CSC 22100
Android Studio Photo Editor Experience

By Josue Flores

Overall Description

- I planned to make a simple Photo Editor with primarily three features and a few other side features.
- Image Cropping
- Filter Application
- Orientation (Rotating)
- Brightness and Saturation
- New/Save

Challenges Faced

- Finding Online Resources that were completely relevant to my project
- Learning about the relevant methods associated with UI Classes
- Overriding certain methods to fit my usage
- BUILDTIME/RUNTIME ERRORS

Learned Concepts

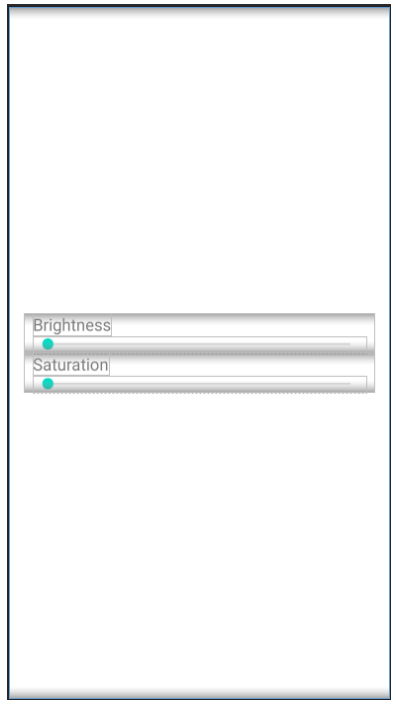
The Significance of Fragments

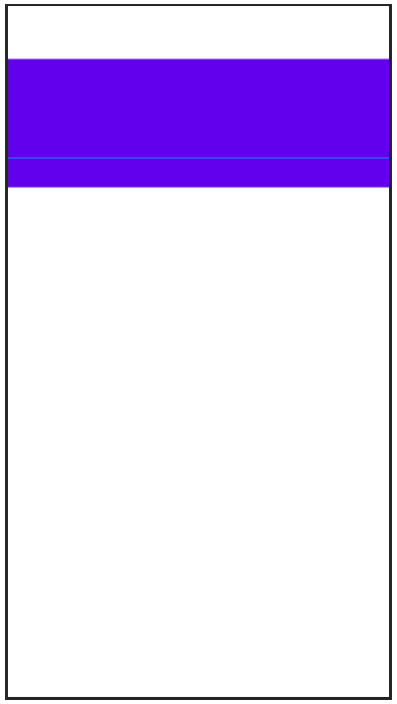
The syntax format for XML files

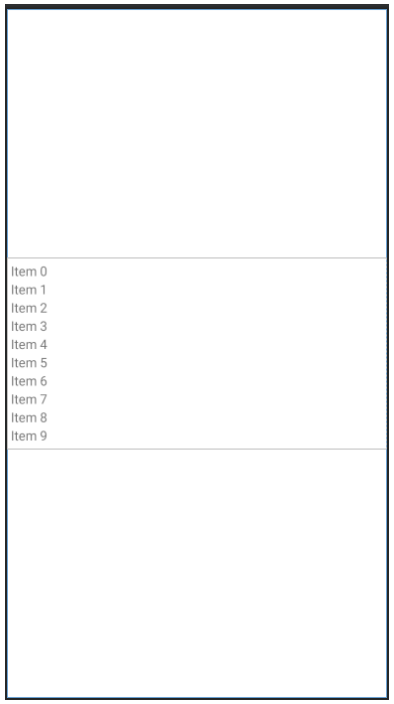
Filter Image Extraction

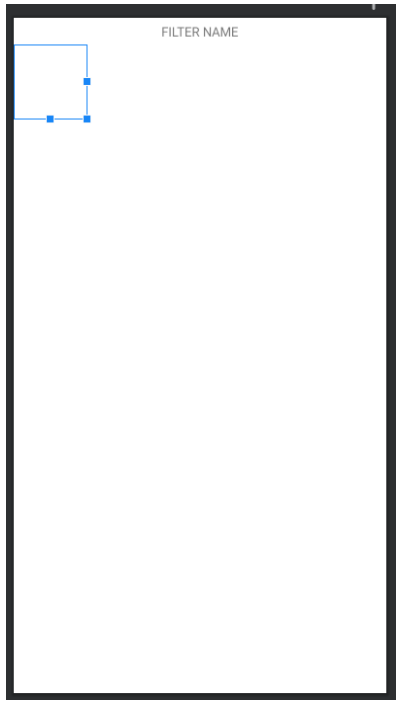
Transferring data between activities (Intents)

