[그래픽스/게임프로그래밍 프로젝트]

폴로라이드

201535002 강보은
201535005 곽소언
201535031 이혜준

목차

1. 개발 배경 및 개발 목적

1.1 기존 어플 분석

- 애플의 App Store에서 먼저 출시된 아날로그 감성의 필름카메라 '구닥'은 일회용 필름 카메라'Kodak'을 오마주한 앱으로써 낡고 오래된 것을 의미하는 '구닥다리'에서 이름을 얻어 개발한 어플이다. 실제 필름카메라와 똑같은 UI를 제공하고 하루에 24장씩 촬영할 수 있으며 3일 후에 촬영한 사진을 확인할 수 있다. '후지캠' 또한 'Fujifilm'의 필름카메라를 오마주한 어플로써 구닥과 마찬가지로 일회용 필름카메라와 비슷한 UI를 제공한다. 두 어플 모두 아날로그적 필터를 입히고 날짜가 삽입되는 등 실제 필름카메라의 사진과 비슷한 효과를 준다.

1.2 개발 목적

- '구닥', '후지캠'과 마찬가지로 'Poloride'는 즉석카메라인 '폴라로이드'에서 착안하여 개발하였다. 실제 폴라로이드와 비슷하게 간단한 조작버튼만을 배치하여 아날로그적 UI를 살리며 사진이 처음 찍히자마자 처음엔 사진이 안보이다가 점점 사진이 뚜렷하게 변하게 하여 마치 즉석카메라에서 사진이 인화 되어 사진이 현상되는 과정을 보고 있는 듯한 효과를 주어 사용자들로 하여금 아날로그적 감성을 불러일으킬 수 있게 하였다. 또한 인화된 폴라로이드 필름처럼 프레임을 씌우고 거기에 날짜, 그리기, 필터 등 꾸미기 요소를 추가하여 사용자들에게 꾸밀 수 있는 선택사항을 주고자 하였다. 완성된 사진은 포토프린터와 연동되어 뽑았을 때 실제 폴라로이드 필름처럼 보이게 한다.

2. 개발 환경

2.1 개발 환경

⑴ 개발 툴

- Android Studio
- Photoshop

⑵ 개발 언어

- JAVA

⑶ 테스트 단말

- 갤럭시 노트4 (2014)
- 갤럭시 A7 (2016)
- LG V30 (2017)

2.2 라이브러리

(1) AndroidPhotoFilters:'info.androidhive:imagefilters:1.0.7

- PhotoFiltersSDK는 모든 이미지 미디어에 멋진 효과를 만들기 위한 빠르고 강력하고 유연한 이미지 처리 도구를 제공하는 것을 목표로 한다. 라이브러리는 API 15 이상의 OS를 지원한다. 라이브러리는 또한 내장 샘플 필터와 함께 제공된다.

(2) com.github.ksoichiro:android-observablescrollview:1.5.2

- 스크롤 가능한 보기에서 스크롤 이벤트를 관찰하는 Android 라이브러리다. Android 5.0 Lollipop에 도입 된 툴바와 쉽게 상호 작용할 수 있으며 머트리얼 디자인 앱의 룩앤필을 구현하는 데 도움이 될 수 있다.

(3) com.jakewharton:butterknife:8.8.1

- 필드에 주석을 달고 뷰 ID를 사용하면 레이아웃에서 해당 뷰를 찾아 자동 캐스팅 할 수 있고, 느린 리플렉션 대신 코드가 생성되어 뷰 조회를 수행할 수 있다.

3. 요구사항

3.1 사용자 요구사항

1) 실제 폴라로이드 카메라처럼 보여야 한다.
2) 조작버튼이 간단함과 동시에 어떤 기능인지 한눈에 알 수 있어야 한다.
3) 사진촬영 후 사진현상과정이 보였으면 한다.
4) 프레임이 합쳐진 사진으로 저장되어야 한다.
5) 어플 내 자체 갤러리에서 사진 리스트를 가로스크롤로 볼 수 있어야 한다.
6) 찍힌 사진에 찍은 날짜를 삽입할 수 있고 직접 그림을 그리고 폰트가 적용된 글자를 삽입할 수 있어야 한다.
7) 원하는 필터를 적용할 수 있어야 한다.
8) 꾸며진 사진을 따로 저장할 수 있어야 한다.
9) 어플 내 자체 갤러리에서 사진을 삭제할 수 있어야 한다.

