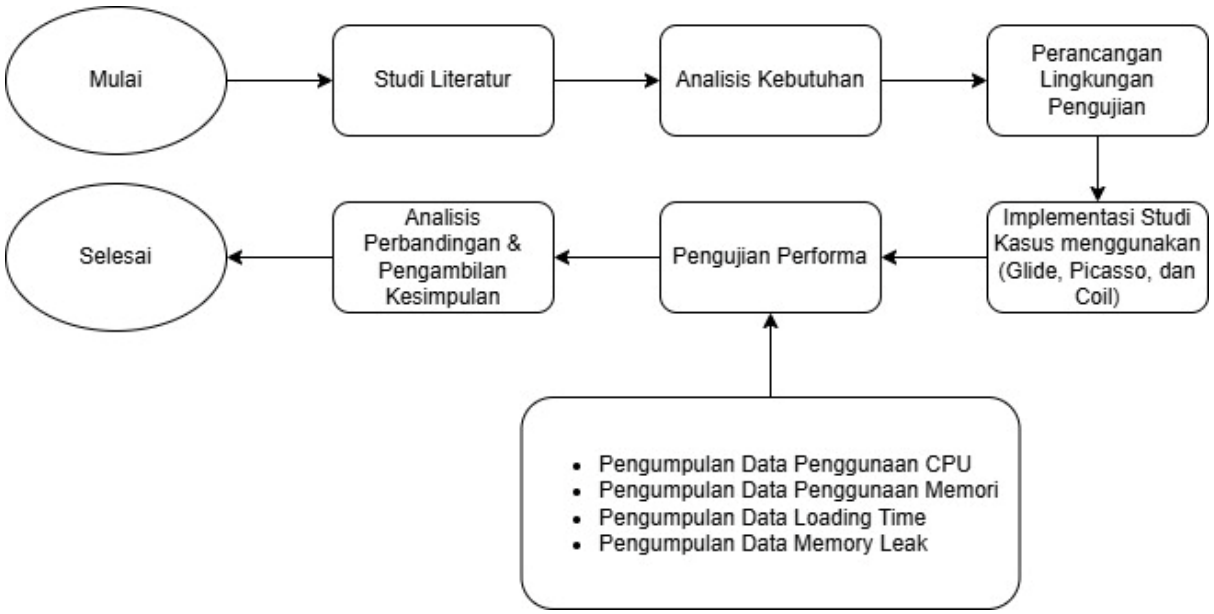
BAB III
DESAIN DAN IMPLEMENTASI SISTEM

3.1 Metodologi Penelitian

Pada sub-bab ini akan menjelaskan alur serta tahapan penulisan dalam melakukan penelitian, tahapan alur penelitian ini dimulai dari proses mencari studi literatur, melakukan analisis kebutuhan, melakukan perancangan lingkungan pengujian, implementasi studi kasus dengan menggunakan pustaka pemuatan gambar Glide, Picasso, dan Coil, melakukan pengujian performa yang mencakup pengumpulan data penggunaan CPU, pengumpulan data penggunaan memori, pengumpulan data *loading time,* dan pengumpulan data *memory leak,* dan alur yang terakhir adalah melakukan analisis perbandingan dan pengambilan kesimpulan.



Gambar 3. 1 Alur Metodologi Penelitian

3.1.1 Studi Literatur

Tahap pertama yang penulis lakukan dalam penelitian ini adalah studi literatur. Studi literatur ini bertujuan untuk mencari dan mengumpulkan referensi yang berkaitan dengan topik penelitian, yang meliputi pustaka pemuatan gambar pada Android yang berfokus pada pustaka pemuatan gambar Glide, Picasso, dan Coil dan metode yang akan diimplementasikan ke dalam sistem yang menjadi studi kasus dalam penelitian ini. Referensi yang dikumpulkan berasal dari berbagai sumber seperti jurnal, buku, artikel ilmiah, dan website dokumentasi resmi. Tahap pertama ini, bertujuan untuk

memberikan pemahaman yang mendalam mengenai konsep-konsep yang akan digunakan dalam penelitian serta membantu penulis dalam membuat kesimpulan atau merumuskan solusi yang tepat dengan berbasis teori yang tepat.

3.1.2 Analisis Kebutuhan

Analisis kebutuhan merupakan sebuah langkah awal yang bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan diimplementasikan dalam penelitian ini. Analisis kebutuhan terdiri dari dua hal, yaitu identifikasi masalah dan identifikasi kebutuhan pengguna.

Pada analisis kebutuhan yang pertama yaitu identifikasi masalah. Dalam konteks penelitian ini mengapa perlu ada penelitian yang membahas tentang analisis perbandingan pustaka pemuatan gambar pada Android berdasarkan parameter yang telah dijelaskan dalam latar belakang penelitian ini. Selanjutnya, siapa yang akan membutuhkan literatur penelitian ini. Kemudian seperti apa analisis perbandingan pustaka pemuatan gambar dengan studi kasus ‘MovieApp’ ini bekerja.

Penelitian analisis perbandingan pustaka pemuatan gambar ini perlu diteliti lebih lanjut, dikarenakan penelitian ini akan sangat bermanfaat untuk para pengembang aplikasi Android dalam pemilihan pustaka pemuatan gambar yang sesuai dengan proses bisnis aplikasi yang akan dikembangkan. Terutama aplikasi yang di dalamnya memerlukan banyak menampilkan gambar atau foto.

Penelitian analisis perbandingan pustaka pemuatan gambar diperuntukkan kepada para pengembang aplikasi Android. Penelitian ini ditujukan untuk para pengembang aplikasi Android dikarenakan sebelum melakukan pengembangan aplikasi Android para pengembang harus menentukan terlebih dahulu pustaka pemuatan gambar mana yang sesuai dengan proses bisnis.

Analisis perbandingan pustaka pemuatan gambar dengan studi kasus ‘MovieApp’ ini bekerja sesuai dengan data konten internet yang sering diakses oleh masyarakat Indonesia yaitu konten media sosial. Oleh karena itu, implementasi aplikasi ‘MovieApp’ ini dirasa sesuai karena dalam konten media sosial terdapat banyak proses untuk pemuatan gambar dan dalam implementasi aplikasi penelitian ini juga memerlukan proses untuk pemuatan gambar. Kemudian dalam penelitian ini diharuskan untuk meguji performa secara berkali-kali untuk melihat hasil secara keseluruhan.

Analisis kebutuhan yang kedua yaitu identifikasi kebutuhan pengguna, dalam konteks penelitian ini pengguna adalah para pengembang aplikasi Android. Identifikasi kebutuhan pengguna terdapat satu hal penting yaitu data atau informasi apa saja yang akan dibutuhkan oleh pengguna dalam penelitian ini.

Data yang dibutuhkan pengguna dalam penelitian ini adalah data *benchmarking* dalam implementasi tiga pustaka pemuatan gambar Glide, Picasso, dan Coil adalah data analisis penggunaan CPU, penggunaan memori, waktu pemuatan gambar *(loading time),* dan potensi kebocoran memori *(memory leak).* Kemudian informasi yang dibutuhkan oleh pengguna dalam penelitian ini adalah informasi mengenai rekomendasi pustaka pemuatan gambar yang lebih sesuai dan lebih efisien berdasarkan studi kasus pengembangan aplikasi.

3.1.3 Perancangan Lingkungan Pengujian

Lingkungan pengujian pada penelitian ini dirancang untuk menguji dan membandingkan performa tiga pustaka pemuatan gambar, yaitu Glide, Picasso, dan Coil, dalam hal penggunaan CPU, penggunaan memori, waktu pemuatan gambar *(loading time),* dan potensi kebocoran memori *(memory leak).* Dengan mengimplementasikan studi kasus 'MovieApp' yang mana data diambil dari API public yang menyediakan informasi film beserta gambar poster.

