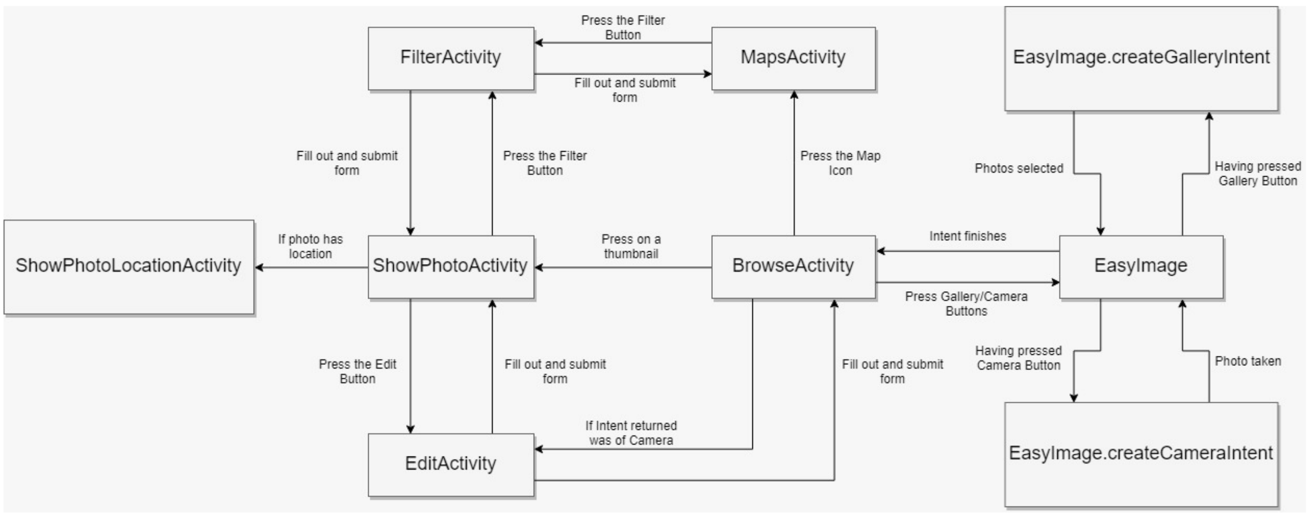
COM6510 Software Development for Mobile Devices
2018/19 final assignment report
Team: “TeamAlwaysLate” Members: Mangesh Hambarde, Akshay Deshpande, Wenbo Yin

1. Overview

- **Completeness.** All the requirements are fully implemented. There are additional developments and special features in some subsections, on which details are included.
- **Flexible layout.** All layouts in the app are designed in a way that they do not use absolute values, so they are flexible and adapt to different screen sizes and orientations.
- **Android version.** Minimum API level is 23 (Android 6.0).
- **Software pattern.** MVVM + Live Data is used to observe data directly returned from a Room database.
- **Flow chart.** Given below is a flow chart of the activities and intents. The default activity is BrowseActivity. (*Mistake: The connection between FilterActivity and ShowPhotoActivity should actually be between FilterActivity and BrowseActivity*)