3.2 기능적 요구사항

1) 사진 촬영 시 촬영한 사진에 프레임을 합성해야 한다.
2) 사진 촬영 후 시간이 흐르거나 단말을 흔들면 사진을 확인할 수 있게 한다.
3) 어플 자체의 갤러리에서 사진을 확인 할 수 있어야 한다.
4) 사진 저장 시 휴대폰의 자체 갤러리에서도 사진이 보여야 한다.
5) 어플 내에서 갤러리의 사진을 삭제 할 수 있어야 한다.
6) SNS나 메신저, 문자로 사진을 공유할 수 있어야 한다.
7) 촬영날짜 삽입, 입력문자열 삽입, 드로우, 필터 등의 꾸미기 기능을 제공한다.
8) 선택한 사진을 포토프린터로 출력 할 수 있어야 한다.

3.3 비 기능적 요구사항

⑴ 호환성

- API 15레벨을 기반으로 개발하여 안드로이드 기기 대부분에서 사용이 가능하다. 마시멜로우(갤럭시 노트4), 누가(갤럭시 A7 2016), 오레오(LG V30) 모두 원활하게 작동함을 확인 했다. ios 지원은 하지 않는다.

⑵ 이식성

- 후지 피킷을 주 프린터로 설정한 어플이므로 후지 피킷 자체 어플의 권장 사이즈인 640*1024에 맞춰 사진이 저장되도록 하였다. 따라서 찍힌 사진을 프레임 까지 포함하여 해당 사이즈로 리사이징 하여 단말에 상관없이 해당 사이즈로 저장된다.

⑶ 사용성

- 라이트, 셔터, 홈 버튼, 꾸미기 등의 버튼들에 직관적인 아이콘을 사용하여 기능을 쉽게 알 수 있도록 했다. 또한 꾸미는 기능들을 한곳에 몰아 화면의 이동을 최대한 하지 않으면서 꾸미기 기능을 한 번의 저장에 모두 담을 수 있도록 하였다.

4.설계

4.1 기능 구현

(1) 메인화면

- 카메라 렌즈 이미지와 카메라 셔터 이미지를 사용해 실제 카메라와처럼 보이게 하는 UI를 사용한다. 카메라 렌즈 가운데에 SurfaceView를 위치하게 하여 촬영할 화면을 볼 수 있게 한다. 사진 촬영 시 플래시를 이용해 어두운 곳에서도 사진을 찍을 수 있게 한다. 셔터 버튼으로는 사진 촬영을, 갤러리 버튼으로는 갤러리로의 이동을 할 수 있다.

(2) 사진 합성 및 사진 표현

- **사진 합성** : 사진 촬영 시 촬영된 화면을 바이트 형식의 배열로 받고, 비트맵으로 변환하여 캔버스를 이용하여 프레임 이미지와 합친다. 합쳐진 이미지를 지정된 양식으로 저장하며, 저장된 이미지를 인텐트에 담아 표현 액티비티로 전달한다.

- **사진 표현** : 표현 액티비티에서는 이미지뷰의 알파 값을 조절하여 촬영된 사진의 인화 표현을 한다. 핸드폰 단말을 흔들거나 일정 시간이 지날 시 뿌옇던 사진이 점차 선명하게 변한다. 단말을 흔드는 것은 가속도센서를 사용하였다.

(3) 갤러리

- **갤러리** : 버튼을 누르면 pola폴더에 이미지 리스트를 읽어 와서 갤러리 뷰 안에 이미지가 꽉 채워지도록 하였고, 이미지를 선택하면 파일 이름을 불러와 넘겨주도록 하였다.

- **사진 저장** : 사진 저장은 단말기 내에 pola폴더를 생성하여 저장되도록 하였고, 저장 시의 이름은 현재 년도, 월, 일, 날짜, 시간, 분, 초를 이용하여 다른 사진들과 이름이 겹치지 않도록 했다. 저장된 사진을 갤러리에서 볼 수 있게 하기 위해서 미디어 스캐닝을 사용하였다.

