untuk menangani berbagai tugas pemrosesan, termasuk pemuatan, transformasi, dan rendering gambar. Setiap kali aplikasi memuat gambar, CPU bertanggung jawab untuk medekode gambar, mengkompresinya, serta menyesuaikan resolusi agar sesuai dengan layar perangkat.

Sistem operasi Android memiliki mekanisme untuk mengelola penggunaan CPU secara dinamis. Ketika sebuah aplikasi memerlukan lebih banyak sumber daya,

Android dapat meningkatkan kecepatan CPU dan memindahkan beban kerja ke inti CPU yang lebih besar untuk menangani permintaan tersebut (Android Developers, n.d.). Hal ini dikenal sebagai *Dynamic Volatage and Frequency Scaling* (DVFC), yang memungkinkan sistem untuk menyesuaikan kinerja CPU berdasarkan kebutuha aplikasi. Dengan menggunakan API Performance Hint, aplikasi dapat memberikan sinyal kepada sistem tentang kebutuhan kerja mereka, sehingga Android dapat mengalokasikan sumber daya CPU secara efisien (Android Developers, n.d.).

Namun, penggunaan CPU yang tinggi dapat menyebabkan beberapa masalah, seperti peningkatan suhu perangkat dan penurunan masa pakai baterai. Penelitian menunjukkan bahwa aplikasi dengan penggunaan CPU yang tinggi dapat berdampak negatif pada kinerja keseluruhan perangkat, karena dapat mengganggu proses lain yang berjalan di latar belakang (Ametova & Lindström, 2023). Selain itu, penggunaan CPU yang tidak efisien sering kali disebabkan oleh implementasi kode yang buruk atau animasi yang tidak dioptimalkan dalam aplikasi (Ametova & Lindström, 2023).

2.2.10 Penggunaan Memori

Penggunaan memori dalam sistem operasi Android adalah aspek krusial yang memengaruhi kinerja aplikasi dan pengalaman pennggguna. Android meenggunakan model manajemen memori yang kompleks, yang mencakup pengalokasian dan pembebasan memori untuk berbagi aplikasi dan proses yang berjalan di latar belakang. Pada dasarnya, Android beroperasi di atas kernel Linux, yang menyediakan fondadi untuk pengelolaan memori, termasuk penanganan alokasi memori, pemantauan penggunaan memori, dan pengelolaan kondisi memori rendah (Das, 2023).

Android mengelola memori dengan menggunakan teknik seperti *paging* dan *memory mapping.* Ketika sebuah aplikasi membutuhkan memori, sistem akan mengalokasikan blok memori dari RAM. Setiap aplikasi memiliki heap-nya sendiri, yang merupakan area memori tempat objek-objek baru ditempatkan di “Young Generation”, sementara objek bertahan lebih lama dipindahkan ke generani yang lebih tua (Android Developers, 2023). Proses pengumpulan sampah *(garbage collection)* secara otomatis mengidentifikasi dan membebaskan memori yang tidak lagi digunakan oleh aplikasi, membantu mencegah kebocoran memori dan memastikan penggunaan sumber daya yang efisien (Das, 2023).

Dalam situasi di mana penggunaan memori mencapai batas maksimum, Android menerapkan mekanisme bernama “low-memory killer” untuk membebaskan ruang dengan menghentikan proses yang tidak aktif atau kurang penting. Proses ini menggunakan skor “out of memory” untuk menentukan prioritas aplikasi mana yang harus dihentikan terlebih dahulu (Android Developers, 2023).

2.2.11 Waktu Pemuatan Gambar (Loading Time)

Waktu pemuatan gambar (loading time) merupakan faktor penting yang memengaruhi pengalaman pengguna. Pengguna biasanya mengharapkan gambar dimuat dengan cepat, terutama dalam aplikasi seperti media sosial atau aplikasi e-commerce yang sering memuat banyak gambar dari internet. Jika pemuatan gambar lambat, hal ini dapat mengurangi respon aplikasi dan menyebabkan pengalaman pengguna yang buruk.

