CSE 5236 Professor Adam Champion
Group 8
Team Member: Jinhee Lee, Jiehui Li

Final Report

Drawing app development

1. Introduction

These days, unlike the monotonous work of the past, there are many people who are looking for self-directed and creative works. With the appearance of independent media, platforms, and the development of IT technology, this trend is accelerating. Among those changes came the spread of the Internet and mobile phone, taking sharing information to a whole new different level. Drawing is one of this self-directed and creative work, and many people are interested in drawing and want to learn. However, it is not easy for people to get professional drawing education. Therefore, we came up with a way to learn using a mobile application which is simple and accessible.

There were already many apps for drawing. These apps provide a variety of features for professional people like illustrators. However, we are focused on people who have very little experience in drawings and we searched drawing apps for beginners. However, it was hard to find an app that helps people to learn drawing, and we planned a drawing app by tracing other pictures. Since all learning comes from imitation, tracing other pictures will be great help in learning to draw. The concept is not an innovation by us, as from ancient to now, people learn to draw by putting a thin paper on others' works and draw by following the works they try to recreate. Our App just makes this technique more accessible to ordinary people who want to learn drawings.

The unique point of our app is that people can set a drawing background from other's work with low opacity. Background image overlaps the user's drawing, and app users can see other's work and trace it. However, drawing should not be disturbed by background images. Also, people can share their drawings from the inner app gallery. So, people can easily approach to other's drawing work and can find their preferred picture from this gallery. With these unique points, seasoned drawers can act as the "teacher" and share their works drawn in this App to our own and unique online gallery. Newcomers can come and browse the works in the online library. They can choose the one they want to imitate, set it as the background in canvas and follow the blueprint.

2. Design process and intended goals

In the first meeting, we decided to make a drawing app for learning and discussed our app's features. Also, we talked about the developing tools and decided to use Android Studio. After then, we brainstormed about functions that are needed for a drawing app. We just enumerated words which are related to our app. At this time, we talked about a multi-layered paint app like photoshop and procreate. We thought that the ‘add layer’ function would be useful for revising paints. Also, we discussed attaching a video file as a background layer. Below list is what we brainstormed in this step.

Words: pen, eraser, color, canvas, download, upload, sharing, fingePpath, recording, recordVideo, newLayer, switchBackground, draw, switchPen, Gallery, choosePicture, name, ID, password, Login and Logout.

Using these words, we wrote each word’s roles for the app and consolidated words by categories. Because if we classify each word as features, it will be easier to make components and flow of our app. The below chart is originally what we categorized the words and their roles.

Category	**Words and roles**
Drawing activity	· Pen: drawing · Erase: eraser setting · Color: change pen color · Layer: layer setting · Canvas: canvas setting · SwitchPen: choose brush · FingerPath: pen’s path · Background: background setting
Canvas setting	· Canvas: canvas setting · NewLayer: add new layer, · Upload: upload picture · ChoosePicture: choose a picture · SwitchBackground: change background
Recording	· RecordVideo: record my drawing procedure
Share	· Upload: upload picture · Download: download picture · ChoosePicture: choose a picture · Name: naming picture
Gallery	· ChoosePicture: choose a picture · Name: naming picture
Login Activity	· Login: user log in · Logout: user log out · ID: user ID · Name: user name · Password: password

Figure 1. Original words and categories

For the login and logout process, we thought about the user's ID, password, and nickname system. Also, for storing this information, we considered the Firebase Firestore Database, one of the most advanced realtime databases on Android provided by Google, to keep the data consistent among all the devices with the app and connected to the Internet. We also plan to store all the drawings (in jpg format) and recordings (in mp4 format) in the Firebase Storage so that we can set up a gallery like features that all our users can see. All users have the right to download or upload drawings and recordings.

The final result of the app is a little different from what we set out in these original categories and words. However, most words and categories are still using and derived from the original idea.

3. Translating Design into Implementation

After we finished the design process, we started studying about Android Studio and making our app[1],[3]. Firstly, we made a home activity that contains three activities which are draw_activity, gallery_activity and share_activity. We made back buttons which makes users can go back to the previous activity. At this time, we did not consider the login process, and other detailed functions.

After implementing a basic part of the main and three activities, we focused on building the drawing activity. We made a new class MyPaint which allows people to draw a picture. Draw_activity calls this class, and the user's drawing can display their actions in the drawing activity. There are three main roles of MyPaint class which are canvas, draw and touchEvent. For the canvas and draw, we initially considered the vector drawable, but the bitmap method was adopted. Because bitmap images are made up of pixels which is more familiar to most painting app users, and there are more resources to implement a drawing app. After that, we implemented touchEvent with drawpath which is based on the x and y axis data. It was working well and finally, we can draw a picture using our app.

As a next step, we tried to make a multi-layered based drawing activity. However, it did not work well. During the storing the image process, there were some errors, For example, drawings were overlapped with background layers, and it was stored just as black image. So, we decided to make a single layered drawing app with a separated background layer.
For the background layer, we considered two ways to import the image. One is getting from the gallery and the other is getting it from camera capturing. Importing from the gallery was simple. To get an image from the gallery, an app gets data as a Uri and sets it into the imageView. However, the camera capturing was not easy to implement. Because, for using a camera, an app needs to get permission from the user. It is an essential requirement for legal work. So, we made a TebPermission part at the beginning of the draw activity, and then, we made a camera capturing part. We used bitmap to get an image data and set it into the imageView as well. We gave opacity for the background image using a setAlpha(). And then, we tried to make a Video background using videoView. For the background video, it should not disturb the user’s drawing, and should be easy to treat the video while drawing. However, when we implemented it, the video stopped and was choppy while the user's drawings. Also, we can’t make a recording method while drawing. So, we decided to remove the video background part.

