CS-E4100 Fall 2017
Mobile Cloud Computing (5 cr)

Group Project – Event-based picture sharing app

Deadline **Sunday, 10 December 2017 at 23:59** (local time in Finland)

Submission Through the Aalto Version Control System (`https://version.aalto.fi`)

Revision `v1.0 (01.11.2017)`

Projects must be submitted according to the instructions in this document (see the "Submission" section for the details). No other form of submission will be accepted. Deadlines are hard. No extensions will be granted. Should you have any issues in accessing MyCourses, A+ or Aalto Version Control System, contact the course staff at `cs-e4100@aalto.fi` as soon as possible. Also refer to the MyCourses workspace for additional information (`https://mycourses.aalto.fi/course/view.php?id=16930`).

Overview

Summary The purpose of the group project is to realize an application to share and synchronize pictures between users. In particular, the application should support the creation of groups of users participating to a certain event for sharing related pictures; this enables all pictures taken by a user within a certain time period to be synchronized in real-time to all other users belonging to the same group. To this end, the application also needs to support authentication of users and the creation of groups. The pictures should be automatically categorized on the cloud based on the presence of people (e.g., for selfies) or the lack thereof (e.g., for landscapes). Finally, the application performs local image processing to avoid sending pictures which contain sensitive information to the cloud, so as to preserve the privacy of the users.

Components The application consists of two components: a frontend and a backend. The *frontend* is the user interface running on a mobile device to access the functionalities of the service. The *backend* consists of a database and functions for processing and storage (see the "Backend" section for more details).

Users The application must support one user per device and a user may be logged in on multiple devices at once.

Frontend

Requirements The frontend must be implemented as an Android application and must realize the following functions.

- Provide a login screen to authenticate users; this can be supported by using FirebaseUI Auth (`https://firebase.google.com/docs/auth/`).
- Allow users to take a picture (with the camera of their mobile device).
- Identify pictures that contain sensitive data by detecting the presence of a 2D barcode.
- Display a gallery with two types of albums: *private* and *group pictures*. The private album contains pictures that are sensitive (i.e., they contain a 2D barcode) and only concerns to the main user; the other type of album stores all the group pictures shared between the members of a group.

- Handle all operations without slowing down or freezing the user interface. Specifically, processing and downloading should be done in the background. Additionally, the application should properly manage loading and displaying large images without running out of memory.

Functionality The frontend should first show a welcome screen prompting for password-based authentication. Upon successful login, the frontend should show a grid menu layout with the following options.

- *Gallery.* The gallery screen should display a grid of albums. Pictures that contain sensitive information should be displayed in the *Private* album. For the group pictures, the application must display albums with the name associated with the group (for instance, Picnic, Wappu) and a picture belonging to that group. Inside each group album, the pictures are divided in two categories: *People*, for the pictures containing faces, and *Not people*, for all other pictures (i.e., those that do not contain faces). This categorization is carried out at the backend. The application should also include an option to show pictures grouped by the user who took them. The content of the albums should be automatically refreshed when other members of the group take pictures as long as these pictures do not contain sensitive information. Synchronization should take place even if the app is not in the foreground or if it is closed. A message in the notification bar should inform the user whenever pictures are added to their gallery. The application must provide an option to open the picture (in both portrait and landscape mode) and support zooming. In addition, the interface should allow to download the picture in its full resolution (refer to "App settings" below for additional details).
- *Take picture.* The frontend should provide a live camera preview and allow the user to take a picture. Non-sensitive pictures taken by users should be synchronized across all users belonging to the corresponding group. Upon taking a picture, an analysis is carried out to detect if there is sensitive data (namely, 2D barcodes, for instance, those that appear in a boarding pass). The detection should be carried out locally on the mobile device and it should run in the background to guarantee that the camera does not freeze.
- *Group management.* The group management screen contains options for creating, joining and leaving (or deleting) a group. Upon group creation the application shows a form with different fields to setup the related properties. Among them, two fields are mandatory: the group name and its expiration, i.e., the time after which the event is over and synchronization is no longer performed. Once the group is created, other members can join that group by scanning a QR code which is displayed on the device of any of the current members. To increase security, the QR code encodes a server-side *single-use token* in addition to a group ID. Single-use means that the token expires after a user has joined a group through that token. The application should automatically refresh the QR code after the token has been used. Once a user is part of a group, he (she) can show the QR code on his (her) device for others to join that group. The application must ensure that each user belongs only to one group at a time. The frontend should display the list of group members too. Each member of the group has the option of leaving the group. When a user leaves a group all other users are notified but his (her) pictures must remain in the devices of the other group members. The creator of the group does not have the option to leave the group but can delete a group instead. This means that the group is no longer valid (i.e., as it instantly expires upon deletion).
- *App settings.* A settings screen should allow users to specify a different application behavior based on wireless connectivity, i.e., WiFi and mobile data. In particular, the set-

tings screen allows to specify the image quality when uploading or synchronizing pictures under WiFi or mobile data network. The image resolution can be *low* (640 by 480 pixels), *high* (1280 by 960 pixels) and *full* (original size).