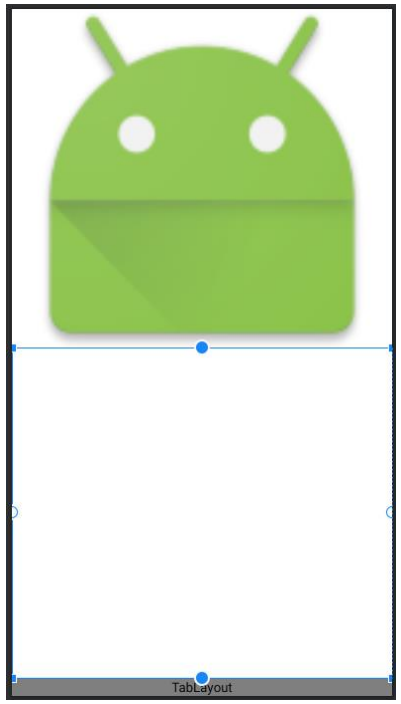
Initial Screenshots of Project

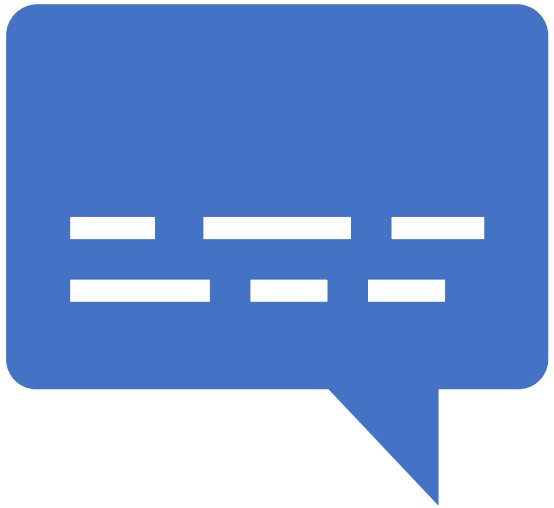










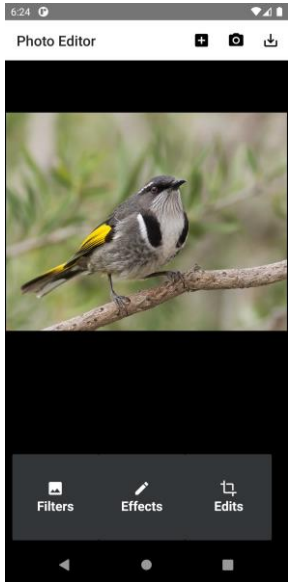


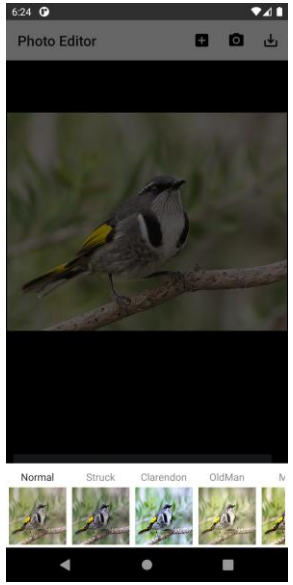
Final Thoughts

I would have liked to add a few more features such as,

- A Text box for comments
- Emojis
- More Image effects

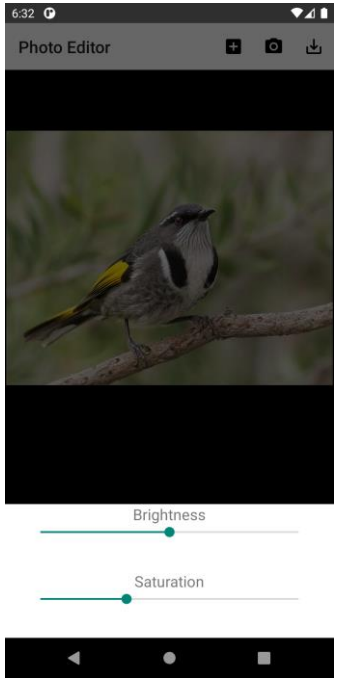
Final Features Implemented

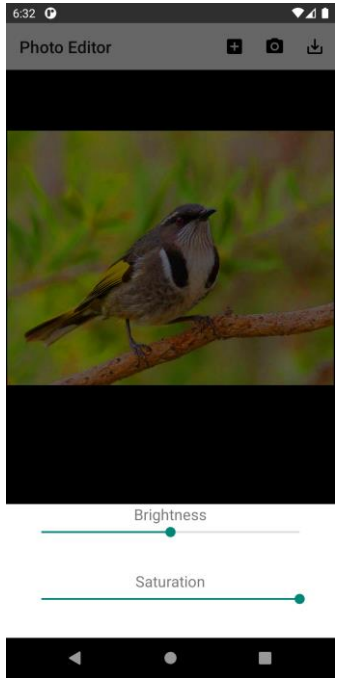


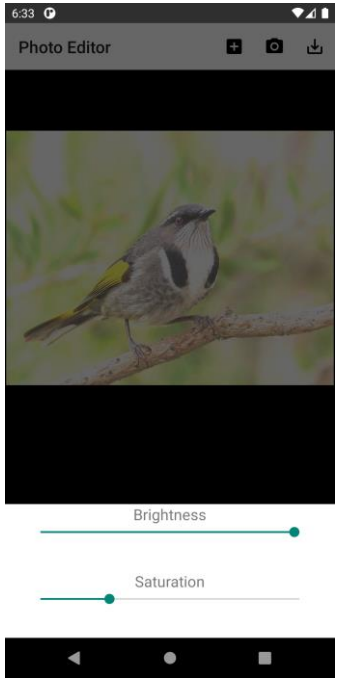


Filters

- As the name suggests, the user can apply a filter on the image currently being edited by selected on the Filter CardView and selecting a specific filter from the list provided within the horizontal layout.
- Initially, I planned to incorporate three filters, but I was able to find a library with multiple pre-built filters and incorporate it within my code.







