# Project 2 Report

File Processing App

CSC 4320/6320- Operating Systems
Spring 2021

Name: Hiren Patel
Email: hpatel126@student.gsu.edu

## Background

Throughout this whole course, we have learned about many different concepts regarding Operating Systems (OS) through various lectures, assignments, and exams. As a result, one of the assignments that students had to complete was developing an Android application that covers certain OS-related functionalities. The app was intended to be developed on Android Studio, which is essentially an integrated development environment (IDE) for Google's Android OS. Android Studio provides an environment which allows people to create and build apps for Android phones, tablets, and a variety of other devices that implement an Android OS. Android Studio allows developers to easily write code as well as easily set up an appealing user interface (UI) for their applications. Prior to creating this Android application, I had never worked with any sort of app development, so I was not too familiar with the idea of creating an Android app. However, after doing more research about Android app creation and Android Studio, I felt that I was in a comfortable enough spot to try and code an application of my own. As a result, with the information that I had just gained due to research, I began the task of creating my first Android application using Android Studio.

## Introduction

The main component of this assignment was the fact that the Android app had to cover some sort of OS-related functionality. Examples of this include things like system calls, which are essentially the methods in which a program requests a service from the kernel of the OS. In my case, the system calls being implemented into my application refer to the kernel of the Android OS. Since this main component of an OS-related function had to be implemented into the app, it was quite tricky figuring out what kind of app I even wanted to create in the first place. As a result, I was not able to create an app that was meant primarily for entertainment purposes such as some sort of game due to the fact that I had to implement some sort of OS functionality into the app.

The project guidelines and instructions showed a variety of different categories and possible options for students to pick from. These categories listed possible topics and themes that students would implement into their apps in order to have some sort of OS-related functionality. There were four different categories and their topics were: Device care, Network, File Manager, and File Processing. The categories regarding file manager and file processing appealed to me the most because I thought it would be interesting to mess around with different files as well as various elements regarding the files. For instance, I thought that it would be interesting to implement some sort of program that allows users to add line numbers into a text file. As a result, I ended up choosing the file processing category and decided to make an application that allows people to modify files and certain elements regarding the files.

With my topic in mind, I decided to create some sort of file processing application, but I was not sure if my idea was too simple or not. I decided to do some more research regarding file processing using Android Studio, and I came to the conclusion that I needed to add more key features to my application. Thus, I decided to implement several features into my app which include: adding line numbers to a file, converting between lowercase and uppercase characters in a file, searching for keywords in a list, and taking images with a camera and storing them as image files. I figured that only messing around with elements in a text file would be too simple and boring, so I decided to implement more features, and the features stated above was my final decision after I completed my research.

## Overview of my App

To start off, I chose to call my File Processing app since the category itself was called file processing. My app contains most of the features regarding file processing, so I felt it made sense to stay true to the category and call it a file processing application. My app allows users to play around with many file processing elements due to many OS-related low-level functions.These low-level functions will be discussed in a bit, but the app essentially is divided into three different activities. Throughout these activities, users will be able to: add line numbers to a text file, convert between lowercase/uppercase characters in a text file, search for keywords in a list, and take pictures with a camera and store them as image files.

Regarding adding line numbers to a text file, the app will allow users to write whatever they want in a text box. Once users have filled in whatever they want in the text box, they have the option to add line numbers to the text file by clicking a button that says "See Line Numbers". The same thing applies to the character conversion because users can convert all characters into lowercase by clicking the button that says "Lowercase", and they can also convert all characters into uppercase by clicking the button that says "Uppercase". Regarding searching for keywords in a list, users can enter whatever they want into a search filter, which will filter the list according to whatever the user types. It is important to note that the list is already predetermined, so typing out an element that does not exist will not filter anything as the element will not be there. Regarding taking pictures with a camera and storing them as image files, the app first asks the user for permission to access the camera. If the user agrees to grant access, the user will be able to use the camera and take a picture with it. The picture will be displayed to the user, and the user will have the option to save the image or discard it. If the user chooses to save it, it will be saved in the photos app on the phone, however, if the user chooses to discard it, the image will just disappear and the user will be able to take another picture.

