Project B Report

Description

This project is of an Android application that shows users a list of popular movies and the details all the movies. In the main screen, we will load a list of popular movies, showing the poster of the movie, it's released year and the ratings (out of 10). The list of popular movies are fetched using the Movie DB (TMDB) API.

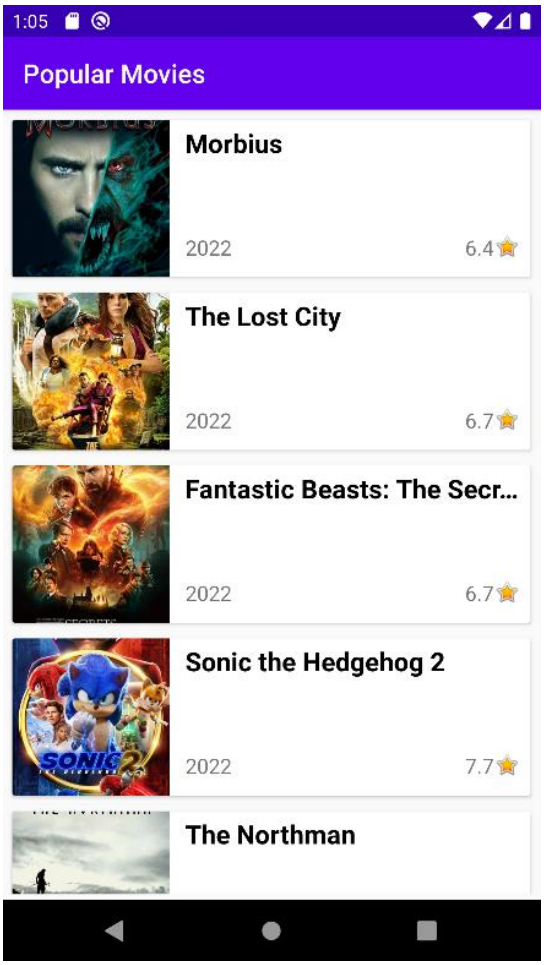

Figure 1: Screenshot of the Movie List

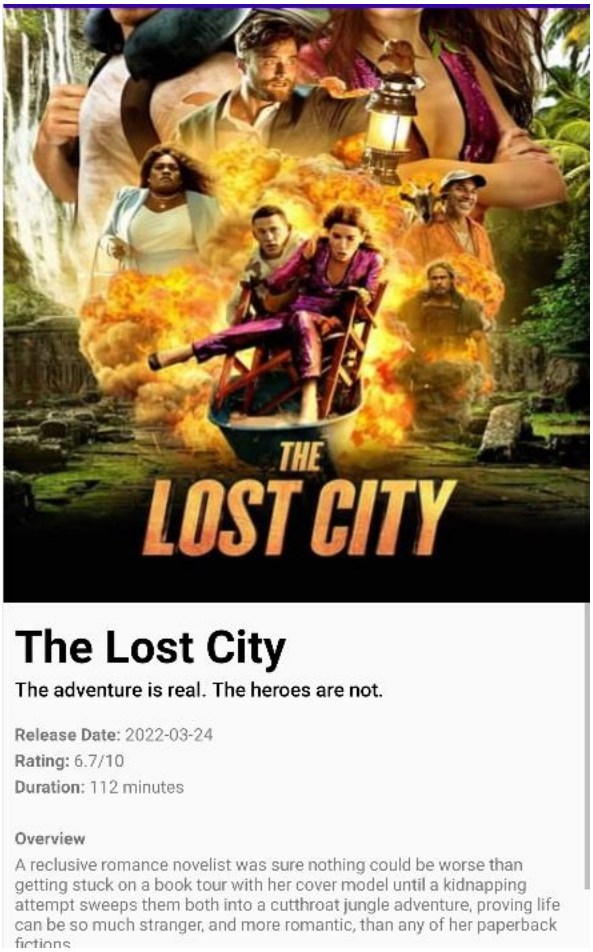

Figure 2: Screenshot of Movie Details Page

Motivation

WIth films being an ever-prominent media, we wanted to create an application that shows a comprehensive, and updated list of movies that are available for viewing in theatres. We were particularly interested in applying what we have learnt on Android application development and API calls throughout the Tiktok Coding Camp in this application. There was also a keen interest in applying our learnings on UI standards to give the page a cleaner, more standard look.

The main feature, ie the movie list, can be easily integrated with further features such as allowing for public comments and ratings. This can potentially make the application more engaging.