Penelitian ini tidak menggunakan *database* lokal, melainkan memanfaatkan API publik sebagai sumber data. API yang digunakan adalah The Movie Database (TMDb) yang menyediakan data berupa informasi film, termasuk judul, deskripsi, dan URL gambar poster. Data yang dikirimkan oleh API memiliki format JSON, yang kemudian diimplementasikan ke dalam studi kasus aplikasi 'MovieApp' dengan membuat tiga aplikasi Android yang masing-masing mengimplementasikan pustaka pemuatan gambar yang berbeda.

API publik dari TMDb digunakan untuk menghubungkan aplikasi studi kasus 'MovieApp' dengan server. Permintaan (request) data film dari aplikasi akan mengirimkan parameter, seperti judul film, kategori film dan lain-lain, dan akan mendapatkan respons dalam format JSON yang berisi informasi lengkap film, termasuk URL gambar poter. Semua pustaka pemuatan gambar akan diperlakukan sama dengan parameter yang sama untuk memastikan kondisi pengujian yang adil.

Untuk menguji performa masing-masing pustaka pemuatan gambar Glide, Picasso, dan Coil, aplikasi akan memuat gambar poster film dari API publik TMDb menggunakan tiga pustaka tersebut. Setiap pustaka akan digunakan dalam kondisi yang sama dengan ukuran gambar, *device,* dan koneksi jaringan yang serupa. Pengujian performa melibatkan beberapa aspek, yaitu penggunaan CPU dengan mencatat penggunaan CPU selama proses pemuatan gambar menggunakan masing-masing pustaka, penggunaan memori dengan mengukur memori yang digunakan oleh aplikasi saat memuat dan menampilkan gambar, waktu pemuatan gambar dengan menghitung waktu yang diperlukan oleh masing-masing pustaka untuk memuat gambar, dan potensi kebocoran memori dengan mengamati potensi kebocoran memori yang terjadi selama atau setelah pemuatan gambar.

3.1.4 Implementasi Studi Kasus Menggunakan (Glide, Picasso, dan Coil)

Penelitian ini mengimplementasikan studi kasus aplikasi Android 'MovieApp' dengan menggunakan tiga pustaka pemuatan gambar, yaitu Glide, Picasso, dan Coil. Studi kasus yang digunakan adalah menampilkan gambar poster film dari API publik TMDb ke dalam Recyclerview di halaman utama aplikasi dan di halaman *watchlist* aplikasi. Setiap pustaka pemuatan gambar diimplementasikan secara terpisah dengan menggunakan parameter dan scenario yang sama.

Implementasi menggunakan pustaka pemuatan gambar Glide. Glide merupakan salah satu pustaka pemuatan gambar yang popular digunakan dalam pengembangan aplikasi Android untuk memuat dan menampilkan gambar. Berikut langkah-langkah implementasi pustaka pemuatan gambar Glide.

```
// Glide
implementation("com.github.bumptech.glide:glide:4.16.0")
annotationProcessor("com.github.bumptech.glide:compiler:4.16.0")
```

Gambar 3. 2 Implementasi Glide dalam Android Studio

Pada gambar 3.3 merupakan implementasi menggunakan pustaka pemuatan gambar Picasso. Berikut langkah-langkah implementasi pustaka pemuatan gambar Picasso.

```
// Picasso
implementation("com.squareup.picasso:picasso:2.8")
```

Gambar 3. 3 Implementasi Picasso dalam Android Studio

Implementasi menggunakan pustaka pemuatan gambar Coil. Berikut langkah-langkah implementasi pustaka pemuatan gambar Coil.

```
// Coil
implementation("io.coil-kt:coil:2.4.0")
```

Gambar 3. 4 Implementasi Coil dalam Android Studio

3.1.5 Pengujian Performa

Pengujian performa dilakukan untuk membandingkan kinerja tiga pustaka pemuatan gambar Glide, Picasso, dan Coil pada studi kasus aplikasi Android 'MovieApp'. Pengujian ini berfokus pada empat parameter utama, yaitu penggunaan CPU, penggunaan memori, waktu pemuatan gambar *(loading time),* dan potensi kebocoran memori *(memory leak).* Setiap pengujian dilakukan dengan kondisi yang sama agar hasil dari pengujian performanya konsisten.

Pengujian dilakukan dengan metode pengukuran langsung pada perangkat Android melalui Android Profile, yang merupakan alat yang telah tersedia di dalam Android Studio untuk memantau performa aplikasi. Pengujian melibatkan beberapa scenario penggunaan aplikasi, seperti membuka halaman utama dan melakukan *scrolling* pada Recyclerview yang memuat gambar poster film dan juga pada halaman *watchlist* dan *favorite-film.*

Pengujian penggunaan CPU diukur ketika aplikasi menjalankan proses pemuatan gambar pada halaman home dan *watchlist* Recyclerview. Aktivitas dilakukan dengan cara melakukan *scrolling* pada halaman tersebut sebanyak 30 kali. Pengujian ini bertujuan untuk mengukur berapa banyak sumber daya CPU yang digunakan setiap library selama pemuatan gambar.

Pengujian penggunaan memori diukur dengan memantau berapa banyak *memory heap* yang dialokasikan oleh aplikasi selama proses pemuatan gambar. Pengujian ini dilakukan untuk mengetahui seberapa besar memori yang dikonsumsi oleh masing-masing pustaka pemuatan gambar, yang memengaruhi kinerja keseluruhan aplikasi.

Pengujian waktu pemuatan gambar *(loading time)* diukur mulai dari saat gambar diminta hingga berhasil ditampilkan di ImageView. Pengujian ini dilakukan dengan membuka halaman home dan halaman *watchlist* aplikasi yang menampilkan poster film dan mencatat waktu yang dibutuhkan untuk memuat seluruh gambar.

Pengujian potensi kebocoran memori *(memory leak)* diuji dengan menjalankan aplikasi selama periode waktu tertentu dan mengukur penggunaan memori dari waktu ke waktu. Jika terjadi peningkatan memori yang tidak dilepakan kembali oleh sistem, maka hal in mengindikasikan adanya kebocoran memori. Alat yang digunakan untuk mengetahui kebocoran memori ini adalah LeakCanary.

3.1.6 Analisis Perbandingan & Pengambilan Kesimpulan

Analisis perbandingan dilakukan setelah melakukan seluruh tahapan pada penelitian ini. Mulai dari studi litearur, analisi kebutuhan, perancangan lingkungan pengujian, implementasi studi kasus menggunakan (Glide, Picasso, dan Coil), dan pengujian performa. Dan tahap ini dilakukan sebelum mengambil sebuah kesimpulan pada penelitian ini. Yang mana tahap ini juga penting bagi peneliti untuk menganalisis perbandingan pustaka pemuatan gambar setelah mengetahui seluruh data.

Kesimpulan berisi hasil perbandingan penggunaan CPU, penggunaan memori, waktu pemuatan gambar *(loading time),* potensi kebocoran memori *(memory leak),* dan pengaruh perbedaan ukuran gambar pada tiap pustaka pemuatan gambar yang diuji. Selain kesimpulan, terdapat juga saran yang bisa digunakan peneliti selanjutnya sebagai acuan dan pertimbangan untuk pengembangan penelitian selanjutnya.

3.2 Pustaka Pemuatan Gambar (Library Image Loading)

Dalam penelitian ini menggunakan tiga pustaka pemuatan gambar yang akan diimplementasikan ke dalam studi kasus aplikasi Android ‘MovieApp’, yaitu Glide, Picasso, dan Coil. Berikut penjelasan mengenai masing-masing versi pustaka pemuatan gambar yang digunakan dalam penelitian ini.

