BAB 3
PELAKSANAAN KERJA MAGANG

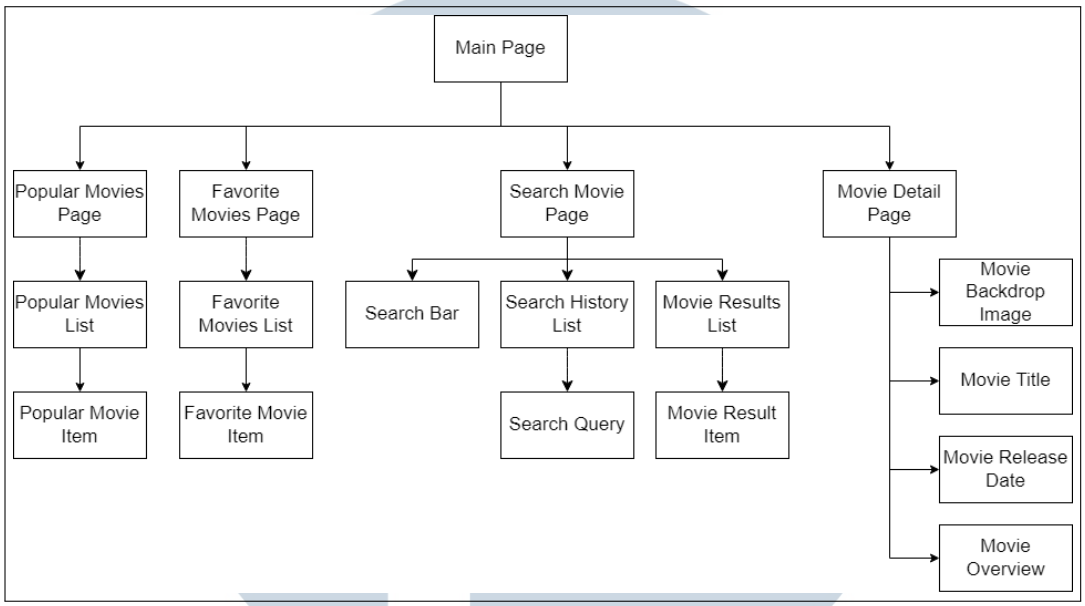
3.1 Kedudukan dan Organisasi

Posisi di PT Global Loyalty Indonesia adalah sebagai *Android Developer Intern* dalam divisi *E-Commerce Application Development*. Program magang ini merupakan bagian dari program pelatihan PT Global Loyalty Indonesia yang disebut ”GLI Academy”. GLI Academy adalah program pelatihan khusus untuk pengembang aplikasi, baik *mobile* maupun *website*. Selama menjalani program magang, komunikasi dan koordinasi dengan rekan satu tim dan mentor dilakukan secara langsung maupun tidak langsung melalui aplikasi Telegram. Setiap peserta magang diberikan *repository* BitBucket untuk mengumpulkan tugas mingguan mereka. Hasil tugas mingguan tersebut dipisahkan dengan *branch* yang dibuat oleh masing-masing peserta magang.

3.2 Tugas yang Dilakukan

Selama melaksanakan program kerja magang di PT Global Loyalty Indonesia, peserta GLI Academy diberikan tugas atau proyek baru di tiap minggunya. Proyek pengembangan aplikasi ”The Movie App” ini adalah proyek terbesar diantara seluruh project yang telah dikerjakan, sehingga proyek inilah yang akan dibahas secara lebih mendalam. Berikut adalah penjelasan menyeluruh mengenai aplikasi ”The Movie App” dan fitur-fitur yang terdapat di dalamnya.

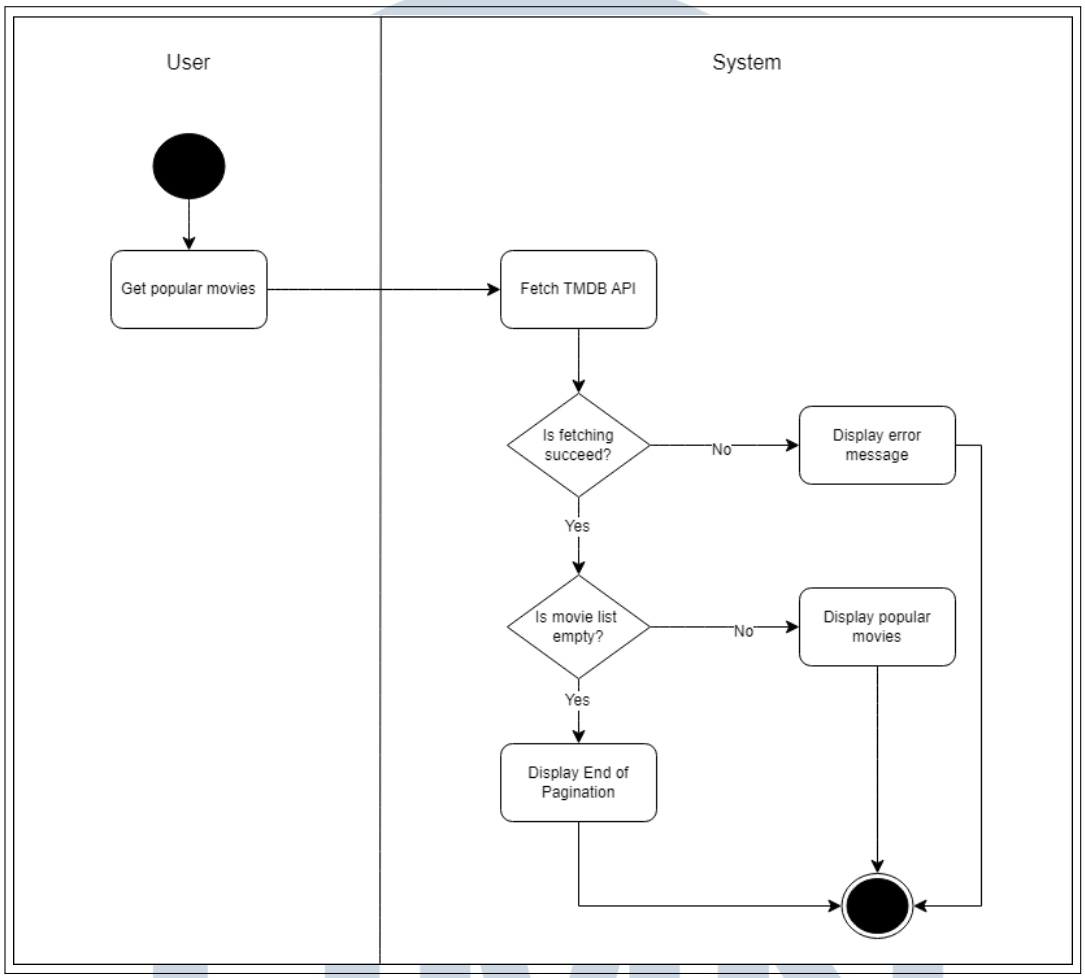
3.2.1 *Sitemap*



Gambar 3.1. *Sitemap* aplikasi ”The Movie App”

Tampilan *Sitemap* untuk aplikasi ”The Movie App” dapat dilihat pada Gambar 3.1. Pada aplikasi ”The Movie App”, terdapat satu halaman utama yaitu *Main Page* yang sudah mencakup tombol navigasi antar halaman *list* film populer, halaman *list* film favorit, dan halaman *search movie*. Ketiga halaman tersebut mempunyai komponen *movie item* yang ketika ditekan oleh pengguna, akan mengarah ke halaman *movie detail*. Halaman *Movie Detail* berisi data dari film yang ditekan oleh pengguna.