2. Details of parts

2.1 Visually browse previews of photos

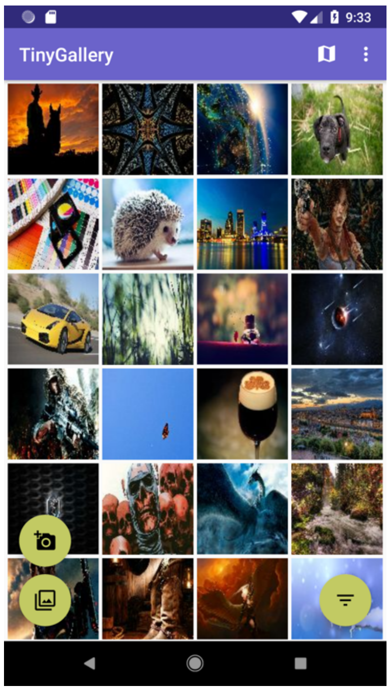


Figure 1. BrowseActivity on mobile

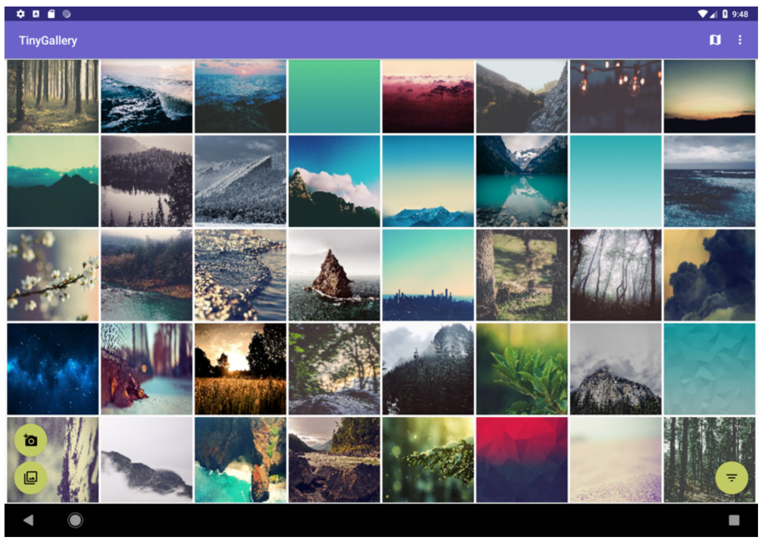


Figure 2. BrowseActivity on tablet

2.1.1 Requirements

- **Views.** Square grid RecylerView and three floating buttons for taking a picture, reading an existing picture, and filtering shown pictures. App bar button to switch to map view, hidden menu item to reset app data.
- **Flexibility.** Layout designed using RelativeLayout, without using absolute values so that layout is adaptive to any device size.
- **Permissions.** The activity asks permissions in onCreate(), and does not set itself up using the setup() method unless all permissions are granted.
- **Scanning for pictures.** In setup, the activity starts an async task to scan the phone for images using MediaStore provider. For each image found, the task enters a dummy entry into the DB, with the image's location and the path where its thumbnail should be built.
- **Live Data and MVVM.** The activity creates a live data observer from the room database of pictures at startup and updates it when a filter is set.
- **Thumbnails.** The thumbnail is actually built when the image is bound to the RecylerView, using an async task. This enables thumbnails to be shown immediately even if the user suddenly scrolls far below in the list. After thumbnail loading, it is cached in memory.
- **Performance.** Images are added to the adapter using DiffUtil [1] to speed up performance in the onChanged callback. Tested for 4000 photos from [2]
- **Deviations from requirements.** None.

2.1.2 Design choices

- **Idea.** The design is meant to be minimalist. It has consistent app colors defined in colors.xml. Colors were chosen according to the recommendations at [3].
- **Advantages of a square grid.** Having square grids allows more information to be shown on the screen at once, reducing the need to scroll too much. Google Photos also uses square thumbnails.
- **Permissions.** All permissions are asked on the first run on the app, to make things simpler.

2.1.3 Separation of concerns

- **Interaction with model.** BrowseActivity only interacts with the model using an instance of the ViewModel class. The only times it accesses it is for setting up an observer, starting an async scan task, and inserting a new picture in the model.
- **Code separation.** There are separate methods for testing and requesting permissions, separate method for setting up views, separate class for indexing images on phone, separate method for updating filters, separate Util class for common utility code.

2.1.4 Special features

- **Splash screen.** 9-patch image consistent with the app colors and icon.
- **Placeholder thumbnails.** Show gray tiles if thumbnails not created yet.
- **Auto hide.** Auto hide floating buttons while scrolling.
- **DiffUtil.** DiffUtil [1] is used to increase performance when lots of new images are added.
- **Sort by newest.** Images are sorted by database column holding last modified time.

2.1.5 Additional development

- **Detection of deleted photos.** Deleted photos are removed from the Room database during the async scan which is run at app startup.
- **Rebuild thumbnail if photo changes.** Photos are detected to be changed using last modified time and thumbnail is rebuilt.
- **Reset app easily.** Hidden menu item to reset app completely.
- **Additional photo source.** Extra floating button to select a file from phone/any other app.

