UDACITY

Popular Movies App Implementation Guide

Project Summary

Most of us can relate to kicking back on the couch and enjoying a movie with friends and family. In this project, you'll build an app to allow users to discover the most popular movies playing.

You'll build the complete functionality of this app in **two** stages which you will submit separately.

Stage 1: Main Discovery Screen, A Details View, and Settings

User Experience

In this stage you'll build the core experience of your movies app.

Your app will:

- Upon launch, present the user with an grid arrangement of movie posters.
- Allow your user to change sort order via a setting:
    - The sort order can be by most popular, or by top rated
- Allow the user to tap on a movie poster and transition to a details screen with additional information such as:
    - original title
    - movie poster image thumbnail
    - A plot synopsis (called overview in the api)
    - user rating (called vote_average in the api)
    - release date

Stage 1 - Implementation Guidance

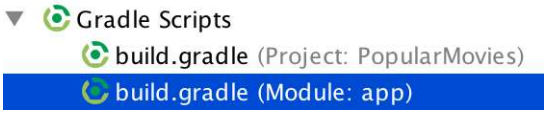
Image Library - Picasso

How to Setup Picasso

We recommend that this project use Picasso, a powerful library that will handle image loading and caching on your behalf. If you prefer, you're welcome to use an alternate library such as Glide.

We've included this to reduce unnecessary extra work and help you focus on applying your app development skills.

You'll need to modify the build.gradle file for your app. These modifications will happen in the build.gradle file for your module's directory, *not* the project root directory (it is the file highlighted in blue in the screenshot below).



In your app/build.gradle file, add:

```
repositories {
    mavenCentral()
}
```

Next, add **compile 'com.squareup.picasso:picasso:2.5.2'** to your dependencies block.

Using Picasso To Fetch Images and Load Them Into Views

You can use Picasso to easily load album art thumbnails into your views using:
`Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);`
Picasso will handle loading the images on a background thread, image decompression and caching the images.

Working with the themoviedb.org API

A note on resolving poster paths with themoviedb.org API

You will notice that the API response provides a relative path to a movie poster image when you request the metadata for a specific movie.

For example, the poster path return for Interstellar is "/nBNZadXqJSdt05SHLqgT0HuC5Gm.jpg"

You will need to append a base path ahead of this relative path to build the complete url you will need to fetch the image using Picasso.

It's constructed using 3 parts:

1. The base URL will look like: http://image.tmdb.org/t/p/.
2. Then you will need a 'size', which will be one of the following: "w92", "w154", "w185", "w342", "w500", "w780", or "original". For most phones we recommend using "w185".
3. And finally the poster path returned by the query, in this case "/nBNZadXqJSdt05SHLqgT0HuC5Gm.jpg"

Combining these three parts gives us a final url of http://image.tmdb.org/t/p/w185//nBNZadXqJSdt05SHLqgT0HuC5Gm.jpg

This is also explained explicitly in the API documentation for /configuration.

Stage 1 - API Hints

1. To fetch popular movies, you will use the API from themoviedb.org.
    - If you don't already have an account, you will need to create one in order to request an API Key.
        - In your request for a key, state that your usage will be for educational/non-commercial use. You will also need to provide some personal information to complete the request. Once you submit your request, you should receive your key via email shortly after.
    - In order to request popular movies you will want to request data from the /movie/popular and /movie/top_rated endpoints. An API Key is required.
    - Once you obtain your key, you append it to your HTTP request as a URL parameter like so:
        - http://api.themoviedb.org/3/movie/popular?api_key=[YOUR_API_KEY]
    - You will extract the movie id from this request. You will need this in subsequent requests.

IMPORTANT: PLEASE REMOVE YOUR API KEY WHEN SHARING CODE PUBLICALLY

If you ever upload your code to a public GitHub repo to share with other students or instructors, it's illegal to include your personal themoviedb.org API key. Please remove it and note in a README where it came from, so someone else trying to run your code can create their own key and will quickly know where to put it. Instructors and code reviewers will expect this behavior for any public GitHub code.

If you happen to commit your API key into version control on accident, check out this GitHub guide on removing sensitive data in order to remove it.

IMPORTANT: PLEASE MAKE SURE YOUR APP CONFORMS TO THE QUALITY GUIDELINES MENTIONED HERE

- You must make sure your app does not crash when there is no network connection! You can see this StackOverflow post on how to do this. **If your app crashes when there is no network connection, you will not pass the project.**

Stage 1 - Additional Resources

These Webcast resources are only available for students enrolled in the Android Developer Nanodegree.

There are some concepts you will need to implement in this project that are not explicitly covered in the course material. We've created these Webcast Videos to supplement your learning. (Note: You need to be in the G+ community to access the webcasts. If you are already a member but are still unable to access them from the links below, try accessing them via *Your Nanodegree home page → Resources → Google+ Webcast*.)

Topic	Notes	Video
Creating and Using a Custom ArrayAdapter	Example Project	Webcast
Parcelables and onSaveInstanceState()	Suggested Article Example Project	Webcast
How to Use the Movie Database API	Example Project	Webcast

Stage 1 - Evaluation Rubric

Your project will be evaluated by a Udacity Code Reviewer according to this rubric for Stage 1.

Stage 2: Trailers, Reviews, and Favorites

User Experience

In this stage you'll add additional functionality to the app you built in Stage 1.

You'll add more information to your movie details view:

- You'll allow users to view and play trailers (either in the youtube app or a web browser).

- You'll allow users to read reviews of a selected movie.
- You'll also allow users to mark a movie as a favorite in the details view by tapping a button(star). This is for a **local** movies collection that you will maintain and does not require an API request*.
- You'll modify the existing sorting criteria for the main view to include an additional pivot to show their favorites collection.