3.2.2 *List* Film Populer



Gambar 3.2. *Activity Diagram* Fitur *List* Film Populer

Tampilan *Activity Diagram* untuk fitur *List Popular Movies* pada aplikasi ”The Movie App” dapat dilihat pada Gambar 3.2. Saat *user* mengakses aplikasi ”The Movie App”, fitur yang terlihat pertama kali adalah fitur daftar film populer yang didapat dari API TMDB. Sistem akan mengambil data dari API ketika aplikasi berjalan untuk pertama kalinya. Jika *fetching* API gagal, sistem akan memberikan *error message* berbentuk Toast yang berisi informasi bahwa *fetching* API telah gagal. Jika pemanggilan API berhasil, sistem akan mengecek apakah data yang didapatkan kosong atau tidak. Jika kosong, sistem akan menampilkan *layout End of Pagination*. Lalu, ketika tidak kosong, sistem akan menampilkan daftar film populer tersebut.

Fitur *list* film populer dibangun menggunakan *library* Retrofit dan Paging 3.

Berikut adalah potongan-potongan kode untuk fitur *list* film populer.

```
class PopularMoviePagingSource(
    private val service: MovieApi
) : PagingSource<Int, MovieItem>() {

    override suspend fun load(params: LoadParams<Int>): LoadResult
    <Int, MovieItem> {
        val page = params.key ?: STARTING_PAGE_INDEX

        return try {
            val response = service.getMoviePopular(page)
            val movieItemResponse = response.results
            val movieItem = movieItemResponse.map {
                MovieItemResponse.transform(it)
            }

            LoadResult.Page(
                data = movieItem,
                prevKey = if (page == STARTING_PAGE_INDEX) null
    else page - 1,
                nextKey = if (movieItem.isEmpty()) throw
    IOException("End of Data") else page + 1
            )

        } catch (e: IOException) {
            LoadResult.Error(e)
        } catch (e: HttpException) {
            LoadResult.Error(e)
        }
    }

    override fun getRefreshKey(state: PagingState<Int, MovieItem>)
    : Int? {
        return state.anchorPosition
    }
}
```

Kode 3.1: PopularMoviePagingSource

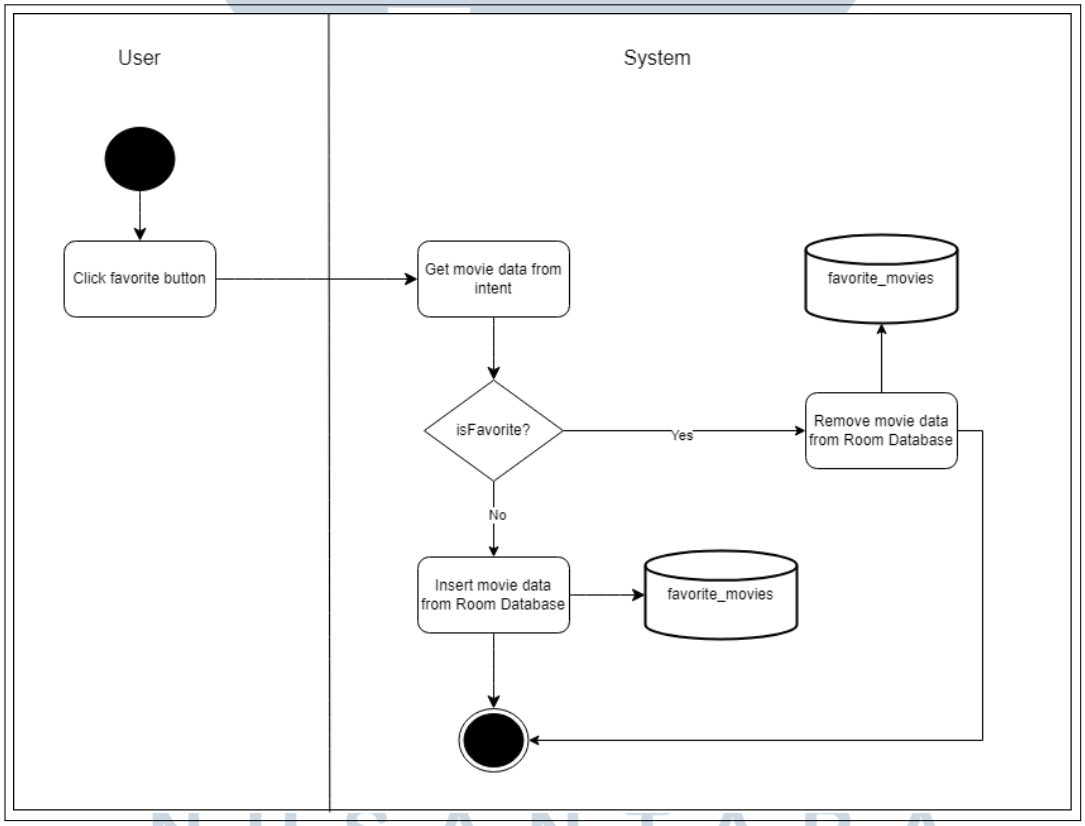
Potongan kode pada lampiran Kode 3.1 merupakan bentuk implementasi kode *library* Paging 3 yang mengatur jalannya *pagination* pada suatu halaman di aplikasi Android. Baris ke-9 dari kode di atas menjalankan aksi *fetching* API dan baris ke-17 hingga baris ke-18 menjalankan aksi validasi data yang telah didapat

dari aksi *fetching* API tersebut. Fungsi getMoviePopular() pada baris 9 adalah fungsi dari *interface* MovieApi yang mengimplementasikan *library* Retrofit untuk mengambil data dari API TMDB. Potongan kode untuk fungsi getMoviePopular() dapat dilihat pada lampiran Kode 3.2.

```
interface MovieApi {
    @GET("movie/popular")
    suspend fun getMoviePopular(
        @Query("page") page: Int
    ): MovieResponse
}
```

Kode 3.2: getMoviePopular

3.2.3 *Favorite Movie*



Gambar 3.3. *Activity Diagram* Fitur *Favorite Movie*

Tampilan *Activity Diagram* untuk fitur *favorite movie* dapat dilihat pada Gambar 3.3. Ketika *user* menekan tombol favorit pada salah satu film yang terdapat

pada daftar film populer, sistem akan mengecek terlebih dahulu apakah film yang dimaksud sudah ada di dalam *database* favorite_movies. Jika data film yang *user* sukai sudah tersedia di dalam database, sistem akan menghapus data film tersebut dari daftar film favorit user. Sebaliknya, jika data film belum ada di *database*, sistem akan memasukkan data tersebut ke dalam daftar film favorit user. Data-data film yang ada pada database favorite_movies akan ditampilkan sebagai daftar film favorit *user* di halaman *Favorite Movies*.