Backend

Architecture The backend must be implemented using Google App Engine Flexible Environment. The application must use the Firebase Realtime Database to sync information across different devices and Firebase Storage to store images.

Requirements The backend should provide APIs for group management as follows.

- *Create a group*, which returns a unique group ID and single-use token, which is used by the frontend to create a QR code (refer to Section "Group management" under "Functionality").
- *Join a group*, given the group ID and single-use token.
- *Delete or leave a group*, given the group ID. The operation depends on whether the user trying to delete the group is the creator of the group. Only the creator of the group is allowed to delete the group (which involves deleting all group information and pictures). If the user attempting the operation is not the creator of the group, the user simply leaves the group and cannot upload or download any further pictures from that group.

Authorization Only authenticated users should be able to carry out the operations listed above. This can be done by verifying the User ID tokens (`https://firebase.google.com/docs/auth/admin/verify-id-tokens`) on the backend. The APIs described earlier should at least return the information described above; you can decide to include further information in the responses sent by the backend.

Other tasks In addition, the backend should perform the following tasks:

- Delete the group when the lifetime of the group has elapsed. This includes deleting any information about the group as well as the corresponding pictures stored in the cloud.
- Label uploaded images based on whether people are present in the picture or not (refer to Section "Gallery" under "Functionality").
- Save low resolution and high resolution versions of each uploaded picture (refer to Section "App settings" under "Functionality") in addition to the full resolution pictures.

Access rules Finally, the Firebase Realtime Database and Firebase Storage must make use of appropriate rules so as to prevent access to unauthorized users. Specifically, the information of a particular group (including the list of members, expiry time and user generated pictures) should be accessible only to members belonging to the group. The pictures should not be available publicly.

Challenges

Overview The following tasks *are not required* to complete the project but give extra points. You can pick any combination of the challenges listed below, however, **the total number of extra points given by the challenges is limited to a maximum of 10 points**.

- Realize a responsive web application that allows to access the photo albums (as a frontend) and include an option to delete pictures from the cloud, which is useful in case a user cannot access the mobile application (**4 points**).
- Allow users to join a group by shake pairing: a user can join a group if he (she) is physically close to a member of that group and they both shake their mobile devices at the

same time. The backend should detect the shake pairing and add the new user to the relevant group (**5 points**).

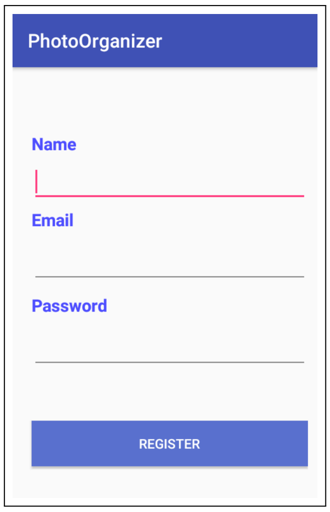
- Free challenge: do you have a good idea on how to improve this application? Are there other features which would turn this into an app that people would like to use? Impress us with these features and get extra points (**1–10 points**).

Submission

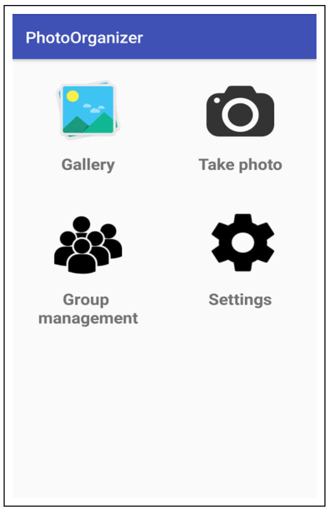
Instructions The project source code must be submitted through the Aalto Version Control System (`https://version.aalto.fi`) on the git repository associated with your group. Specifically, the name of the repository is '`mcc-fall-2017-gXX`', where `XX` is your group number. Additional information about using the Aalto Version Control system and the setup of group-specific repositories is available in MyCourses under the "Software project" section (`https://mycourses.aalto.fi/course/view.php?id=16930&section=4`). No explicit action is required to submit the code: **the last commit before the submission deadline will be used for evaluation**.