(4) 사진 삭제

- **사진 삭제** : 갤러리에서 이미지를 선택하고 사진 삭제 버튼을 누르면 파일경로를 읽어와 이미지 uri에 저장되는데 uri를 파일 시스템을 통해 이미지를 지울 수 있게 하였다.

(5) 사진 꾸미기 (폰트, 드로우, 날짜, 필터)

- **날　짜** : 사진의 이름에 촬영 시의 일자가 들어있기 때문에 uri에서 사진을 찍은 날짜를 구하고 drawText를 이용하여 사진의 오른쪽 하단에 넣는다. paint를 이용하여 폰트와 색상을 즉석카메라처럼 보이도록 설정한다.

- **텍스트** : editText를 이용하여 입력 받은 문자열을 변수에 담고 사용자가 입력을 완료 했을 때 입력된 문자열을 프레임에 넣는다. 버튼 최초 클릭 시 입력을 돕는 힌트 문구가 화면 가운데에 출력되고, 입력 시 힌트 메시지는 사라진다. 입력 완료시에는 입력부분과 버튼이 비활성화 된다.

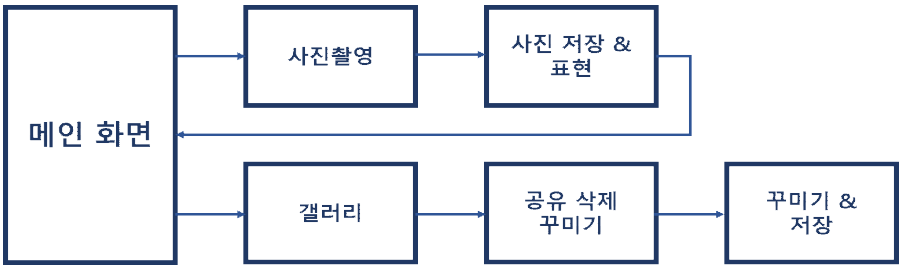
- **그리기** : 터치 리스너와 drawPath를 이용해 구현한다. 이미지 뷰에 그려진 비트맵에 바로 선을 긋게 하였다.

- **필　터** : 갤러리에서 이미지를 선택하고 하단의 필터를 선택하게 된다. 받아온 파일명의 사진 파일을 접근해서 하단의 필터 이미지를 클릭하면 적용할 필터의 종류를 받아와서 새로운 비트맵 파일로 변수에 할당해 주고 만든 비트맵을 Canvas View 에 비트맵 이미지로 다시 보여줘 사진 영역에 필터 적용된 미리 보기를 보여준다.

(6) 사진 공유

- Intent의 ACTION_SEND를 이용해 다른 어플과 사진을 공유할 수 있다. 사용자는 기기에 설치되어있는 어플을 선택하여 해당 어플로 사진을 전달 할 수 있다.

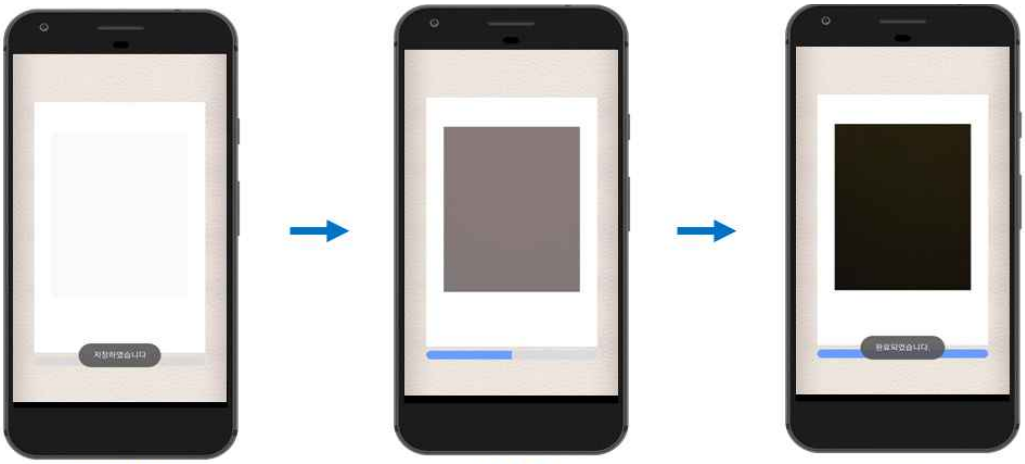
4.2 구동 설명



1. 메인화면

◀ [메인화면]에서 사용자가 카메라 SurfaceView를 통해 촬영할 화면을 확인한다. 어두울 시 <플래시>스위치로 기기의 플래시를 조작할 수 있고 <셔터>버튼으로 사진을 촬영한다. <갤러리>버튼을 누르면 사용자가 촬영한 사진을 확인 할 수 있다.

