A4: Mobile Layout (Java/Android) -- Fotag

Synopsis

You will create an interactive mobile application that allows users to load images over the internet, and display these images in a dynamic layout that responds to device orientation. Users can rate images (1-5), and filter them based on this rating.

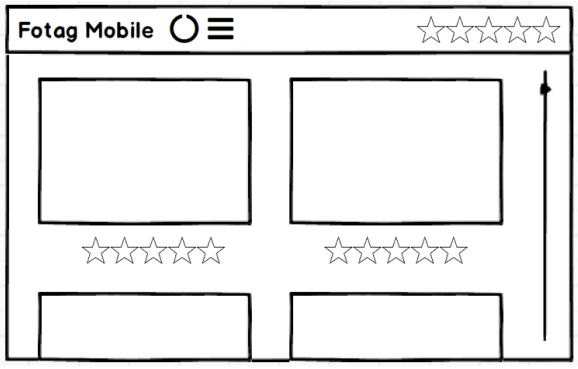
Learning Goals

This assignment gives you an opportunity to build an interactive mobile application, and gain experience managing dynamic layout on a mobile device.

Requirements

General

Design and implement a mobile application called *Fotag*, which allows users to load and display a grid (or list) of thumbnail images, rate each image, and filter the list of images by this rating. The application should look something like this:



Fotag Mockup

When first launched, the application should show an empty list (image grid). There should be a toolbar presenting the following functionality:

- A **load** button which will prompt the user to enter (paste) a URL, load a single image from that location, and add it to the list of images being displayed.
- A **load image set** button, which loads a large set of images over the network from this URL: https://www.student.cs.uwaterloo.ca/~cs349/w20/assignments/images (https://www.student.cs.uwaterloo.ca/~cs349/w20/assignments/images), and adds them to the existing list of images (note: we will use these hard-coded image paths and filenames, but will replace them with new images when testing. You *must* load them over the internet and not just hard-code them in your application.)
- A **clear** button, which removes all images from the list. Clearing the list will also clear the ratings for those images (i.e. you do not need to save the ratings).
- A **filter** widget, showing 5-stars, used to filter the image list based on rating. You should support displaying "all" images, or "1-5" where the filter shows all images that have that rating or higher (hint: you might find the Android RatingBar widget helpful, but make sure to support all of the required functionality).

If you wish, you can split functionality across multiple toolbars, or put one or more functions in a floating toolbar. Note that you do not have to use the image icons shown in the example below, but you should use images on your widgets, and not just text labels.

Your application needs to support JPG and PNG image formats. The sample images (above) include at least one image of each type, so a working application should be able to load all images from this URL. You do not need to load other image types, but if the user enters a URL pointing to an invalid image, you should display an error.

The grid should be a regular size, so that the images and ratings line up horizontally. This means that you need to scale your images to fit into the row. You should be careful to not scale unevenly and distort the images while resizing them.

Each image in the grid should be shown as a thumbnail, with a set of 5-stars beneath it. Users can touch a star to select a rating, and they should be able to clear the rating on an individual image.

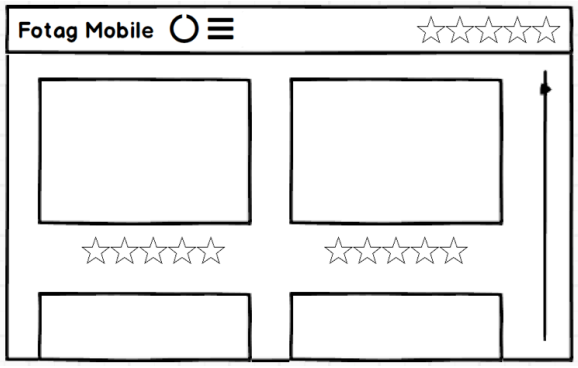
Touching the image thumbnail in the grid should launch a new activity that displays the image full-screen with the rating bar below it. Users should be able to rate the image from this activity, and there should be a means of closing it and returning to the main activity (i.e. grid).

Your application should not hang or freeze-up when loading images. Consider loading the image in the background to prevent this problem. See hints below.

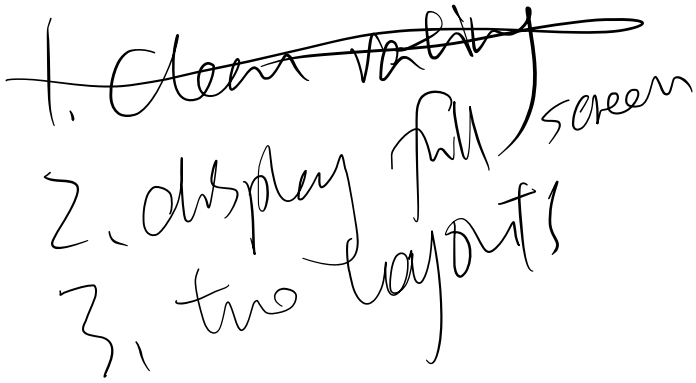
Supported Layouts

You need to support two layouts: one that is shown when the phone is held vertically, and one when it is held horizontally. The user does not manually change the layout, but instead your application should respond to changes in phone orientation so that the appropriate layout is always shown. No data should be lost when the phone is rotated (note: the emulator uses a hotkey to change orientation, typically Ctrl-F11).

