위치기반 앱 프로그래밍
#Personal Project

12141104
서정훈

1. Application 소개

(1) App 배경
여행, 관광, 명소, 맛집 등 추억의 장소에서 자신의 모습과 순간을 남기기 위해 사진을 이용한다. 이러한 추억의 사진들은 시간이 지나 나중에 보면 이 장소가 어디였는지, 그 주변은 어떤 분위기였는지 생각이 나지 않고 2차원 평면상에서 고정된 Frame만 볼 수 있다. 기억이 없는 장소를 재방문 의사가 있지만 거리와 시간, 경비의 문제로 방문하지 못하는 불편함이 있다. 이러한 문제를 해결하기 위해 Street View를 이용하여 사진이 찍힌 장소가 어디에 있는지 그 주변을 감상 할 수 있다.

(2) App 목적
최우선의 목적은 2차원으로 제공하는 사진의 시각적인 한계를 넓혀준다. 이용자가 경험하고 추억인 장소에 있는 느낌과 사진에서 볼 수 없는 정보를 제공해준다.

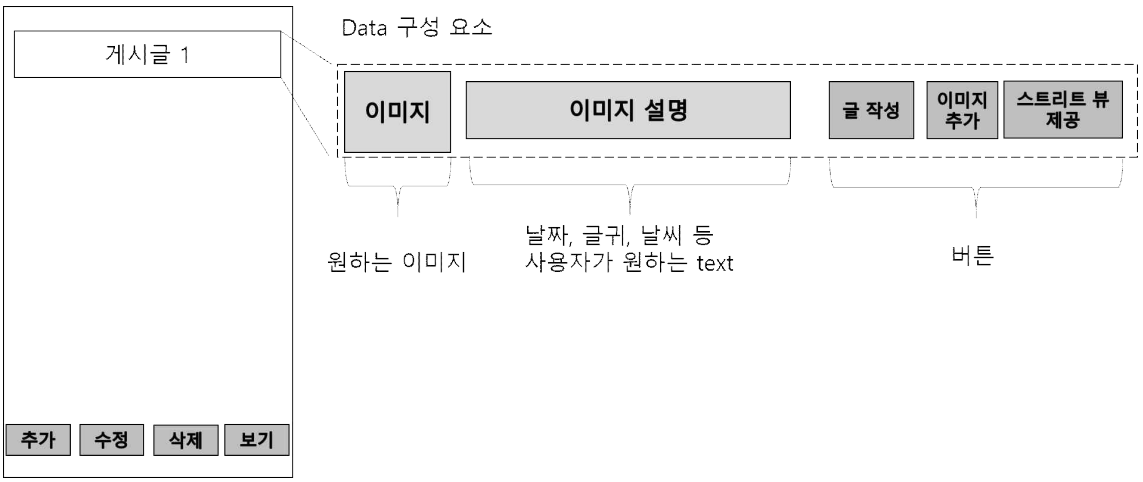
(3) App 기능
자신의 추억을 저장할 수 있는 게시글 기능과 사진의 Meta-data를 읽어와서 사진이 촬영된 장소를 Street View를 이용해 주변 풍경을 감상 할 수 있다.

2. APP 구성 요소

(1) 필수 요소

1) 게시글의 추가, 수정 및 삭제 기능
2) 게시글에 사진 추가
3) 사진의 좌표 추출과 스트리트 뷰 연결

(2) 화면 구성 및 작동



게시글을 추가, 수정, 삭제를 할 수 있으며 게시글 안에는 이미지와 이미지에 대한 설명, 이미지 설명을 작성할 수 있는 글 작성 버튼, 이미지를 가져오는 버튼, 가져온 이미지의 위치를 Street View 서비스를 제공하는 버튼으로 구성한다. 스트리트 뷰 제공 버튼을 누르면 이미지가 저장한 위치의 장소를 제공한다.

(3) 작동 Flow

휴대폰에서 촬영된 이미지를 열면 사진에서 저장된 메타데이터 Exif을 추출한다. 추출된 Exif의 위경도 좌표를 추출하여 Street View에 넘겨주면 휴대폰 화면에 이미지가 찍힌 장소에 거리의 풍경을 보여준다.

(4) Manifest, Activity 및 layout

1) Manifest

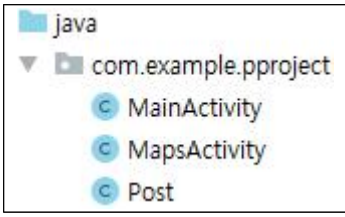
```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

앱에서 이미지 및 이전에 작성했던 게시글 불러오고 저장하기위해 WRITE, READ _EXTERNAL_STORAGE와 위치정보를 사용하기위 ACCESS_FINE_LOCATION를 권한을 허용했다.

