Final Report

Brooke Claroni, Jekko Syquia, Giana Fiore

Section 1

Brooke Claroni

- Vector graphics
- Login UI
- Signup UI
- Navigation Screen UI
- Portfolio UI
- Upload Photo
- Firebase services
    - Authentication
    - Storage
    - Database
- Pixabay API
- Picasso library
- Anko library
- Okhttp3 library

Jekko Syquia

- Results Activity UI
- Image Overlay UI
- Photo Composer Library
- Subject Detection
    - Subject Selection
    - Automatic Subject Selection
- Subject Labeling (MLKit)
- Camera UI
- Permissions

Giana Fiore

- Grading UI
- Details UI
- Grading algorithms
    - Rule of thirds
    - Framing
    - Exposure
    - Clarity
    - Level
- Detailed recommendations
- Application logo

Section 2

PhotoComposer is an Android smart-camera app that teaches users how to *take* better photos. While there are many photography applications on the market, they are broadly geared toward *making* better photos with editing and filters. PhotoComposer enables users to actually capture better images, minimizing postprocessing and delivering a superior product. To accomplish this, the application carries out the following three steps:

1. The application accepts a photo from the user via upload or live capture and automatically detects the photo's subject.
2. The application analyzes the photo's composition and provides the user with an overall grade, along with a detailed evaluation based on five technical photography criteria: rule of thirds, framing, exposure, clarity, and level.
3. The application provides the user with actionable suggestions so the user may better capture the subject next time, and also offers reference images of well-received photos of the same subject for the user to find inspiration.

Section 3

Our website includes:

- Brief description of our project
- Bio and photo of each team member
- Project presentation screencast
- Commercial screencast
- Links to writing assignments
- Link to Final Report (this document)

Section 4

Libraries, packages, and APIs:

- ML Kit
- TensorFlow
- Firebase
- OpenCV
- Pixabay API
- Picasso
- Anko
- Okhttp3

Section 5

PhotoComposer comprises the following activities: login and sign up, navigation menu, portfolio, photo selection, photo grading, and detailed feedback. Activities communicate synchronously via Intents.

Photos may be selected by the user from the device photo gallery or captured with the device camera. Our application leverages machine learning modules (ML Kit and TensorFlow) to identify the primary subject of the photo and extract its relative location and label. Computer vision (OpenCV) is harnessed to analyze the image and provide grades for the following criteria: rule of thirds, framing, exposure, clarity and level. In addition, tailored recommendations are provided, alongside popular images of the same subject (Pixabay API) for inspiration. Finally, the user's photos and corresponding grades, as well as account credentials, are stored with a cloud-hosted database (Firebase).

Section 6

Brooke Claroni

I would consider optimizing image loading. I would choose to use Glide rather than Picasso for image loading and/or I would incorporate an image compressor since the portfolio is slow to load. I would also consider improving the security rules on the Firebase services I implemented. Additionally, I would consider creating the skeleton on a cross-platform framework such as Cordova, perhaps. Lastly, I would consider constructing a new photo uploader from scratch rather than including Android's native photo uploader since it has its own issues.

Jekko Syquia

If I was to work with this again I would fully work with just using tensorflow for the whole process instead of switching around with different APIs. I would have focused on the camera more and also a better algorithm in terms of finding the main subject since the main subject is mostly determined by the largest area. I would also consider adding threading to speed up the process of post processing after a picture is taken. The camera application would definitely benefit from adding more basic features. Lastly, instead of focusing on the object detection training I should have focused on the image classification instead since the label is more important. Finally, I would allow the option of selecting multiple subjects instead of one.

