CS478: Software Development for Mobile Platforms

Project #5

Due time: 11:59 pm on 4/29/2022
Submit using Blackboard web site
Total points: 100
Instructor: Ugo Buy

You are to code two Android apps. The first app, named *MovieCentral*, stores information about motion pictures. This information includes for each movie (1) the title of the movie, (2) the name of the movie director, and (3) a string denoting the URL of a web site containing a (short) video clip of the movie. *MovieCentral* stores information about *n* movies numbered 1 through *n*, with $n \geq 5$. The app exposes a service that supports functionality for downloading information about each movie.

The API of *MovieCentral*'s service should expose the following 3 pieces of functionality in an appropriate AIDL file:

1. Retrieve all information for all movies stored in the service.
2. Retrieve all information for one specific movie by its number (passed as a parameter).
3. Retrieve the URL string of the site with the movie's video file.

The second app, *MovieClient*, contains a main activity that supports functionality for using the *MovieCentral* app by starting and binding to its service. Once bound to the service, *MovieClient* can use the service's API to retrieve information about movies. Once the service is bound, *MovieClient* allows an interactive user to download information about each movie, display that information, and play the video clip at the URL obtained from *MovieCentral*.

In particular, the interactive user of *MovieClient*'s main activity uses key presses on appropriate buttons to bind and unbind from the service. The status of the service (i.e., whether bound or unbound) should be clearly displayed in the main activity as well. An additional UI element in the main activity allows the user to select a movie clip from the service if the service is bound. The clip will be played in *MovieClient*. In addition, when bound to the service, the client can request a list of the movies stored in the service or information about a specific movie. The list of movies is displayed in a *ListView* of a second activity of *MovieClient*. (You may use class *ListActivity*, if this convenient. For each movie (i.e., list item) the following information is displayed: (1) title of movie, and (2) name of the director. A click on a list item starts the playback using the predefined Android class *MediaController* to manage the playback in a *VideoView* UI element.

Hints. You are at liberty to choose the video clips from segments publicly-available (and not copyrighted or otherwise protected) on the Internet (e.g., YouTube). When testing your application, make sure to upload *MovieCentral* app first, or else the client app may fail to initialize properly. Use methods *startService()* and *bindService()* to start the service and to bind to the service. In app *MovieClient* use Android's built-in *MediaController* to play the clip in a *VideoView* view. You will have to set the *VideoView*'s controller to the *MediaController* instance that you created and set the *MediaController*'s anchor view to your *VideoView*. The *onPrepared()* callback of the *VideoView*'s *OnPreparedListener* will start the playback of the video stream.

Implementation notes. As with the previous projects, use a Pixel 3a XL virtual device running the usual Android platform (API 30—Q). Define the layout of your client app in such a way that it will display best in portrait mode. You are not required to provide backward compatibility with previous Android versions or to support phone reconfigurations. Use the AIDL to expose the service's functionality. Make sure that the implementation of the service is *thread-safe*. This means that access to data structures shared by multiple clients should be protected by suitable mutual exclusion locks.

Be advised that we will stress test the robustness of your client app if the service suddenly dies. In this case, the user of the client app should be given the opportunity to restart the service without restarting the client app.

You must work alone on this project. Submit a zip archive containing two root directories; each directory contains the full Android Studio repository of the corresponding app. No late submissions will be accepted.