```
<activity android:name=".MapsActivity"></activity>
<activity android:name=".Post"></activity>
```

사용하는 Activity는 3개이다. Activity간 이용하기 위해 Manifest에 명시해야 한다.

2) Activity

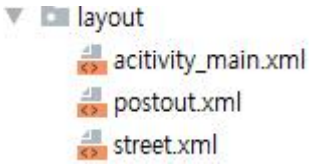


MainActivity	초기 화면에서 작동하는 activity이다.
MapActivity	street view를 작동하는 activity이다.
Post	게시글에서 작동하는 activity이다.

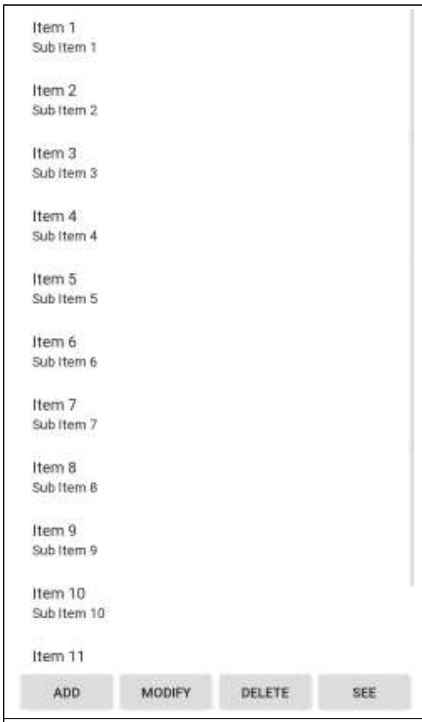
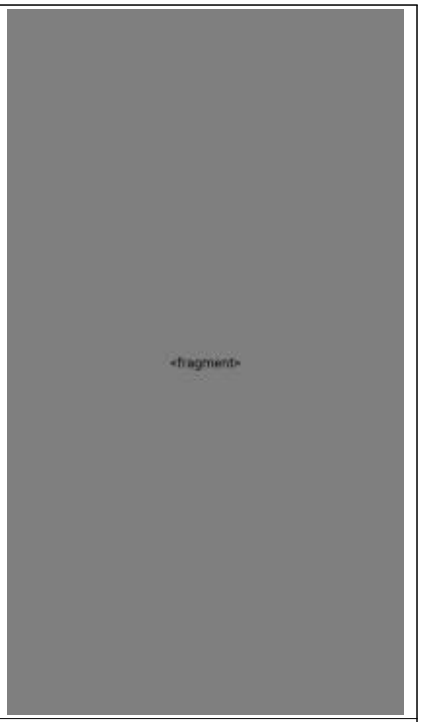
현 App에서 총 3개의 Activity를 구성하고 있다. 처음 화면에서 MainActivity가 작동되며 초기에 필요한 앱의 기능를 담당하며 게시글 추가, 수정, 삭제, 보기, 이전 게시글 가져오는 기능한다. Post에서는 게시글 중 하나에 들어와서 이미지 설정과 이미지의 대한 설명 기능을 담당한다. 이후 MapActivity와 연결하여 Street View를 제공한다.

Map Activity는 Street view를 제공해주며 이전 Activity인 Post에서 이미지의 위경도를 수신하여 화면에서 이미지의 위치를 보여준다. Street View는 구글에서 제공하는 API를 이용한다.

3) Layout

	activity_main	초기 화면을 출력하는 Layout이다.
	postout	게시글을 출력하는 Layout이다.
	street	street view를 출력하는 Layout이다.

화면을 구성하는 Layout들로서 activity_main에서 ListView로 게시글을 표현하고 버튼 4개를 구성한다. postout에서 이미지를 출력할 ImageView와 이미지의 설명 TextView 그리고 버튼 3개로 구성한다. street에서 Street View를 출력할 fragment로 구성된다. 아래의 그림과 같이 Layout 구성된다.

		
activity_main	postout	street

3. 코드 설명

(1) 구글 API 발급 및 코드 적용

https://cloud.google.com/maps-platform/

위의 링크에서 구글에서 제공하는 지도 서비스를 이용하기 위해 API를 발급받는다.

이후 발급받은 코드는 res/values/google_maps_api.xml 파일에서 발급받은 API 코드를 적용해준다.

```
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyDn3zbMByyQLggngs-rKjTcfEfTSkS_DoM</string>
</resources>
```

(2) 게시글 DB 형태

	저장 Data	변수명	자료형
변수 1	제목	title	String
변수 2	이미지 설명	txt	String
변수 3	이미지 경로	filepath	String

위의 표는 App가 재실행될 때 이전의 게시글 가져오기 위한 txt파일에 저장되는 data이다.

이전 게시글을 저장하여 App가 재실행 될 시 이전 작성했던 게시글을 보여줘야 한다. 그렇기 때문에 이전에 작성된 게시글을 저장할 필요가 있다. 게시글을 이루는 필수 요소를 txt파일 형식으로 저장한다. 위의 txt파일은 이전에 App을 작동하여 게시글을 저장된 형태이며 이를 MainActivity에서 불러들어와서 초기화면에 출력해준다.

(3) MainActivity

1) 뒤로가기로 앱 종료 시 이전의 게시글 저장

```
@Override
public void onBackPressed(){
    String save="";
    int count;
    count = adapter.getCount() ;
    try{
        BufferedWriter writer = new BufferedWriter(new FileWriter(DBWrite+"/DB.txt"));
        for(int i=0;i<count;i++){
            if(txt.get(i)==null){
                txt.set(i," ");
            }
            if(filepath.get(i)==null){
                filepath.set(i," ");
            }
            String temp= title.get(i)+"₩t"+txt.get(i)+"₩t"+filepath.get(i)+"₩n";
            save=save+temp;
        }
        writer.write(save);
        writer.close();
    }
    catch(Exception e){
        e.printStackTrace();
    }
    super.onBackPressed();
}
```

뒤로가기로 App 종료시 작동되는 코드이다. 현 앱 이용하면서 만든 data를 저장한다. BufferedWriter를 이용하여 저장할 파일경로 지정해준다. 그러면 지정해준 경로에 txt파일이 생긴다. count로 게시글 개수를 받고 반복문을 이용해 게시글 개수만큼 txt파일에 저장해준다. 그리고 게시글만 만들고 글이나 이미지를 입력하지 않는 경우 null값을 같기 때문에 임의로 값을 저장하여 반환한다.

2) 저장된 게시글 불러오기

