Post Mortem - MultiFuncApp

When assigned this Android application project. I was fairly excited and confident when I discovered that it was related to a project that I had already done in a previous class. Being that the application had to do with a multi-view application, meaning that there had to multiple windows (or views) that would open while using the app, I was a little confused and wary as to what that entailed in this assignment. However, when starting and going through this assignment, I didn't focus on the fact that it had to be multi-view, and rather just focused on the other requirements, where we needed to integrate the camera in a device, to take a picture, have options to manipulate that picture, and to save the image pre-manipulation and post-manipulation in the device's gallery. Another requirement that I had to integrate, was another Android hardware function of my own choosing, in which I chose to use the microphone, to listen for a user's voice and convert their speech on ending the recording, and convert the speech to text. I had the code down for this function, but while working on this assignment I was having a hard time trying to find a place to integrate it into the main activity of the app itself. I didn't want the individual buttons and activities running into the other functions and buttons on the initial screen, because I knew that the application had to be multi-view. I was having a lot of trouble with this, until it was covered in class and demonstrated that the application can have an activity file of its own, and can be called upon within the activity. I was finally able to get this working, and the strange errors that I could not seem to find a solution for anywhere online, were finally resolved. One particular issue I was having, which took up more of my time than I would like to admit, was that I was trying to find an online resource demonstrating how to make a round button. I was going to use this round button for my recording speech-to-text function button, and after many

failed iterations of trying to get this working, I found one that required a lot of additional xml files and cross references between them, as well as importing a package in order to have the proper formatting objects within those files. Strange thing is, when using the Android emulator within Visual Studio, the formatting of that button worked perfectly as I wanted it to, but when using the app on my Android device, the button did have the formatting I wanted but the text within the button was skewed. I figure that this may be because the version of Android that is running on my personal device is Android 6.0 Marshmallow vs. the emulator running Android Nougat 7.1.1. I did find it very useful and relaxing to be able to use my notes and already written code for image manipulation from a previous class, and to be able to use it and integrate it into an Android application, which is far more user friendly and intuitive than running it manually through an IDE and image-viewer application.