Aim

The aim of our application is to provide users a list of new popular movies and their details so that users can getsome ideas of what to watch during their free time.

User Stories

To specify the functional requirements of our application, we have brainstormed and listed down some user stories which can help us identify the necessary features for our minimum viable product (MVP). From then on, this list can be progressively added so that we can take an iterative development as our software development approach.

These are some of the user stories we have come up with for our application:

1. As a user, I can see a visual list of updated popular movies so I can get some recommendations of what to watch.
2. As a user, I can see the ratings of the movies so that I can get a rough gauge of how good a movie is based on others' reviews.
3. As a user, I can see a brief overview of the movies so that I can get a gist of what the movie is about.

Feature List

1. Home page: Shows a list of popular movies (currently)
2. Details page: Tapping on a movie item in the list will navigate user to a Movie Detail page which shows more necessary information about the movie, namely:
    a. Sub-title
    b. Release Date
    c. Ratings
    d. Duration
    e. Gist of movie

Non-Functional Requirements

1. Navigation from home page to details page should not take more than 1 second.
2. Loading of movie posters upon non-scrolling movement should not take more than 1 second.

Technical Overview

The application is built using Android Software Development Kit (SDK) on Android Studios, and the language used was Kotlin. The reason why our group chose Kotlin is because Kotlin is simpler to use and less rigid than Java, which makes development much easier for beginners like us. Kotlin also has higher performance and scalability, as mentioned by the course conductors for Android development. Lastly, Kotlin also has null safety feature, which prevents errors that are usually caused by undefined types or parameters that are not initialized correctly, which can help to reduce compile time and run time errors.

For libraries, we have employed Retrofit for secure HTTP requests in our application. This enables the REST API to be specified as an interface, and we can simply use annotations to create the HTTP functions and manage its headers, body, and query parameters.

We also implemented lazy loading for our main activity with the help of Glide, which helps to ensure that we are able to scroll through a long list of images without a major reduction in

performance. RecyclerView together with lazy loading ensures that the our main activity will not be too laggy and not all images are rendered at the same time.

The API used in our application was TMDB, and this is the link to its documentation: https://www.themoviedb.org/documentation/api. TMDB is a free open source API that allows us to fetch a list of movies, which can be of different categories, such as “popular movies”, “now playing movies”, “top rated movies” etc. It also support different types of operations, which can be found in this link: https://developers.themoviedb.org/3. Requesting of API Key was a simple process and it took only 1 day for the application process to be approved. We also decided to proceed with using this API because the documentation is very verbose and concise, which is very easy for us to understand and implement. This API also supports pagination, which helps in implementation of lazy loading.

We also used okhttp library to use the interceptor to inject the API key to the API requests with the API key query parameter. We found that this is a popular way for developers to inject API key to the requests. The way we hide our API key was to use local properties and adding the API key to the build configuration of our application. We have also tried adding it to gradle properties but we realised that the gradle properties file was pushed to the repository and it did not work out for us in restricting the API key from the public. Since the local properties file are not meant to be pushed to the repository, it served as a way for our team to restrict the API key from the public. However, we do note that the local properties file stated that the file should not be edited, so we do believe that this is not the best way to hide the API key and there are other ways that we have not explored due to the lack of time.

For our project, we followed the Model-View-ViewModel architecture.

Product Overview

The application home page features a list of movies. Each movie is reflected along with its title, release year and up-to-date ratings (figure 1).

Upon tapping on a movie, the application will navigate to the movie details page which has the essential information about the movie.

- Release Date
- Duration
- Overview

This is a short demonstration of our application:
https://drive.google.com/file/d/1qIneY77Mfer3bZNeCYfVoh5fkIpSAA-D/view?usp=sharing

This video also show cases the lazy loading of the main page where there is no lag when scrolling quickly because images are not rendered immediately.

Highlights

The most challenging part of the application development was to resolve certain errors caused by dependencies being too new. Since none of us had experience in Android development, we faced a lot troubles when we tried to add dependencies and update it to the latest version, which caused certain errors that we were unable to figure out. We took it

straight to StackOverflow and followed the solutions to downgrade the versions, but we have no idea why downgrading those versions worked.

There were also difficulty building the APK file because of an error caused by OS independent files in META-INF.

Building an android application is also a whole new experience for us, and we found it a lot more difficult to learn as compared to Web development.

Project Management

Boilerplate Code Setup	Marcus
Home Page UI	Micaella
Movie API Interface	Marcus
Movie Detail Page UI	Ming Hao
Unit Test	Gloria