1. Glide

   Pustaka pemuatan gambar Glide yang akan digunakan dalam penelitian ini menggunakan versi terbarunya yaitu 4.16.0 dan tidak menutup kemungkinan apabila penelitian ini belum selesai Glide telah mengembangkan versi yang lebih baru lagi dari versi 4.16.0.

2. Picasso

   Pustaka pemuatan gambar Picasso yang akan digunakan dalam penelitian ini menggunakan versi 2.8 dan tidak menutup kemungkinan apabila penelitian ini belum selesai Picasso telah mengembangkan versi yang lebih baru lagi dari versi 2.8.

3. Coil

   Pustaka pemuatan gambar Coil yang akan digunakan dalam penelitian ini menggunakan versi 2.4.0 dan tidak menutup kemungkinkan apabila penelitian ini belum selesai Coil telah mengembangkan versi yang lebih baru lagi dari versi 2.4.0.

3.3 Spesifikasi Perangkat Keras (Hardware)

Dalam penelitian ini memerlukan dua perangkat keras untuk mencapai tujuan penelitian yang diinginkan oleh peneliti. Perangkat keras yang dibutuhkan yaitu Laptop untuk pengembangan dan implementasi studi kasus aplikasi Android dan *smartphone* Android untuk menjalankan aplikasi.

1. Laptop

   Tabel 3. 1 Spesifikasi Laptop

Processor	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz
GPU	NVIDIA GeForce GTX 1650 Ti
Sistem Operasi	Windows 11 Home Single Language
Penyimpanan	512 GB SSD
RAM	2 x 8 GB DDR4 2 x 8 GB DDR4 3200MHz

2. Smartphone Android

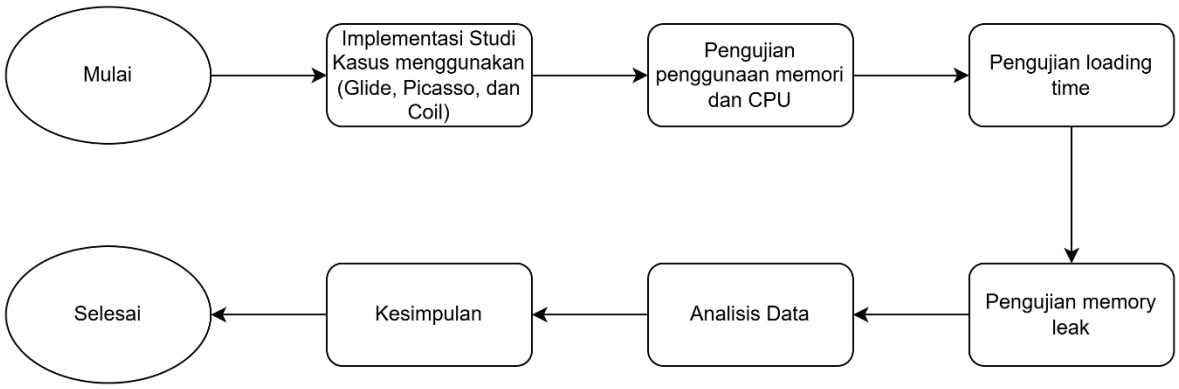
   Tabel 3. 2 Tabel Smartphone

Model	Samsung Galaxy A71
Sistem Operasi	Android 13

Processor	Qualcomm Snapdragon 730G
Penyimpanan	128 GB
RAM	8 GB

3.4 Rencana Pengujian

Pada sub bab rencana pengujian ini menjelaskan mengenai alur pengujian yang berfokus dalam pengujian untuk menganalisis perbandingan pustaka pemuatan gambar Glide, Picasso, dan Coil. Berbeda dengan alur yang berada di sub bab 3.1 Metodologi Penelitian, yang mana dalam sub bab tersebut menjelaskan alur mengenai keseluruhan penelitian ini.



Gambar 3. 5 Rencana Pengujian

3.4.1 Implementasi Studi Kasus Menggunakan (Glide, Picasso, dan Coil)

Untuk penjelasan implementasi studi kasus aplikasi Android 'MovieApp' menggunakan pustaka pemuatan gambar Glide, Picasso, dan Coil telah dijelaskan di dalam sub bab 3.1.4.

Studi kasus yang digunakan dalam penelitian ini adalah studi kasus 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.

3.4.2 Pengujian penggunaan CPU

Pada tahapan ini menjelaskan mengenai pengujian-pengujian yang dilakukan dalam menentukan analisis perbandingan pustaka pemuatan gambar dengan parameter penggunaan cpu.

Pengujian ini dilakukan dengan menggunakan *automated UI testing* dan testing secara manual. *Automated UI testing* dilakukan untuk menjaga konsistensi dalam pengujian dan pengambilan data. Untuk testing secara manual juga berguna untuk memberikan variasi dalam proses pengujian. Pengujian antarmuka aplikasi menggunakan *framework* untuk *automated UI testing* yaitu Espresso.

Proses pengujian di jalankan dengan melakukan *scrolling* pada halaman yang menampilkan daftar poster film, halaman *watchlist,* dan halaman *favorite film.* Setelah pengujian ini peneliti merekam penggunaan memori dan cpu menggunakan *tools* yang ada di dalam Android Studio yaitu Android Profiler.

Tabel 3. 3 Pengujian pengguaan CPU

No	Test Case	Pustaka	Langkah Pengujian	Hasil Aktual	Status
1	Pengujian penggunaan CPU	Glide	Pantau penggunaan CPU menggunakan Android Profiler saat aplikasi memuat gambar di halaman utama dan watchlist.	persentase penggunaan CPU (%)	Penggunaan CPU stabil di bawah batas maksimal
2	Pengujian penggunaan CPU	Picasso	Pantau penggunaan CPU menggunakan Android Profiler saat aplikasi memuat gambar di halaman utama dan watchlist.	persentase penggunaan CPU (%)	Penggunaan CPU stabil di bawah batas maksimal
3	Pengujian penggunaan CPU	Coil	Pantau penggunaan CPU menggunakan Android Profiler	persentase penggunaan CPU (%)	Penggunaan CPU stabil di bawah batas maksimal

			saat aplikasi memuat gambar di halaman utama dan watchlist.		

3.4.3 Pengujian penggunaan Memori

Pada tahapan ini sama seperti dengan pengujian penggunaan CPU pada sub-bab 3.4.2 yang menjelaskan mengenai pengujian-pengujian yang dilakukan dalam menentukan analisis perbandingan pustaka pemuatan gambar dengan parameter penggunaan memori.

Tabel 3. 4 Pengujian penggunaan Memori

No	Test Case	Pustaka	Langkah Pengujian	Hasil Aktual	Status
1	Pengujian penggunaan memori	Glide	Pantau penggunaan memori menggunakan Android Profiler saat aplikasi memuat dan menampilkan gambar di Recyclerview.	MB (Megabyte)	Penggunaan memori stabil tanpa lonjakan signifikan
2	Pengujian penggunaan memori	Picasso	Pantau penggunaan memori menggunakan Android Profiler saat aplikasi memuat dan menampilkan gambar di Recyclerview.	MB (Megabyte)	Penggunaan memori stabil tanpa lonjakan signifikan
3	Pengujian penggunaan memori	Coil	Pantau penggunaan memori menggunakan Android Profiler saat aplikasi memuat dan menampilkan	MB (Megabyte)	Penggunaan memori stabil tanpa lonjakan signifikan

			gambar di Recyclerview.		