Proses pemuatan gambar pada Android melibatkan beberapa tahapan, mulai dari pengunduhan gambar dari server hingga rendering gambar di layer. Tahapan ini dapat memperlambar waktu pemuatam gambar jika tidak dioptimalkan dengan baik. Misalnya, penggunaan pustaka pemuatan gambar seperti Glide, Picasso, dan Coil dapat signifikan mengurangi pemuatan gambar karena mereka melakukan caching otomatis dan pengaturan *thread* yang canggih. (Mukherjee & Mondal, 2014) menekankan bahwa implementasi pustaka ini secara substansial mengurangi kompleksitas kode dan mempercepat proses pengembangan.

2.2.12 Kebocoran Memori (Memory Leak)

Kebocoran memori atau *memory leak* adalah kondisi di mana memori yang telah dialokasikan tidak dibebaskan ketika sudah tidak diperlukan, sehingga menyebabkan berkurangnya ketersediaan memori secara bertahap. Pada aplikasi Android, kebocoran memory disebabkan oleh kesalahan dalam menangani siklus hidup Activity, di mana pengelolaan yang kurang tepat dapat menurunkan knerja aplikasi dan bahkan mengalami crash (Amalfitano et al., 2020).

Pustaka seperti Glide dan Coil dilengkapi dengan mekanisme yang lebih baik untuk mengelola siklus hidup gambar, yang membantu mengurangi risiko kebocoran memori. Glide, misalnya, secara otomatis mengelola siklus hidup komponen gambar melalui integrasi dengan lifecycle Android, sementara Coil menggunakan pendekatan berbasis coroutines untuk memastikan gambar dilepaskan dengan benar ketika tidak

lagi dibutuhkan. Sebaliknya, Picasso memerlukan lebih banyak intervensi manual dari pengembang untuk memastikan bahwa gambar dilepaskan dengan benar.

Studi oleh (Song et al., 2021) menyoroti pentingnya menggunakan alat seperti IMGDroid untuk mendeteksi cacat pemuatan gambar yang dapat menyebabkan kebocoran memori pada aplikasi Android. Studi ini mengidentifikais berbagai pola yang menyebabkan kebocoran memori dalam pemrosesan gambar dan menunjukkan pentingnya pengelolaan memori yang efektif.

2.2.13 Movie App

Studi kasus dalam penlitian ini adalah aplikasi Android 'MovieApp'. MovieApp adalah sebuah aplikasi yang dikembangkan untuk menampilkan informasi tentang film, termasuk poster film, sinopsis, tanggal rilis, dan ulasan yang mana data film berasal dari API publik *The Movie Database* (TMDb). Salah satu fitur utama aplikasi ini adalah kemampuan untuk memuat dan menampilkan gambar poster film yang diambil dari API publik. Setiap kali pengguna membuka aplikasi atau menelusuri daftar film, MovieApp harus memuat gambar dari server eksternal yang kemudian ditampilkan di antarmuka pengguna.

Studi kasus ini sangat relevan untuk penelitian ini karena MovieApp bergantung pada pemuatan gambar secara dinamis dari API, yang menuntut efisiensi dalam penggunaan sumber daya, termasuk CPU dan memori. Setiap gambar yang ditampilkan di aplikasi dapat memiliki resolusi yang berbeda, yang memerlukan pengelolaan gambar yang efisien agar aplikasi tetap responsif. Dalam aplikasi seperti ini, penggunaan pustaka pemuatan gambar seperti Glide, Picasso, dan Coil menjadi sangat penting karena mereka mempengaruhi kecepatan pemuatan gambar, penggunaan memori, dan pengelolaan cache.

MovieApp cocok sebagai studi kasus untuk penelitian ini karena aplikasi ini mencerminkan tantangan nyata yang sering dihadapi dalam pengembangan aplikasi Android modern, terutama dalam hal pemuatan gambar dari API publik. Pengujian pustaka pemuatan gambar dalam konteks MovieApp memberikan wawasan yang mendalam tentang efisiensi pengelolaan gambar dalam aplikasi yang memerlukan pemrosesan gambar secara dinamis. Melalui studi kasus ini, peneliti dapat membandingkan performa pustaka Glide, Picasso, dan Coil dalam hal penggunaan CPU, penggunaan memori, serta kecepatan pemuatan gambar. Dengan demikian,

MovieApp menjadi platform yang ideal untuk menilai efisiensi pustaka image loader di lingkungan yang mendekati aplikasi nyata.