Notifications Lab

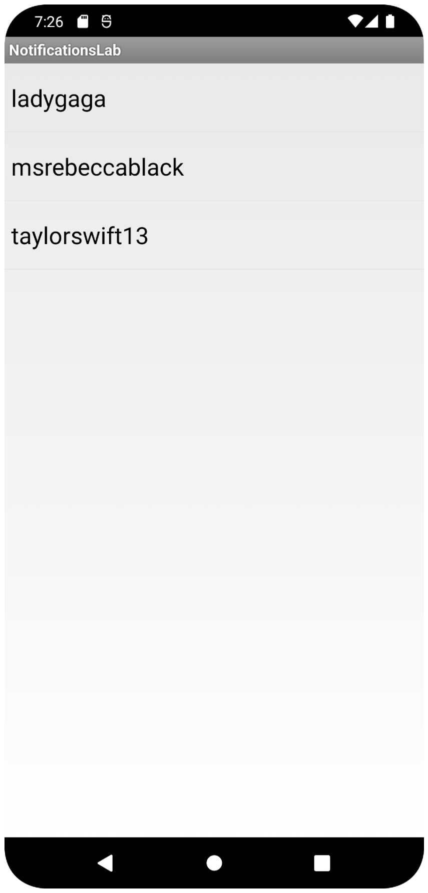
Notifications and BroadcastReceivers

Objectives:

This week's Lab explores User Notifications and Broadcast Receivers. After finishing this Lab, you will have a better understanding of how to create and display different types of User Notifications to inform users of your application's actions. You will also learn how to broadcast and receive Intents.

Exercise A:

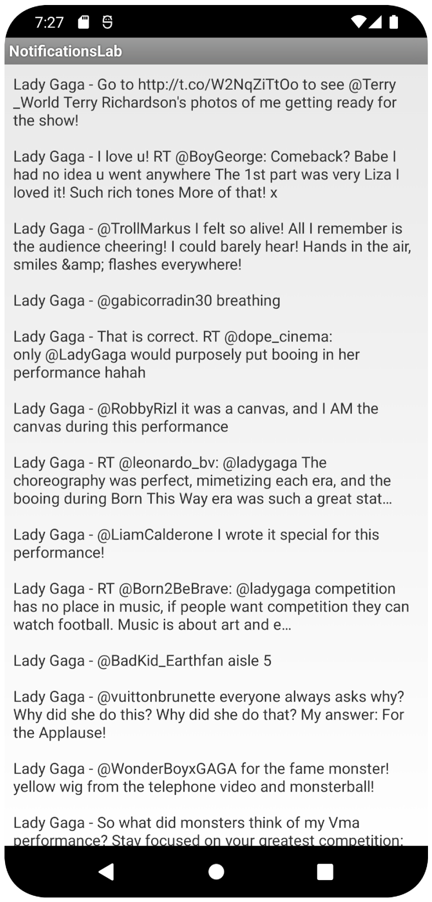
This Lab involves an app that displays locally stored Twitter data from a set of friends. The basic interface appears below.



ladygaga

msrebeccablack

taylorswift13



Lady Gaga - Go to http://t.co/W2NqZiTtOo to see @Terry_World Terry Richardson's photos of me getting ready for the show!

Lady Gaga - I love u! RT @BoyGeorge: Comeback? Babe I had no idea u went anywhere The 1st part was very Liza I loved it! Such rich tones More of that! x

Lady Gaga - @TrollMarkus I felt so alive! All I remember is the audience cheering! I could barely hear! Hands in the air, smiles &amp; flashes everywhere!

Lady Gaga - @gabicorradin30 breathing

Lady Gaga - That is correct. RT @dope_cinema: only @LadyGaga would purposely put booing in her performance hahah

Lady Gaga - @RobbyRizl it was a canvas, and I AM the canvas during this performance

Lady Gaga - RT @leonardo_bv: @ladygaga The choreography was perfect, mimetizing each era, and the booing during Born This Way era was such a great stat…

Lady Gaga - @LiamCalderone I wrote it special for this performance!