3.4.4 Pengujian *Loading Time*

Pada tahapan ini menjelaskan mengenai pegujian waktu pemuatan gambar *(loading time).* Perhitungan untuk pengujian waktu pemuatan gambar dilakukan dari awal aplikasi dibuka dan peneliti mencatat waktu pemuatan gambar dengan menggunakan *tools* yang ada di dalam Android Studio yaitu Android Profile. Kemudian peneliti akan mencatat hasil waktu pemuatan gambar dari masing-masing aplikasi yang diimplementasikan dengan menggunakan tiga pustaka pemuatan gambar yang berbeda, yaitu Glide, Picasso, dan Coil.

Tabel 3. 5 Pengujian Loading Time

No	Test Case	Pustaka	Langkah Pengujian	Hasil Aktual	Status
1	Pengujian waktu pemuatan gambar	Glide	Ukur waktu dari gambar diminta hingga tampil di ImageView menggunakan Android Profiler.	detik (seconds)	Waktu pemuatan gambar konsisten dan cepat
2	Pengujian waktu pemuatan gambar	Picasso	Ukur waktu dari gambar diminta hingga tampil di ImageView menggunakan Android Profiler.	detik (seconds)	Waktu pemuatan gambar konsisten dan cepat
3	Pengujian waktu pemuatan gambar	Coil	Ukur waktu dari gambar diminta hingga tampil di ImageView menggunakan Android Profiler.	detik (seconds)	Waktu pemuatan gambar konsisten dan cepat

3.4.5 Pengujian Memory Leak

Pada tahapan ini menjelaskan mengenai pengujian potensi kebocoran memori yang ada pada studi kasus aplikasi ‘MovieApp’. Pengujian potensi kebocoran memori menggunakan Leak Canary dengan cara menjalankan aplikasi beberapa kali.

Tabel 3. 6 Pengujian Memory Leak

No	Test Case	Pustaka	Langkah Pengujian	Hasil Aktual	Status
1	Pengujian potensi kebocoran memori	Glide	Gunakan LeakCanary untuk mendeteksi apakah ada memori yang tidak dibebaskan setelah pemuatan gambar.	[Diisi saat uji]	Ada/ Tidak Ada
2	Pengujian potensi kebocoran memori	Picasso	Gunakan LeakCanary untuk mendeteksi apakah ada memori yang tidak dibebaskan setelah pemuatan gambar.	[Diisi saat uji]	Ada/ Tidak Ada
3	Pengujian potensi kebocoran memori	Coil	Gunakan LeakCanary untuk mendeteksi apakah ada memori yang tidak dibebaskan setelah pemuatan gambar.	[Diisi saat uji]	Ada/ Tidak Ada

3.4.6 Analisis Data

Untuk mengetahui perbedaan performa antara pustaka pemuatan gambar Glide, Picasso, dan Coil, data akan dianalisis menggunakan dua uji. Uji pertama yang dilakukan adalah uji normalitas. Uji normalitas adalah uji untuk mengetahui apakah data sampel berdistribusi normal atau tidak (Nugroho et al., 2023). Selanjutnya akan menjadi penentu apakah uji yang dilakukan selanjutnya menggunakan uji parametrik atau non-parametrik. Setelah data terkumpul dari langkah-langkah pengujian sebelumnya, maka akan dianalisis menggunakan uji Shapiro-Wilk. Uji Shapiro-Wilk digunakan karena jumlah sampel sebanyak 30 dan data berbentuk angka (Nugroho et al., 2023). Jika terdapat distribusi data yang tidak normal maka nantinya data yang tidak normal akan di uji menggunakan Uji Mann-Whitney U. Uji Mann-Whitney U digunakan apabila terdapat data yang tidak berpasangan.

3.4.7 Kesimpulan

Pada tahap ini peneliti akan memberikan kesimpulan mengenai analisis perbandingan pustaka pemuatan gambar Glide, Picasso, dan Coil. Isi dari kesimpulan tidak hanya mengenai pustaka pemuatan gambar mana yang lebih efisien secara performa akan tetapi peneliti akan memberikan rekomendasi kepada para pengembang aplikasi Android untuk memilih pustaka pemuatan gambar yang cocok untuk pengembangan aplikasi Android lainnya.

3.5 Test Case (Black Box Testing)

Test case atau pengujian kasus yang digunakan dalam penelitian ini adalah menggunakan Black Box Testing. Black Box Testing adalah pengujian yang dilakukan untuk mengamati hasil dari aplikasi yang siap untuk menguji fungsionalitas input dan output, sehingga dapat berfungsi dengan baik tanpa mengetahui struktur kode, dan pengujian ini dapat diterapkan pada pengujian *unit, integration, system,* dan *acceptance testing* (Hidayat et al., 2023).

Dalam Black Box Testing terdapat beberapa Teknik yang bisa digunakan untuk menguji perangkat lunak, yaitu *All pair testing, boundary value analysis, cause-effcect graph, equivalence partitioning, fuzzing, orthogonal array testing,* dan *state transition.* Pada penlitian ini untuk Black Box Testing akan menggunakan Teknik *Equivalence Partitioning.*

Teknik Equivalence Partitioning adalah Teknik yang sering digunakan dalam pengujian Black Box Testing yang dikelompokkan atau berdasarkan domain saat menguji aplikasi, terutama saat menguji antarmuka. Tujuan dari penelitian ini adalah untuk mengetahui kelemahan dalam sistem ketika diuji atau memastikan bahwa input yang dimasukkan akan menghasilkan output yang diharapkan sehingga dapat menghindari kesalahan atau kekurangan dalam aplikasi (Hidayat et al., 2023).

3.5.1 Test UI

Test antarmuka ini bertujuan untuk melakukan pengujian dalam antarmuka aplikasi studi kasus 'MovieApp' agar keseluruhan antarmuka pengguna berjalan dengan lancar.

Tabel 3. 7 Tabel Test UI

No	Nama Test Case	Deskripsi	Langkah Pengujian	Input	Ekspetasi Hasil	Hasil Aktual	Status
1	Pengujian Pemuatan Gambar halaman home (API TMDb)	Menguji apakah gambar poster film berhasil di muat dalam halaman home aplikasi	1. Buka aplikasi Movie App 2. Mengamati apakah daftar poster film tampil pada layar	Daftar film dengan poster dari API TMDb	Gambar film berhasil tampil di layer pengguna dengan baik.	[Diisi sat uji]	Berhasil/ Gagal
2	Pengujian pemuatan gambar	Menguji apakah ketika memilih poster film, detail	1. Pilih salah satu film pada halaman home 2. Mengamati	Detail dari masing-masing poster film dari	Halaman detail dapat dimuat dengan baik ketika memilih	[Diisi sat uji]	Berhasil/ Gagal

	halaman detail	film berhasil ditampilkan dengan baik	apakah halaman detail berhasil dimuat dengan baik	API TMDb.	salah satu poster film.		
3	Pengujian Pemuatan Gambar halaman watchlist (API TMDb)	Menguji apakah gambar poster film berhasil di muat dalam halaman watchlist aplikasi	1. Menambahkan watctlist film dari halaman home yang menampilkan daftar poster film. 2. Apakah penambahan watchlist berhasil ditambahakn di halaman watchlist 3. Mengamati apakah daftar poster film tampil pada layer di halaman watchlist	Daftar watchlist film	Halaman watchlist dapat menambahkan watchlist dari poster film pada halaman home	[Diisi satuji]	Berhasil/ Gagal

4	Pengujian Pemuatan Gambar favorite film watchlist (API TMDb)	Menguji apakah gambar poster film berhasil di muat dalam halaman favorite film aplikasi	1. Menambahkan favorite film dari halaman home yang menampilkan daftar poster film. 2. Apakah penambahan favorite film berhasil ditambahakn di halaman favoritefilm 3. Mengamati apakah daftar poster film tampil pada layer di halaman favorite film	Daftar favorite film	Halaman favorite film dapat menambahkan favorite film dari poster film pada halaman home	[Diisi sat uji]	Berhasil/ Gagal
5	Pengujian Scroll	Menguji apakah aplikasi tetap berjalan	1. Membuka halaman daftar poster film	Daftar film dengan gambar	Scroll berjalan dengan lancar tanpa lagging	[Diisi sat uji]	Berhasil/ Gagal

		dengan lancar ketika pengguna melakukan scroll pada halaman daftar film yang memuat banyak gambar	2. Melakukan Scroll secara berkelanjutan 3. Mengamati apakah ada jeda atau lagging selama proses scroll		signifikan meskipun banyak gambar yang dimuat		

3.6 Data yang Digunakan

3.6.1 Sumber Data

Pada penelitian ini data yang digunakan diambil dari API publik *The Movie Database (TMDb).* TMDB merupakan laayanan basis data film yang menyediakan berbagai informasi tentang film, termasuk sinopsis, genre, tanggal rilis, rating, dan gambar poster film.