2.1.6 Limitations

- **Flickering photos on first app run.** On first app run, the thumbnails can flicker while scrolling until all thumbnails are built. As a workaround, scrolling can be done slowly.

2.2 Showing pictures on a map

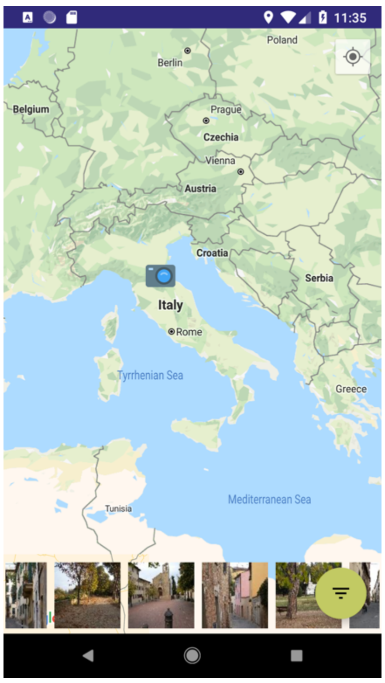
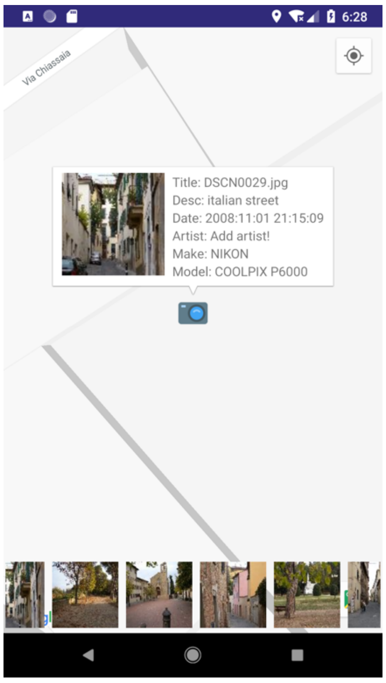
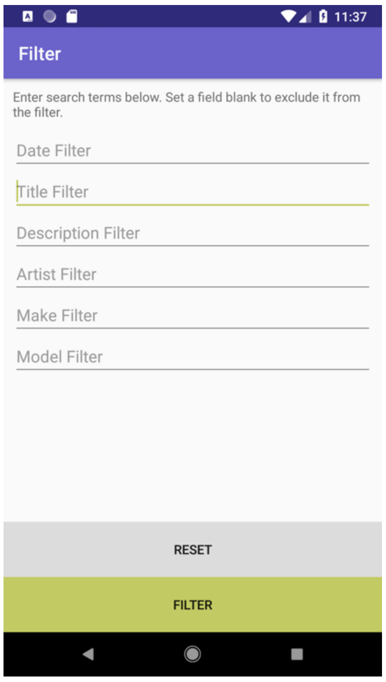


Figure 3. MapActivity



Title: DSCN0029.jpg
Desc: italian street
Date: 2008:11:01 21:15:09
Artist: Add artist!
Make: NIKON
Model: COOLPIX P6000

Figure 5. MapsActivity markers



Enter search terms below. Set a field blank to exclude it from the filter.

Date Filter

Title Filter

Description Filter

Artist Filter

Make Filter

Model Filter

Figure 4. Filtering

2.2.1 Requirements
- **Views.** Google Map fragment with markers indicating the position of pictures with GPS data, button for filtering images, horizontal RecyclerView of thumbnails that can be clicked on to go to a particular marker on the map.
- **Filtering.** Images can be filtered based on their date taken, title, description, artist, make, model.
- **Current location.** Current location can be shown by clicking on a button.
- **Testing.** Activity was tested using images available at [4]
- **Deviations from requirements.** None.

2.2.2 Design choices
- **Idea.** It is advantageous to use as much screen space as possible when there is a map present. App bar is hidden. Place thumbnail strip and floating button at bottom to avoid interfering with map.

2.2.3 Separation of concerns
- **Interaction with model.** Only interacts with model using instance of ViewModel at only once place – for setting up filtered live data observer.
- **Filtering.** Filtering is implemented by a starting an intent for a separate activity, FilterActivity.