Most of the data and information regarding my app is found as a result of many system calls. For instance, the whole aspect of taking pictures and saving images is implemented using system calls because users are able to access the camera and save images as a result of different system calls. The only downside of the whole camera concept is that an emulated version of an Android phone does not fully support camera usage. What I mean by this is that users on an emulator will not be able to actually see what a normal camera would see on a normal Android phone, and they will only see a base emulated image that Android Studio provides for users. Nevertheless, users will still be allowed to capture images and save them, but they will only really be able to store emulated images if using an emulated version of an Android phone.

To be fair, my app is not that complex and is quite simple if you think about it. Since this whole assignment was considered a practice project, the whole point of even completing this assignment was for students to get familiar with OS-related functionalities in the Android System utilizing Android Studio. As a result, I would say that my app is quite straightforward. However, there was still a great amount of research and work required in order to even make my app work properly as intended. The overall function of my application can be decided by how users choose to utilize it. For instance, some users may want to play around with elements regarding text files. On the other hand, some users may want to take pictures and save them on the phone using the camera function, so it really just depends on what users' intentions are regarding app usage.

## OS-Related Low-Level Functionalities

As previously stated, most of the data and information is found by the application due to various different system calls that are primarily implemented capturing and saving of images. The app is divided into three different activities, which allow for different features to be implemented. The first activity contains the text file processing aspect of the app, which lets users add line numbers and convert between lowercase and uppercase. The second activity implements a search filter for a list, which allows for searching of keywords in the list. The third activity is the activity which primarily invokes system calls, which implement the camera feature allowing users to capture images and store them as image files.

Regarding the first activity, various different functions are implemented in order to make it work. For example, the getText() function returns the text that is contained in the EditText component and the toString() function converts the text obtained from the getText function into String. In addition to toString(), the length() function returns the string length of the string found by the toString() function. In order to add line numbers into the text file, the getLineCount() function is called in order to figure out how many lines of text are being utilized in the text file. This line count is then displayed to the user by looping through the line count and printing the different line numbers on the text file. In order to be able to convert between lowercase and uppercase characters, the toLowerCase() and toUpperCase() functions are called when their respective buttons are clicked.

Regarding the second activity, there are not many low-level functions being implemented in order to create a search filter that will search for keywords in a list. A ListView is created in order to display the various elements in the list, and an array list is created in order to add elements on the list. An array adapter is utilized in order to display the various elements of the array list on the ListView component. A search filter is created, which allows users to enter whatever they want into the search field, and the getFilter() function is called in order to hold data with a filtering pattern. Finally, the filter(CharSequence s) function is called in order to actually filter the list according to whatever the user chooses to type.

The third and final activity is the one in which most of the significant system calls and low-level functions are implemented in my app. To start off, the Build.VERSION.SDK_INT system call returns the Software Development Kit(SDK) version of the software currently running on the device. The Build.VERSION_CODES.M checks if the device is running on a certain SDK version. These two system calls are implemented to essentially check that the device is running and executing on a certain version and is not on an outdated or old SDK version. The Manifest.permission.CAMERA system call allows for the application to access the camera, while the PackageManager.PERMISSION_DENIED system calls checks to see if permission has been granted or not to the given package. These two system calls are implemented in order to check whether or not the user has been granted access to utilize the camera function. The Manifest.permission.WRITE_EXTERNAL_STORAGE system call allows the application to write to external storage, which is also implemented with the PackageManager.PERMISSION_DENIED system call to check if the application has been granted access to write to external storage. The MediaStore.Images.Media.TITLE and MediaStore.Images.Media.DESCRIPTION system calls creates a read-only value that describes the media item. The MediaStore.Images.Media.EXTERNAL_CONTENT_URI system call is used to allow the Media provider to access all image type files on the external storage of the device. The MediaStore.ACTION_IMAGE_CAPTURE system call allows the camera to capture an image and display to the user while the MediaStore.EXTRA_OUTPUT system call is used to store images on the device. The Toast.LENGTH_SHORT system call is used to create a fast and small message for the user, which in my case, a “Permission Denied” message is shown to the user for a short amount of time if he/she chooses to deny access at any point.