```
//파일 저장 및 불러오기 경로 설정
DBWrite=getExternalCacheDir();
//파일 불러오기
try{
    BufferedReader reader = new BufferedReader(new FileReader(DBWrite+"/DB.txt"));
    String line = "";
    if(reader !=null){
        while((line=reader.readLine())!=null){
            String[] temp;
            String TITLE;
            String TXT;
            String FILEPATH;
            temp=line.split("₩t");
            TITLE=temp[0];
            TXT=temp[1];
            FILEPATH=temp[2];
            //불러온 DATA 추가
            title.add(TITLE);
            txt.add(TXT);
            filepath.add(FILEPATH);
        }
    }
}
catch (Exception e){
    e.printStackTrace();
}
```

코드 계속...

DBWrite는 현 코드에서 사용되는 폴더에서 Cache 폴더 경로을 설정해준다.
DBWrite에서 설정한 경로에서 DB.txt가 있을 경우 작동되며 txt파일 한줄 씩 읽어서 null이 아닌 경우 "₩t"을 구분하여 temp에 저장된다. 그 후 title, txt, filepath에 추가한다. 만약 DB.txt가 없을 경우 catch가 작동되서 코드에서 파일이 없어서 생기는 오류를 막아준다.

3) 보기 버튼

```
checked = listview.getCheckedItemPosition();
Intent intent=new Intent(this,Post.class);
intent.putExtra("TXT",txt.get(checked));
intent.putExtra("Title",title.get(checked));
intent.putExtra("Path",filepath.get(checked));
intent.putExtra("Index",checked);
startActivityForResult(intent,REQUEST_CODE);
```

int형 checked으로 선택된 게시글이 listview의 위치값으로 반환되면서 index값으로 사용할 수 있다.
보기 버튼 클릭 시 Intent를 이용해 Post Activity를 실행시켜주고 현 Activity에서 사용되는 data txt,

title, filepath, index 넘겨준다. 그리고 REQUEST_CODE 넘겨주면서 Intent가 끝날 경우 사용된다.

4) Intent 결과 반환

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data){

    if(resultCode==RESULT_OK){
        int index=0;
        index=data.getIntExtra("INDEX",0);
        String temptxt=data.getStringExtra("txtout");
        String temppath=data.getStringExtra("FILEPATH");
        txt.set(index,temptxt);
        filepath.set(index,temppath); }}
```

Post activity가 끝날시 반환되는 코드 RESULT_OK 일 경우 Post에서 반환되는 값을 올바른 index 에 txt, filepath을 저장한다.

5) 게시글 Control 버튼

```
//게시글 추가 기능
                filepath.add(" ");
                // listview 갱신
                adapter.notifyDataSetChanged();
//게시글 수정 기능
  modi(title,checked,count,adapter,listview,value);
    public void modi(ArrayList<String> items,int checked,int count, ArrayAdapter adapter, ListView
listview,String value) {

        if (count > 0) {
            // 현재 선택된 아이템의 position 획득.
            if (checked > -1 && checked < count) {
                // 아이템 수정
                items.set(checked, value) ;
                // listview 갱신
                adapter.notifyDataSetChanged();
            }
        }
    }
//삭제 버튼 기능
   checked = listview.getCheckedItemPosition();
   filepath.remove(checked);
```

게시글을 Control하는 기능가진 버튼 3개로 add()을 이용해 data를 추가해주고 remove를 통해 int 형 checked에 입력되는 선택되는 게시글 인덱스를 받아 remove()를 통해 삭제한다. 수정 버튼은 함수 modi로 구현했고 코드에 있는 필요한 변수를 넘겨줘서 set()함수를 통해 값을 수정한다.

(4) Post

1) 파일 경로에 있는 이미지 불러오기

```
File imgFile = new File(filepath);
if(imgFile.exists()){
    Bitmap bm = BitmapFactory.decodeFile(imgFile.getAbsolutePath());

    try{
        ExifInterface exif=new ExifInterface(filepath);
        getexif(exif);
    }catch(IOException e){
        e.printStackTrace();
        Toast.makeText(this,"이미지를경로를확인해주세요!",Toast.LENGTH_LONG).show();
    }
    degree=ExiforientationToDegrees(exiforientation);
    bm=rotate(bm,degree);
    imageView.setImageBitmap(bm);}
```

MainActivity에서 전달 받은 filepath에 이미지가 있을 경우 Exif 추출한다. 영상이 회전되 있을 경우 회전시켜주고 올바른 영상 orientation를 적용하여 ImageView에 이미지를 보여준다.

