Photo Management Application For Android Phones Documentation

1. Introduction

“Photo Manager” which is an app that helps user manage their personal photos. The interface is comfortable in a blue style and it can work both in portrait and landscape mode. The system has four main interfaces, namely gallery, album, search and map. The gallery part is used to browse the preview of photos and inspect the details of photos. The album part includes album list and a window for creating an album with a new name. The search part includes the search interface, the calendar for selecting the date, and the searching result will be displayed in the interface of gallery.

This app meets all the required functions which contain five main functions, and the details of these section are detailed in the third part. Furthermore, it can work on multiple devices with different screen size. We use MVVM pattern that allows fast reaction to design changes without modifying all the data structure. Meanwhile, using Rooms to store data locally and async processes to do worker thread in the background. We met all the requirements of the assignment.

2. System Design

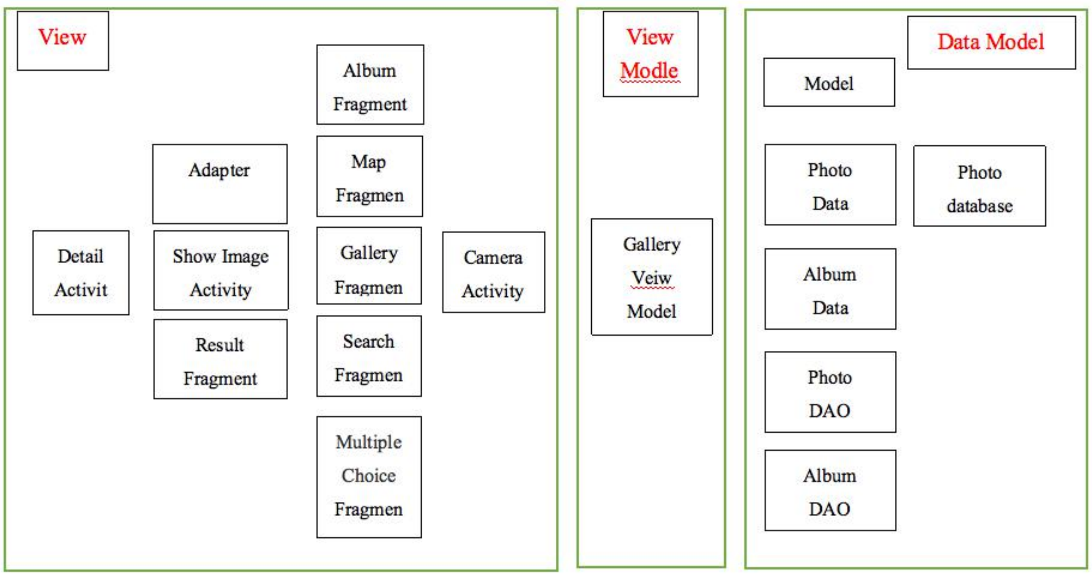

Figure 2.1 MVVM Model

MVVM pattern tries to remove the amount of glue code that we have to write to connect the view and the model. The view binds to observable variables and actions exposed by the viewModel in a flexible way.

3. Layout Design

There are three activities in our design, that is the main activity containing replaceable fragments for switching, one activity to show the full screen size image and one to show the detail of photo.