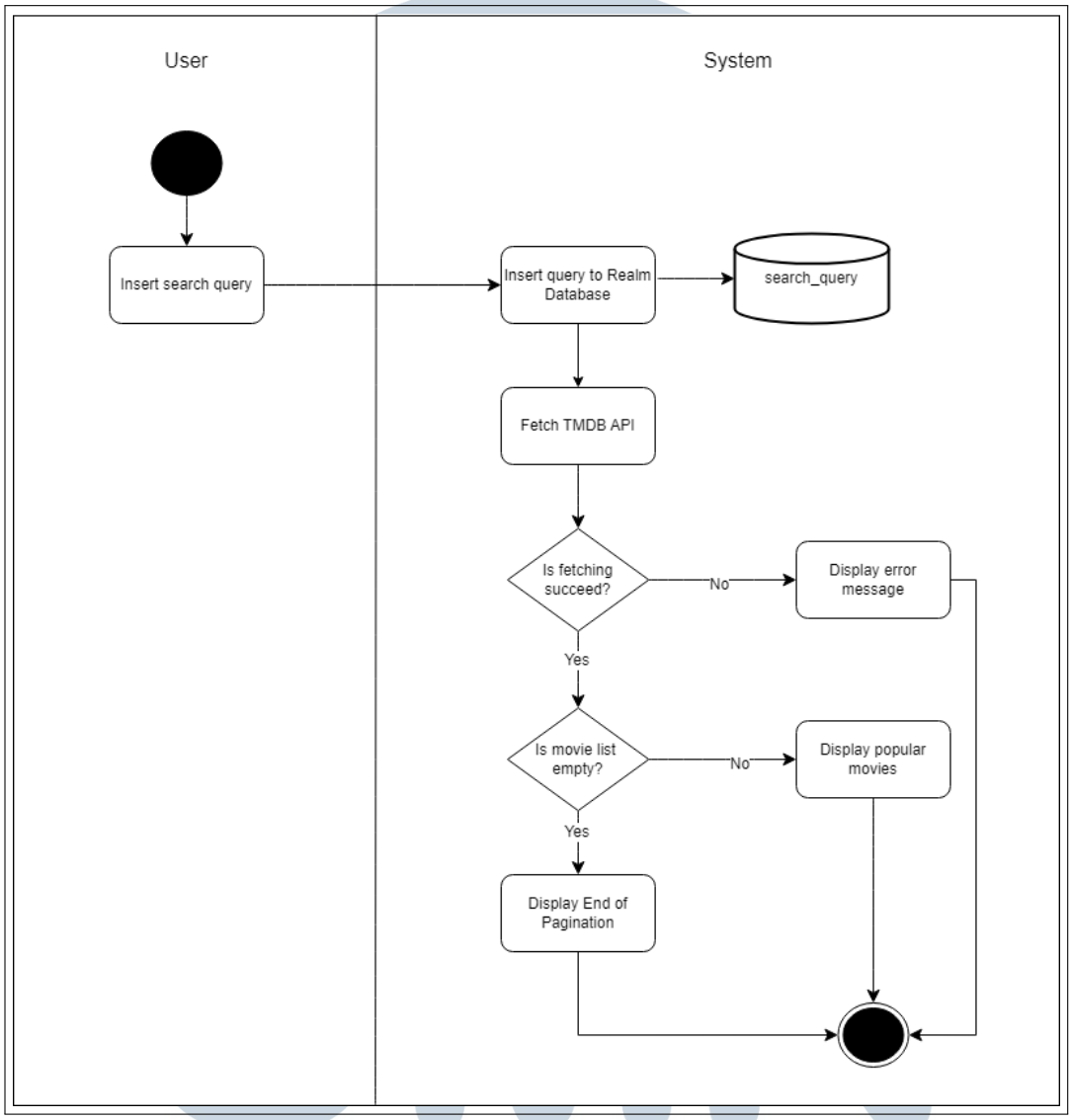
Fitur *favorite movie* dirancang menggunakan Room Local Database. *Database* tersebut dipakai untuk menyimpan data film-film yang telah disukai oleh user. Berikut adalah potongan kode implementasi Room.

```
@Dao
interface MovieDao {
    @Insert(onConflict = OnConflictStrategy.IGNORE)
    suspend fun insertFavouriteMovie(movie: MovieEntity)

    @Query("SELECT * FROM favourite_movies")
    fun getAllMovies(): LiveData<List<MovieEntity>>

    @Query("SELECT EXISTS (SELECT 1 FROM favourite_movies WHERE id
    = :id)")
    suspend fun getMovieById(id: Int): Boolean

    @Delete
    suspend fun deleteFavouriteMovie(movie: MovieEntity)
}
```

Kode 3.3: MovieDao

Potongan kode pada lampiran Kode 3.3 adalah fungsi-fungsi untuk menjadi perantara antara interaksi *user* dengan database. Ketika *user* menekan tombol hati berwarna putih, fungsi insertFavouriteMovie() akan dijalankan. Selanjutnya, jika *user* menekan tombol hati berwarna merah, fungsi deleteFavouriteMovie() akan dijalankan. Fungsi getAllMovies() dijalankan pada saat *user* membuka halaman Favourite Movies. Terakhir, fungsi getMovieById() dijalankan pada saat pengecekan apakah data film yang dimaksud *user* sudah tersedia di database favourite_movies.

3.2.4 Search Movie



Gambar 3.4. Activity Diagram Fitur Search Movies

Gambar 3.4 di atas merupakan tampilan *Activity Diagram* untuk fitur search movie. Saat *user* mengakses halaman *Search Movie*, *user* dapat memasukkan judul film yang ingin dicari di *search bar* yang tersedia di halaman tersebut. Ketika *user* menekan tombol Enter, sistem akan mengambil data dari API TMDB sesuai dengan *query* judul film yang dimasukkan oleh user. Jika *fetching* API gagal, sistem akan memberikan error message berbentuk Toast yang berisi informasi bahwa *fetching* API telah gagal. Jika pemanggilan API berhasil, sistem akan mengecek apakah data yang didapatkan kosong atau tidak. Jika kosong, sistem akan menampilkan layout

yang menginformasikan bahwa judul film yang dimasukkan *user* tidak tersedia di database API TMDB. Lalu, ketika tidak kosong, sistem akan menampilkan daftar film populer tersebut. *query* yang *user* masukkan juga akan dimasukkan ke dalam database search_query untuk ditampilkan pada halaman Search Movie.