As stated earlier, there are not that many system calls being implemented in the first two activities of my app, but a great deal of system calls are used in order to make the third activity to work efficiently. System calls definitely proved to be advantageous for me during the creation of my app because I was able to easily access the camera as well as store images on the phone. I most likely could have also implemented system calls into the first two activities of the application, but I never really found time to implement them unfortunately. I guess it is pretty counterintuitive to not add them in each activity since the whole point of this assignment was to get familiar with OS-related low-level functionalities, but I felt that I added a sufficient amount of system calls in the third activity to make up for the lack of system calls in the previous activities.

## Design of my App and How it Works

The design of my app plus was not too complicated, and the same applies to how it works too. Since the first activity primarily deals with altering text file elements, I figured that it would be ideal for users to be able to type whatever they chose to in some sort of text box or text field. As a result, I created an EditText element for users to input whatever they wanted as well as a TextView element to display line numbers. Regarding the first activity, it is quite simple because users can enter what they want in the text field, and then they are able to alter whatever is contained in the text field depending on the button they click after finishing with user input. Clicking the “See Line Numbers” button results in the getLineCount() function being called, which will ultimately display the amount of line numbers in the text file to the user. It is important to note that some sort of random character needed to be entered after pressing this button in order for the line numbers to appear. For instance, entering a space with the spacebar or entering another character will make the line numbers appear after pressing the button. This was an unfortunate bug that I encountered, which I was sadly unable to resolve. Regardless, the main feature still works, and line numbers will still be displayed to the user. Clicking the “Lowercase” button results in the toLowerCase() function being called, which changes all the characters in the text file to lowercase, and clicking the “Uppercase” button results in the toUpperCase() function being called, which changes all the characters in the text file to uppercase. The user then has the option to go to the next activity by clicking the “Next Activity” button, which calls the Intent() function. The Intent() function allows for users to switch between different activities, and in this case, users are able to switch from the first activity to the second activity.

For the second activity, a predetermined list is created beforehand, which utilizes a ListView component as well as an EditText component for user input. An array list is created in order to add elements to the list, and an array adapter is used in order to display whatever is in the array list in the ListView component, which is then displayed to the user. Users will then be able to type whatever they want in the EditText element, which calls the addTextChangedListener() function. This function essentially filters the list according to what the user types by calling the filter() function. Depending on what the user inputs, the list will be filtered accordingly, and users will be able to see the filtered outcome as a result. In addition to the search filter, users have the option to switch between activities by pressing the “Previous Activity” and “Next Activity” buttons. Pressing either of these buttons results in the Intent()

function being called, which results in users being able to switch between the first and second activities by pressing the “Previous Activity” button or switch between the second and third activities by pressing the “Next Activity” button.

For the third activity, users will have the ability to capture images using the camera on the emulated/real device. Users will begin by clicking the “Capture Image” button, which calls the setOnClickListener() function for the “Capture Image” button. This function checks whether the device has a compatible SDK version, and it also asks the user to grant permission in order to access the camera component as well as external storage of the device. If users agree to grant permission to both aspects, they will be able to utilize the camera and capture images as well as save and store the captured images as image files.. However, if they disagree to grant permission to either aspect, they will not be able to access the camera component or save and store images, and they will have to agree to grant permission in order for everything to work. Once permissions have been granted, the displayCam() function is called, which displays the camera on the device to the user. If users are using an emulated device, Android Studio will show some sort of emulated picture on the camera since an emulated device is not fully compatible with the camera component. On the other hand, if users are using a real Android device, the camera will be fully functional, and users will be able to capture images in real time just like a standard camera on a regular phone/smartphone. Once the image is shown on the camera, users will then be able to capture images. When an image is captured, users will have the option to save or discard the image by clicking the small check mark button or “X” button. Clicking the check mark button stores and saves the image on the device due to various system calls, and clicking the “X” button simply discards the captured image and restarts the camera allowing for users to capture more images. Once an image has been saved and stored, users can actually find the image on the device by accessing the Photos application on the device. Users will be able to see that the image was saved as a JPEG image, and they will also be able to see the date and time that the image was captured, which corresponds with the present date and time. Once users are finished with this activity, they will have the option to go back to the second activity by clicking the “Previous Activity” button. Pressing this button calls the Intent() function again, which allows users to switch between the third and second activities.

