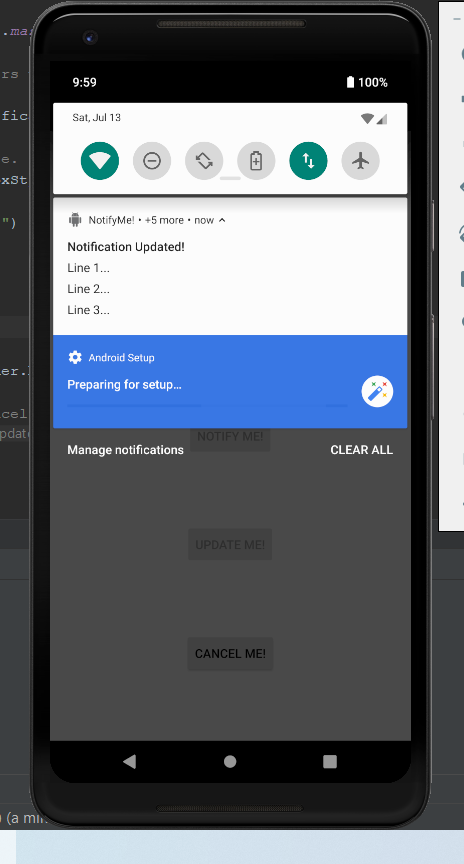
**Notifications Codelab Overview**

A notification is a way to show information to users even when the app is not running on the foreground. In other words, a notification is a message that your app displays to the user outside of your app’s normal UI. This user notification when the app is not in the foreground can be achieve by using the Android notification framework. Notifications normally appear as icons in the device’s notification area, located in the status bar. On devices running Android 8.0 and higher, an app icon automatically shows a badge when it has a notification to show to the user.

In this codelab, I learned how to create a notification using the notification builder, how to use pending intents to respond to notification actions, and how to update or cancel existing notifications. It also taught me how to create an app that sends a notification when the user taps a button and then update the notification from a button in the app and from an action button inside the notification. The “Notify Me!” app allows users to trigger, update and cancel a notification using three buttons. The app also experimented with notification styles, actions and priorities. Each notification channel represents a type of notification. You can group several notifications in each notification channel.

The Android system uses the NotificationManager class to deliver notifications to the user. Notifications are created using the NotificationCompat.Builder class, which allows the content and behavior of the notifications to be set. I also learned how to add a content intent for notifications, which is similar to the intents I have used throughout the training. Content intents can be explicit intents to launch an activity, implicit intents to perform an action, or broadcast intents to notify the system of an event. The difference is that a content intent must be wrapped inside a PendingIntent. PendingIntent allows the Android notification system to perform assigned action on behalf of your code.

As I mentioned before, the codelab also showed me how to implement update and cancel to the notification. It taught me how to cancel a notification by calling the cancel() method on the NotificationManager and by passing in the notification ID. Additionally, it also showed me how to create an update for the notification, which is far more complex than to cancel one. To update a notification, this codelab uses the BigPictureStyle, which allows you to include an image in the notification. Moreover, the codelab also showed me how to track the state of the notification by enabling and disabling the buttons depending on the notification’s state. For example, in the app, the Update and Cancel buttons should only be enabled/visible once the notify button is clicked by the user. Lastly, I also learned how to implement a broadcast receiver that calls the updateNotification() method and an update action to the notification. The purpose of it is to be able to update the notification from the homepage without opening the app.



**The above screenshot is homework section of the Notifications Codelab. The answers to the question below are bold and underlined**.

**Answer these questions**

**Question 1**

Select all that are true for notification channels:

* You use notification channels to display notifications to the user in the device status bar.
* **You use notification channels to group multiple notifications so that the user can control the notifications' behavior.**
* Notification channels are available in older devices, those running Android 7.0 Nougat (API 24) and lower.
* Notification channels are not yet available in the Android Support Library package.

**Question 2**

Which API do you use to show a notification in the notification drawer and in the device's status bar?

* Notification.notify()
* **NotificationManager.notify()**
* NotificationCompact.notify()
* NotificationCompat.Builder.notify()

**Question 3**

Which component is *not* needed when you add a notification action?

* Icon that represents the action
* Title that describes the action
* **Click listener for the action button click event.**
* PendingIntent that's sent when the user taps the action button.

**Question 4**

Which API do you use to add an action button to a notification?

* NotificationCompat.addActionButton()
* **NotificationCompat.Builder.addAction()**
* Notification.Builder.addActionButton()
* NotificationManager.addAction()

**Question 5**

Suppose that you create an app that downloads a work of art on the user's device every day. Once the day's image is available, the app shows a notification to the user, and the user can download the image or skip the download. What PendingIntent method would you use to start a service to download the image?

* Activity.startService()
* **PendingIntent.getBroadcast()**
* PendingIntent.getActivity()
* PendingIntent.getService()