Clarapplai

Thank you for applying to Clarifai for a position in Android development. For this task, we would like to see your real-world engineering skills in action.

You're going to make a facial-recognition app. This sounds really daunting, but you have Clarifai's API to make it simpler!

What does the API do for me?

For every image that Clarifai's API analyzes, it returns a value called an **embedding**. This is just a fancy term for a 1024-dimension vector (which we represent to you in our API as a `double[]` with length of 1024). The similarity of two images can be determined by simply taking the Euclidean distance between their embedding vectors (similar to how you'd determine the distance between two points on a 2D graph, or between two points in 3D space, but for 1024 dimensions).

So you just have to pass images of faces to the API, and compare the embeddings on two images to determine the likelihood that they're the same face. Embeddings that are very close to one another are very likely to be of the same face, and embeddings that are very far apart are very likely to be of different faces.

So, what is the exact goal?

We want you to write a selfie-detector app. So the app will:

- Allow the user to train the app on pictures of their own face. You will reach out to the Clarifai API, get embeddings for these images, and use these as the owner's face.
- Allow the user to take a picture of *any* face. You will reach out to the Clarifai API, get embeddings for these images, and then compare them against any embeddings that the user has trained the app on, and tell the user if you think the image contains their face or not.

The minutiae of the implementation is left up to you. You can have two buttons, a "train" button and a "judge" button, that will do each of the two actions above. You could do something else if you think there's a more elegant or simple way to implement it.

You can make the user upload a picture they've already taken in their camera app, or just have them take a picture right then and there and upload that. Or give them the option between the two. Either way is fine.

Sounds like a lot of work

It is, as would be making any app from scratch! So we made this boilerplate app for you to build on, and took care of some of some annoying time-wasters, such as:

- The Clarifai API doesn't take care of cropping around faces for you! We have included this library

for your convenience. All you have to do is `faceCropper.getCroppedImage(bitmap)` to get an image that is cropped to the correct area, which you can then send to the Clarifai API.
  - Don't worry about correcting inaccuracies in the output of `.getCroppedImage`; it's a thin wrapper around the built-in Android face-detection libs, and we will just trust its output.
  - You can assume the user will always take a picture with exactly one person's face in the frame. In real-life, we would have to guard against this, but for the sake of this exercise, don't fret over what to do if there are no people, or two people, in the image.
    - *Of course, if you have extra time and* **want** *to try to guard against these scenarios, please do so!*
- We will provide you with an API ID and secret to use via email.
- We've provided you with a "Java client" to hit our API. Start by constructing an instance of `ClarifaiAPI` and go from there.
- We've provided you with an `ImagePicker` that consists of a few static methods that help with image-related intents.

How polished should this be, and what's allowed?

- Don't reinvent the wheel! If there is a library that you would use to solve this problem in your day-to-day life, go grab that library and include it. **You can use any library you want.**
  - If you're curious, some libraries I'd use to get through this as quickly as possible:
    - Glide
    - Otto or EventBus
    - ButterKnife
    - Timber
- In a similar vein, copying code from StackOverflow or some other site **is allowed.** Again, you'd be doing that on the job all the time, anyway. *If you do copy some code from online though, try to remember to put a comment above it with a URL to visit the source.* StackOverflow does require you to properly credit the author of any code you take from them.
- Feel free to use **any language** you want. Here at Clarifai, we use Java and Kotlin in our Android apps. But if it runs, we'll accept it for this exercise. If you want to use Scala, go ahead.
- Bump up the minimum Android version if you have to. Even minSdkVersion 23 is fine with us.
- The design can be as sparse or as fancy as you want it to be. Any extra effort you put into making things look nice will be taken into account, but make sure that you solve the problem at hand before you pretty it up.
- We can't expect you to spend a whole day on this, so we also can't expect the app to be totally polished. Act like you're at a hackathon! For example, it's totally fine if:
  - there are memory leaks when you close the app, rotate the phone, etc
  - the app crashes when there's a network error
  - the app violates Material Design principles or has janky-looking Views. Don't bother testing on different screen sizes or anything either. We'll just test it on our end on an emulator with the same resolution and Android version as your emulator.
  - all of your training data is stored in memory and thus lost when the app is killed. Persistence is annoying and we trust you know how to do it. Don't waste your time on SQL.

- **Feel free to surprise us!** If you have an idea for how something should be implemented that sounds like it diverges from the spec slightly, go for it.
- **If you have a more fun idea than this app, go for it**, we want to see what you're capable of!

How should I get started?

- Start in `MainActivity` and work from there. Also look at all of the `public class`es in the `api` and `util` packages. You probably don't need to look at the non-public ones, but go ahead if you want to.
- We will be available if you have any questions throughout the process. Send an email to kmost@clarifai.com if you get stuck at any point.
- Try to keep it around 2-3 hours.