2) 버튼 기능

```
public void monclick(View v){
    switch (v.getId()){

        //img read
        case R.id.button4:
            getImageFromAlbum();
            break;
        // street view
        case R.id.button5:
            Intent intent=new Intent(this,MapsActivity.class);
            intent.putExtra("LAT",latlon[0]);
            intent.putExtra("LON",latlon[1]);
            startActivityForResult(intent,TEST);
            break;

        case R.id.button6:
          MainActivity에 제목 수정과 같은 코드이다.
}
```

id 버튼 4를 누르면 함수 getImageFromAlbum() 갤러리에 있는 이미지를 불러온다.
id 버튼 5는 MapsActivity를 호출하고 이미지의 Lat,Lon 값을 전달해준다.
id 버튼 6은 이미지 설명을 적어주는 기능이며 Title(제목)을 바꾸는 코드가 같기 때문에 생략한다.

3) 앨범에서 사진 불러오기

```
private void getImageFromAlbum(){
    Intent intent=new Intent(Intent.ACTION_PICK);
    intent.setType(MediaStore.Images.Media.CONTENT_TYPE);
    intent.setData(MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(intent,PICK_FROM_ALBUM);
}
```

ACTION_PICK 설정으로 갤러리에서 이미지를 가져온다. setType는 원하는 형식의 Type를 설정한다. 예를 들면 이미지나 동영상 등의 형태로 설정한다. CONTENT_TYPE으로 content 형태로 설정했다. setData로 intent의 Data를 설정한다. URI형태로 설정했다. 갤러리에서 이미지 선택됐을 때 URI로 반환이 되고 URI을 절대 파일 경로 바꾸면 이미지가 저장된 파일 경로를 알 수 있어 나중에 이미지를 불러올 때 파일 경로만 있으면 된다. requestcode를 PICK_FROM_ALBUM으로 설정하여 나중에 Intent가 끝나고 onActivityResult에서 이후 작동할 코드를 적용할 수 있다.

4) 갤러리에서 이미지 선택 후

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent){
    if(requestCode==PICK_FROM_ALBUM){
        Uri photo=intent.getData();
        //이미지 설정

        //파일 경로 가져오기1
        String pp=getPath(photo);
        filepath=pp;

      이후 (4)-1)에서 적용되는 코드와 같다.


    }
```

갤러리에서 선택된 이미지를 가져오면 Uri의 형태로 반환된다. 함수 getPath()로 현 Uri의 절대 파일 경로를 가져온다. 파일 경로를 이용해 (4)-1)에서 적용된 코드와 같이 이미지를 설정해주면 된다.

5) 절대 파일 경로 반환

```
public String getPath(Uri uri){
    String[] projection={MediaStore.Images.Media.DATA};
    Cursor cursor=managedQuery(uri,projection,null,null,null);
    startManagingCursor(cursor);
    int columnIndex=cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
    cursor.moveToFirst();
    String imgpath=cursor.getString(columnIndex);
    return imgpath;
}
```

갤러리 이미지 선택 후 반환되는 Uri로 절대 파일 경로로 변환하는 함수이다.

Cursor는 data를 DB의 Table 형태로 다루는 Class이다.
managedQuery 절대 파일경로와 uri를 DB형태로 지정한다. startManageingCursor를 통해 Cursor를 이용한다. getColumnIndexOrThrow을 통해 DB형태로 저장된 파일경로에서 절대 경로를 가져온다. moveToFirst는 Cursor를 제일 첫 번째 행으로 이동시킨다. 이후 String 형태로 반환하여 사용한다.

6) Exif – 위경도, 회전값 가져오기

```
public void getexif(ExifInterface exif){
    exif.getLatLong(latlon);
    Log.i("TAG","latitude : "+latlon[0]+",longtitude : "+latlon[1]);

exiforientation=exif.getAttributeInt(ExifInterface.TAG_ORIENTATION,ExifInterface.ORIENTATION_NORMAL);}
```

사진에서 저장된 Meta-data에서 위경도 값과 오리엔테이션 값을 저장한다.
함수 getLatLong()을 통해 위경도 값을 저장하고 함수 getAttributeInt() 통해 사진에서 저장된 orientation값을 저장한다.

7) 회전값 Degree 반환

```
public int ExiforientationToDegrees(int exiforientation){
    if(exiforientation==ExifInterface.ORIENTATION_ROTATE_90){
        return 90;
    }else if(exiforientation==ExifInterface.ORIENTATION_ROTATE_180){
        return 180;
    }else if(exiforientation==ExifInterface.ORIENTATION_ROTATE_270){
        return  270;
    }
    return 0;
}
```

Exif에서 회전의 회전값은 degree 형태가 아니므로 degree형태로 반환해준다. 이후 이미지를 출력할 때 회전하여 보여준다.

8) 이미지 회전

```
public Bitmap rotate(Bitmap bitmap, int degrees) {
    if(degrees != 0 && bitmap != null)
    {
        Matrix m = new Matrix();
        m.setRotate(degrees, (float) bitmap.getWidth() / 2,
                (float) bitmap.getHeight() / 2);

        try
        {
            Bitmap converted = Bitmap.createBitmap(bitmap, 0, 0,
                    bitmap.getWidth(), bitmap.getHeight(), m, true);
```

```
                if(bitmap != converted)
                {
                    bitmap.recycle();
                    bitmap = converted;
                }
            }
            catch(OutOfMemoryError ex)
            {
            }
        }
        return bitmap;
    }
```

Matrix를 이용하여 Bitmap를 회전시켜준다. createBitmap를 통해 회전될 Matrix와 맵 사이즈 bitmap를 넘겨줘 회전된 converted를 생성 시켜준다. bitmap과 converted가 같지 않으면 회전이 되었으므로 bitmap에 회전된 영상을 저장하고 반환한다.

9) 뒤로가기 버튼을 이용해 값 반환