Lady Gaga - RT @Born2BeBrave: @ladygaga competition has no place in music, if people want competition they can watch football. Music is about art and e…

Lady Gaga - @BadKid_Earthfan aisle 5

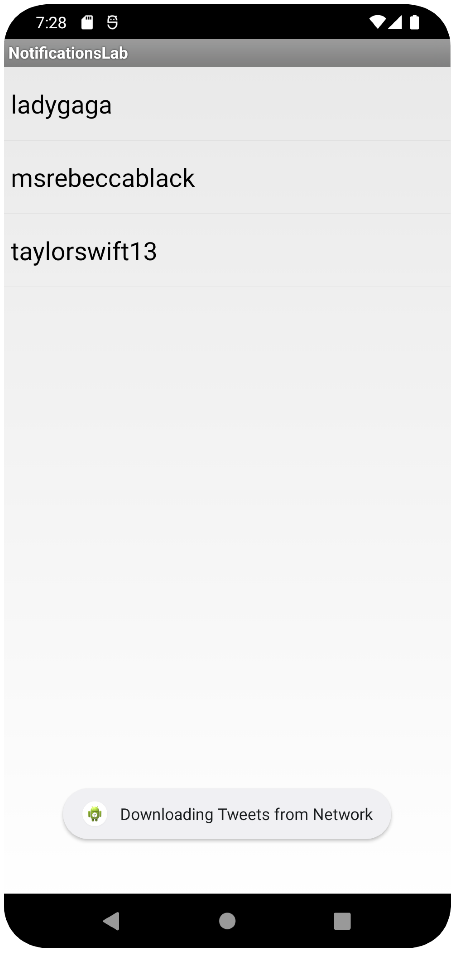
Lady Gaga - @vuittonbrunette everyone always asks why? Why did she do this? Why did she do that? My answer: For the Applause!

Lady Gaga - @WonderBoyxGAGA for the fame monster! yellow wig from the telephone video and monsterball!

Lady Gaga - So what did monsters think of my Vma performance? Stay focused on your greatest competition:

When the application's MainActivity begins running, it determines whether Tweet data has been downloaded within the last two minutes. If not, it considers any existing data to be stale and therefore downloads fresh Twitter data (the downloading process is simulated). To do the "downloading" step the Activity causes an CoroutineAsyncTask to execute, which will load the Twitter data off the main Thread.

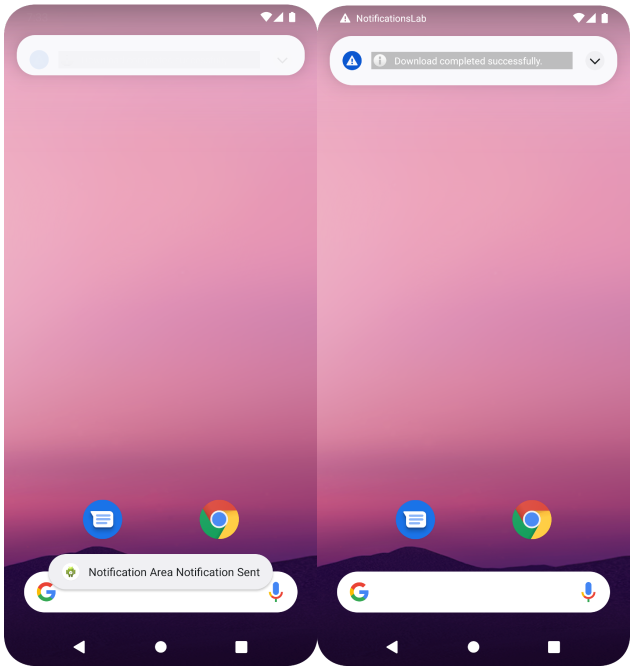
As the download process begins, the application creates and displays a Toast message, alerting the users that the download is starting. An example Toast message is shown below:



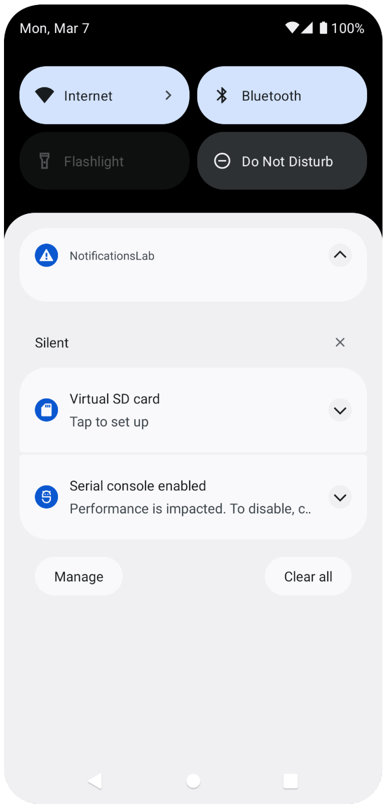
In practice, the download process could take a noticeable amount of time. Therefore, it's possible that the user might exit the application while the download is in progress. If that happens, the application will generate a Notification Area Notification to inform the user once the download finishes. To do this, the CoroutineAsyncTask broadcasts an Intent when it finishes downloading the tweet data. If the application's MainActivity is not running and in the foreground when the Intent is broadcast, then the CoroutineAsyncTask creates a Notification Area Notification and places it in the Notification area. However, if the MainActivity is running and in the foreground, then the CoroutineAsyncTask does not create this Notification. Instead, it displays a Toast message informing the user that the download is finished.

Specifically, the CoroutineAsyncTask uses the sendOrderedBroadcast() method to broadcast an Intent once downloading has finished. It also passes its own BroadcastReceiver into this call so that it will receive the result of the broadcast. The MainActivity will dynamically register a BroadcastReceiver so that it receives this Intent only when the MainActivity is visible and in the foreground. If the BroadcastRecevier receives this Intent, then it will return a specific result code. The presence of this code lets the CoroutineAsyncTask know that the MainActivity was visible and in the foreground. The absence of this result code means that the Activity was not visible and in the foreground. In the latter case, the CoroutineAsyncTask will create and send the Notification Area Notification.

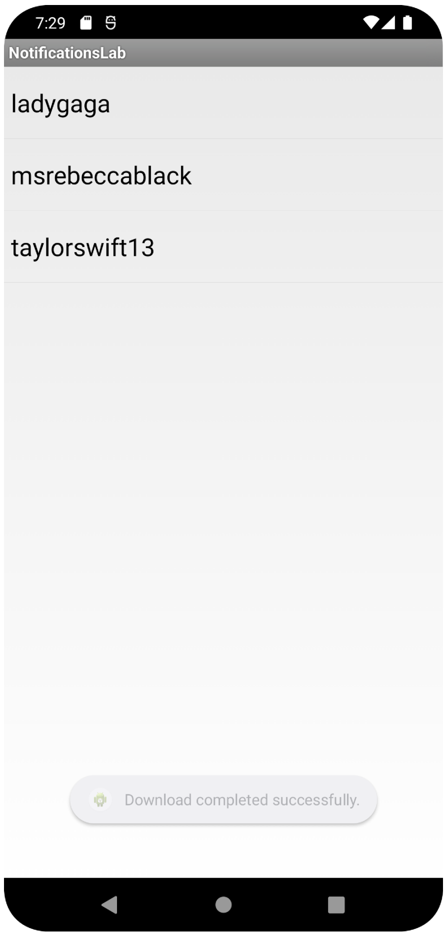
If the CoroutineAsyncTask does send the Notification Area notification, an icon will appear in the Notification Area and the user can pull down on the Notification Ares to further examine the notification.


Notification Area Notification Sent
NotificationsLab
Download completed successfully.

When the user pulls down on the Notification drawer, he or she will see an indication that the data was successfully downloaded. An example is shown below:



If the user clicks on this Notification's View, the MainActivity should re-open. Again, if the MainActivity starts more than two minutes after the data was downloaded, then MainActivity should download the data again. Otherwise, it should not.
Finally, if the download finishes while the MainActivity is visible and in the foreground, the app will display a Toast message to that effect.



ladygaga

msrebeccablack

taylorswift13

Download completed successfully.

Implementation Notes:

1. Download the application skeleton files and import them into your IDE.
2. Implement the TODO comments found in the MainActivity.kt and DownloadTaskFragment.kt files

   In MainActivity.kt:

   a. Create a BroadcastReceiver that returns a result code (MainActivity.IS_ALIVE) to inform the CoroutineAsyncTask that the MainActivity's active and in the foreground, and therefore, the CoroutineAsyncTask should not send the Notification Area Notification.
   b. Register the broadcast receiver in the onResume() method after the super.onResume() call.
   c. Unregister the broadcast receiver in the onPause() method before the super.onPause() call.