Giana Fiore

In terms of technical lessons, developing with the OpenCV image processing library was a significant learning experience. Though I don't regret choosing OpenCV, as I believe it was the best fit for our project, it is particularly confusing and time consuming to get running. While there is extensive online documentation, errors and solutions vary greatly by platform and OpenCV version. As such, it was challenging to get set up properly, before any algorithms could be developed. To any future teams using OpenCV for Android development, I would recommend the following steps to use the library (OpenCV 3.4.10 and Android Studio 4.1.2):

- Download OpenCV for Android here
- In Android Studio, File → New → Import Module
- Choose OpenCV-android-sdk/sdk from Downloads

- Click Finish
- File → Project Structure
    - Add module dependency to app
- sdk → build.gradle
    - Make compileSdkVersion and targetSdkVersion the same as app → build.gradle compileSdkVersion and targetSdkVersion
- Ensure the following user permissions are added to the manifest
    - READ_EXTERNAL_STORAGE
    - WRITE_EXTERNAL_STORAGE
- Allow the permissions in device settings
- Use OpenCVLoader.initDebug() to test for proper set up
- If you encounter difficulties, Gradle Sync in Android Studio

As for what I would do differently if I did this project over again, I think that clearly defining the photo grading criteria early on, as well as the grading measure, would have been helpful. Because we went through three different iterations of grading (letter grade, sliding scale, and, finally, stars), the process was ultimately more time consuming than it needed to be. Had I settled on the grading criteria at the beginning and created a detailed mock up of the UI to follow as a template, I would have spent far less time reworking the algorithms to accommodate for a new grading scale each time the UI was changed.

Section 7

How to

Before beginning make sure the following requirements are met to build and run the application.

- Make sure to register and acquire your Pixabay Api Key here. This is required to build the source code.
    - Copy the API key create a new file called `keys.xml` under PictureComposer > app > src > main > res > values

```
<?xml version="1.0" encoding="utf-8"?>

<resources>

    <string name="pixabay_api_key">PASTE PIXABAY API KEY HERE</string>

</resources>
```

- Please use on a Pixel 3 device running API 28 + or emulate Pixel 3 API 28 through Android Studio's AVD Manager
- If running on a physical device make sure to enable Developer options.
- Android Studio 4.13 or newer to build the source codes

To build and run on an emulator, make sure the requirements are met in the important notes above before proceeding. Download our zip folder of code and unzip. Open an existing project in Android Studio and select PictureComposer from your documents.

Pixel 3 API 28 should be selected beside the `app` configuration toolbar, if not please select *Pixel 3 API 28* as seen on the screenshot below.

Run `Gradle Sync` by clicking the small elephant icon in the top right corner of Android Studio.

To `Run` the application press the green play button next to the emulator name (Pixel 3 API 28 in this example)

This will perform a Gradle Build and if it is successful a pop up will display saying “Success. Operation Succeeded” and the app will install and launch on your emulator on a separate emulator window.

What works, what doesn’t, what to be aware of

When testing on physical devices, we noticed some photos would automatically rotate 90 degrees when displaying. This did not occur with the emulator’s camera. To our knowledge, all of our other functionality should work across any device Android 5.0 or greater.

Next steps

To continue with development on our project, you may want to gain access to this project’s Firebase console. To do so, email a current admin and request permission to join the project as a Firebase admin. Using a personal email for this is recommended since Firebase treats student accounts differently.

Some of our ideas for potential project upgrades are included in Section 6. Other ideas include:

- Adapting subject detection, leveling, and grading to better accommodate photos that don’t have a clear subject. For example, landscapes without any defining feature. Currently, photos for which a subject cannot be detected are not graded- the user is simply advised to choose a different photo by the subject labeling activity. This is an excellent opportunity for expansion.
- Allowing the user to manually box and label a subject (for example, though a text submission field), would be an interesting feature. Though we didn’t have the time to implement this, it would allow for outliers that cannot be detected to still be graded with some user input.
- Adding iOS support would be a useful addition to the project. Though we chose Android development deliberately (as the team had some experience and there are more global Android users), it would be great if PhotoComposer could be accessible to iPhone users as well.

Relevant Documents

All relevant documents have been included.