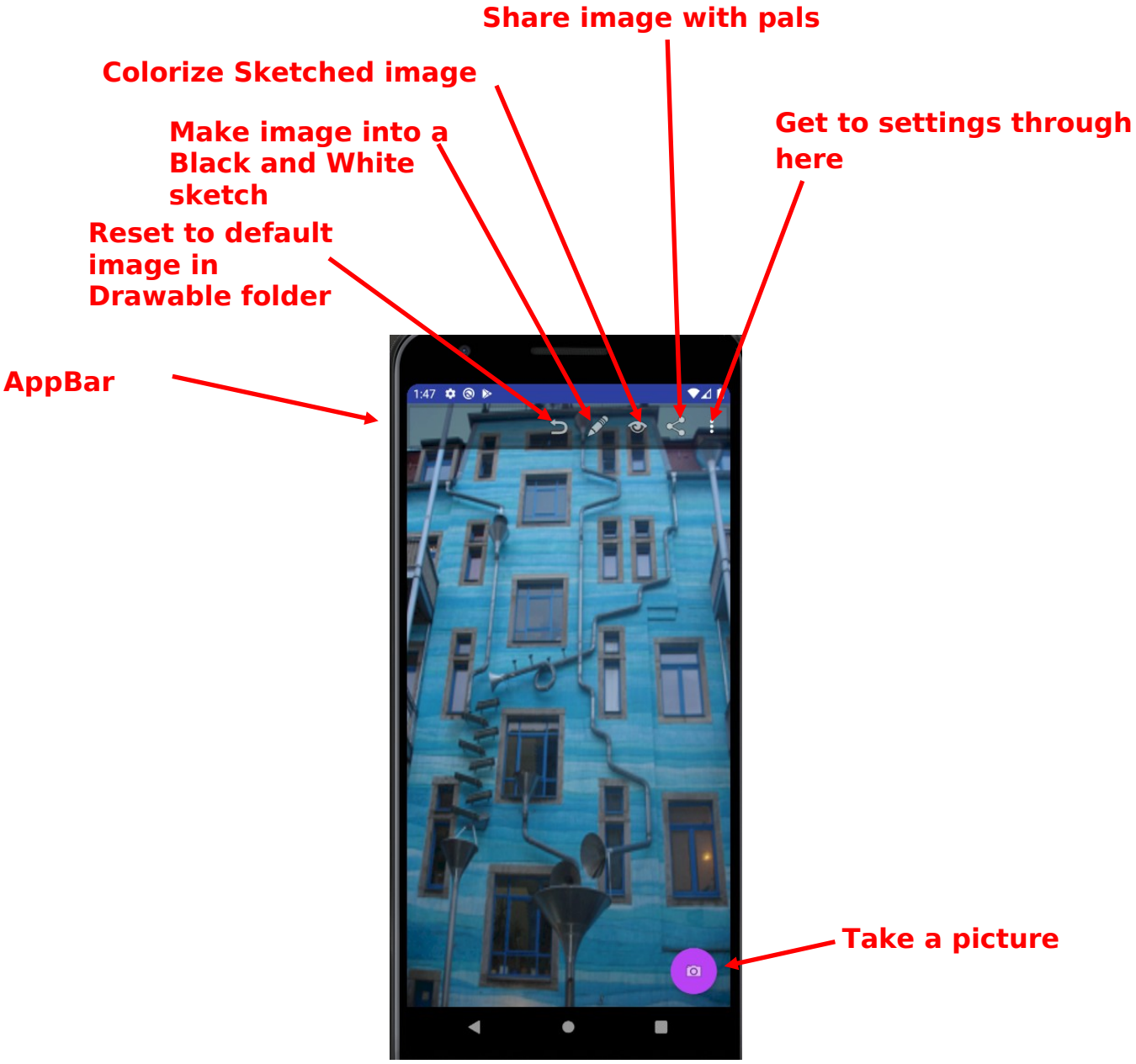
CPEN 475
Project 2: Implicit intents-bitmap manipulation-libraries
Target Environment: Pixel 3a or C running API 29

Overall Description
Please develop an application that can take a picture using the device camera, turn it into a Black and White sketch, colorize it, and share this image with others. On startup your GUI will should look like the following. (Note that the image goes from edge to edge)