```
@Override
public void onBackPressed(){
    intent.putExtra("txtout",txt);
    intent.putExtra("INDEX",index);
    intent.putExtra("FILEPATH",filepath);
    setResult(RESULT_OK,intent);
    this.finish();
    super.onBackPressed();
}
```

뒤로 가기 버튼을 눌렀을시 intent에 Post에서 사용된 txt와 index, filepath를 반환해 준다.

(5) MapsActivity

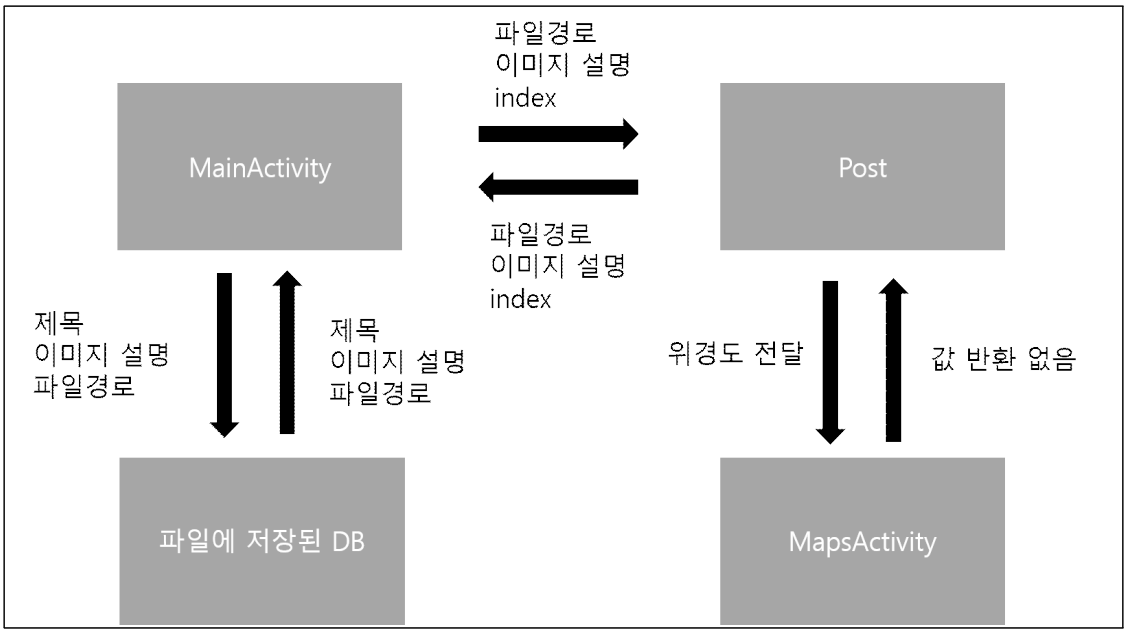
1) 위경도 값 수신 후 적용

```
    //onCreate 함수 안에서 작동한다. Post에서 위경도 수신
    Intent intent=getIntent();
    latlon[0]=intent.getFloatExtra("LAT",36);
    latlon[1]=intent.getFloatExtra("LON",127);