2.2.4 Special features
- **Auto hide.** Auto hides floating button while scrolling.
- **Custom marker.** Camera icon used for markers indicating position of photo.

2.2.5 Additional development
- **Filtering map markers.** Implemented exactly the same way as BrowseActivity, reusing code.
- **Thumbnail strip.** Showing thumbnails of images with location data. Clicking zooms in to the location.

2.2.6 Limitations
- **Images not shown on map.** Images will not be shown on map unless they are explicitly indexed by scrolling down in the main RecyclerView in BrowserActivity, since image metadata is only read when it is bound to the RecyclerView.

2.3 Inspecting details of a photo

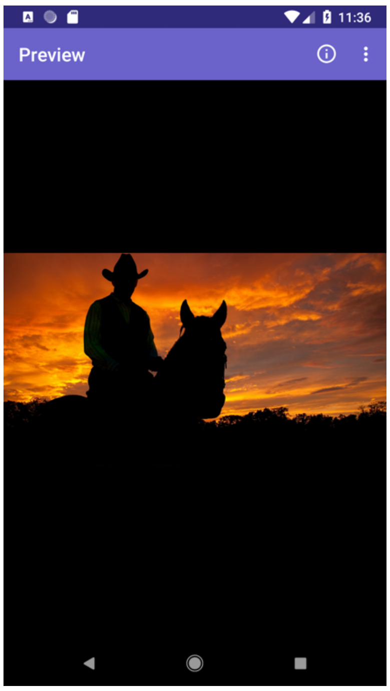


Figure 6. Preview

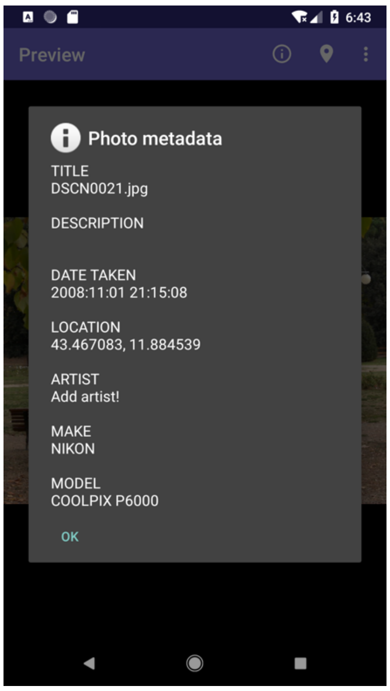


Photo metadata

TITLE
DSCN0021.jpg

DESCRIPTION

DATE TAKEN
2008:11:01 21:15:08

LOCATION
43.467083, 11.884539

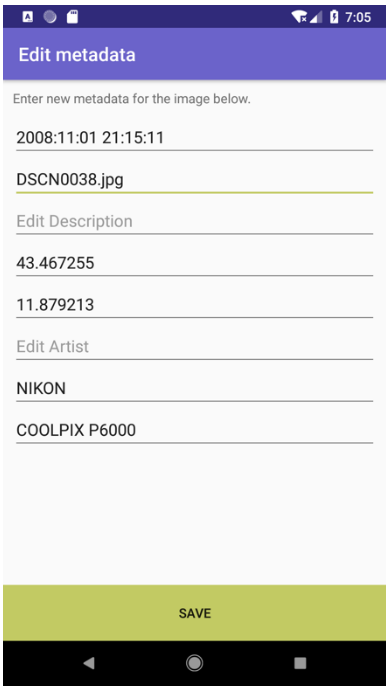
ARTIST
Add artist!

MAKE
NIKON

MODEL
COOLPIX P6000

OK

Figure 7. Show metadata



Enter new metadata for the image below.

