CS 371M

Homework 3: Music Player

Submission:

All submissions should be done via git. Refer to the git setup, and submission documents for the correct procedure. Within your Android Studio directory there will be a README file that you should fill in.

Overview:

Without music to decorate it, time is just a bunch of boring production deadlines or dates by which bills must be paid.

—Frank Zappa

Sometimes you have to play a long time to be able to play like yourself.

—Miles Davis

There's always a dissonance between what you wish was happening and what is actually happening. That's the nature of creativity, that there's a certain level of disappointment in there.

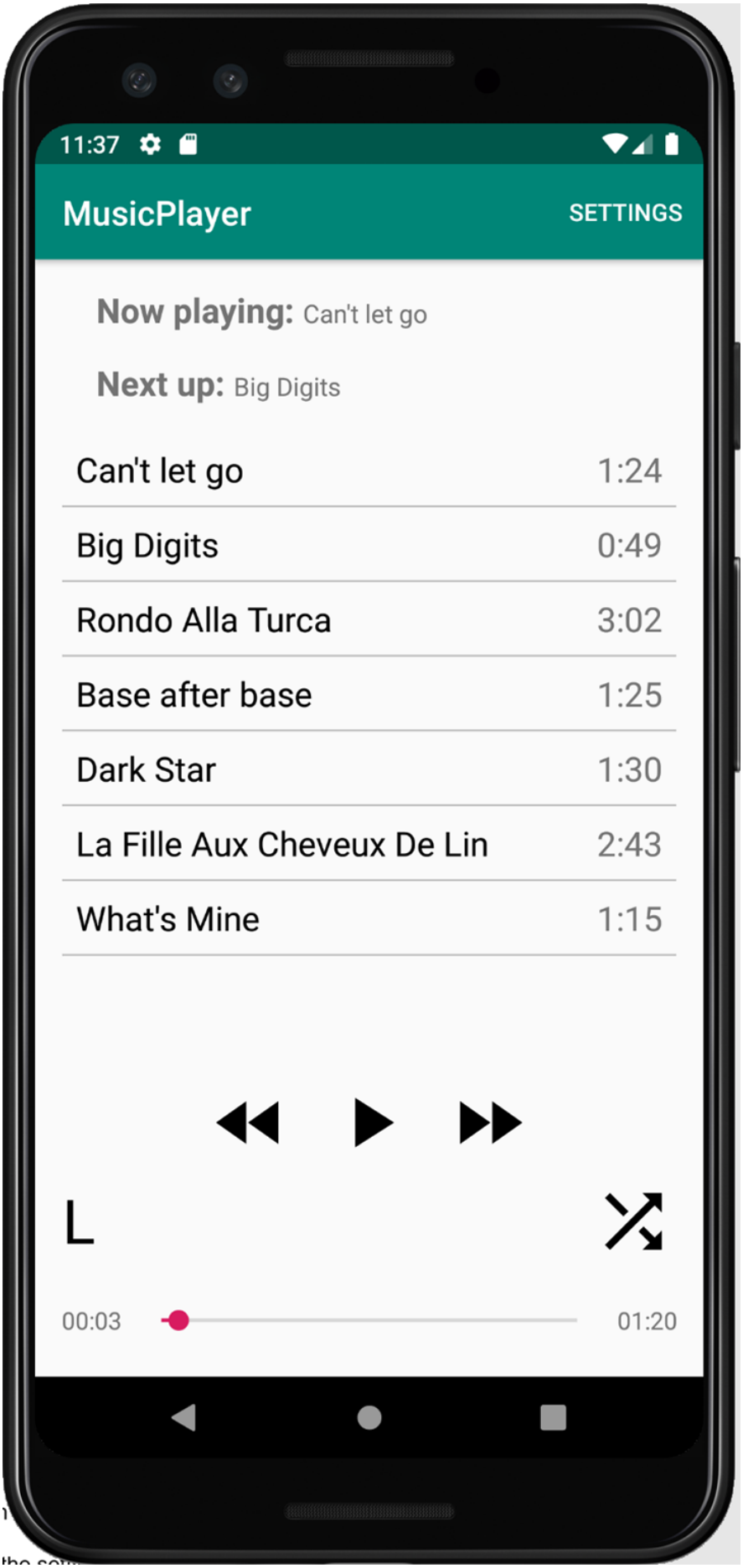
—Jerry Garcia

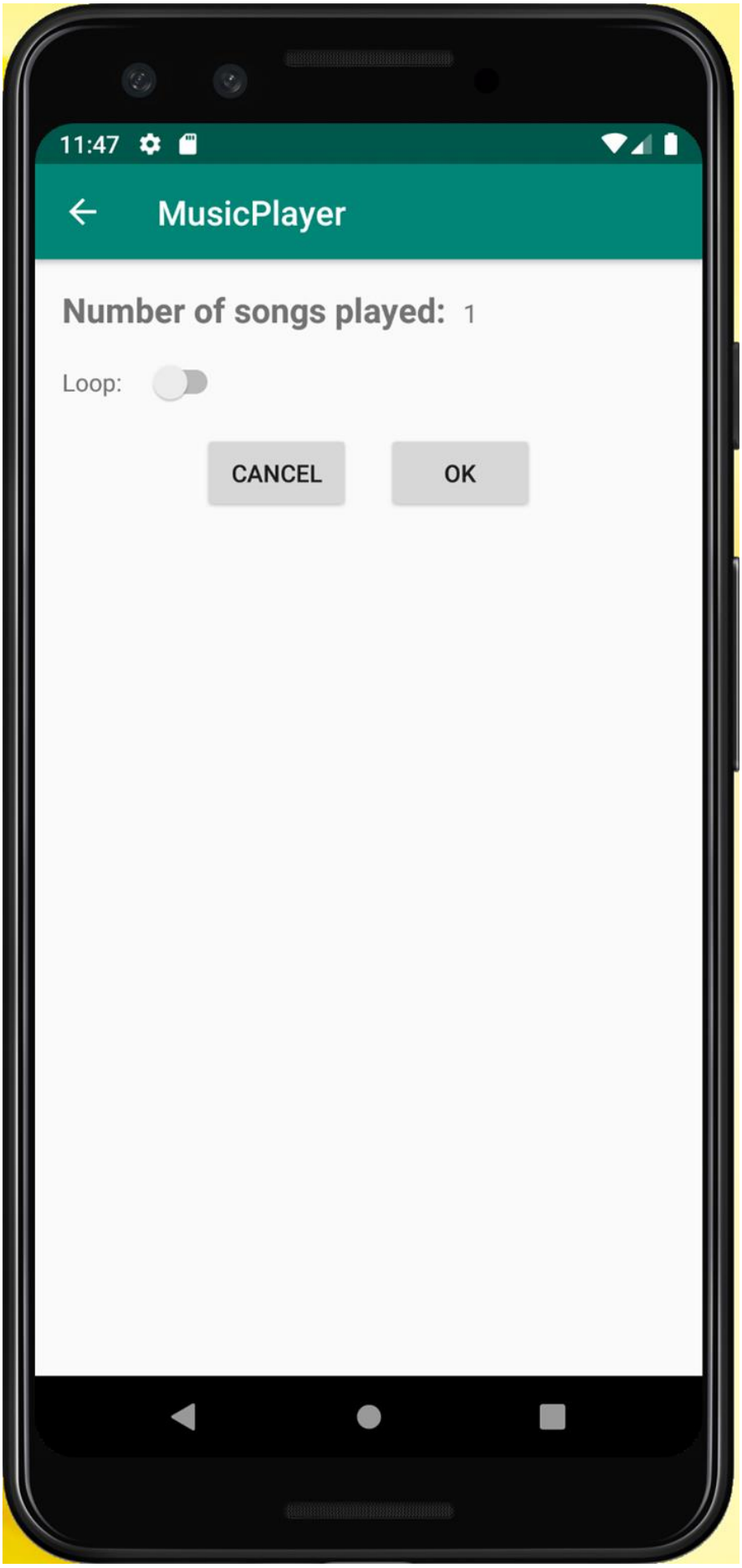
I remember one time - it might have been a couple times - at the Fillmore East in 1970, I was opening for this sorry-ass cat named Steve Miller. Steve Miller didn't have his shit going for him, so I'm pissed because I got to open for this non-playing motherfucker just because he had one or two sorry-ass records out. So I would come late and he would have to go on first and then we got there we smoked the motherfucking place, everybody dug it.

—Miles Davis

For this assignment, you will be implementing a simple music player.

On the next page are some screenshots of an example app.


11:37
MusicPlayer
SETTINGS
Now playing: Can't let go
Next up: Big Digits
Can't let go 1:24
Big Digits 0:49
Rondo Alla Turca 3:02
Base after base 1:25
Dark Star 1:30
La Fille Aux Cheveux De Lin 2:43
What's Mine 1:15
00:03
01:20


11:47
MusicPlayer
Number of songs played: 1
Loop:
CANCEL
OK

Specifications:

Elements

Your app should contain the following elements:

- Text showing the name of the current song that is playing.
- Text showing the next song that will be played.
  - If a song is selected from the list, the names of the current and next song update.
- A RecyclerView containing a list of songs, and their total time.
  - When a song in this list is tapped by a user, that song starts playing.
- A play/pause button
  - Pressing this button plays/pauses the music
  - The button switches between the play and the pause symbol each time it is clicked.
- Skip forward and skip backwards buttons.
  - Pressing skip forward causes the next song to be played. The song after the last song is the first song.
  - Pressing the skip backwards button causes the previous song to be played, even if the current song is well on its way. The previous button never restarts a song. The song before the first song is the last song.
  - Skip forward/backward does not change the play/pause state of the player.
- Loop indicator (far left) and shuffle button (far right)
  - The L for Loop indicator indicates if the current song will restart once it finishes. If the loop condition is true, the background of the L is red and the current song will loop when it finishes. Otherwise the background of the L is white.
  - When the shuffle button is pressed, the songs are randomly permuted. The current song stays the same, but the next song can change as well as the overall song order. The recyclerview should update to show the new order, and the next song text indicator should also update.
- A slider that shows the progress through the song.
  - The slider should update as the song plays.
  - If the user drags the slider then the song should skip to the appropriate position
  - Hint: It can be implemented by SeekBar.
  - Hint: If you “grab and hold” the slider, it will jiggle a bit between where you are holding it and the current play position. That is acceptable behavior.
- Text (to the left of the slider) showing how long a song has been playing in the format “MM:SS” (see convertTime)
- Text (to the right of the slider) showing how long until a song finishes in the format “MM:SS”
- There is a settings button in the action bar. When pressed this opens the settings activity.
  - The settings are for “loop” mode, and there are two buttons, cancel and ok.
  - The values displayed when this activity opens should mirror the current settings of the app
  - The Cancel button

        - Pressing this goes back to the main activity, discarding any modifications
    - The OK button
        - Pressing this goes back to the main activity and uses the provided modifications
    - If loop mode is enabled, press forward/backward button to play the next/previous song and then loop on that song.
- If a song is playing and the activity is paused/settings activity is invoked, you can either keep playing the song or pause the song and resume playing when the main activity comes back.
- You will need controls for your app. We recommend Android's built in vector graphics, which are very versatile, look good at any resolution and are easy to add to your project. You can see the course video on icons and images for more information. Once you add them to your project, they will have names like @drawable/ic_play_arrow_black_24dp and @drawable/ic_fast_forward_black_24dp.
- The music files are in res/raw directory.

Initialization

When the app starts it should begin playing the first song in the list.

When a song finishes playing

When a song finishes playing the next song in the list should be played (unless loop is enabled). If the last song in the list is played then wrap around to the beginning and play the first song.

Layout

Here are your guides for layout.

- You can use LinearLayouts or ConstraintLayouts.
- I use 8dp of margin for most layout elements.
- Center the (back/play/forward) controls and separate them by 20dp. Separate the buttons on the settings layout by 20dp.
- All drawable buttons are 50dp by 50dp, except the L which is width 30dp and height of 50dp. The L is 36sp.
- "Now playing", "Next Up", and "Number of songs played" are all bold, 20sp. The text following them is not bold and 15sp
- Song titles are black and 20sp. Times are default color and 20sp.
- TextView time displays use the default font size and are 50dp in width.

Settings

This activity displays the number of songs played so far. It counts on song starts—you don't need to listen to the whole song to increase the counter. It also has a switch element to control whether loop mode is on. The switch should be set correctly (i.e., if loop mode is on when we enter the settings mode, the switch will be on). The user can commit their changes by clicking ok, or by returning to the main display by hitting the "up" arrow in the upper left corner. If the user clicks cancel, then whatever loop mode setting was in effect when the activity started remains in effect.

RecyclerVew Adapter

The code for the RecyclerView adapter is copied almost exactly from our demo code. I encourage you to pattern match.

Coroutines

We have talked a bit about coroutines in class, and they are present in your first two programming assignments. We use one here for updating the time and seek bar. Coroutines are light-weight threads that are independent and can execute concurrently (at the same time). Coroutines are useful for when you need to do something like update the time display. Having code that just wakes up, sees what time it is and update the display is simple to write and to reason about.

The code in your coroutine should mostly read state, but it will update some state. Just go for it. You do not need any synchronization between the main thread and the coroutine (even though that lack of synchronization will cause the jiggle if you hold the seek bar at a new location). We have included the code to launch and manage the coroutine.

Hints

If our music is driving you crazy, you can replace the songs. You also need to update the Repository.

You need to modify the AndroidManifest.xml.

The MediaPlayer object has internal state and methods that can be called only in a specific state (https://developer.android.com/reference/android/media/MediaPlayer#state-diagram and https://developer.android.com/reference/android/media/MediaPlayer#valid-and-invalid-states have more details). We use it in a pretty simple way, so you shouldn't need all these details.

I recommend setBackgroundColor, rather than the background property when controlling the loop indicator.

If you think clicking the L should change the loop mode and that going to a separate settings activity is silly, well, I have to say I agree with you. That partitioning of functionality is a bit pedagogical.

Work on one feature at a time. There are a lot of tiny elements in this project, many of them do not depend on each other.

When it comes time to "get rid" of a MediaPlayer, call reset() and then release(). Otherwise you will get errors about unhandled events.

We will press lots of buttons on your app to exercise all of its functionality.

This project will require a little bit of outside research. We do not cover SeekBars (for example) in class, nor do we cover MediaPlayers. But we cover the things you need to know to figure these things out on your own. Google, the Android documentation, Stack Overflow,

and Piazza are your friends.