Enpoint API yang diakses dalam penelitian ini menggunakan endpoint (/movie/popular) dari TMDb untuk mendapatkan daftar film yang popular saat ini. Endpoint tersebut mengembalikan data dalam format JSON yang berisi informasi dasar film serta URL gambar poster film dalam beberapa resolusi.

Format data yang diterima dari API TMDb dikirim dalam format *JavaScript Object Notation* (JSON), yang mudah dibaca dan diproses oleh aplikasi Android. Setiap respons JSON berisi atribut seperti, *title, overview, poster_path, release_date,* dan *vote_average.*

Jenis informasi yang diambil dalam penelitian ini berfokus dalam gambar poster film, yang diambil melalui atribut *poster_path* dari respons JSON. Gambar ini

kemudian diunduh dan ditampilkan menggunakan pustaka pemuatan gambar Glide, Picasso, dan Coil.

3.6.2 Spesifikasi Data Gambar

Resolusi gambar diambil dari API TMDb menyediakan beberapa resolusi gambar untuk setiap poster film.

Resolusi	Lebar (px)	Jenis	Format
w154	154	poster	.jpg
w185	185	poster	.jpg
w780	780	backdrop	.jpg
w92	92	person	.jpg

Gambar dengan resolusi yang berbeda digunakan untuk menguji bagaimana masing-masing pustaka pemuatan gambar menangani gambar dengan ukuran file yang berbeda.

Format gambar yang digunakan dalam penelitian ini adalah JPEG(.jpg) yang merupakan format standart untuk poster film yang disediakan oleh TMDb.

Untuk resolusi w154 menggunakan endpoint API https://image.tmdb.org/t/p/w154/63xYQj1BwRFielxsBDXvHIJyXVm.jpg

Gambar 3. 6 Gambar Poster Resolusi w154

Untuk resolusi w185 menggunakan endpoint API https://image.tmdb.org/t/p/w185/63xYQj1BwRFielxsBDXvHIJyXVm.jpg

Gambar 3. 7 Gambar Poster Resolusi w185

Untuk resolusi w780 menggunakan endpoint API https://image.tmdb.org/t/p/w780/63xYQj1BwRFielxsBDXvHIJyXVm.jpg

Gambar 3. 8 Gambar Backdrop Resolusi w780

Untuk resolusi w92 menggunakan endpoint API https://image.tmdb.org/t/p/w92//bHHn3krbHyxQIWb4JbHkPlV6Uu1.jpg

Gambar 3. 9 Gambar Person Resolusi w92

3.6.3 Kondisi Pengujian

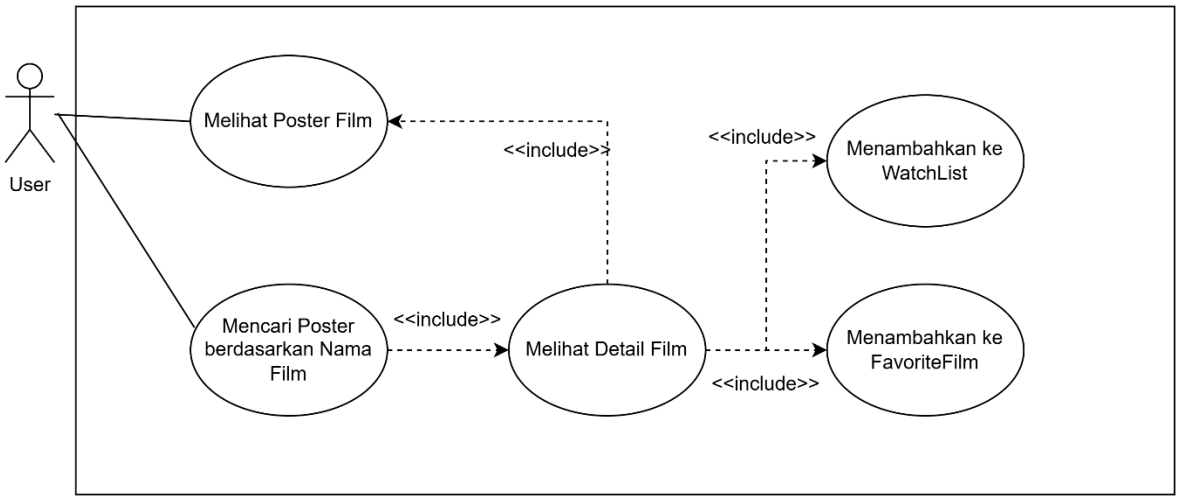
Terdapat beberapa kondisi pengujian dalam penelitian ini, yang pertama adalah kecepatan internet yang digunakan. Dalam pengujian ini dilakukan dalam lingkungan dengan kecepatan internet stabil, yaitu sekitar 50 Mbps, untuk memastikan bahwa variasi waktu pemuatan gambar tidak dipengaruhi oleh kondisi jaringan.

Kemudian kondisi pengujian *cache* dilakukan dalam dua kondisi, yaitu *fresh install* dan dengan *cache. Fresh install* di mana kondisi aplikasi tidak memiliki *cache* sebelumnya dan setiap gambar dimuat dari server. Dengan *cache* di mana gambar yang telah dimuat sebelumnya disimpan dalam *cache* lokal untuk menguji bagaimana setiap pustaka menangani caching gambar.

3.7 Desain Studi Kasus aplikasi Android 'MovieApp'

Desain sistem untuk studi kasus aplikasi Android 'MovieApp'. Seperti yang telah dijelaskan pada sub-bab 2.2.13 untuk penjelasan mengenai studi kasus dalam penelitian ini. Berikut merupakan sebuah desain sistem untuk lebih memperjelas penjelasan mengenai studi kasus aplikasi Android 'MovieApp' yang berisikan *use case, use case scenario, activity diagram, dan class diagram*.