2008:11:01 21:15:11

DSCN0038.jpg

Edit Description

43.467255

11.879213

Edit Artist

NIKON

COOLPIX P6000

SAVE

Figure 8. Edit metadata

2.3.1 Requirements

- **Views.** Single full size image fitted to width/height. App bar with button to view metadata, show map location (if available) and hidden button to edit metadata.
- **Show metadata.** Image metadata can be seen as a popup by clicking on the info app bar button.
- **Show map location.** Location on map can be seen by clicking an app bar button that appears when there is location data available. It opens another activity ShowPhotoLocationActivity which has a full screen map with a single marker.
- **Edit metadata.** Hidden menu item allows editing all image metadata using EditActivity.
- **Deviations from requirements.** None.

2.3.2 Design choices

- **Idea.** Minimal uncluttered view to focus on the image. No information is shown unless explicitly asked by user, making experience pleasant. Google Photos has a similar experience.
- **Black background.** Images look nicer and stand out.

2.3.3 Separation of concerns

- **Separate activity for editing metadata.** EditActivity is called for editing metadata.

2.3.4 Special features

- **Full screen.** Clicking image makes it full screen.

2.3.5 Additional development

- None.

2.4 Taking pictures

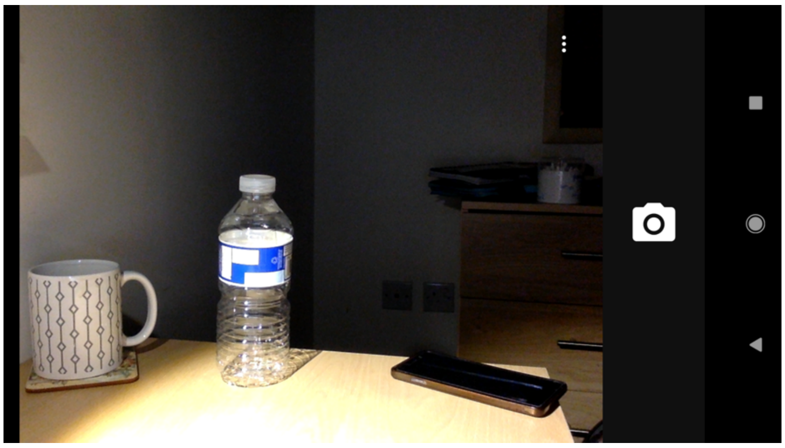


Figure 9 Take camera picture

2.4.1 Requirements

- **Camera detection.** Floating button is hidden if phone does not have a camera.
- **Location metadata.** Current location metadata is stored in the image file and database.
- **Dummy location on emulator.** If run on emulator, UOS DCS coordinates are provided.
- **Deviations from requirements.** None.

2.4.2 Design choices

- **Idea.** Floating button is used for taking pictures from camera. It is the recommended Android way of feeding in new information into an app.

2.4.3 Separation of concerns

- N/A

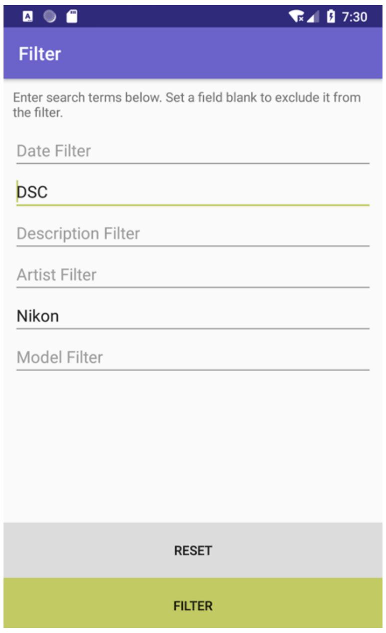
2.4.4 Special features

- **Edit metadata.** EditActivity is called just after a photo is clicked.

2.4.5 Additional development

- None

2.5 Saving metadata to local database



7:30

Filter

Enter search terms below. Set a field blank to exclude it from the filter.