Evaluation Each group must show a demo of the software project to the teaching assistants. A time for the demo has to be booked according to the instructions available in MyCourses under the "Software project" section (`https://mycourses.aalto.fi/course/view.php?id=16930&section=4`). The evaluation criteria for the software project are available in MyCourses under the "Software project" section (`https://mycourses.aalto.fi/course/view.php?id=16930&section=4`).

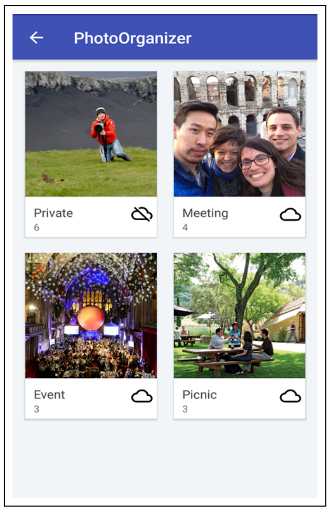
Reference user interface



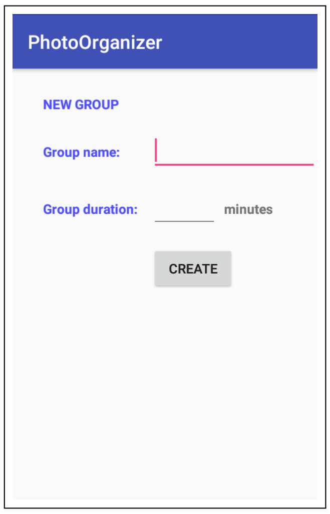
Sign up



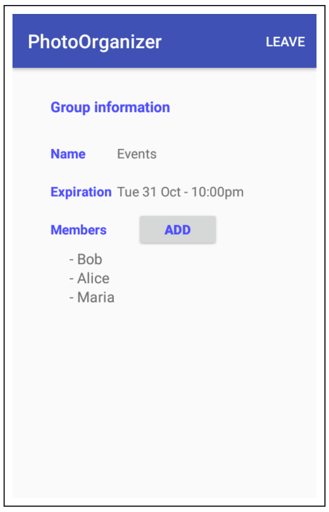
Grid Menu



Gallery



Create group



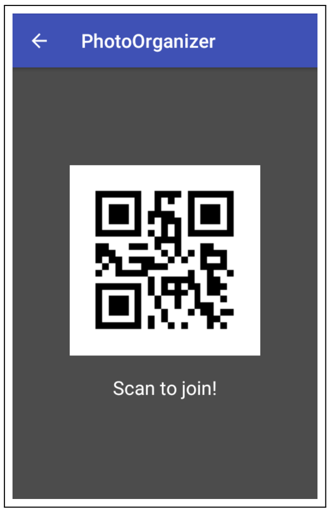
Name Events

Expiration Tue 31 Oct - 10:00pm

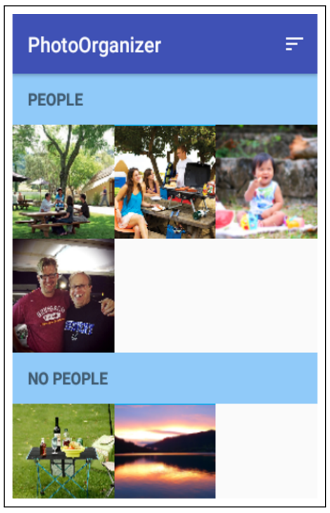
Members ADD

- Bob
- Alice
- Maria

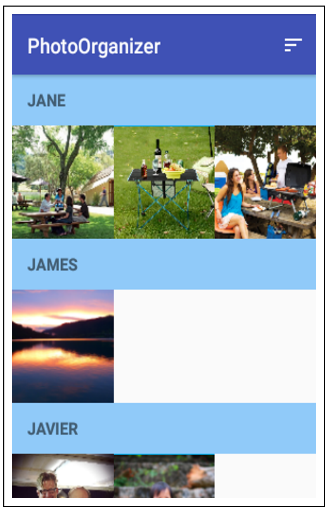
View group



Join group



Album sorted by people



Album sorted by author