## User Interface (UI) + Screenshots

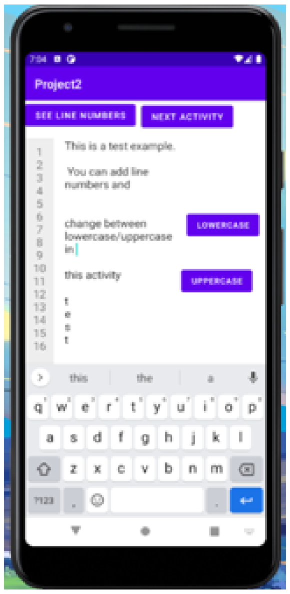
Regarding the user interface (UI), I wanted to make the app look visually appealing to users, but I did not want to have such complex elements to the point where the user would not be able to interact with the app efficiently. As a result, I chose to go with more basic elements regarding the UI, which was then applied to each activity.

To start off, for the first activity, I wanted users to be able to input whatever text they wanted using the keyboard on the device, so I first added an EditText element in the middle of the screen, which allows for user input. As a result, users will be able to easily type whatever it is they want, and they can also type as much as they want because the EditText element is large for users to type on multiple lines. In addition to the EditText element, I created a TextView element and positioned it a bit to the left of the EditText element. This TextView element is responsible for counting and displaying the line numbers on the text file, so I figured it would be most efficient to display these line numbers to the side of the EditText component. I then created buttons for the first activity and positioned them on the top and right side of the EditText component. Each button has its own functionality as discussed earlier and calls its respective onClickListener() function depending on which button is clicked. For instance, pressing the “See

Line Numbers" button will display line numbers to the user, but pressing the "Next Activity" button will send the user to the second activity. The UI for the first activity was quite basic and primarily utilized EditText, TextView, and buttons.

The UI for the second activity is also not that complex because it is just a simple search filter being implemented on a list. A ListView element is first created in order to display a vertical list of items to the user. An ArrayList and ArrayAdapter are then created in order to display the contents of the ArrayList in ListView format. It is important to note that the contents of the ArrayList are predetermined and cannot be changed unless programmers choose to add more elements to the ArrayList in the source code of the Java files. Once the whole ListView was sorted out, I created an EditText element in order to allow users to type and search for elements on the list. The EditText element then calls the addTextChangedListener() function, which filters the list according to what the user types. Similar to the first activity, I also added buttons to the second activity, which allow for users to switch between the second and first activities as well as the second and third activities. All in all, the UI for the second activity is also quite simple because it uses ListView, EditText, and buttons.

For the UI for the third activity, I first created an ImageView element, which shows a basic image picture provided by Android Studio to indicate that an image is going to be displayed on the ImageView element using the camera. I then created a button which says "Capture Image", that is responsible for granting permission from the user in order to access the camera as well as external storage of the device. Pressing this button allows users to agree or disagree to granting permissions, and if permissions are granted, the camera will be displayed and users will be able to capture images and save or discard these images depending on what button they choose to press. It is important to note that these buttons automatically appear after the image is captured, and they were not created by me when I was designing the UI for this activity. If users choose to save the captured image, the image is then displayed to the user in the ImageView element and is also saved as an image file in the Photos application on the device. Once an image is captured, users can go back to the second activity by pressing the "Previous Activity" button, which I created and added a bit below the "Capture Image" button. Similar to the previous activities, the UI is also not complex for this activity and makes use of ImageView and buttons. Below are some screenshots, which show the basic UI and design of my app.



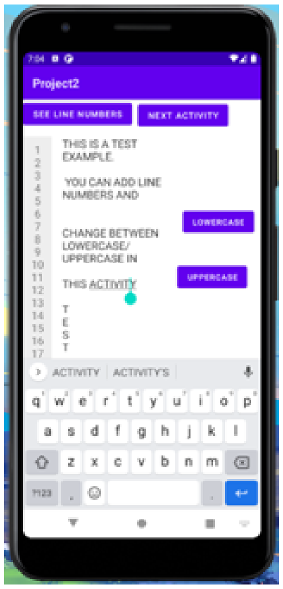
This is a test example.