3.7.1 Use Case



Gambar 3. 10 Use case

Tabel 3. 8 Use Case Scenario Melihat Poster Film

Use Case Name	Melihat Poster Film	
Actor	User	
Purpose	Melihat keseluruhan poster film	
Typical Course of Event	Aktor	Sistem
	1. Membuka aplikasi	2. Menampilkan seluruh poster film
Alternative Course	1. Tidak terhubung koneksi internet	2. Tidak bisa menampilkan poster film
Pre Condition	Membuka aplikasi	
Post Condition	Melihat poster film dalam halaman *home*	

Tabel 3. 9 Use Case Scenario Mencari Poster berdasarkan Nama Film

Use Case Name	Mencari Poster berdasarkan Nama Film	
Actor	User	
Purpose	Menemukan judul poster film	
Typical Course of Event	Aktor	Sistem

	1. Mencari judul poster film	2. Menampilkan hasil pencarian judul poster film
Alternative Course	1. Tidak mencari judul poster film	2. Tidak menampilkan hasil pencarian poster film
Pre Condition	Mencari judul poster film	
Post Condition	Melihat hasil poster film yang dicari	

Tabel 3. 10 Use Case Scenario Melihat Detail Film

Use Case Name	Melihat Detail Film	
Actor	User	
Purpose	Melihat Detail Film	
Typical Course of Event	Aktor	Sistem
	1. Memilih poster film yang ingin dilihat detail film nya	2. Menampilkan halaman detail film
Alternative Course	1. Tidak memilih poster film	2. Tidak menampilkan halaman detail film
Pre Condition	Memilih poster film	
Post Condition	Melihat halaman detail poster film	

Tabel 3. 11 Use Case Scenario Menambahkan ke FavoriteFilm

Use Case Name	Menambahkan ke FavoriteFilm	
Actor	User	
Purpose	Menambahkan poster film ke favorite film	
Typical Course of Event	Aktor	Sistem

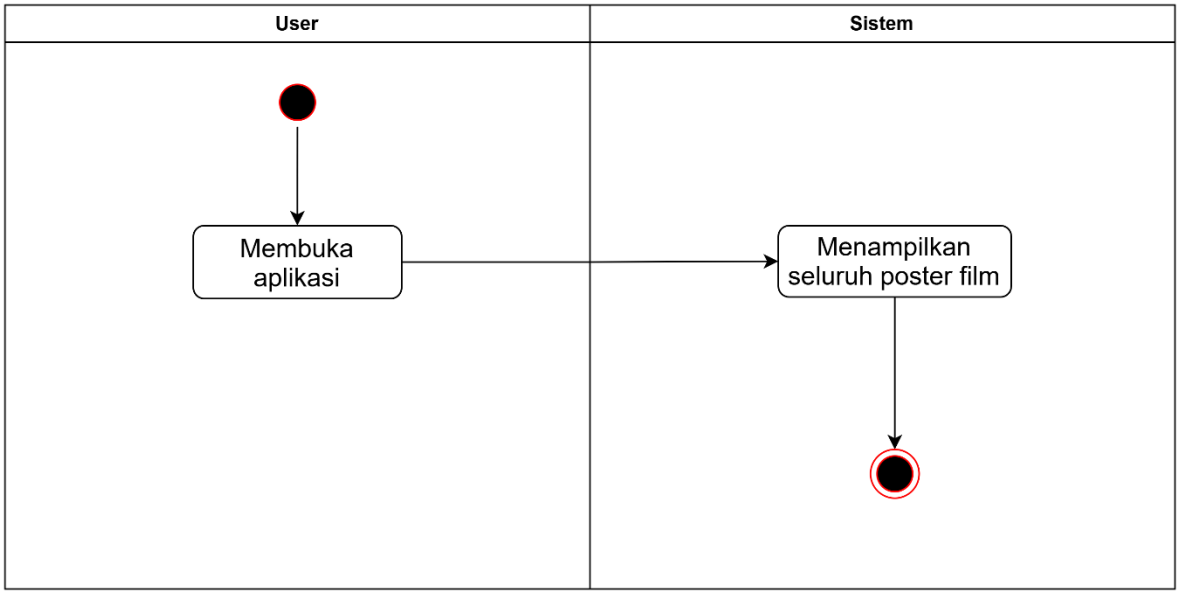
	1. Memilih poster film yang ingin dilihat detail film nya	2. Menampilkan halaman detail poster film
	3. Melihat detail poster film	
	4. Menambahkan poster film ke dalam favorite film	5. Menyimpan poster film ke dalam favorite film
Alternative Course	1. Tidak memilih poster film	2. Tidak menyimpan poster film ke dalam favorite film
Pre Condition	Memilih poster film	
Post Condition	Poster film tersimpan ke dalam favorite film	

Tabel 3. 12 Use Case Scenario Menambahkan ke WatchList

Use Case Name	Menambahkan ke WatchList	
Actor	User	
Purpose	Menambahkan poster film ke watch list	
Typical Course of Event	Aktor	Sistem
	1. Memilih poster film yang ingin dilihat detail film nya	2. Menampilkan halaman detail poster film
	3. Melihat detail poster film	
	4. Menambahkan poster film ke dalam watch list	5. Menyimpan poster film ke dalam watch list

Alternative Course	3. Tidak memilih poster film	4. Tidak menyimpan poster film ke dalam watch list
Pre Condition	Memilih poster film	
Post Condition	Poster film tersimpan ke dalam watch list	

