**Background process analysis**

**App permissions**:

Depending on our application, only the necessary permission is requested from user. Since the application is developed on Android version 6.0 Marshmallow, the permission is requested only at runtime of the application rather than the installation. User has the right to deny the permission requested by the app. App should be not be broken because of this action. [1]

**Camera API:**

In To Do application, the need to access camera and save in the To Do notes is necessary. There are two ways of implementing this; by using the Camera API, and by using system intents. If the need to access Camera is more frequent, the permission needs to be requested only once from the user.

However, if the need is not that frequent, then system intents can be used. System intents will ask for the user’s permission directly and not the app. For this application, we will use the Camera API, as the user needs to grant permission only once. This will save the user’s time and help when creating tasks.

**Location:**

Using android.location package, the location of user is fetched. Two classes of this package; LocationManager and LocationListener are used. For this, google.play.services dependency is needed.

To fetch location update from user, requestLocationUpdates function is used. To prevent device’s power consumption, location update can be fetched at limited time intervals. This reduces battery drainage and improves app’s efficiency. Also, another parameter can be set for minimum distance. The location is fetched at a specified minimum distance achieved.

ACCESS\_FINE\_LOCATION permission is set for accessing the user’s location using GPS. This permission is requested in form of asking for accessing location, when the app is run and not while it is installed. Since android 6.0 onwards, these permissions are set only when the app is opened. However, the fallback strategy when the user declines such permissions need to be considered. [6]

If the user has not moved from its location for a longer duration, there is no need to frequently request a location update. Use of TYPE\_SIGNIFICANT\_MOTION is needed for controlling the location update frequency. This prevents from frequent updates and only requests when there is a significant change in the location. Overall, it reduces the battery consumption for getting drained. [8]

**Flow for obtaining user location**

Here's the typical flow of procedures for obtaining the user location:

Once the app is installed and opened, it requests permission for accessing the user’s location. When the user accepts the permission, it starts polling for location. The location update is requested at the specified minimum time / distance intervals. Sometime later, start listening for updates from desired location providers. Maintain a "current best estimate" of location by filtering out new, but less accurate fixes. Stop listening for location updates when user turns off the feature. Take advantage of the last best location estimate.

Listening for a long time consumes a lot of battery power, so as soon the information is received, you should stop listening for updates by calling *removeUpdates(PendingIntent).*

**Location Update Frequency:**

Choosing a sensible value for minimum time is important to conserve battery life. Each location update requires power from GPS, WIFI, Cell and other radios. A minimum time value should be selected which should be as high as possible while still providing a reasonable user experience. When the application is not in the foreground, instead of continuously polling requests, seta time interval in seconds. If the application is in the foreground and showing location to the user, then it is appropriate to select a faster update interval.

The minimum distance parameter can also be used to control the frequency of location updates. If there is a distance change, then it checks for minimum distance specified and minimum time. However, it is more difficult for location providers to save power using the minimum distance parameter, so minimum time should be the primary tool to conserving battery life. [4]

You can reduce the rate of updates by increasing the parameter values in *requestLocationUpdates*() that specify the interval time and minimum distance change.[5]

On Android 8.0 (API level 26) and higher, if an app is running in the background when it requests the current location, then the device calculates the location only a few times each hour.

Another factor affecting the power consumption is the APK size. Size of and APK tell how fast the app loads and how much memory is needed. Overall APK size can be reduced to reduce the app’s memory usage. [2]

**Performance**

Using performance tools such as Systrace and Traceview to determine bottlenecks in app's responsiveness. [7]

**Doze and Standby mode:**

In Android 6.0, two features have been added: Doze and Standby mode. The app needs to be tested in these two scenarios as well, when the app is not running. [3]

**References:**

[1] Developers, G. (2018). App permissions best practices | Android Developers. Retrieved from <https://developer.android.com/training/permissions/usage-notes>

[2] Developers, G. Reduce the APK size | Android Developers. Retrieved from https://developer.android.com/topic/performance/reduce-apk-size

[3] Developers, G. Optimize for Doze and App Standby | Android Developers. Retrieved from <https://developer.android.com/training/monitoring-device-state/doze-standby>

[4] LocationManager  |  Android Developers. Retrieved from <https://developer.android.com/reference/android/location/LocationManager.html#requestLocationUpdates(long,%20float,%20android.location.Criteria,%20android.app.PendingIntent)>

[5] Location and context overview  |  Android Developers. Retrieved from https://developer.android.com/training/location/receive-location-updates

[6] Location strategies  |  Android Developers. (2018). Retrieved from <https://developer.android.com/guide/topics/location/strategies#java>

[7] Analyze power use with Battery Historian  |  Android Developers. Retrieved from <https://developer.android.com/topic/performance/power/battery-historian>

[8] Sensor  |  Android Developers. Retrieved from <https://developer.android.com/reference/android/hardware/Sensor#TYPE_SIGNIFICANT_MOTION>