Fitur ini dibangun menggunakan *library* Realm Database yang digunakan sebagai wadah untuk menampung *query* yang telah *user* masukkan di search bar pada halaman search movie. Berikut adalah potongan kode implementasi Realm Database pada fitur search movie.

```
class SearchHistoryRepositoryImpl @Inject constructor(
    val realm: Realm
) : SearchHistoryRepository {

    override fun getSearchHistory(): RealmResults<SearchQuery>? {
        return realm.where(SearchQuery::class.java).findAll()
    }

    override fun insertQuery(query: String) {
        try {
            realm.executeTransactionAsync { realm ->
                val searchQuery =
                    realm.createObject(SearchQuery::class.java,
    UUID.randomUUID().toString())
                searchQuery.query = query
                realm.insert(searchQuery)
            }
        } catch (e: Exception) {
            Timber.tag("SearchHistoryRepositoryImpl").e("
    insertQuery: %s", e.message)
        }
    }
}
```

Kode 3.4: SearchHistoryRepositoryImpl

Potongan kode yang dapat dilihat pada lampiran Kode 3.4 adalah fungsi-fungsi yang dijalankan pada halaman search movie. Fungsi getSearchHistory() digunakan untuk mendapatkan daftar *query* yang telah *user* masukkan sebelumnya untuk ditampilkan pada halaman search movie. Sedangkan fungsi insertQuery() adalah fungsi yang digunakan untuk memasukkan input *query user* pada halaman search movie.

3.3 Uraian Pelaksanaan Magang

PT Global Loyalty Indonesia memberikan tugas-tugas yang harus diselesaikan oleh para pekerja magang GLI Academy dalam batas waktu yang telah ditentukan. Rangkuman pekerjaan yang diberikan dapat dilihat pada Tabel 3.1.

Tabel 3.1. Uraian Pekerjaan yang Dilakukan Selama Masa Magang

Minggu Ke-	**Pekerjaan yang dilakukan**
1	*Training softskill* tentang *Presentation Skill* diikuti untuk meningkatkan kemampuan presentasi yang formal namun tidak kaku. Selain itu, dasar-dasar Kotlin seperti *Basic Syntax*, *Data Types*, dan *Control Flow* juga dipelajari. Tugas yang diberikan meliputi pembuatan *function* untuk menampilkan informasi produk, menghitung harga total, diskon, dan validasi input pengguna. Implementasi pengecekan stok produk dan nominal pembayaran juga dilakukan.
2	*Training softskill* tentang *Effective Communication* diikuti untuk memahami cara komunikasi yang baik dan efektif. Kotlin *Data Classes*, *Collections*, *Object Oriented Programming*, serta dasar-dasar *Android Development* turut dipelajari. Beberapa aplikasi seperti kalkulator, *login system*, dan penghitungan volume bangun ruang dengan validasi *username* dan *password* dikembangkan. Selain itu, penggunaan *list*, *map*, dan *collection operations* juga diaplikasikan dalam tugas-tugas tersebut.
3	*Training softskill* tentang *Work Ethics* di PT Global Loyalty Indonesia diikuti untuk menyelaraskan visi dan misi. Selain itu, komponen Android seperti *Style and Theme*, *Toolbar*, *Menu*, *Intent*, dan *RecyclerView* serta prinsip *SOLID* dipelajari. Aplikasi mirip Alfagift yang menampilkan produk, *banner*, dan *official store* dikembangkan. Implementasi navigasi dan detail produk pada aplikasi juga dilakukan.
Lanjut pada halaman berikutnya	

Tabel 3.1 Uraian Pekerjaan yang Dilakukan Selama Masa Magang (lanjutan)

Minggu Ke-	**Pekerjaan yang dilakukan**
4	*Training softskill* tentang *Team Work* diikuti untuk memahami cara bekerja efektif dalam tim. Penggunaan *SnackBar*, *BottomSheet*, *AlertDialog*, *ViewBinding*, dan aspek-aspek modern *Android Development* turut dipelajari. Aplikasi yang menggunakan API TMDB dengan *Fragment* dan *View Pager* untuk navigasi halaman dikembangkan. View Binding digunakan untuk mempermudah pengelolaan layout aplikasi.
5	*Training softskill* tentang *Result Oriented* diikuti untuk memahami cara mencapai dan mempertahankan hasil kinerja yang baik. Selain itu, *Background Task*, *Notification*, *Permission*, dan *Local Data* juga dipelajari. Pengembangan aplikasi ”The Movie App” dengan fitur offline dan penyimpanan favorit di Local Database dilanjutkan. Implementasi penanganan koneksi internet dan penyimpanan data film favorit dilakukan.
6 dan 7	*Training softskill* tentang *Analytical Thinking* diikuti untuk mengembangkan pemikiran analitik dan kritis. Metode *Fish Bone* dan *5 Whys* digunakan untuk menyelesaikan masalah. *Clean Architecture*, MVVM, Kotlin Coroutines, *Dependency Injection*, dan Paging 3 turut dipelajari. Implementasi MVVM, Paging 3, dan Dependency Injection dalam proyek dilanjutkan.
8	*Training softskill* tentang *Awareness* diikuti untuk menjadi pribadi yang lebih peka terhadap lingkungan sekitar. Pengerjaan aplikasi ”The Movie App” dengan penanganan *End Of Pagination* menggunakan Paging 3 dilanjutkan. Selain itu, implementasi MVVM untuk mempermudah pengelolaan folder, fitur, dan kode aplikasi dilakukan. *Dependency Injection* juga diterapkan pada setiap *View Model*, *Activity*, dan *Fragment* yang digunakan.
Lanjut pada halaman berikutnya	

Tabel 3.1 Uraian Pekerjaan yang Dilakukan Selama Masa Magang (lanjutan)

Minggu Ke-	**Pekerjaan yang dilakukan**
9	Pengembangan aplikasi ”The Movie App” dengan implementasi Paging 3 dan *Dependency Injection* dilanjutkan. Perbaikan setiap komentar dari mentor di BitBucket untuk mengoptimalkan aplikasi juga dilakukan. *Weekly review* diadakan untuk meninjau pemahaman materi dan mengidentifikasi kesulitan yang dihadapi. Kolaborasi antara mentor dan peserta ditingkatkan melalui *review* ini.
10	Penambahan fitur penyimpanan di *Local Database* menggunakan Room dan Realm untuk fitur *Favorite* dan *Search* pada aplikasi ”The Movie App” dilakukan. Selain itu, perbaikan setiap komentar dari mentor untuk meningkatkan kinerja aplikasi juga dikerjakan. *Weekly review* diadakan untuk mengevaluasi perkembangan dan pemahaman. Fokus pada minggu ini dialokasikan pada optimalisasi aplikasi berdasarkan masukan dari mentor.
11 dan 12	Pengembangan aplikasi ”My Nearest Gas Station” menggunakan Google Maps API untuk menampilkan pom bensin terdekat dilakukan. Penanganan izin lokasi dan pergerakan *user* di MapView diimplementasikan untuk mengikuti lokasi terbaru. Selain itu, perbaikan setiap komentar dari mentor untuk meningkatkan kinerja aplikasi, terutama pada penanganan perizinan lokasi, dilakukan. *Weekly review* diadakan untuk mengevaluasi kemajuan dan menerima masukan dari mentor.
13	Pembuatan aplikasi kamera yang dapat mengambil foto dan menyimpannya di galeri dengan implementasi *Local Database* Room dilakukan. Fitur *scan* menggunakan Google MLKit dan *QR Code Generator* menggunakan *library* ZXing ditambahkan berdasarkan input pengguna. Perbaikan aplikasi berdasarkan komentar yang diberikan oleh mentor juga dilakukan. *Weekly review* diadakan untuk mengevaluasi dan mengimplementasikan masukan dari mentor agar aplikasi menjadi lebih optimal.