Lastly, you'll optimize your app experience for tablet.

Stage 2 - Implementation Guidance

Working with the themoviedb.org API

Stage 2 - API Hints

1. To fetch trailers you will want to make a request to the /movie/{id}/videos endpoint.
2. To fetch reviews you will want to make a request to the /movie/{id}/reviews endpoint
3. You should use an Intent to open a youtube link in either the native app or a web browser of choice.

IMPORTANT: PLEASE MAKE SURE YOUR APP CONFORMS TO THE QUALITY GUIDELINES MENTIONED HERE

- At this point, you should have a working understanding of recreating activities using onSaveInstanceState. If you need a refresher check out this post in the Android Documentation. **If your app crashes when the screen is rotated, you will not pass the project.**

Stage 2 - Additional Resources

The Webcast resources are only available for students enrolled in the Android Developer Nanodegree.

There are some concepts you will need to implement in this project that are not explicitly covered in the course material. We've created these Webcast Videos to supplement your learning. (Note: You need to be in the G+ community to access the webcasts. If you are already a member but are still unable to access them from the links below, try accessing them via *Your Nanodegree home page → Resources → Google+ Webcast*.)

Topic	Notes	Video
Content Providers	Example Project Stetho guide - *an Android app debug tool* Stetho Tutorial Video	Webcast
RecyclerView	Example Project	Webcast
AppCompat		Webcast

Activity Lifecycle	Example Project	Webcast
Content Provider Libraries	Example Project Code to Auto-generate a ContentProvider Gist of CursorRecyclerViewAdapter	Webcast

---

Stage 2 - Evaluation Rubric

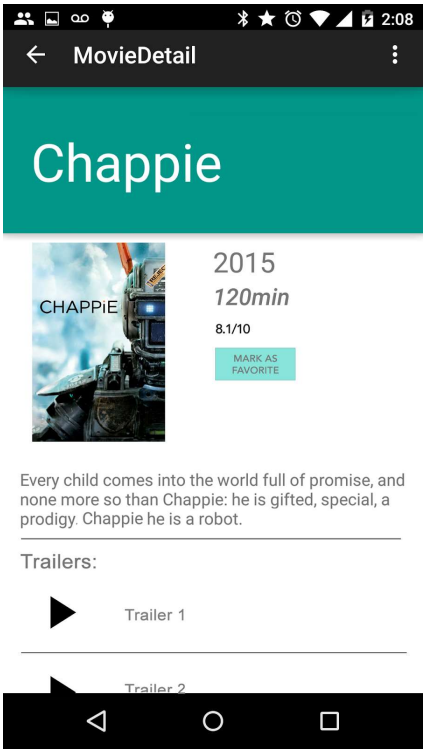
Your project will be evaluated by a Udacity Code Reviewer according to this rubric for Stage 2.

Again, please remember to remove your API key before you submit through GitHub.

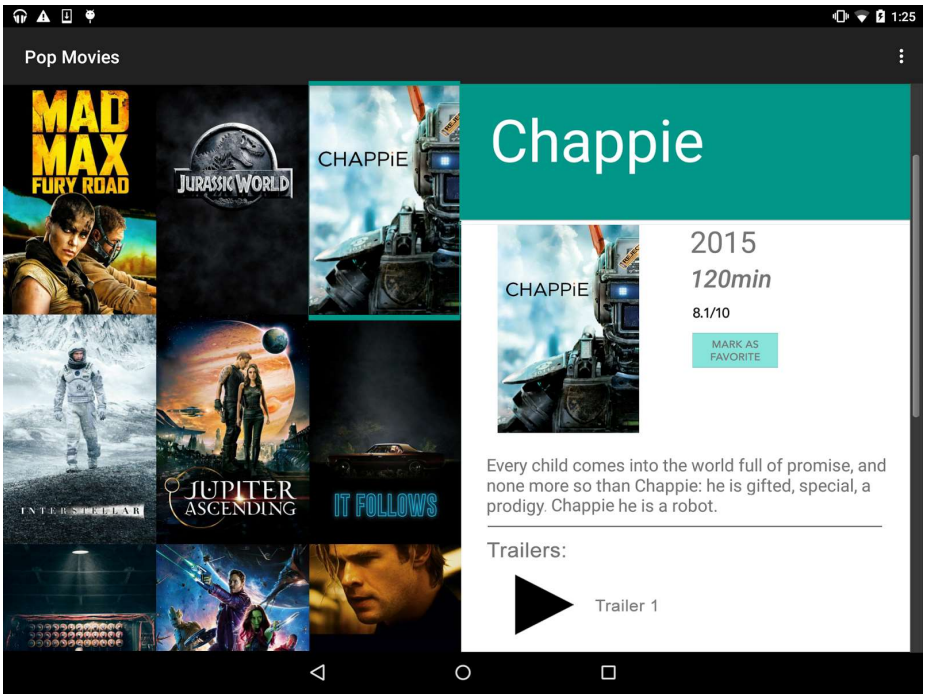
Be sure to review it thoroughly before you submit. All criteria must "meet specifications" in order to pass.

UX Mockups (Complete App)

Phone UX



Tablet UX


1:25
Pop Movies
MAD MAX FURY ROAD
JURASSIC WORLD
CHAPPiE
JUPITER ASCENDING
IT FOLLOWS
Chappie
CHAPPiE
2015
120min
8.1/10
MARK AS FAVORITE
Every child comes into the world full of promise, and none more so than Chappie: he is gifted, special, a prodigy. Chappie he is a robot.
Trailers:
Trailer 1