public void onStreetViewPanoramaReady(StreetViewPanorama streetViewPanorama) {
    LatLng test=new LatLng(latlon[0],latlon[1]);
    streetViewPanorama.setPosition(test);
}
```

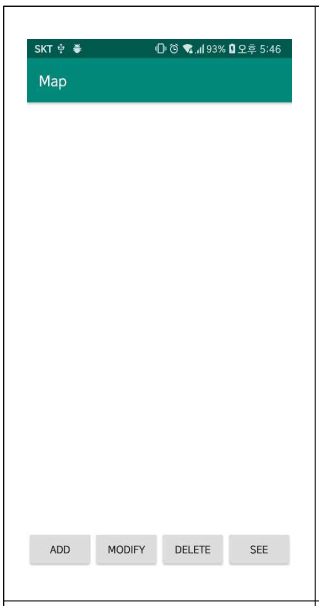
Post에서 전달받은 위경도 값을 저장하고 Street View에 위경도 위치값을 설정해준다.

(6) Activity간의 송수신 Data

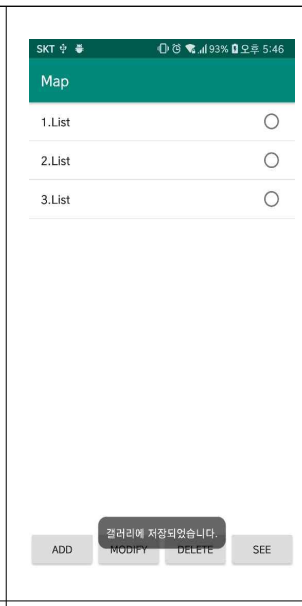


4. 작동 및 결과

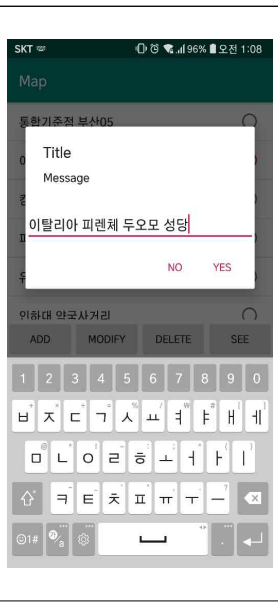
(1) 초기화면 및 버튼 작동



초기 화면

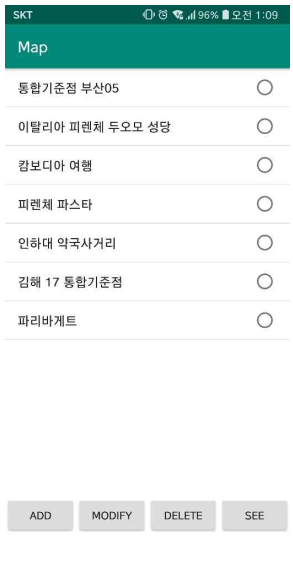


추가 버튼 클릭



수정 버튼 클릭

제목 수정 결과



삭제 버튼 클릭

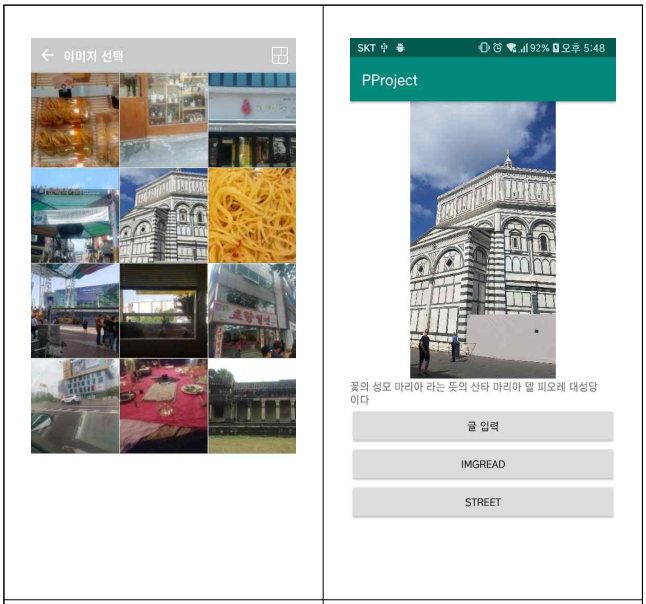
보기 버튼 클릭

이미지 설명 입력

제목 수정 결과

초기화면은 아무것도 없는 상태이며 추가 버튼을 클릭 시 클릭한 만큼 글이 생긴다. 이후 수정 버튼은 게시글 선택 후 제목을 바꿀 수 있다. 삭제 버튼도 똑같이 게시글 선택 후 삭제하면 글이 지워진다. 현 그림에서 유럽 공항이라는 제목의 글이 삭제되었다. 이후 보기 버튼 클릭 시 Post가 실행되어 postout.xml를 출력하여 글 입력 기능, 이미지 입력, Street View 기능 제공한다. 글 입력 버튼 클릭 시 다이얼로그가 나오고 글을 수정할 수 있다.

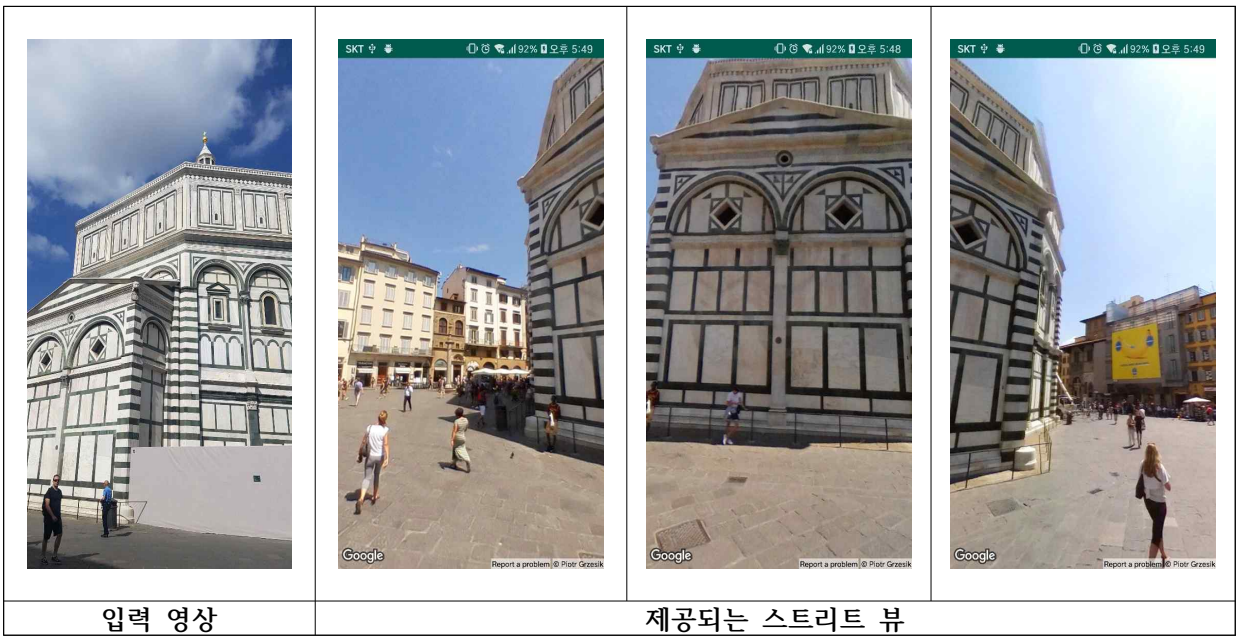
(2) 이미지 입력 및 출력



갤러리에서 이미지 입력	이미지 입력 결과

IMGREAD 버튼를 누르면 갤러리에 있는 사진을 선택할 수 있다. 사진을 선택하면 위의 그림과 같이 이미지가 게시글에 이미지가 출력된다.

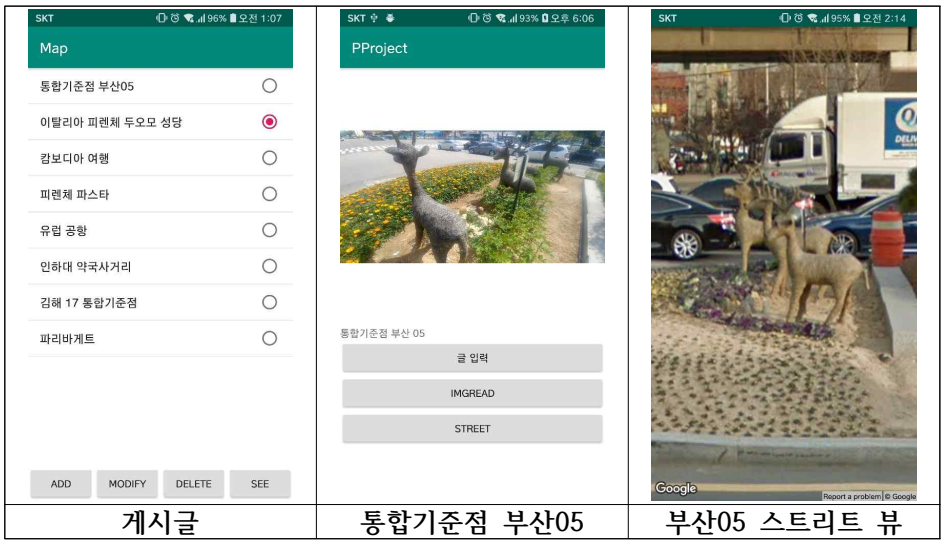
(3) 스트리트 뷰 제공



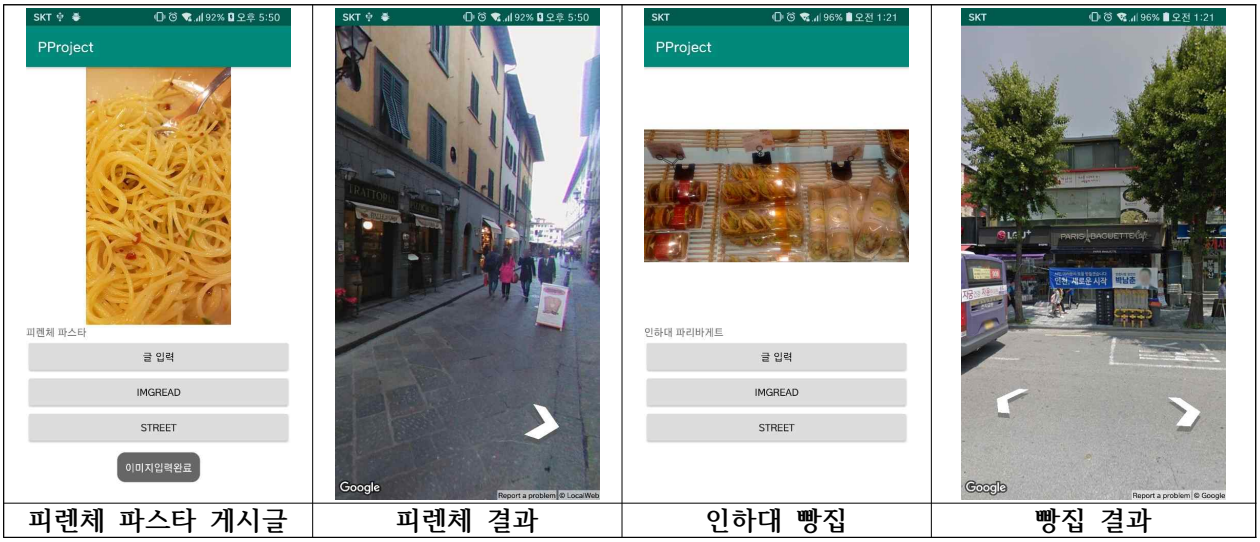
입력 영상	제공되는 스트리트 뷰

위의 그림은 이탈리아 피렌체 두오모 성당이다. 입력된 사진과 같은 건물을 볼 수 있고 그 주변의 거리와 상점까지 관찰이 가능하다.