Lanjut pada halaman berikutnya

Tabel 3.1 Uraian Pekerjaan yang Dilakukan Selama Masa Magang (lanjutan)

Minggu Ke-	Pekerjaan yang dilakukan
14	Refaktorisasi struktur kode aplikasi ”The Movie App” dari MVVM (*Model-View-View Model*) ke MVVM dan *Clean Architecture*. Refaktorisasi kode ini dilakukan untuk menguasai pengetahuan dan implementasi *Clean Architecture* yang akan menjadi bekal untuk pengerjaan projek-projek Alfagift di masa depan.

3.4 Perangkat Penunjang

Selama pelaksanaan kerja magang di PT Global Loyalty Indonesia, terdapat perangkat lunak dan perangkat keras yang digunakan untuk mengerjakan tugas atau proyek tiap minggunya. Daftar perangkat lunak yang digunakan adalah sebagai berikut:

1. Android Studio
2. Bitbucket
3. Google Chrome
4. Postman
5. Telegram

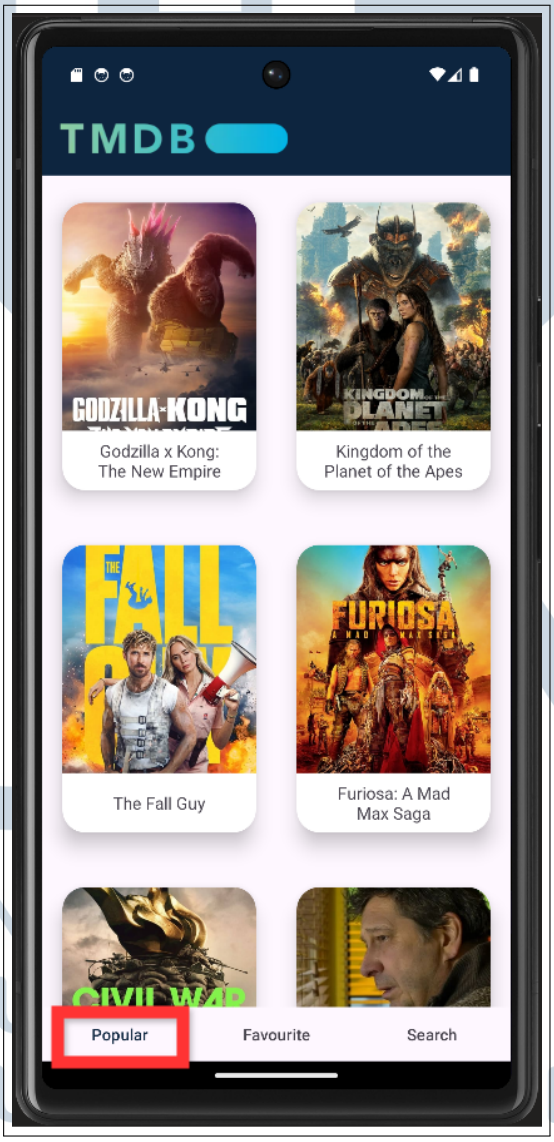
Lalu, daftar spesifikasi perangkat keras yang digunakan adalah sebagai berikut:

1. Operating System: Windows
2. RAM: 16 GB
3. Storage: 512 GB SSD
4. Processor: 11th Generation Intel Core i7-1165G7

3.5 Implementasi

3.5.1 List Film Populer

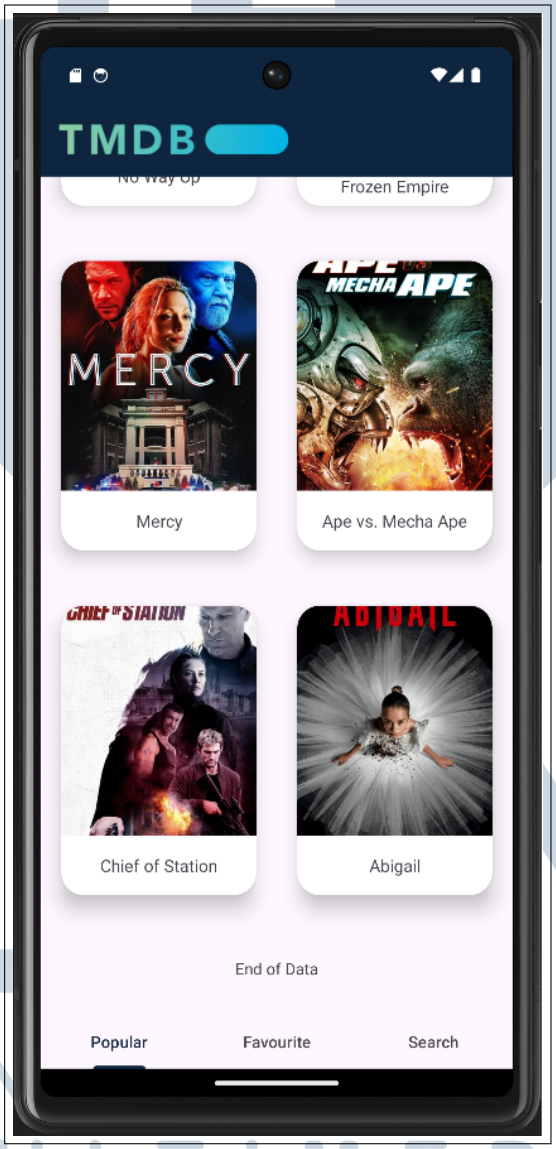
Fitur *list* film populer adalah halaman pertama yang terlihat ketika *user* membuka aplikasi ”The Movie App”. *user* juga dapat mengakses halaman fitur *list* film populer dengan cara menekan tombol bertuliskan ”Popular” yang terdapat pada bagian kiri bawah halaman tersebut. Tombol ”Popular” dan tampilan dari fitur *list* film populer dapat dilihat pada Gambar 3.5.