2. 촬영 직후/사진 저장&표현



▲ 촬영한 직후에는 촬영된 사진과 프레임이 합쳐져서 [갤러리]에 저장이 되고 프레임이 씌워진 사진은 사용자에게 보여 진다. 사용자는 단말기를 흔들어서 사진이 빨리 현상 될 수 있게 한다. 저장된 사진은 [메인화면]의 <갤러리>버튼에서 확인가능하다.

3. 갤러리

갤러리 화면

◀ [갤러리]에는 그동안 찍은 사진이 가로 스크롤로 저장되어 사용자가 넘기면서 사진을 확인할 수 있다. [갤러리]의 사진 리스트 중에서 하나의 사진을 선택하여 새로운 화면으로 가져간다.

4. 공유/삭제/꾸미기

갤러리에서 사진 선택 시

◀ 새로운 화면에서는 [메인화면]으로 돌아갈 수 있는 <홈>버튼, 다른 어플과 연동하여 사진을 공유할 수 있는 <공유>버튼, 꾸밀 수 있는 창으로 넘어가는 <꾸미기>버튼, 해당 사진을 삭제할 수 있는 <삭제>버튼 중에서 원하는 기능을 선택하여 실행할 수 있다.

◀ <공유>버튼을 누를 시에는 해당기기에 설치되어 있는 앱과 사진 공유가 가능하다. <꾸미기>버튼 클릭 시에는 해당 사진을 꾸밀 수 있는 새 창으로 넘어가간다.

5.꾸미기&저장

<꾸미기>버튼 클릭 시　　필터 선택화면

▲ <꾸미기>버튼 클릭 시 넘어온 새 창에는 날짜를 삽입할 수 있는 <달력>버튼, 원하는 텍스트를 입력하여 사진에 삽입할 수 있는 <텍스트>버튼, 원하는 그림을 그릴 수 있는 <드로우>버튼이 있다.

▲ <필터>버튼을 누르면 새 창으로 넘어가서 다양한 필터 적용이 가능하다. 필터 선택이 완료되면 <적용>버튼을 눌러서 필터가 입혀진 사진을 가지고 [꾸미기] 창으로 되돌아간다. 마지막으로 <저장>버튼을 누르면 지금까지 꾸민 사진을 [갤러리]에 새롭게 추가할 수 있다. 최종적으로 이미지를 저장하면 자동적으로 [갤러리]화면으로 돌아가서 꾸민 이미지가 저장된 것을 확인할 수 있다.

4.3 주요 기능 코드

(1) 프레임 합성

- 찍힌 사진을 바이트 배열로 받아 비트맵으로 변환시키고, 후지 피킷(포토프린터)과의 호환을 위해 리사이즈 한다. 리사이즈 된 이미지를 프레임과 합치기 위해 combineImages 함수를 이용한다.

```
//byte array를 bitmap으로 변환
Bitmap CameraBitmap = BitmapFactory.decodeByteArray( data, offset: 0, data.length, options);

CameraBitmap =  Bitmap.createBitmap(CameraBitmap, x: 0, y: 0, w, h, matrix, filter: true);

CameraBitmap = resizeBitmap(CameraBitmap, resizeWidth: 510);
Bitmap resultBitmap = combineImages(CameraBitmap);
```

- 촬영된 사진과 프레임의 크기를 감안하여 Canvas의 크기를 설정한다. 프레임은 캔버스에 꽉 차게, 찍힌 사진은 (65, 120)위치부터 시작되도록 한다.

```
public Bitmap combineImages(Bitmap cameraBitmap) {
    Bitmap cameraFrame = BitmapFactory.decodeResource(getResources(), R.drawable.polaback);

    int width, height = 0;
    width = cameraBitmap.getWidth();
    height = cameraBitmap.getHeight();

    Bitmap result = Bitmap.createBitmap( width: width+130,  height: height+344, cameraFrame.getConfig());
    Canvas canvas = new Canvas(result);
    canvas.drawBitmap(cameraFrame,  left: 0f,  top: 0f,  paint: null);
    canvas.drawBitmap(cameraBitmap,  left: 65,  top: 120,  paint: null);

    saveBitmaptoJpeg(result);
    return result;
}
```