You can add line numbers and

change between lowercase/uppercase in

this activity

t
e
s
t



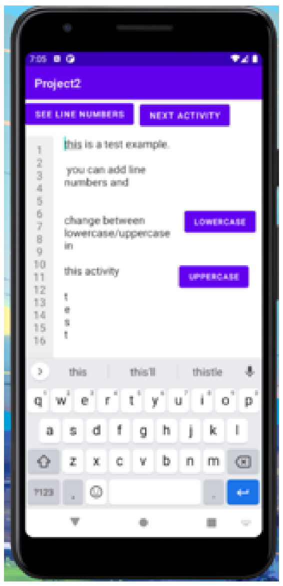
THIS IS A TEST EXAMPLE.

YOU CAN ADD LINE NUMBERS AND

CHANGE BETWEEN LOWERCASE/ UPPERCASE IN

THIS ACTIVITY

T
E
S
T



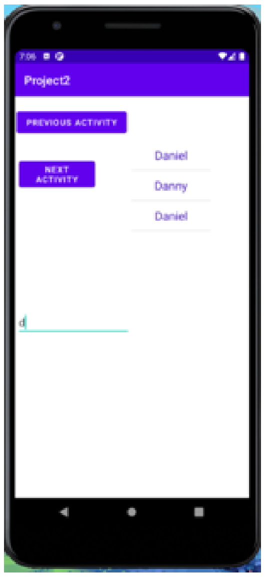
this is a test example.

you can add line numbers and

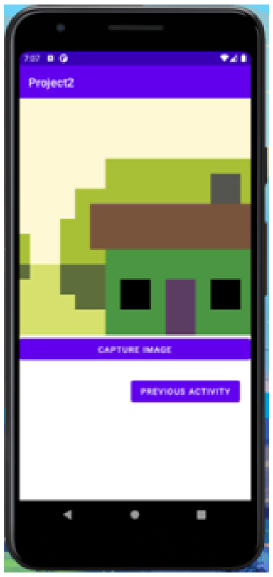
change between lowercase/uppercase in

this activity

t
e
s
t



Daniel
Danny
Daniel
d



Going from left to right, the first three screenshots essentially show the basic UI for the first activity such as EditText and TextView. They also show several key features being implemented including adding line numbers and conversion of characters between lowercase and uppercase. The fourth screenshot shows the simple UI for the second activity such as ListView and EditText, and it also shows the searching for keyword feature being implemented. Finally, the fifth and last screenshot shows the basic UI for the third activity. Notice how the captured image is displayed in the ImageView element above the "Capture Image" button. In my case, the image is an emulated image provided by AndroidStudio since I used an emulated device rather than a real Android device to test and run my app.

Source Code (OS-related functionalities):

```
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.third_act_layout);

   imgShow=findViewById(R.id.image_show);
   capture=findViewById(R.id.capture_btn);

   Button btn5= (Button) findViewById(R.id.btnGoBackAct2);

   capture.setOnClickListener(new View.OnClickListener() {
       @Override
       public void onClick(View v) {
           if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M){
               if(checkSelfPermission(Manifest.permission.CAMERA)==
                       PackageManager.PERMISSION_DENIED ||

checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE)==
                       PackageManager.PERMISSION_DENIED ){
                   String[] access={Manifest.permission.CAMERA,
Manifest.permission.WRITE_EXTERNAL_STORAGE};
                   requestPermissions(access,ACCESS_CODE);
               }
               else{
                   displayCam();
               }
           }
           else{
               displayCam();
           }
       }
   });
```

```java
private void displayCam() {
   ContentValues cv=new ContentValues();
   cv.put(MediaStore.Images.Media.TITLE, "New Image");
   cv.put(MediaStore.Images.Media.DESCRIPTION, "From Camera");

uriImg=getContentResolver().insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,cv);

   Intent camera=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
   camera.putExtra(MediaStore.EXTRA_OUTPUT, uriImg);
   startActivityForResult(camera, CAPTURE_CODE);

}


@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
   switch (requestCode){
       case ACCESS_CODE: {
           if(grantResults.length > 0 && grantResults[0]==
           PackageManager.PERMISSION_GRANTED){
              displayCam();
           }
           else{
               Toast.makeText(this, "Permission Denied", Toast.LENGTH_SHORT).show();
           }
       }
   }
}
```