Gambar 3.5. Fitur *list* film populer pada aplikasi ”The Movie App”

Pada Gambar 3.5, data film yang ditampilkan

didapatkan dari hasil pemanggilan API TMDB dengan *query* ”https://api.themoviedb.org/3/movie/popular”. Setelah data film-film berhasil didapatkan, daftar film tersebut akan ditampilkan menggunakan *library* Paging 3. Ketika data film populer sudah habis, akan ada tampilan End of Pagination seperti pada Gambar 3.6.

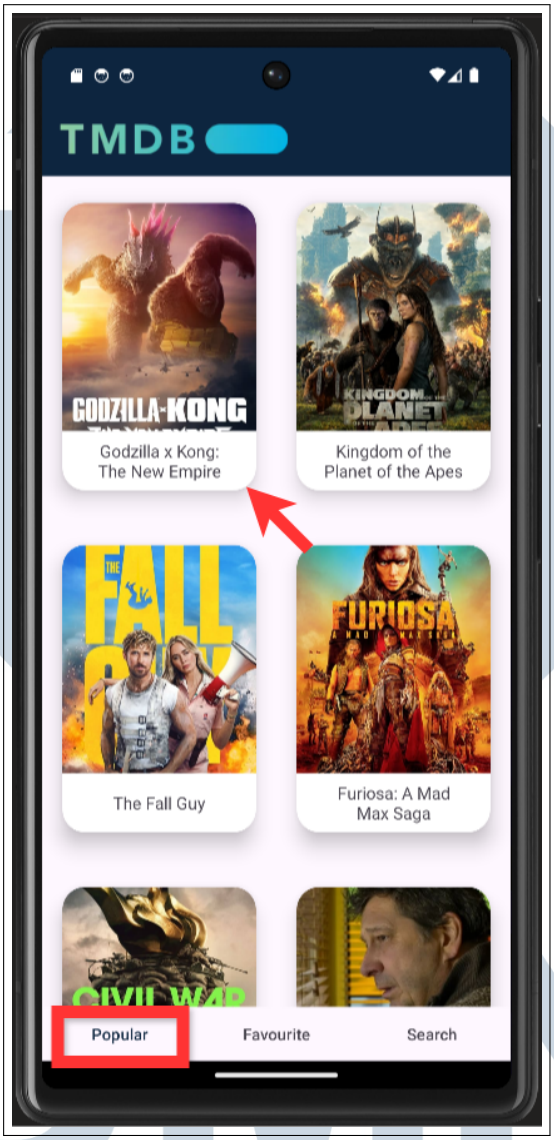


Gambar 3.6. Tampilan End of Pagination pada fitur *list* film populer

3.5.2 Favorite Movie

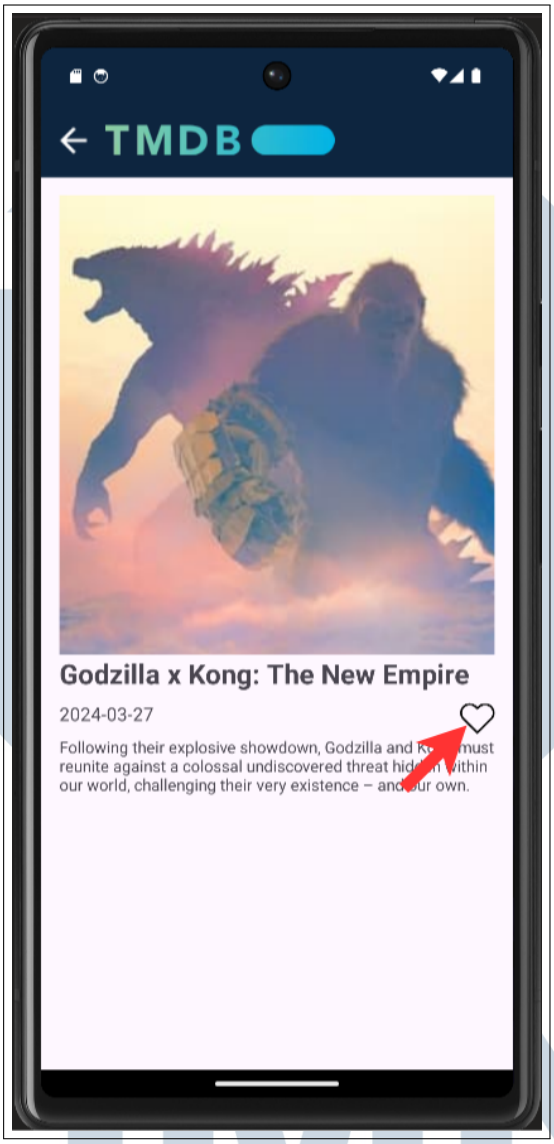
Fitur *favorite movie* terdapat pada halaman movie detail yang dapat diakses dengan cara menekan item film yang terdapat di halaman fitur *list* film populer.

Langkah ini dapat dilihat pada Gambar 3.7.



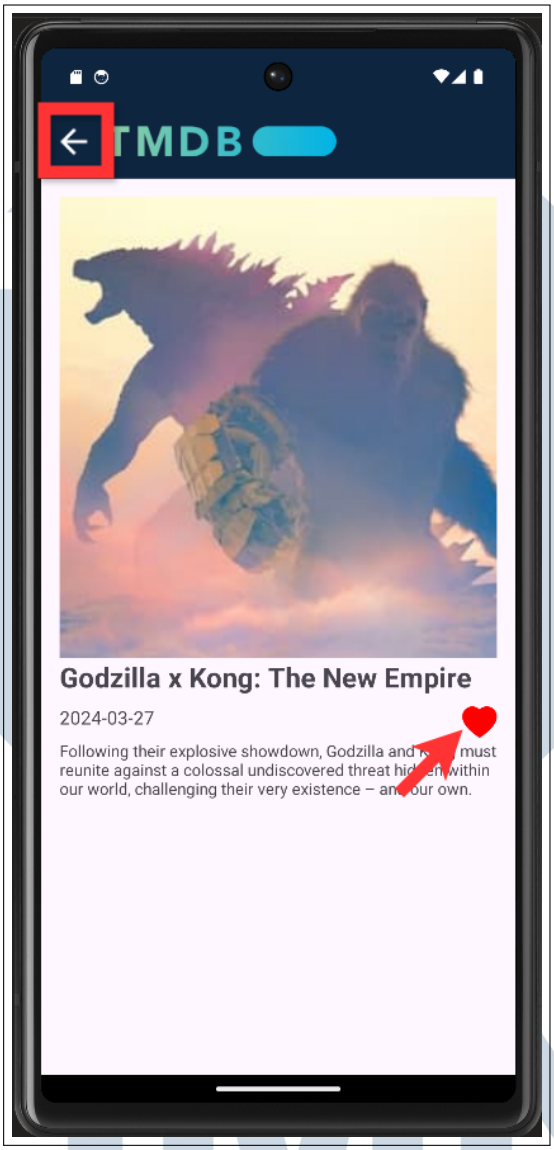
Gambar 3.7. Langkah pertama untuk mengakses halaman movie detail

Setelah menjalankan aksi pada Gambar 3.7, halaman movie detail yang berisi poster, judul, tanggal rilis, dan overview film yang ditekan *user* akan tampak pada layar. Untuk menjalankan fitur *favorite movie*, *user* dapat menekan tombol berbentuk hati yang terdapat di bagian tengah kanan halaman tersebut. Tampilan dari aksi yang dapat dijalankan *user* untuk fitur *favorite movie* dapat dilihat pada Gambar 3.8.