(2) 갤러리

- 갤러리 버튼을 누르면 pola폴더의 이미지 리스트를 읽어 와서 갤러리 뷰 안에 이미지가 꽉 채워지도록 하고, 이미지를 선택하면 파일 이름을 불러와 넘겨주도록 한다.

```
list = imageReader( new File( pathname: Environment.getExternalStorageDirectory() + File.separator+ "pola"));

for(int i=0; i< list.size(); i++) {
    ImageView imageView = new ImageView( context: this);

    imageView.setAdjustViewBounds(true);
    Log.v( tag: "Gallery" , msg: "IMAGEPARAM1: "+ layout.getHeight());

    imageView.setOnClickListener((view) → {
            String uri = list.get(view.getId()).toString();

            Intent intent = new Intent(getApplicationContext(), SelectedPicture.class);
            intent.putExtra( name: "Data", uri);
            startActivity(intent);
    });
```

- 사진은 단말의 pola 폴더에 저장되며 이름은 현재의 년, 월, 일, 시, 분, 초를 받아서 정해지게 된다. 또한 갤러리에서 사진을 확인 할 수 있도록 미디어 스캐닝을 위한 코드를 작성한다.

```
public  void saveBitmaptoJpeg(Bitmap bitmap){
    String folder = Environment.getExternalStorageDirectory() + File.separator+ "pola" + "/";
    String file = "pola_" + getDateString() + ".jpg";

    File file_path;
    try{
        file_path = new File(folder);
        if(!file_path.isDirectory()){
            file_path.mkdirs();
        }

        FileOutputStream out = new FileOutputStream( name: folder+file);
        //사진 저장
        bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 100, out);
        //갤러리에 보이게 함
        sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, Uri.parse("file://"+ folder + file)));
```

```
public String getDateString()
{
    SimpleDateFormat df = new SimpleDateFormat( pattern: "yyyy-MM-dd-HH-mm-ss", Locale.KOREA);
    String str_date = df.format(new Date());

    return str_date;
}
```

(3) 사진 삭제

- 갤러리에서 이미지를 선택하고 사진 삭제 버튼을 누르면 파일 경로를 읽어와 이미지 uri에 저장되는데 uri 파일 시스템을 통해 이미지를 지운다. 아래는 이미지 삭제를 위한 코드의 일부이다.

```
final Button btn_del = (Button)findViewById(R.id.btn_del);
btn_del.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), GalleryActivity.class);

        File file = new File(Uri.parse(uri).getPath());
        System.out.println("file Deleted :" + uri);
        boolean isFileDeleted = file.delete();
```

(4) **사진 꾸미기** (폰트, 드로우, 날짜, 필터)

- 촬영한 사진을 전달받고 크기를 구해 canvas를 생성하고 사용한다.

```
result = Bitmap.createBitmap(width, height, image.getConfig());

canvas = new Canvas(result);
canvas.drawBitmap(image, left: 0f, top: 0f, paint: null);
```

- **날짜 입력** : 사진이 찍힌 연, 월, 일을 구하고 drawText를 이용하여 canvas에 그린다.

```
String strTime = "'"+ year + "  " + month+ "  "+day+ "  ";

int timeColor = getResources().getColor(R.color.colorTime);

time_font = Typeface.createFromAsset(getAssets(), path: "font_digital.ttf");
Paint tPaint = new Paint();
tPaint.setTextSize(25);
tPaint.setTypeface(Typeface.create(Typeface.DEFAULT, Typeface.BOLD));
tPaint.setColor(timeColor);
tPaint.setStyle(Paint.Style.FILL);
tPaint.setTypeface(time_font);
canvas.drawText(strTime, x: 455f, y: 775f, tPaint);
```