(4) 결과



게시글	통합기준점 부산05	부산05 스트리트 뷰



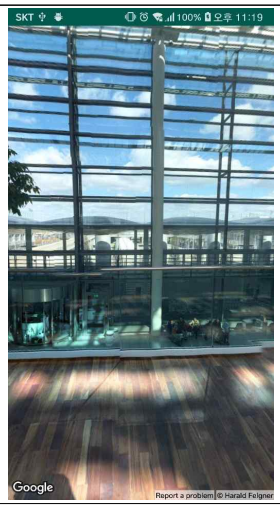
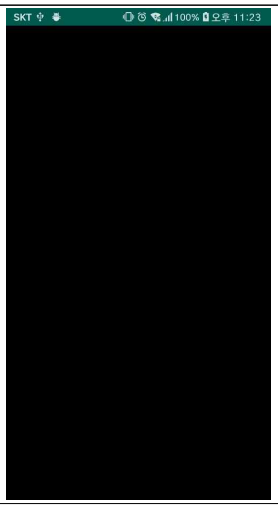
피렌체 파스타 게시글	피렌체 결과	인하대 빵집	빵집 결과

캄보디아 여행 글	캄보디아 결과	인하대 후문 글	인하대 후문 결과

위의 결과 화면을 보면 야외에서 사진을 찍은 경우 상대적으로 Street View가 제공 지역일 시 올바른 위치에서 서비스를 제공한다. 또한 서비스를 제공하는 거리에서 가까운 상점들은 실내에서 촬영한 이미지도 서비스를 제공받을 수 있음을 알 수 있다. 사진이 찍힌 지역으로부터 Street View가

이동이 가능하며 주변 환경이나 근처 상점, 도시 분위기를 알 수 있다.
이미지에 태깅된 Meta data를 이용하기 때문에 단말기의 위치 정확도가 중요함을 알 수 있다. 저 위의 피렌체 파스타 사진의 결과를 보면 좁은 길이 나온다. 이는 멀티패스 오차를 발생하기 때문에 위치 좌표값을 신뢰할 수 없다. 그러므로 저 거리는 사진을 촬영한 사람이 아니면 알 수 없다. 따라서 정확한 위치 센싱이 중요함을 알 수 있다.
이 앱의 목적은 사진이 촬영된 지역을 Street View로 서비스 해주고 그 장소 재방문하지 않아도 시간과 경비를 아끼면서 여행지의 추억을 기억할 수 있게 하며 사진의 한계점인 2차원 영상 형태에서 벗어나야 한다. 위의 사진 중 유럽을 제외한 사진은 모두 작성자가 찍은 사진이다. 개인적인 평가로 캄보디아 고대 유적지가 모두 비슷하게 생겨서 사진만으로 어디서 찍은 사진인지 알 수 없었다.
하지만 사진에서 저장된 위경도 값을 이용해서 그 주변을 둘러보고 이동하면서 여기가 앙코르 와트임을 알 수 있었다. 2차원 영상에서 알 수 없는 정보를 이 앱에서 제공 받을 수 있다. 따라서 2차원 사진의 시각적인 한계를 Street View로 해결했고 사진만 보고 어디 지역인지 알 수 없는 문제를 이 App을 통해 사진이 찍힌 지역을 돌아다닐 수 있었다. 사진에서 볼 수 없는 정보를 관찰하면서 그 지역에서 걸어다니는 효과가 있다.

5. 문제점

<table>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
</td>
<td>  </td>
</tr>