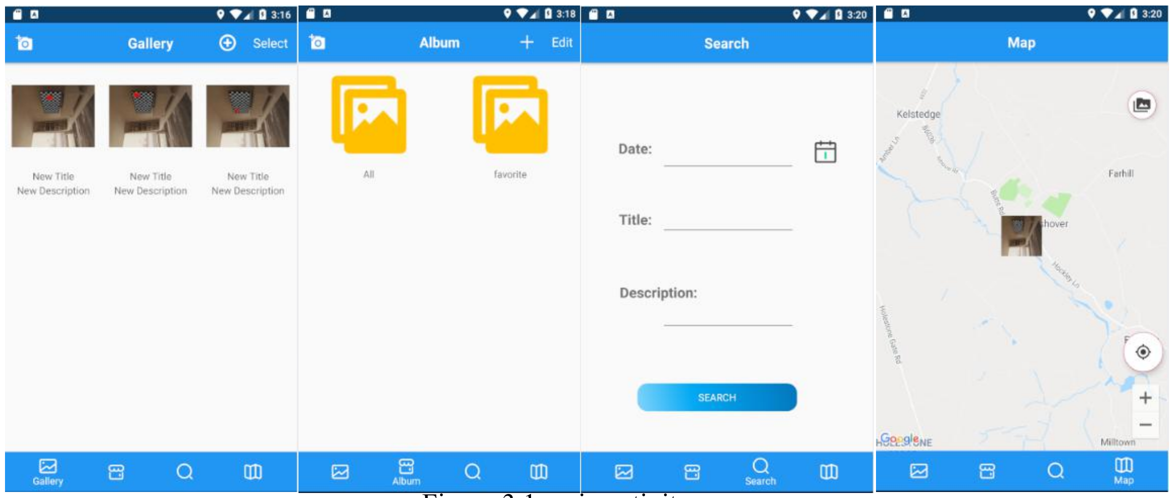

Figure 3.1 main activity

2.1 main activity

As shown in figure 3.1, main activity consists a menu toolbar, a bottom navigation bar, and a replaceable gallery, album, search, map fragments in the middle of the screen, selected by navigation bar at the bottom. Items in menu toolbar are slightly different between different fragments.

In gallery fragment, there are three menu items, to add new photo through camera, to add new photo from gallery of device, “Select” to choose multiple photos then choose to delete or to add them to the same album together.

In album fragment, menu toolbar slightly changed from gallery. ＋ is to create a new album, and “Edit” to choose multiple albums to delete.

In search and map fragments, there is no toolbar items but some buttons in fragments. gives users a way to choose date they want to search. is to show photos bounded and shown on the map now, and
go back to the users’ current place.

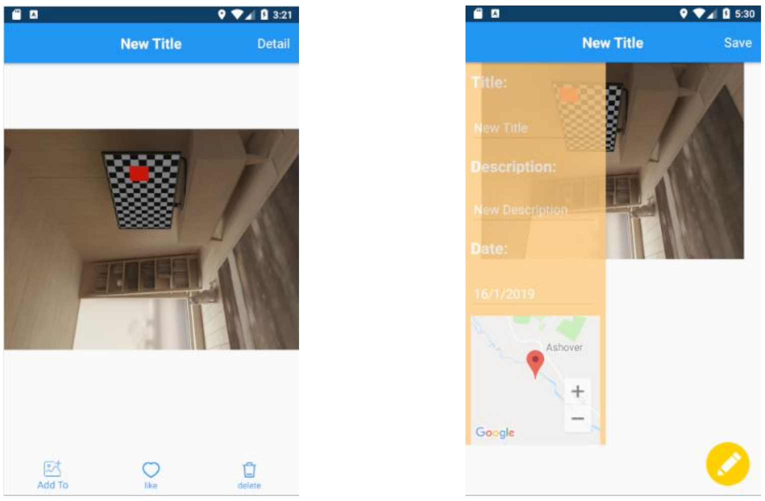

Figure 3.2 show image and detail activity

Show image activity's navigation bar has three items, to add the selected image to specific album,
 to add a favorite mark, to delete image. "Detail" button in menu skip to Detail activity.

Detail activity has a to edit information of photo and "Save" to save the modification.

4. Software Requirements Specification

The function of this system is mainly divided into five parts:

1. Visually browse previews of photos.
2. Showing pictures on a map.
3. Inspecting the details of a photo.
4. Taking pictures.
5. Saving metadata to a local database.

The purpose of this section is to introduce the details of the design, each function will be described. It includes the design details, interface design and the way we perform each function will be elaborated. In addition, we have added an album function that can be used to categorize images and browse, and a multi-select function to operate a batch of photos, such as delete or add to album in one time.

4.1 Visually browse previews of photos

We accomplish functions of previewing photos and looking up for detail. In terms of photo review, we used the LiveData, which can be considered as a tube between view and database. When there is a new photo added into or deleted from the database, the LiveData can be observed by the onChange() method with a callback. We can add the new data to the adapter, then show it on the recycleView. The adapter is the bridge between our data set and the ShowImageActivity. It should also be highlighted that the recycle view do a great job on the scrollable view and saved lots of memory.

4.2 Showing pictures on a map

The map fragment, applied the Google map api, which can show the feature of the map and we applied the location service which can return the current location (after we granted the location permission). We made an icon on the map to represent where photo was taken (re-size the match the map view and set the bitmap as icon place at the latitude and longitude stored in photo data), and the marker also shows the title of the photo. A single click on the icon can switch to the next photo (user may took many photo at a same place), furthermore, a single click on the title label can lead the user to the detail page of the photo by switching to ShowImageActivity. We also appended two buttons on the user interface, one will lead to the gallery which only contains the photo shown in the bound of the map which is accomplished by searching method comparing with the bounded latitude and longitude. The second one will take the user back to the current location with moving the camera to the current location callback from Location Service.