- **텍스트 입력** : editText를 이용하여 프레임에 입력된 문자열을 넣는다. 버튼 최초 클릭 시 문자를 입력할 수 있고, 한 번 더 클릭하면 문자열이 프레임에 적용된다.

```
if (btnState==1) {
    btn_inputfont.setBackgroundResource(R.drawable.font_icon_activ);
    editText.setHint("문자를 입력하세요");
    editText.setTypeface(input_font);
    editText.setHintTextColor(Color.RED);
    editText.setEnabled(true);

}else if(btnState==2) {
    btn_inputfont.setBackgroundResource(R.drawable.font_icon_finish);
    if (editText.getText().toString().length() != 0) {

        inputString = editText.getText().toString();


        fontpaint.setTextSize(70);
        fontpaint.setTypeface(Typeface.create(Typeface.DEFAULT, Typeface.BOLD));
        fontpaint.setColor(Color.BLACK);
        fontpaint.setStyle(Paint.Style.FILL);
        fontpaint.setTypeface(input_font);


        canvas.drawText(inputString,  x: 75f,  y: 925f, fontpaint);
```

- **그리기** : 터치 리스너와 drawPath를 이용해 이미지 뷰에 선을 그을 수 있게 한다. 터치와 실제로 그려지는 것에 차이가 있어 x와 y에 각각 보정을 위한 값을 나눠 사용한다.

```
float x = event.getX() / (imageView.getWidth() / 640f);
float y = event.getY() / (imageView.getHeight() / 1024f);

switch (event.getAction() & MotionEvent.ACTION_MASK) {
    case MotionEvent.ACTION_DOWN:
        path.reset();
        path.moveTo(x, y);
        canvas.drawPoint(x, y, DrawPaint);
        backX = x;
        backY = y;
        imageView.invalidate();
        return true;
    case MotionEvent.ACTION_MOVE:
        path.quadTo(backX, backY, x, y);
        canvas.drawPath(path, DrawPaint);
        backX = x;
        backY = y;
        imageView.invalidate();
        return true;
}
return false;
```

- **필터** : 갤러리에서 이미지를 선택하고 하단의 필터를 선택하게 된다. 받아온 파일명의 사진 파일을 접근해서 하단의 필터 이미지를 클릭하면 적용할 필터의 종류를 받아와서 새로운 비트맵 파일로 변수에 할당해 주고 만든 비트맵을 Canvas View 에 비트맵 이미지로 다시 보여줘 사진 영역에 필터 적용된 미리 보기를 보여준다.

```
public void onThumbnailClick(Filter filter) {

    filteredImage = image.copy(Bitmap.Config.ARGB_8888, isMutable: true);
    Bitmap filterImage = filter.processFilter(filteredImage);

    Bitmap result = Bitmap.createBitmap( width: 640, height: 1024, nocropImage.getConfig());
    Canvas canvas = new Canvas(result);
    canvas.drawBitmap(nocropImage, left: 0f, top: 0f, paint: null);
    canvas.drawBitmap(filterImage, left: 65, top: 120, paint: null);

    placeHolderImageView.setImageBitmap(result);
                    .
}
```

(5) 사진 공유 및 프린트

- Intent의 ACTION_SEND를 이용해 이미지뷰의 이미지를 다른 어플로 공유한다. 공유할 컨텐츠의 타입을 setType에서 image로 설정하고 형식은 *로 설정하여 모든 형식의 이미지를 보낼 수 있게 한다. 현재 액티비티에 있는 이미지뷰의 이미지를 uri선택하여 putExtra를 통해 다른어플로 해당 이미지를 전송한다. 공유할 어플을 선택할 수 있는 목록은 startActivity를 이용하여 기기에 설치 되어있는 어플을 띄워준다.

```
btnSharing.setOnClickListener((view) → {
        Intent intent = new Intent(Intent.ACTION_SEND);
        intent.setType("image/*");
        intent.putExtra(Intent.EXTRA_STREAM, Uri.parse(uri));
        startActivity(Intent.createChooser(intent, title: "공유"));

});
```

5. 난제

- Bitmap과 Bitmap을 이용한 이미지 합성 지식 부족
- 드로우 기능 구현 시 터치 리스너에 대한 이해 부족
- 드로우 기능 구현 시 이미지 뷰와 터치좌표 사이의 보정
- 필터기능 사용 시 레이아웃까지 필터가 입혀지는 현상
- android api와 필터 라이브러리build.gradle 설정

6. 일정