3.7.2 Activity Diagram

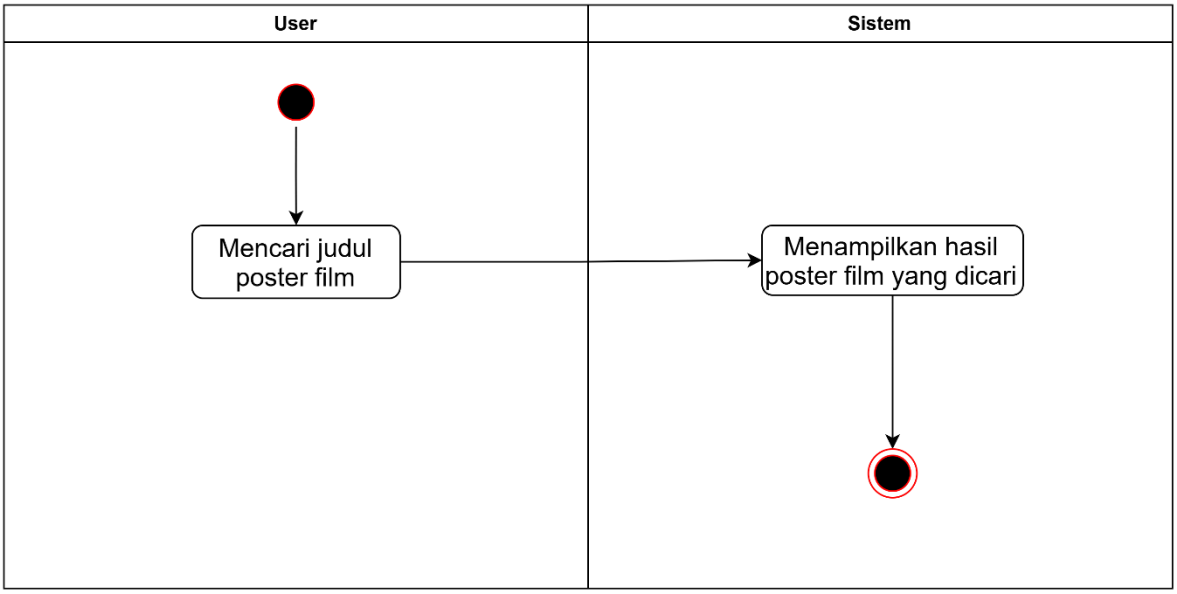


Gambar 3. 11 Activity Diagram 1

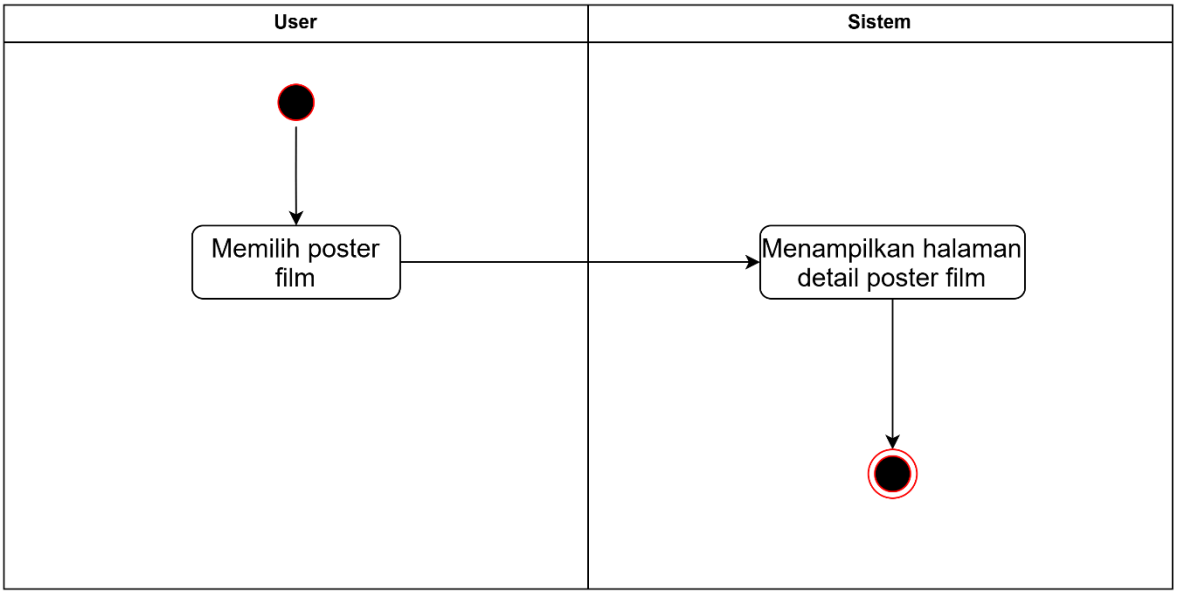
Pada gambar 3.7 menggambarkan diagram yang menunjukkan proses pengguna membuka aplikasi untuk pertama kali dan sistem akan menampilkan seluruh poster film.

Pada gambar 3.8 menggambarkan diagram yang menunjukkan proses pengguna mencari judul poster film dan sistem akan menampilkan hasil pencarian poster film.

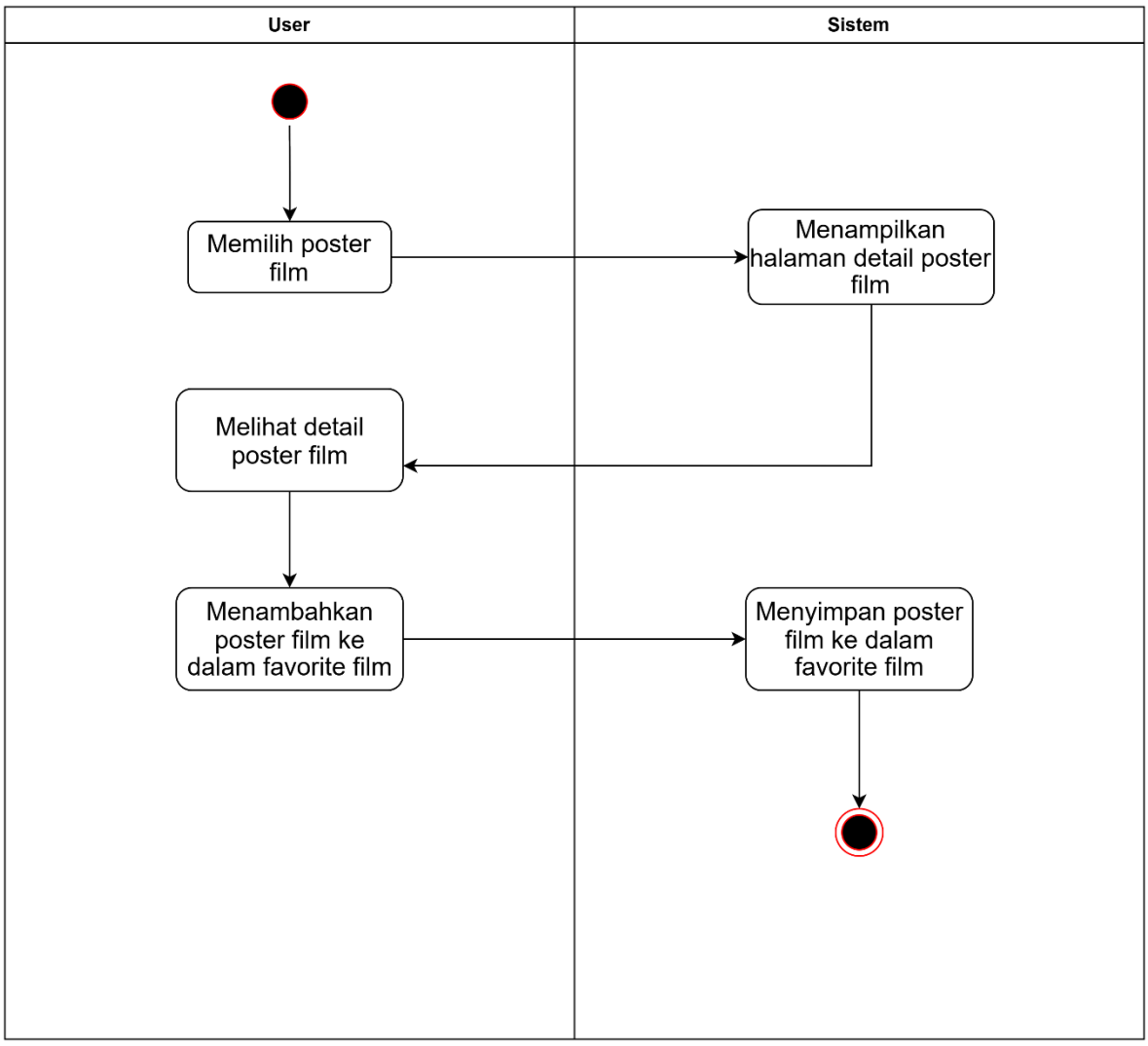
Pada gambar 3.9 menggambarkan diagram yang menunjukkan proses pengguna memilih poster film dan sistem akan merespon untuk menampilkan halaman detail film.