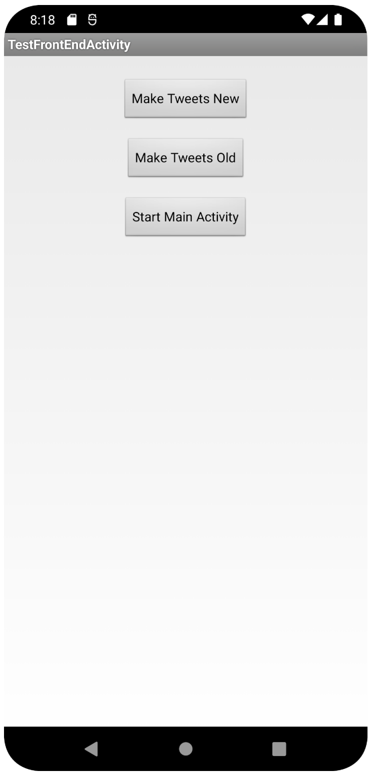
   In DownloadTaskFragment.kt:

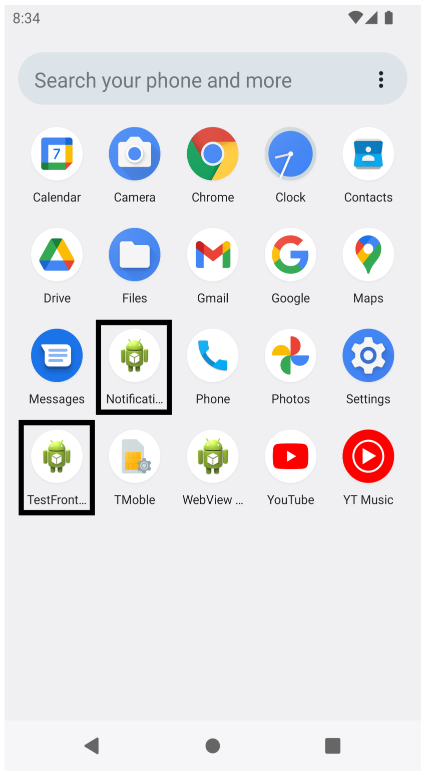
   d. Implement the logic to notify the user that the feed has been downloaded using sendOrderedBroadcast(). You will need to create a BroadcastReceiver to receive the result of this broadcast. If that result is not MainActivity.IS_ALIVE then this BroadcastReceiver should create a Notification Area Notification and create a NotificationChannel.

Testing:

We will be manually looking at the code and testing the lab for 4 cases.

Please note that for testing instead of the MainActivity, we will use another activity called, TestFrontEndActivity. If you look into the AndroidManifest.xml file for this application, you'll see that both of these Activities are main entry points for the app. In fact, when you install this app, two icons will appear in the launcher (Main Menu). This approach allows us to modify the age of already downloaded Tweet data before starting the MainActivity. The interface for this Activity is shown below on the left and the two icons in the launcher on the right.





Test Case 1.1:

1. Start the TestFrontEndActivity app
2. Click on the "Start Main Activity" button
3. There should be no Toast or Notification

Test Case 1.2:

1. Start the TestFrontEndActivity app
2. Click on the "Make Tweets New" button
3. Click on the "Start Main Activity" button
4. There should be no Toast or Notification

Test Case 2.1:

1. Start the TestFrontEndActivity app
2. Click on the "Make Tweets Old" button
3. Click on the "Start Main Activity" button
4. There should a Toast indicating downloading has started (use the "download_in_progress_string" from string resources as the message displayed)
5. The app should be kept open till another Toast indicating whether the download succeeded or failed (use "download_succes_string" and "download_failed_string" respectively from string resources as the message displayed)

Test Case 2.2:

1. Start the TestFrontEndActivity app
2. Click on the "Make Tweets Old" button
3. Click on the "Start Main Activity" button
4. There should a Toast indicating downloading has started (use the "download_in_progress_string" from string resources as the message displayed)
5. Hide the app (let the app run in background)
6. There should be a notification indicating whether the download succeeded or failed

Note:

After following the steps for each test case, the rest of the app should function in the same way fragments lab did i.e., look at the first 2 images in this doc. Also take a look at the video attached for how the testing flow would be.

Warnings:

1. These test cases have been tested on a Pixel 5 AVD emulator with API level 31. To limit configuration problems, you should test your app against a similar AVD.

Once you've passed all the test cases, submit your project to GitLab.

Submission

To submit your work, you will need to commit your solution to your repo on GitLab by running the following command: git push origin main.