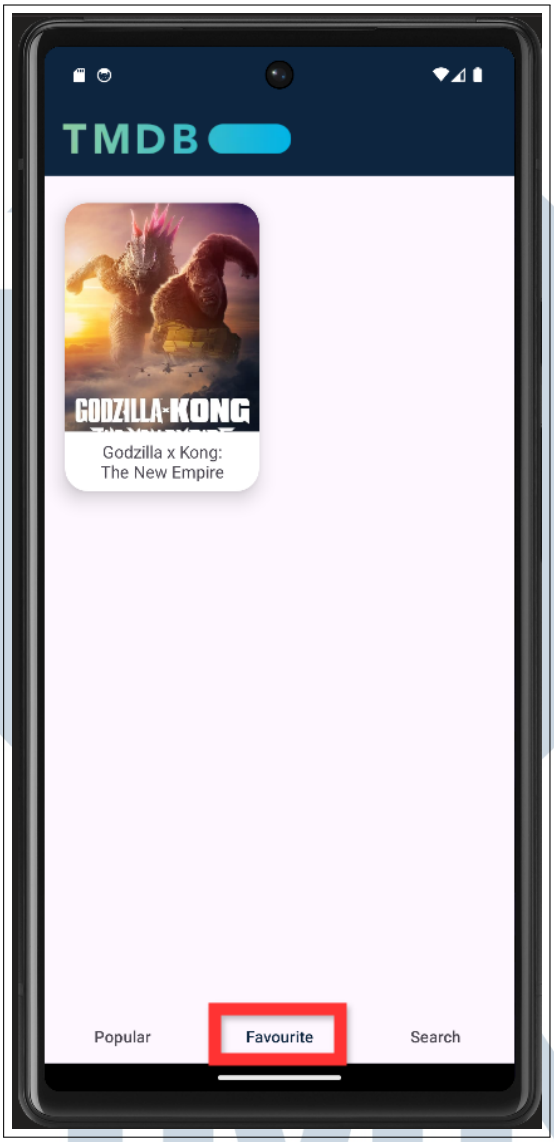
Gambar 3.8. Tampilan halaman *movie detail* dan tombol hati fitur *favorite movie*

Tombol hati fitur *favorite movie* akan berubah menjadi warna merah ketika *user* selesai menekan tombol tersebut. Tombol hati yang sudah berwarna merah juga menandakan bahwa data film yang terlihat sudah masuk ke dalam *database* favorite_movies. Tampilan dari tombol hati yang berubah menjadi warna merah terlihat pada Gambar 3.9



Gambar 3.9. Tampilan tombol hati berwarna merah pada fitur *favorite movie*

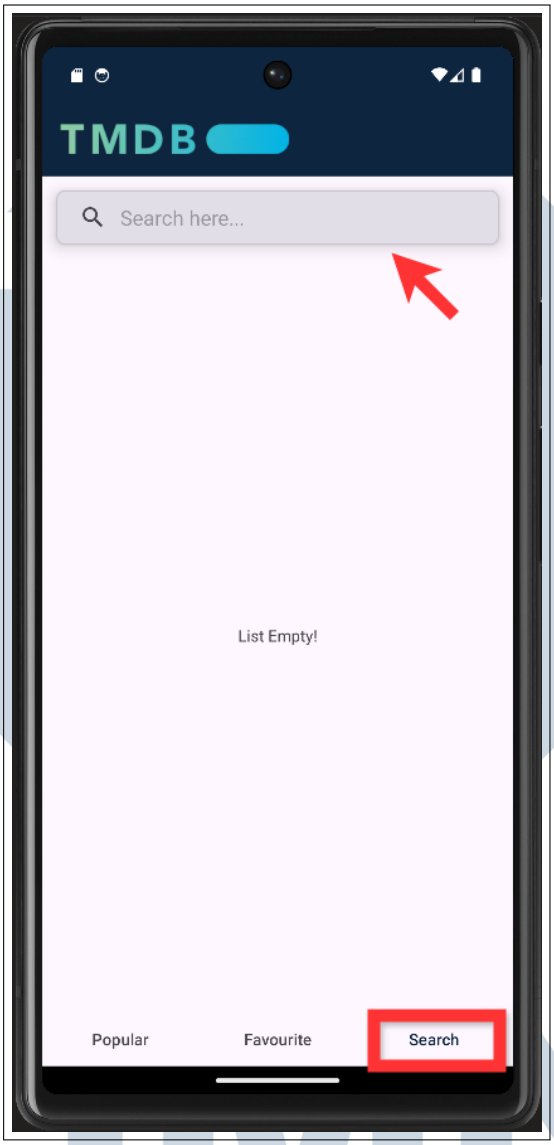
Untuk melihat daftar film yang sudah *user* favoritkan, *user* dapat mengakses halaman *favorite movies* dengan cara menekan tombol bertuliskan ”Favourite” di posisi tengah bawah layar perangkat *user*. Tampilan dari halaman *favorite movies* dapat dilihat pada Gambar 3.10.



Gambar 3.10. Daftar film yang didapatkan dari fitur *favorite movie*

3.5.3 Search Movie

Fitur terakhir yang terdapat pada aplikasi ”The Movie App” adalah pencarian film berdasarkan judul. *user* dapat mengakses halaman yang berisi fitur ini dengan cara menekan tombol bertuliskan ”Search” pada bagian kiri bawah halaman *Main*. Tampilan dari halaman ini dapat dilihat pada Gambar **??**.