Gambar 3. 12 Activity Diagram 2



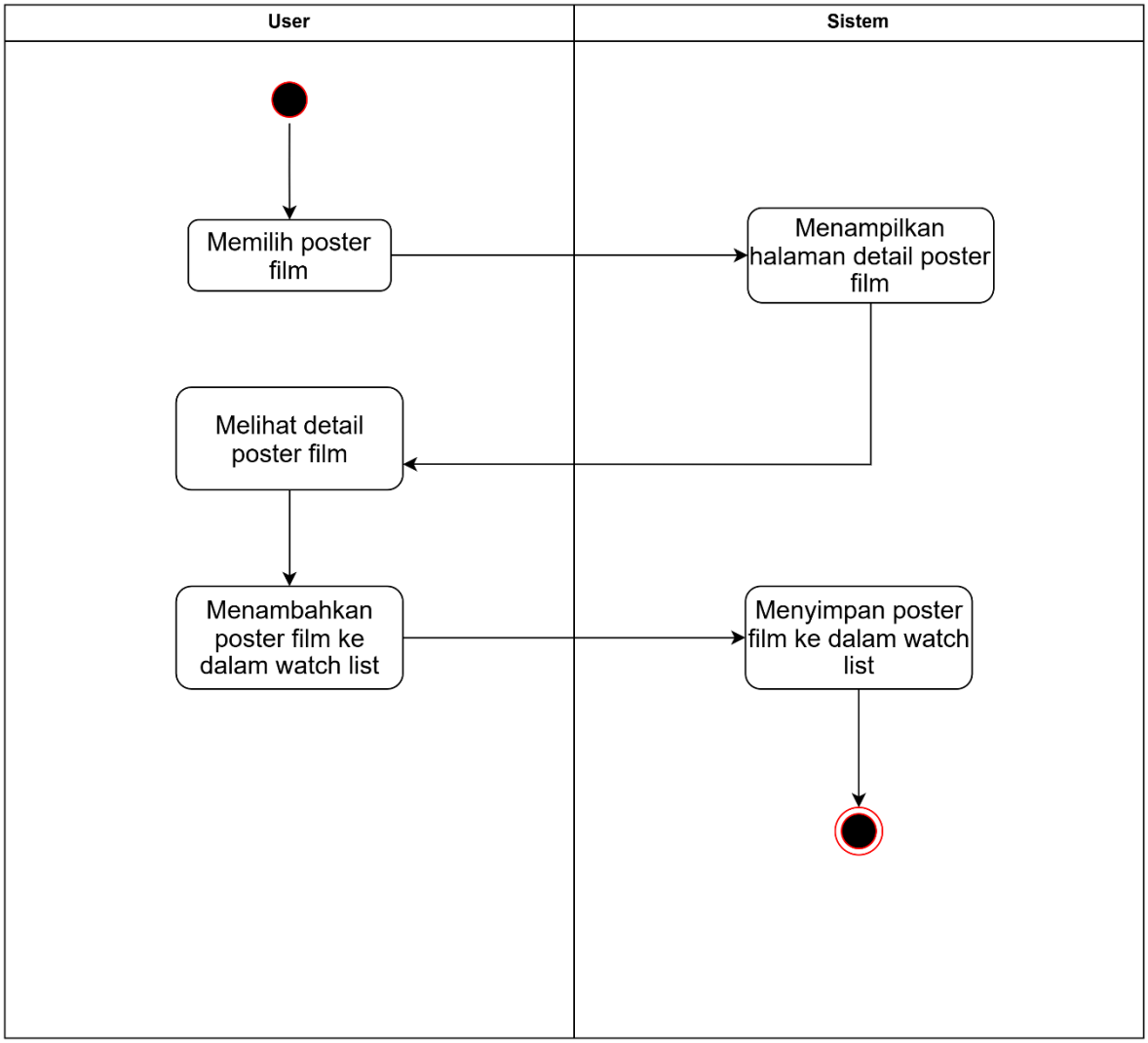
Gambar 3. 13 Activity Diagram 3



Gambar 3. 14 Activity Diagram 4

Pada gambar 3.10 menggambarkan alur diagram yang menunjukkan proses pengguna untuk memilih poster film kemudian sistem akan menampilkan halaman detail poster film, kemudian pengguna dapat melihat halaman detail film, dan pengguna juga dapat menambahkan poster film ke dalam *favorite film* yang nantinya sistem akan menyimpan poster film ke dalam *favorite film.*

Pada gambar 3.11 menggambarkan alur diagram yang menunjukkan proses pengguna untuk memilih poster film kemudian sistem akan menampilkan halaman detail poster film, kemudian pengguna dapat melihat halaman detail film, dan pengguna juga dapat menambahkan poster film ke dalam *watch list* yang nantinya sistem akan menyimpan poster film ke dalam *watch list.*



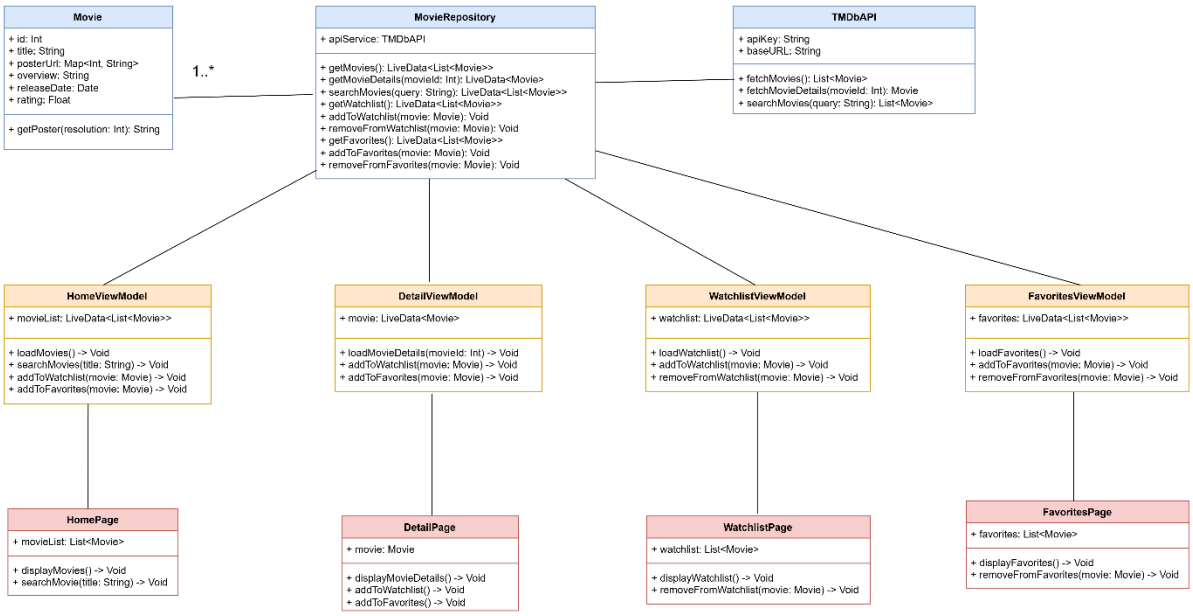
Gambar 3. 15 Activity Diagram 5

3.7.3 Class Diagram

Dalam pengembangan aplikasi studi kasus 'MovieApp' menggunakan arsitektur MVVM (Model-View-View Model). Model berfungsi untuk menangani data dan logika bisnis aplikasi. Data model dalam aplikasi ini mencakup informasi seperri judul, sinopsis, poster, tanggal rilis, dan lain-lain yang diambil dari API TMDb.

View digunakan untuk merepresentasikan antarmuka yang berinteraksi dengan pengguna, misalnya Activity dan Fragment yang menampikan daftar film, detail film, daftar *watch list,* dan daftar *favorite film.*

ViewModel digunakan untuk menghubungkan Model dan View yang bertindak sebagai lapisan logika aplikasi yang mengelola data dan merespon interaksi dari View.



Gambar 3. 16 Class Diagram

Gambar diagram kelas menunjukkan arsitektur aplikasi yang terdiri dari 4 lapisan, yaitu lapisan UI (User Interface), lapisan ViewModel, lapisan Repository, dan lapisan API (Application Programming Interface).

Lapisan UI terdiri dari empat halaman: HomePage, DetailPage, WatchlistPage, dan FavoritesPage. Setiap halaman bertanggung jawab untuk menampilkan informasi tertentu kepada pengguna. Lapisan ViewModel terdiri dari empat ViewModel: HomeViewModel, DetailViewModel, WatchlistViewModel, dan FavoritesViewModel. Setiap ViewModel bertanggung jawab untuk mempersiapkan data untuk halaman yang sesuai di lapisan UI. Lapisan Repository terdiri dari satu kelas: MovieRepository. MovieRepository bertanggung jawab untuk mengakses data film dari lapisan API dan menyediakannya ke ViewModel. Lapisan API terdiri dari satu kelas: TMDbAPI. TMDbAPI bertanggung jawab untuk berkomunikasi dengan API TMDb untuk mengambil data film.

Diagram kelas menunjukkan bagaimana setiap lapisan berinteraksi satu sama lain untuk menampilkan data film kepada pengguna.

Secara sederhana, aplikasi ini mengambil data film dari API TMDb melalui lapisan API dan kemudian menyediakan data tersebut ke lapisan UI melalui ViewModel dan Repository.