Week 3 – Daily 2

1. Broadcast receivers

Broadcast is a core component of Android, with its function pertaining to communication between apps and the operating system. Given an event within the system, a registered broadcast receiver within an app can be triggered to read and react to the changes (i.e. airplane mode, Bluetooth, batter level); for which may affect the overall efficiency of the app. Furthermore, aside from the system itself, apps can notify other apps in receiving information that may be necessary in its function – downloaded content, messages, etc.

The broadcast is sent within an intent that uses an action identifier to indicate the type of receiver to track, returning a Boolean value to show to the application of its receival. In order to declare a receiver, it must be established within the Manifest; capable of taking in multiple receivers if necessary, by the developer.

In tracking the broadcast receiver, the class extends the Broadcast Receiver class and implements the OnReceive method that enables for a log display of the content within the broadcast – returning a status either active or not. To ensure efficient data handling of the operating system resources, after the app is completed or closed, should end or ‘unregister’ the receiver as it would no longer be needed.

During an instance of needing to a certain component to be on regarding the Broadcast Receiver, can explicit delegate the task into an Async thread that will allow for a background threat throughout the application lifecycle

An event within the application can trigger itself to send a broadcast to other application in 3 ways: ‘sendOrderedBroadcast’, ‘sendBroadcast’, ‘LocalBroadcastManager’. In the method, sendOrderedBroadcast, a broadcast is explicitly sent to certain applications within a specific order. This broadcast can be controlled given a priority attribute that will derive the extent of the order sent. Another method, sendBroadcast, is similar to the ‘sendOrderedBroadcast’ with the difference of not stating an explicit order to which is sent in – although more efficient, it lacks the ability to read the results from the other receivers. Lastly, ‘localBroadcastManager’ is local specific regarding the broadcast sent that is only capable of being received given locality of application.

To ensure security and user awareness of the app’s necessary functionality to effectively function, a permission parameter can be attached to the receiver tag within the manifest. This would correlate to the intent, as it would trigger the system to display user permission response tag.

1. Services

Service is another core component of the android device, that performs operations in the background given the time necessary for the task to be complete. Generally, under most circumstances, this background task will continue to run regardless of the user changing application interactions. Furthermore, as this ‘service’ is not necessary handled by the user there is not interface for it; however, there are 3 different capacities for which it can apply to – Foreground, background, and bound. In a foreground service, the operation is visible to the user yet can continue to run in the background, i.e. music player or notifications – regardless of user interaction with the application. Background service apply to operations that aren’t directly noticed by the user and is rather handled by the system; i.e. storing information to the application. Lastly, a service can pertain to a bound, to which can ‘bind’ to a component by utilizing bindService. In this case, the interface directly communicates with the IPC as long as there is an application bound to it.

To instantiate a service to be handled, it must be declared within the manifest and can be set as a component that will not enable start up without having the user start the service, regardless of intent sent. Furthermore, depending on the service provided by the application, the developer can track this component by running it within the foreground after startService and onStartCommand.

Service can be started within an activity or other applications by an Intent that creates specification regarding the data in its use. The activity would proceed to deliver the intent through the startService from the call onStartCommand – enabling for any instance of online data transaction to be completed and in turn destroy itself upon completion. Furthermore, threads can also be utilized to extend the service to handle multiple threads that may be necessary aside from the main.

Throughout a service lifecycle, it must manage itself in regard to its creation to that of its own destruction, happening in two different ways. At startService, the service runs continuously until called within the stopSelf method or stopServices – for then would destroy itself. In bound service, another component calls the bindService method, communicating to the service the need to unbind itself – which would then implicitly destroy the service as opposed to explicitly needing to destroy itself. In either case, they can be mixed in tracking the service operation – managing different tasks while the user is interacting with multiple application.