**Udacity Android Developer Nanodegree - Core App Quality Guidelines**

**Introduction**

This guideline acts as the official document to follow in your projects. Udacity reviewers will use this guide to grade your projects(along with the respective rubrics). All Android apps that you code in your projects should meet the criteria listed below. For a more detailed set of rules to assess the basic quality of your apps with, refer to [this](https://developer.android.com/distribute/essentials/quality/core.html) link.

|  |
| --- |
| **Visual Design and User Interaction** |
| **Standard Design** |
| * App does not redefine the expected function of a system icon (such as the Back button). * App does not redefine or misuse Android UI patterns, such that icons or behaviors could be misleading or confusing to users. |
| **Navigation** |
| * App supports standard system Back button navigation and does not make use of any custom, on-screen "Back button" prompts. * All dialogs are dismissible using the Back button. * Pressing the Home button at any point navigates to the Home screen of the device. |
| **Functionality** |
| **Permissions** |
| * App does not redefine or misuse Android UI patterns, such that icons or behaviors could be misleading or confusing to users. * App does not request permissions to access sensitive data or services that can cost the user money, unless related to a core capability of the app. |
| **User/App State** |
| * App correctly preserves and restores user or app state, that is , student uses a bundle to save app state and restores it via onSaveInstanceState/onRestoreInstanceState. For example,   + When a list item is selected, it remains selected on rotation.   + When an activity is displayed, the same activity appears on rotation.   + User text input is preserved on rotation.   + Maintains list items positions on device rotation. * When the app is resumed after the device wakes from sleep (locked) state, the app returns the user to the exact state in which it was last used. * When the app is relaunched from Home or All Apps, the app restores the app state as closely as possible to the previous state. |
| **Performance and Stability** |
| **Stability** |
| * App does not crash, force close, freeze, or otherwise function abnormally on any targeted device. |
| **Google Play** |
| **Content Policies** |
| * All content is safe for work content. * App adheres to the [Google Play Store App policies](https://play.google.com/about/developer-content-policy.html). * App’s code follows standard Java/Android Style Guidelines. |

P1 - Popular Movies App, Stage 1 Rubric

The Rubric

**Common Project Requirements**

To “meet specifications”, your app must fulfill all the criteria listed in this section of the rubric.

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Does Not Meet Specifications** | **Meets Specifications** |
| **General Project Guidelines** |  |  |
| App conforms to common standards found in the [Android Nanodegree General Project Guidelines](https://www.google.com/url?q=http://udacity.github.io/android-nanodegree-guidelines/core.html&sa=D&ust=1485636803632000&usg=AFQjCNE_I-NohdVwix_4W5sxiOL7Mxfp-A) (**NOTE: For Stage 1 of the Popular Movies App, it is okay if the app does not restore the data using onSaveInstanceState/onRestoreInstanceState**) |  |  |

**Required Components**

To “meet specifications”, your app must fulfill all the criteria listed in this section of the rubric.

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Does Not Meet Specifications** | **Meets Specifications** |
| User Interface - Layout |  |  |
| Movies are displayed in the main layout via a grid of their corresponding movie poster thumbnails |  |  |
| UI contains an element (i.e a spinner or settings menu) to toggle the sort order of the movies by: most popular, highest rated |  |  |
| UI contains a screen for displaying the details for a selected movie |  |  |
| Movie details layout contains title, release date, movie poster, vote average, and plot synopsis. |  |  |
| **User Interface - Function** |  |  |
| When a user changes the sort criteria (“most popular and highest rated”) the main view gets updated correctly. |  |  |
| When a movie poster thumbnail is selected, the movie details screen is launched |  |  |
| Network API Implementation |  |  |
| In a background thread, app queries the /movie/popular or /movie/top\_rated API for the sort criteria specified in the settings menu.  **OR**  **Acceptable until April 8, 2016:**In a background thread, app queries the /discover/movies API with the query parameter for the sort criteria specified in the settings menu. (Note: Each sorting criteria is a different API call.)  This query can also be used to fetch the related metadata needed for the detail view. |  |  |

Top Rated Movies

https://api.themoviedb.org/3/movie/top\_rated?api\_key=f4d478c237eb3ff26f0f228d47c9327f&language=en-US&page=1

Popular

https://api.themoviedb.org/3/movie/popular?api\_key=f4d478c237eb3ff26f0f228d47c9327f&language=en-US&page=1

Image Example

You'll notice that movie, TV and person objects contain references to different file paths. In order to generate a fully working image URL, you'll need 3 pieces of data. Those pieces are a base\_url, a file\_size and a file\_path.

The first two pieces can be retrieved by calling the [/configuration](https://developers.themoviedb.org/3/configuration/get-api-configuration) API and the third is the file path you're wishing to grab on a particular media object. Here's what a full image URL looks like if the poster\_path of /kqjL17yufvn9OVLyXYpvtyrFfak.jpg was returned for a movie, and you were looking for the w500 size:

https://image.tmdb.org/t/p/w500/kqjL17yufvn9OVLyXYpvtyrFfak.jpg

<https://image.tmdb.org/t/p/w500/kqjL17yufvn9OVLyXYpvtyrFfak.jpg>

JSON

Get images = get Object Root > get Array "results" > For Each get Object[i] > get String "poster\_path"

**PROJECT SPECIFICATION**

**Popular Movies, Stage 2**

User Interface - Layout

| MEETS SPECIFICATIONS |
| --- |
| UI contains an element (e.g., a spinner or settings menu) to toggle the sort order of the movies by: most popular, highest rated. |
| Movies are displayed in the main layout via a grid of their corresponding movie poster thumbnails. |
| UI contains a screen for displaying the details for a selected movie. |
| Movie Details layout contains title, release date, movie poster, vote average, and plot synopsis. |
| Movie Details layout contains a section for displaying trailer videos and user reviews. |

User Interface - Function

| MEETS SPECIFICATIONS |
| --- |
| When a user changes the sort criteria (most popular, highest rated, and favorites) the main view gets updated correctly. |
| When a movie poster thumbnail is selected, the movie details screen is launched. |
| When a trailer is selected, app uses an Intent to launch the trailer. |
| In the movies detail screen, a user can tap a button(for example, a star) to mark it as a Favorite. |

Network API Implementation

| MEETS SPECIFICATIONS |
| --- |
| In a background thread, app queries the /movie/popular or /movie/top\_rated API for the sort criteria specified in the settings menu. |
| App requests for related videos for a selected movie via the /movie/{id}/videos endpoint in a background thread and displays those details when the user selects a movie. |
| App requests for user reviews for a selected movie via the /movie/{id}/reviews endpoint in a background thread and displays those details when the user selects a movie. |

Data Persistence

| MEETS SPECIFICATIONS |
| --- |
| The titles and ids of the user's favorite movies are stored in a ContentProvider backed by a SQLite database. This ContentProvider is updated whenever the user favorites or unfavorites a movie. |
| When the "favorites" setting option is selected, the main view displays the entire favorites collection based on movie ids stored in the ContentProvider. |

General Project Guidelines

| MEETS SPECIFICATIONS |
| --- |
| App conforms to common standards found in the [Android Nanodegree General Project Guidelines](http://udacity.github.io/android-nanodegree-guidelines/core.html). |

**Suggestions to Make Your Project Stand Out!**

* Extend the favorites ContentProvider to store the movie poster, synopsis, user rating, and release date, and display them even when offline.
* Implement sharing functionality to allow the user to share the first trailer’s YouTube URL from the movie details screen.