Final Features Implemented...

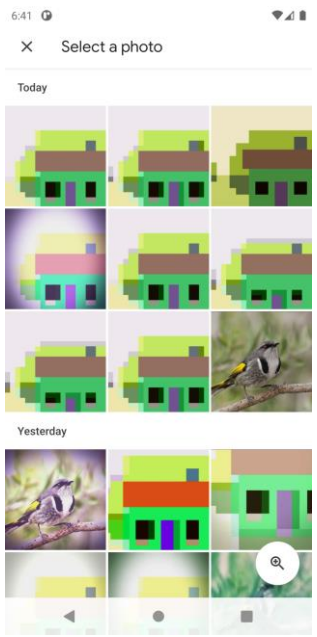
Effects Button

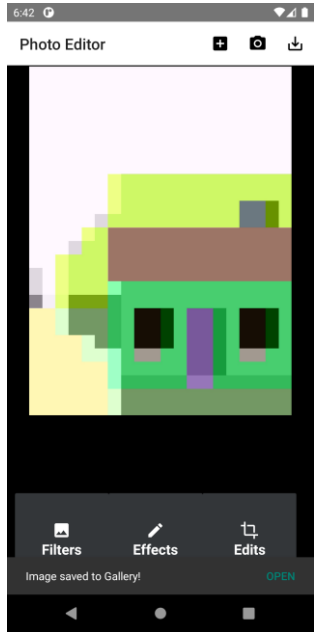
- This features allows the user to adjust the brightness and saturation levels associated with the image by adjusting the slider's position for the corresponding attribute.
- This is done by pressing the Effects cardview and a prompt will appear showing the sliders for both the brightness and saturation.

Final Features Implemented…

Edits Button

- This feature allows the user to select from four different ratios of rectangular crops and apply it to the image. Not to mention, if the user selects on one of the ratio, the ratio of the crop inverts (e.g. 3:4 becomes 4:3)
- In addition, the user can rotate the orientation of the image with a 90-degree button that rotates it 90 degrees to the right or it can be adjusted to obtain a different rotation angle. There is also a reset button that returns the angle to 0.
- Lastly, the user can zoom in and out of the image with the scaling feature. However, the scale automatically limits itself once the boundaries of the zoom matches the max image boundaries.





Final Features Implemented…

Minor Features

- As a prerequisite for the completed project, I felt the need to include a new, camera, and save feature, situated at the top of the application.
  - The new option allows the user to select a new photo from their gallery and load the image into the image view.
    - The camera feature grants the user the ability to take a photo and load it into the image view.
- Lastly, the save feature allows the user to save all the changes they have made to that image back into the gallery as a new .jpg file.

Data Save / Retrieval Methods

For both the data save/retrieval process, I employed the Dexter Library to facilitate the permission process.

Afterwards, I used the saveAsBitmap method, taken from a different photo editor library, to save the data into the gallery.

Likewise, I used intents to pick an image from the gallery and transfer it to the photo editor view.

Data Structures Used

For this project, I mainly utilized the Array List/List data structure to complete my photo editor

Specifically, I used the List data structure to extract the individual filter data from the Zomato library into the FiltersListFragment. This enables the android user to select the filter they want to apply from that library.

Missing Features

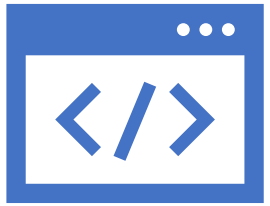
- In my case, some missing features for my project would be the circular crop and the flip feature associated with the orientation of the photo.
- However, after some thought, I realized those features were not necessary, considering that the project contains four(or seven) dimension-based cropping rectangles built into the editor. Not to mention, that the flip feature can be treated as a non-fundamental component when it comes to photo editing.

Challenges Faced

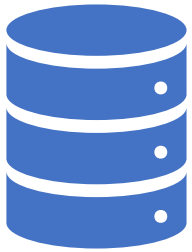
Some Challenges I have faced include:

- Resetting the effects slider to its default animation when a filter was selected
- Cropping an image with the filter and image effects remaining in effect
- An error that saves a blank photo to the gallery when the camera feature is selected, and a photo is not taken.

Solutions Found



To resolve the filter and effect slider issue, I would need to changing the API of the project from 21 to API 24, which supports the animation reset. After updating to API 24, I simply made a new instance of the Edit Image Fragment whenever a filter was selected.



I chose to abandon the cropping issue since the cropping feature extracted its data from a URI instead of the Photo Editor View. So, if I truly wanted to perform a crop with the filter and effects intact, I would have to save the image into a new file and apply the crop to the URI taken from that file.

I chose to abandon the camera null photo error because I could not discover the source as to why this null image was being saved into the gallery. My guess is that the intent data from the camera was being read as null every time the user didn't take a photo after entering the camera application.

Demo Recording