Date Filter

DSC

Description Filter

Artist Filter

Nikon

Model Filter

RESET

FILTER

Figure 10. FilterActivity

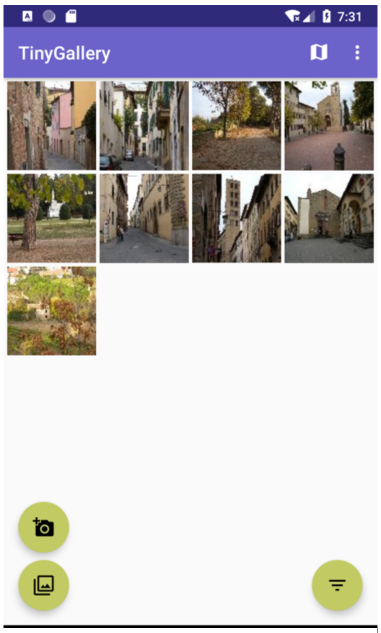
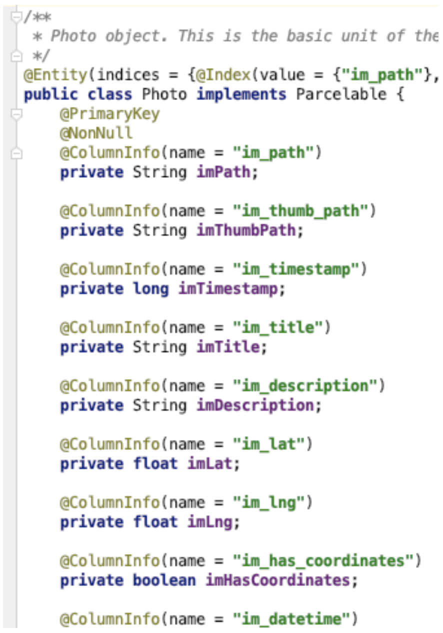


Figure 11. Filtered pictures



```
/**
 * Photo object. This is the basic unit of the
 */
@Entity(indices = {@Index(value = {"im_path"},
public class Photo implements Parcelable {
    @PrimaryKey
    @NonNull
    @ColumnInfo(name = "im_path")
    private String imPath;

    @ColumnInfo(name = "im_thumb_path")
    private String imThumbPath;

    @ColumnInfo(name = "im_timestamp")
    private long imTimestamp;

    @ColumnInfo(name = "im_title")
    private String imTitle;

    @ColumnInfo(name = "im_description")
    private String imDescription;

    @ColumnInfo(name = "im_lat")
    private float imLat;

    @ColumnInfo(name = "im_lng")
    private float imLng;

    @ColumnInfo(name = "im_has_coordinates")
    private boolean imHasCoordinates;

    @ColumnInfo(name = "im_datetime")
```

Figure 12. Photo table declaration

2.5.1 Requirements

- All image data is stored in a Room database table called “Photo”. It has columns for image location, thumbnail location, file last modified timestamp, EXIF fields etc.
- **Searching.** Searching of images is done by setting a dynamic live data observer using the filter action buttons. Searching is possible by date taken, title, description, artist, phone manufacturer and model.
- **Deviations from requirements.** None.

2.5.2 Design choices

- **Dynamic filtering.** Filtering is done by creating a live data observer on the fly. This allows for new search results to pop up without having run the filter again.
- **Reuse of components.** FilterActivity is shared with MapsActivity.
- **Layout.** Textboxes and buttons in FilterActivity are designed to be flexible to different screen sizes and resize automatically.

2.5.3 Separation of concerns

- N/A

2.5.4 Special features

None

2.5.5 Additional development

- None

3. Division of Work

Mangesh
1.1.1 (major), 1.1.2 (minor), 1.1.4 (major), 1.1.5 (major)

Akshay
1.1.2 (major), 1.1.5 (minor), documentation (major)

Wenbo
1.1.3 (major), testing (major), documentation (minor)

Mark split - Mangesh should get a relatively higher share of marks for working on core parts of the app and setting UI style.

4. Libraries used

EasyImage [5], Google Maps, Android Room

5. Bibliography

[1] "DiffUtil," [Online]. Available: https://developer.android.com/reference/android/support/v7/util/DiffUtil.

[2] "imgur album," [Online]. Available: https://imgur.com/gallery/Enxdo.

[3] "ColorHexa," [Online]. Available: https://www.colorhexa.com/6b63ca.

[4] "EXIF samples," [Online]. Available: https://github.com/ianare/exif-samples/tree/master/jpg/gps.

[5] "EasyImage," [Online]. Available: https://github.com/jkwiecien/EasyImage.