The Daily Selfie with Concurrent Image Processing

The Concurrent Daily Selfie application is an extension of one of the mini-projects in Professor Porter's MOOCs, which enables users to create selfies of themselves over an extended period of time. It periodically reminds the user to take a selfie and presents the selfies in a list that makes it easy to see how the user has changed over time. This extended implementation also allows users to process their selfies to add effects, such as blurring or charcoaling. The image processing is done concurrently in the background to ensure that the UI thread is not interrupted.

Basic Project Requirements

Any Capstone project must support multiple users and should leverage services running remotely in the cloud. Each project's specification clearly outlines the app's intended high-level behavior, yet leaves substantial room for individual creativity. Students will therefore need to flesh out many important design and implementation details. Basic requirements for all Capstone MOOC project specifications include:

- Apps must support multiple users via individual user accounts. At least one user facing operation must be available only to authenticated users.
- App implementations must comprise at least one instance of at least two of the following four fundamental Android components: Activity, BroadcastReceiver, Service and ContentProvider.
- Apps must interact with at least one remotely-hosted Java Spring-based service over the network via HTTP.
- At runtime apps must allow users to navigate between at least three different user interface screens.
    - e.g., a hypothetical email reader app might have multiple screens, such as (1) a ListView showing all emails, (2) a detail View showing a single email, (3) a compose view for creating new emails, and (4) a Settings view for providing information about the user's email account.
- Apps must use at least one advanced capability or API from the following list covered in the MoCCA Specialization: multimedia capture, multimedia playback, touch gestures, sensors, or animation. Experienced students are welcome to use other advanced capabilities not covered in the specialization, such as BlueTooth or Wifi-Direct networking, push notifications, or search. Moreover, projects that are specified by commercial organizations may require the use of additional organization-specific APIs or features not covered in the MoCCA Specialization. In

these cases, relevant instructional material will be provided by the specifying organization.

- Apps must support at least one operation that is performed off the UI Thread in one or more background Threads or a Thread pool. Communication between background Threads and the UI Thread should be handled by one of Android concurrency frameworks, such as the HaMeR or AsyncTask framework.

There may also be additional project-specific requirements (e.g., required use of a particular project-specific API or service).

Basic Functional Description and App Requirements for Concurrent Daily Selfie

1. A *Selfie* is an image captured using the Daily Selfie application by the *User*.

2. A *Reminder* is a unit that holds a time. The *User* is informed daily to take a *Selfie* at the time specified in the *Reminder.*

3. A *SelfieRecord* is a data structure holding a selfie, which stores the date that the image was taken and the file path to the image's storage location, in addition to the Selfie image itself.

4. The *User* creates selfies by taking a picture using the phone's built in camera app, which saves the image to the device at a designated location.

5. The *User* can view their past *Selfies* in an Android ListView. Clicking on a *Selfie* in the ListViewwill open a large view in a separate Activity, showing the *Selfie* in a larger form.

6. The *User* can process *Selfies* using ImageMagick library capabilities, which includes adding noise, blurring, or adding a charcoal effect to the image. The *User* selects any number of *Selfies* to process and then selects an editing button on the action bar. This selection, in turn, creates one or more *ImageProcessingTasks* to concurrently apply the ImageMagick processing to subsections of the selected selfies. Once processed, the images are saved and added to the ListView.

7. An *ImageProcessingTask* is a subclass of the Android *AsyncTask* that takes a list of *SelfieRecords* as input, processes them in a background thread, and then displays the new *Selfies* in a ListView after the processing is finished.

Implementation Considerations

- How will *Remainders* be used to inform the *User* that it's time to take a *Selfie*? For example, will an Android notification be used, so that clicking on the notification will bring the user to the Daily Selfie application?
- How will you store the images? For example, will you use a directory of files, a ContentProvider, etc.?
- What will the user interface look like for the Daily Selfie app so that it is quick and simple to use?
- How will *Reminders* be delivered to a *User* in a way that will help the User to use the app more frequently and consistently?
- What user preferences can the *User* set? How will the app be informed of changes to these user preferences?
- How will the app handle concurrency issues, such as how to process images concurrently via one or more AsyncTasks?
- How will the app use at least one advanced capability or API listed above? For example, will you create an animation to explain the app? How will you allow *Users* to take pictures? Will you use push notifications to prompt *Users* when to take *Selfies*?
- Does your app really require two or more fundamental Android components? If so, which ones? For example, this app might benefit from using a ContentProvider or from using a background Service that synchronizes local and remote images to a cloud-based server only when the device is connected to a WiFi network.

Installing the ImageMagick library on Android

The Concurrent Daily Selfie application uses the ImageMagick library to provide users with its various image processing effects to apply to their selfies.

- To use ImageMagick, you will need to download and install the latest NDK, as described at this link
    - https://developer.android.com/tools/sdk/ndk/index.html
- Use the Android-ImageMagick implementation at the following repository to install and use the ImageMagick library:
    - https://github.com/paulasiimwe/Android-ImageMagick

Follow the instructions in the README file to compile the binaries and add the required source files to your Android project