CS 5450: Networked and Distributed Computing
Lab 4
Authentication and Mobile
Fall 2017
Instructor: Professor Vitaly Shmatikov
TA: Yunhao Zhang

Due: 11:59PM Thu, Dec 7th 2017

Introduction

In this lab we will work with Google Firebase as a backend for Android platform and develop an app that utilizes its functions. The app will be a cloud photo management app, where users can store their photos and share them with others. You would need to develop an application and then record a *short demo* demonstrating the functionality of the app and integration with Firebase.

Different from previous labs, you will focus on gluing different platforms and libraries together instead of implementing the functionalities with your own code logic. For example, Firebase has provided storage service, account management and authentication. Be sure to read the mannuals or documents of existing systems and understand what they provide before you start coding.

You may need to read several "Hello World" instructions, such as those for Android app, Firebase and Docker. Feel free to share useful and related materials on Piazza.

App Description

The app helps users to store and share photos. There are two types of photos: Public (available to everyone) and Private (available to the user). Each photo has a short description. The app has five views:

1. **Photos** — This part allows a user to look through photos that were uploaded by anyone and have a tag Public. If the user is authenticated then there should be additionally Private photos that were uploaded by this user. You don't need to implement preview of photos, just make it a small rectangle with description (simple is better).
2. **Upload** — A user can upload photos, provide description and tag them either Public or Private. You should use Firebase data storage and database. Additionally, Security Rules should be used to manage access for photos on the Firebase level.

   NOTE: This section is only available if the user is already authenticated. Unauthenticated users can't upload anything.
3. **Search** — The user should look up the image by description. If the user is authenticated, then search is enabled for Private images, otherwise only Public images are available.
4. **Auth** — User authenticates with the app using Email or Google Sign-in (OAuth). The authentication doesn't happen at the start of the app and is voluntarily.

5. **Post-Processing Photos** — Similar to the first view, this view allows a user to look through photos that were uploaded by anyone with Public tag and applied some post-processing effect. You are free to choose the post-processing effects and hints for implementation are described later.

IMPORTANT: As this lab is not dedicated for UI design, you don't need to implement much of it, and it won't impact the grade, but during the demo you would need to explain us, how it works. Assume no more than 5 photos for simplicity.

Milestone I

It's a good idea to implement view1-4 first and then consider post-processing service. The steps for this milestone should be:

1. Install Android Studio (or Eclipse) and create a sample project. There are many online instructions about how to create a "Hello World" Android application.
2. Setup a Firebase account.
3. Design a dumb UI.
4. Create Sign-in and Sign up with Email, Google Account (using OAuth) through Firebase.
5. Implement upload of private and public photos by the user, using Firebase Storage service.
6. Implement photo search by description (with Public and Private results). Show only one result. You may need Firebase Database for help.
7. Write two test cases that test your Firebase Security Rules. You can do it by hand, or in Test Lab using Espresso, Robotium, or UI Automator 2.0.

Milestone II

Most real-life applications require post-processing, such as adding watermarks, increaseing brightness, etc. After Milestone I is working, you can start to add post-processing functionality:

1. Your service will be packaged as a Docker container running on an instance in the cloud of your choice.
2. Your service will check for new data in the Firebase data store.
3. For each new picture, your service will download it, perform the post-processing transformation locally and upload the picture (or some metadata) back to Firebase Storage.
4. In the Android app, users can view these modified photos in "Post-Processing Photos" view.

You can run Docker on your own computer first for fast debugging. You do **NOT** need to implement image processing yourself. Instead, you need to find some existing photo processing libraries and wrap them as a service inside the Docker image. You will practice how to deploy, run and debug others' open-sourced project in this milestone.

For an example solution of this post-processing service, the steps are:

1. Install Docker on local machine (or AWS EC2).
2. Follow the instructions *here* (https://github.com/AVatch/ascii-image-service) and deploy the ASCII-image-service inside Docker.
3. Modify the ASCII-image-service so that it works well with your Firebase configuration and Android app.

Feel free to implement any post-processing feature you like. Some other examples are:

1. Use *Google Vision API* to extract text from a picture (assuming your pictures contain text).
2. Use *Google Vision API* to extract features from a picture.
3. Use *FacePlusPlus API* for face detection.
4. Use *ASCII art* to transform pictures into ASCII.

Grading Criteria

Record a demo of the working app (showing different use cases). There are many online instructions about how to run Android applications on PC/Mac simulators.

Demo should include a recording of your laptop screen, where you demonstrate work of your app, as well as demonstrate that Firebase correctly creates users, stores photos and uses security Rules.

Your grade will mostly be based on the video demo that you have to submit. Additionally we would want to see your code to verify your code style. During upload, be careful to eliminate the unnecessary binary files and contain the source files.