4.3 Inspecting the details of a photo

For looking up for details, the preview on the view can all be selected to be an entrance to ShowImageActivity, which shows a full screen photo with a detail. To do this, we pass the data of the picked photo to this activity and bind the information with the view (also contains a Google map with a mark of location). We can do this because our PhotoData implements Parcelable, which helps pass data with bundle from activity to a fragment. After switching to the ShowImageActivity, the detail can be modified through the text input layout via a Broadcast. The entered new value will be passed to the database and being updated.

4.4 Taking pictures

We use the EasyImage library to get access to the device camera and gallery (after we granted the camera permission).We can add a new photo by taking a new one with the camera or choose from the gallery (get the returned photo in onActivityResult()), and capture the location callback and time at the same time to create our new photo data to store into database.

4.5 Saving metadata to a local database

Our database extends RoomDatabase which contains all the basic function of insert, delete, search, update. We store our self-design PhotoData model and AlbumData model into it, contains entity of information (id, title, description, location, image path label of album name) with setter , getter, and override the Parcelable function for data transferring with bundle.

To contact with the database, we have two abstract class called PhotoDAO and AlbumDAO. PhotoDAO contains queries to specify action to the database. (insert/update/delete/search), and the albumDAO do the same as it.

Searching can be classified as search by pattern, search by the bound between a and b, or equivalence (use Like for a specific pattern, between for retrieving number between a and b, = to get the equivalence of keyword).

It is very important that we use the MVVM model to construct the whole app. Our class Model (Data Model) extends viewModel and has a PhotoDao and AlbumnDAO instance to use the function in DAO. The class Model also contain asyncTask for inserting or deleting data... etc to do action in the background to prevent from access to database from the UI. Our GalleryViewModel (view model) extends androidViewModel, which has instance of model for updating view.

4.6 Album

We wrote a new AlbumData model to store the information of album to database. To implement this function, we gave PhotoData a new attribute called album by assigning the album name to it. As a result, in the view of album, the action of clicking an album is doing a search to get photos having the same album label from database. In addition, we also add a button to add like in ShowImageActivity in order to provide a convenient way to categorize photos. Moreover, in our design of the view of AlbumFragment, there are two unchangeable album, one showing the full gallery and the other showing the favorite photos with like label.
We also add a multi-select function to delete a batch of albums in one time.

5. Division of Work

We have a total of 3 members. The division of labor for each person is as follows：

Weishi Li: (1) Complete the overall style design and layout design;
(2) Navigator bar, calendar design.
Chennan Luo: (1) Make most menu buttons' actions, connect the front-end UI and the back-end data.
(2) Implement search algorithm, design of function of multiple choices of photos and albums, delete action and add-like algorithm.
Yu-Chen Hung: (1) Database/photoData/PhotoDAO/albumData/albumDAO/adapter design
(2) Map fragment design and function.

We all agree that marks should be split equally among members.

6. Library Planned/Used

```
implementation 'com.github.jkwiecien:EasyImage:2.0.3'
implementation "android.arch.persistence.room:runtime:1.0.0"
implementation "android.arch.persistence.room:rxjava2:1.0.0"
implementation 'com.google.android.gms:play-services-maps:16.0.0'
implementation 'com.google.android.gms:play-services-location:16.+'
```

7. Conclusion

In conclusion, we finished all the requirements. We using MVVM to organize the separation of concerns. Then Rooms was used to store data locally successfully and use async processes. In addition, the jump between interfaces of the entire Android application layout was successfully completed. The interfaces can display the all information with simple and clear interface style. In terms of function of the implementation, we finish (1) browsing the preview and view details; (2) map view photos, (3) according to date, title and description of each photo, we can search them and display in the result interface; (4) gallery creation and more. After testing, these features can be implemented successfully in the simulator (PIXEL XL API 25).

In the whole process of development, the three team members all perform their duties and cooperate well. The entire R&D process is basically done together for communication and modification. This is really a great experience.

8. Bibliography

1. week3_lab: RecyclerView
2. week4_lab: EasyImage
3. week6_lab: LiveData
4. week8_lab: Google Map API
5. LiveData/ViewModel:https://android.jlelse.eu/android-architecture-components-room-livedata-and-viewmodel-fca5da39e26b
   https://android.jlelse.eu/android-architecture-components-livedata-1ce4ab3c0466
6. MVVM Model: https://www.jianshu.com/p/d9b10a3963ff
7. NewInstance between fragment:
   https://stackoverflow.com/questions/9245408/best-practice-for-instantiating-a-new-android-fragment
8. Parcelable: https://stackoverflow.com/questions/49249234/what-is-parcelable-in-android
9. Broadcast:
   https://stackoverflow.com/questions/10032480/how-to-pass-data-to-broadcastreceiver
10. Google map bound:
   https://stackoverflow.com/questions/16056366/android-google-maps-how-to-get-the-area-which-is-currently-shown-in-screen-devi