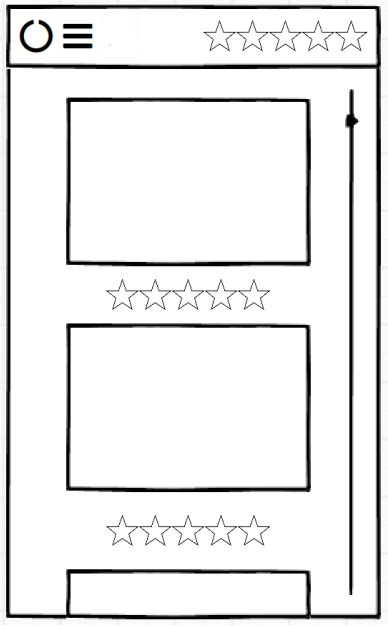
Horizontal orientation should show a fixed number of columns (2 or 3 columns, your choice), and sufficient rows to show all of the images. Adding images should add more rows as required (i.e. it may show a vertical scrollbar if required, but it should never show a horizontal scrollbar).



Landscape (Horizontal)



Vertical orientation should show a single column, and sufficient rows to accomodate the number of images (i.e. a vertical scrollbar may appear if needed to show all of the images, but it should never show a horizontal scrollbar).



Portrait (Vertical)

Technical Requirements

- You must test using a Pixel (Google Phone) AVD using API 29.
- You need Android SDK API 29 to build your application.
- You must use IntelliJ with the Android plugin. The course examples use IntelliJ 2019.3, which is the latest version at the time this assignment was written. Instructions to install and configure IntelliJ are included in the Android development slides.
- You can use any functionality included in the Android SDK, and are free to use code snippets from the Android sample code

included with the SDK (i.e. a snippet being a small segment of code). You can also use any code that we introduce in the course (i.e. from the CS 349 Git repo). If you do this, put comments in your code referencing the original source. You are not allowed to use any other third-party code or libraries, unless you obtain permission from the instructor (permission on Piazza is fine).

- You are not required to use Model-View-Controller, but you *must* save your data in such a way that no data is lost during orientation changes, or when switching between activities.

Getting Started

Here's some suggestions if you're new to Android and mobile development.

1. Create your project (recommended to use the "Empty" project). Setup your AVD, and configure the project to launch and run in the AVD. Projects are **very** difficult to get setup and build properly, so you want to make sure it's working before you start modifying anything.
2. Commit and push changes to Git frequently! Anytime you make a substantial change (and your code compiles/works), then commit and push to Git.
3. Use the XML-layouts to create two layouts, corresponding to the two orientations. If you set this up properly, Android will handle the orientation change for you. (Hint: make sure to save and restore data during the orientation change.)
4. Create a Main activity for your grid, and a second activity for the photo details screen. Use intents or a shared model to pass information between these activities.
5. Look at the Volley library (https://developer.android.com/training/volley) for fetching data over the network (it's part of the Android SDK and you can use it). Make sure to request internet access permissions in your manifest file!
6. Consider using an AsyncTask (https://www.tutorialspoint.com/android-asynctask-example-and-explanation) to fetch data in the background.

Submission

Your directory structure for your assignment submission should be in subdirectories under your a4/ directory in your Git repository. It should use a standard file layout for an Android IntelliJ project.

Your submission needs to include:

- All source code and resources required to build and execute your program.
- An IntelliJ project that compiles everything using the specified JDK and Android SDK, and which has a Run target that will execute your program.
- A `readme.md' file with any details required to help the TA grade your assignment.
- An APK file at the top-level of your directory structure. You will likely need to manually create this (Build -> Build APK) and copy to this location. This will help the TA to load and test your application without needing to load the project.

Your `readme.md' file needs to include, at a minimum:

- Your name,
- Your student number,
- Your operating system & version,
- The version of Java and the Android SDK that you used,
- The source for your image and sound assets, if appropriate.

Assessment

Late assignments will not be accepted. Markers will test your application by running your APK file against a Pixel AVD. Your submission will be assessed roughly as follows (detailed marking scheme to follow):

5%
Code compiles and runs, readme, working project.
10%
Application loads, and screen includes a toolbar with required functionality.
25%
Images shown in a layout, with ratings. Changing phone orientation loads the appropriate layout.
25%
User can load images from URL above. Images appear in the grid, reasonably sized for the device.
15%
Selecting an image shows a full-screen image view, with rating support.

20%
Ability to rank individual images, and filter based on ranking.

Versions

1.0. Mar 3, 2020. Initial release.

CS 349 User Interfaces (Winter 2020)

Cheriton School of Computer Science

University of Waterloo