What I have provided

- A starter project with;
    - All of the image processing code
    - A module library (bitmap_utilities) contains
        - Bitmap_Helpers – A wrapper class which simplifies and exposes image manipulation routines.
        - Other imaging processing static functions
    - A module Project2_Solution_Color which is where you will do most of your work and is also the module that you run the project from
- An apk of the project, make sure your final application does all that my apk does

Your job:

- set up permissions
- set up seekbar preference
- set up app bar with buttons
- correctly share processed image
- correctly grab and use image returned from camera app
- do media scan to find files
- setup pref change listener

Things I will look for;

1. If you have taken a picture with the app or processed it using the sketch or colorize button then when you start the app the next time this should be the image it shows you
2. When you hit the reset button the app will again show the default image
3. You must check Permissions before any action that would access images on external storage or invoking the camera

To Start

You have to use logcat to get past the initial bugs in the project. Look for the exceptions that your program throws.

Grading:

20% permissions
20% implicit camera intent set up with URI for file
20% image sharing – meaning onActivityResult handles both requestCode and resultCode as well as URI correctly
20% settings and preference change listener
20% correctly implemented program flow

Permissions

Please place all images (those generated by the camera and those that you generate) in EXTERNAL Storage. Make sure you set proper permissions in your manifest (the app is set up to handle runtime permissions once this is done) make sure you ask the user for permissions, the ones you need for this app are;

android.permission.CAMERA
android.permission.WRITE_EXTERNAL_STORAGE
android.permission.READ_EXTERNAL_STORAGE

Locking Orientation

Usually the only time you want to lock orientation is when you are writing a game, where the phone needs to stay in landscape while you plink away at targets. For this app however, I am only going to test in portrait mode. In the real world, phones with slider keyboards would have a problem with this as keyboards are used in landscape mode. If your app is locked in portrait then it will appear sideways on the screen. Use the following code in your manifest for the activity to lock your app into portrait mode if you wish.

```
android:screenOrientation="portrait"
```

Emulator

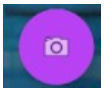
The emulator is excellent for this. Its backed by your machines horsepower so things will be quite a bit snappier than a physcal device.

UI explanation
AppBar (also known as toolbar)

 You have to set up an AppBar like this for this project. You can make the appbar transparent by placing the following in See lecture notes and projects for help;

 **Reset**

`calls doReset()` See this function for more info and tasks.

 **Take a picture**

Implemented as a Floated Action Button and calls doTakePicture()

In doTakePicture(), create and start an implicit intent to take a picture. Please attach a Uri for where the camera should save the photo to this intent.

The Uri should point to a location in the first external media directories. Use the call getExternalMediaDirs() to get these directories.

Please use startActivityForResult(…) and properly handle what the camera returns, including the case of nothing (user chose to cancel).

If a photo was taken, you will need to get the image from where it was left on the filesystem by the camera. I used

Camera_Helpers.*loadAndScaleImage*(**originalImagePath**, **screenheight**, **screenwidth**);

Also be sure to invoke the mediascanner so it shows up in the devices Photos app.

Please see example projects and lectures online for further help.

 **Make the image Sketchy**

Calls doSketch(). See this function for more info and tasks.

If you are interested in the process of how this was done see the following link for Adobes description of the process

http://www.createblog.com/paintshop-pro-tutorials/14018-sketch-effect/

Colorize the image

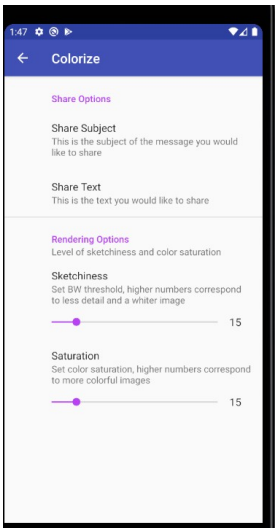
Calls doColorize(). Uses the member vars `saturation and bwPercent which are preferences.`These come from preferences set in the Settings Activity and are updated via the preference change listener.
`See this function for more info and tasks.`

Invoking a communications app to send picture

Calls doShare(). Uses the member vars shareSubject and shareText. These come from preferences set in the Settings Activity and are updated via the preference change listener. Be sure to attach the processed image to the message.
See this function for more info and tasks. See example projects and lectures online.

Settings

Clicking the 3 dots brings up the settings preference Activity:



Calls up an EditText that allows you to customize your share info

Seekbarpreference to set the saturation and 'sketchiness' of image

Bug Bounty
This app resets the initial image size incorrectly on first install. If you can provide a fix I will give you 5% on this projects grade.