Next, we want a place where users can display and share the drawings they created, so we created the gallery activity. The gallery still works like the one we proposed during the design process, but the layout as well as the core behind it is completely different during implementation. Our first iteration of the gallery is pretty simple. It is a local gallery without the ability to share with other users. The main focus of this iteration is to display images in RecyclerView. We wrote a singleton class that returns a list of addresses of the images stored in internal storage. We use a binder and an adapter for the recycler view to help us display the images and their names. The second iteration makes the gallery an online shared gallery among all users of our app. We use Firebase Storage to help store those images online.

Firebase Storage will not only store the image but also the metadata that include the painter and the name of the drawing. We updated the recycler view according to deal with the changes. Finally, we add two buttons to the recycler view, giving users freedom to delete their images from the online gallery as well as to export anyone's works on the gallery to their devices.

We also implemented a painter account system to let users create personal profiles for themselves and upload drawings using their profiles. During our design process, we did design a database schema that belonged to painter, but we didn't include an account system, so this is something completely new that we came up with during implementation. First, we made the launch activity of the app to be the login activity. Users enter their username and password here to login and use the app. We also implement an activity where users can create their profiles, by entering a username(that can't be repetitive to the ones that existed in the database), their favorite color and a password. We use Firebase Firestore Database to store those information, so that the data can be consistent on all devices that use our app. The profile is useful when users try to upload an image to the shared gallery, their username will be attached to the image that is stored online. So in the gallery, you can see the drawings are drawn by which users.

For the final process, we considered the layouts of each activity. We drew a few sample UI interfaces while we were in the design process, but we needed to revise those designs knowing how Android views work. Here are the changes. It is important that the views are being displayed correctly no matter the phone is in horizontal or landscape view, or what the size of the screen is. We wrote landscape layout resource files for profile activity, main activity and gallery activity because they are the ones that the organization of layouts will be different in landscape mode. We also made some slight modifications for existing horizontal layout files to account for the different screen sizes our users may have. We use features like layout_weight and <Space> to help achieve the compatible UI interface we want.

We refined our app by working on the non functional requirements of the app, which we didn't envision much in our design process, but is a very important part in our implementation. First, we implement support for other languages. We introduce kr_strings.xml and cn_strings.xml to our project and we do the translation to Korean and Chinese ourselves. When the users of our app set their system language to those two languages, the app will be displayed in their system language. Next, we consider the rotation of the app. First, we apply a no-rotation constraint to our draw activity as we don't want the user to be in landscape mode while drawing. Thanks to the landscape layout files we implemented earlier, the testing of rotation of other activities went very well. Then we try to deal with the problem that there is no network connection, as a lot of features in our app are internet-based. When there is no internet connection on users' devices, we stop them from creating and updating profiles, giving them an option to login as guests and suggesting them to export their drawings to their devices as they can't upload it to the online gallery. Last but not least, we worked on performance enhancement. We set the anti alias to the false setting. Anti alias is the way to smoothing images, but our app does not use high resolution images, and we turned off the anti alias. It increased cpu performance by about 2%.

4. Suggested Changes and Improvements

One of the most important takeaways from this project is that you need to be very familiar with the tools that you are going to use and what they are capable of doing before you start your design process. In our case specifically, we need to be familiar with the Android framework itself as well as external tools like Firebase. In reality, when we are working on Checkpoint 2, which is the design process, we have little knowledge about those two and we include a lot of designs on the basis that we don't know how things work. For example, we aren't that familiar with Fragment and Activity in Android yet. We are not very familiar with the way that views are constructed that we design some UI interface that is hard to achieve by simple Android views. We only knew about the name of Firebase but didn't have a clear idea of what it can do and what it can't. As you can imagine, the design coming out of this is really hard to translate into the implementation. We succeed in implementing some, as described in the section above, while also failing in some, as described below. If we can do this project all over again now, we are already very familiar with the Android framework as well as Firebase, so our design process under this circumstance will be much easier to translate into implementation than the one we did this semester. The same holds true for programming projects in other fields, as we need to be very familiar with the tools we are going to work when you are in the design process.

In addition, we need to cover more non-functional requirements during our design process. Looking back at our Checkpoint 2, we barely cover any useful non-functional requirements in our document. All of those NFR we implemented in the project are added during our implementation process. While the design plan can always change when the development is in process, we still think that it's a better idea to cover those NFRs in the design process. For example, if we pointed out that localization is a priority NFR in our design process, we might have an app setting that can switch language, instead of adapting to the system language. Mentioning NFRs during our design process will certainly help improve the quantity and quality of usability and scalability of the final product, and thus is a change we are willing to make if we are starting the project all over again.

5. Reference

[1] Android Open Source Project, Android, http://developers.android.com

[2] Firebase, https://firebase.google.com/

[3] Bill Phillips, Chris Stewart, Brian Hardy, and Kristin Marsicano, Android Programming: The Big Nerd Ranch Guide, Big Nerd Ranch LLC, 3rd edition, 2017. (Notation: [BNR])