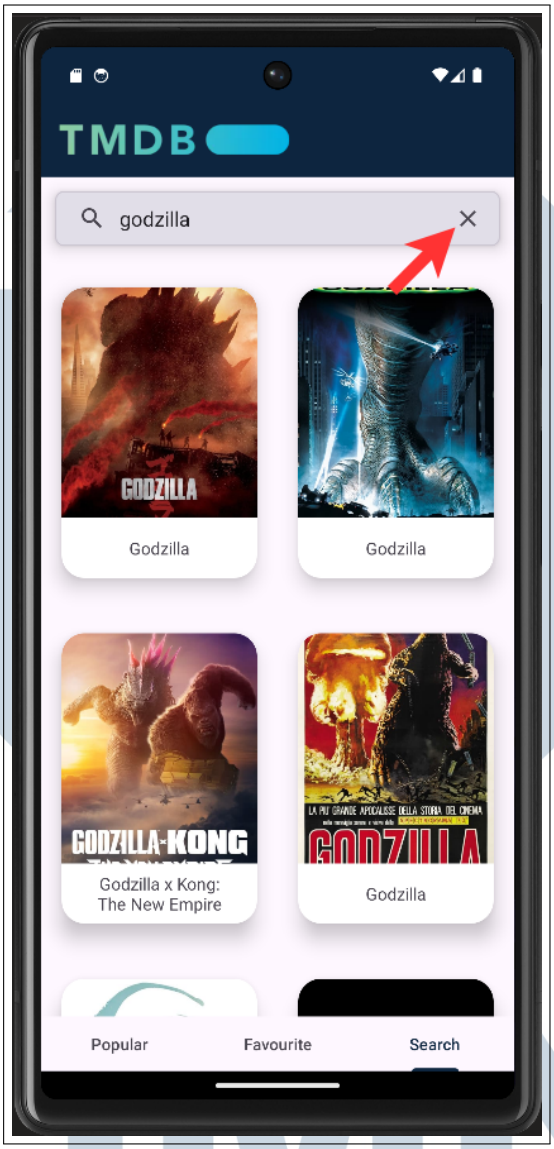


Gambar 3.11. Halaman *search movie* pada aplikasi ”The Movie App”

Pada Gambar 3.11, terdapat dua elemen yang akan terlihat pertama kali, yaitu *search bar* dan *query history list*. *user* dapat mulai mencari judul film dengan cara menekan *search bar* yang berada di atas halaman *search movie* dan memasukkan judul film yang ingin dicari dengan cara diketik. Tampilan dari langkah selanjutnya terdapat pada Gambar 3.12.

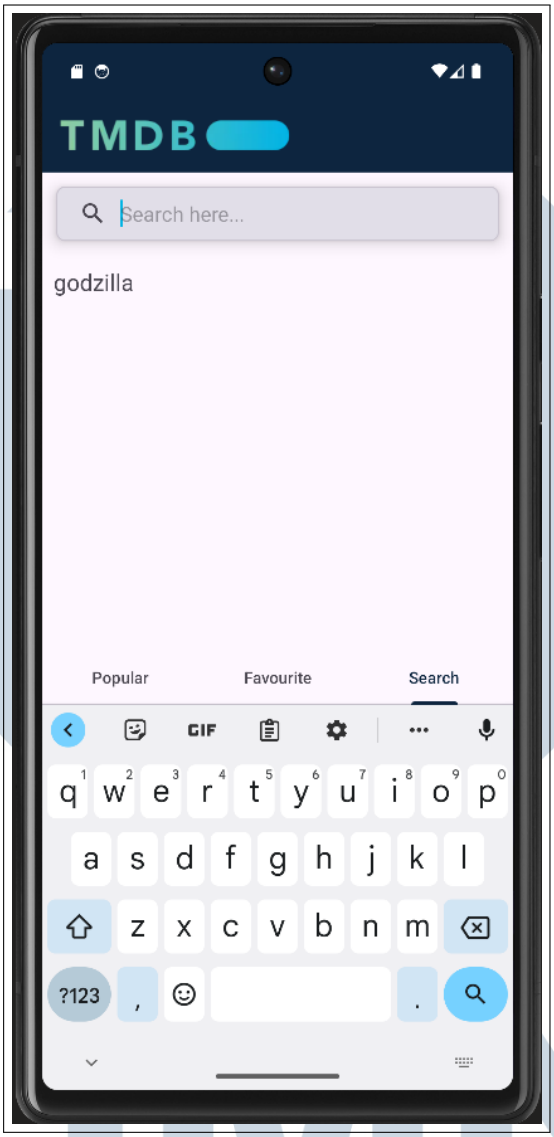
Gambar 3.12. Tampilan ketika *user* memasukkan *query* di *search bar* halaman *search movie*

Pada Gambar 3.12, kata ”godzilla” telah diketik *user* pada *search bar* yang tertera di halaman search movie. Langkah selanjutnya untuk mencari film dengan judul yang mempunyai kata ”godzilla” di dalamnya adalah dengan menekan tombol ber-*icon* kaca pembesar yang terdapat pada *keyboard handphone* yang sedang digunakan *user*.



Gambar 3.13. Tampilan dari daftar film yang didapatkan dari *query* ”godzilla”

Tampilan daftar film yang didapatkan dari *query* ”godzilla” dapat dilihat pada Gambar 3.13. Masing-masing *item* dari daftar tersebut dapat ditekan untuk mengakses halaman *movie detail* seperti pada Gambar 3.8. Untuk menampilkan daftar riwayat *query* yang *user* masukkan sebelumnya, *user* dapat menekan tombol ”X” pada bagian kanan *search bar* halaman *search movie*.



Gambar 3.14. Fitur search movie pada aplikasi ”The Movie App”

Seperti yang dapat dilihat pada Gambar 3.14, *query* ”godzilla” terdapat pada urutan pertama *list* riwayat *query user*. *Search bar* pada halaman *search movie* pun sudah dapat diisi lagi dengan *query* yang baru untuk mencari daftar film dengan judul yang diinginkan *user*. Daftar riwayat *query* pencarian *user* akan terus bertambah seiring dengan input *query* yang *user* masukkan pada *search bar* halaman *search movie*.

3.6 Kendala dan Solusi yang Ditemukan

Selama menjalani program magang di PT Global Loyalty Indonesia, kendala signifikan dialami pada tahap awal pengembangan aplikasi berbasis Android. Kendala ini meliputi perlunya pemahaman dan penerapan konsep penting seperti Kotlin *Code Convention*, prinsip-prinsip SOLID (*Single Responsibility Principle*, *Open-Closed Principle*, *Liskov Substitution Principle*, *Interface Segregation Principle*, *Dependency Inversion Principle*), *Modern Android Development*, *Android Architecture Component*, MVVM (*Model-View-ViewModel*), *Dependency Injection*, dan *Clean Architecture*. Proses pengembangan menjadi terhambat karena banyak waktu yang harus dialokasikan untuk mempelajari alur serta struktur tersebut terlebih dahulu sebelum dapat berkontribusi secara efektif dalam proyek.

Solusi yang telah dilakukan untuk mengatasi kendala tersebut adalah dengan mempelajari dan menerapkan Kotlin *Code Convention* terlebih dahulu untuk menjamin konsistensi penulisan kode. Lalu, melalui proyek-proyek kecil, prinsip-prinsip SOLID dikuasai secara bertahap. Setelah itu, pemahaman mendalam tentang *Modern Android Development*, *Android Architecture Component*, MVVM, *Dependency Injection*, dan *Clean Architecture* diperoleh melalui pembelajaran intensif dari berbagai sumber seperti YouTube, Udemy, dan dokumentasi-dokumentasi resmi. Diskusi rutin dengan rekan, mentor, dan supervisi magang juga menjadi bagian penting dalam memperoleh pemahaman yang lebih baik dan solusi praktis terhadap masalah yang dihadapi.