<tr>
<td>유럽 공항</td>
<td>잘못된 위치 센싱</td>
<td>제주도 카페</td>
<td>제공 불가 지역</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
<td></td>
<td></td>
</tr>
<tr>
<td>인하대 후문 꽃집</td>
<td>시기 차이</td>
<td></td>
<td></td>
</tr>
</table>

위에 3가지 경우는
1. 잘못된 위치 센싱으로 인한 잘못된 지역을 제공한다.
2. Street View가 제공 불가 지역이다.
3. 촬영된 시기가 달라서 새로 생긴 가게는 제공이 안된다.
3 가지 문제점이 있다. 1 번 문제점은 정확한 위치를 센싱할 수 없어서 발생한 문제점이다. 이는 앞으로 해결하기 어려운 문제이다. 하지만 영상에서 제공하는 위경도 좌표를 임의로 변경하여 사진이 찍힌 위치를 조절할 수 있다.
2, 3 번 Street View의 관리와 Update 문제점으로 서비스 지역이 제공이 불가하며 전 세계적으로 서비스하기 때문에 Update 주기가 짧을 수 없어 발생한 문제이다. 아래의 그림은 구글 Street View 에서 만든 것이 아니라 사용자가 직접 촬영하여 업데이트한 지역들이다. 인하대학교 안은 Street View를 제작하지 않았지만 이렇게 사용자가 직접 제작하여 Update가 가능하기 때문에 이용자가 많아진다면 2, 3의 문제점은 해결된다.

5호관	사용자 제공	60주년	사용자 제공

6. 발전 가능성

현재 App은 갤러리에 저장되어 있는 이미지로만 서비스를 제공받을 수 있다. 그렇기 때문에 이미지를 구해야 했다. 지금 위에서 사용된 유럽 이미지는 해외에서 여행 중인 지인한테 받았다. 이미지를 주고 받는 상황에서 이미지가 촬영된 지역의 분위기와 상점에 대해서 대화를 했다. 이러한 특징은 현 SNS에서 잘 보여주고 있는 특징이다. 사진과 글 올리고 많은 사람과 소통한다. 여기서 Street View를 제공한다면 사진에서 보이는 것 뿐 만 아니라 주변의 상점과 분위기에 대해 대화가 가능해진다. 그렇기 때문에 공유 기능이 추가 될 경우 사용자 간의 소통이 활발할 것 이다.
또한 Street View 장점 중 하나가 단말기 하나로 그 지역을 탐방 할 수 있다. 하지만 실내, 사유지 등 다양한 이유로 서비스가 제공되지 않는 지역이 많다. 이러한 지역들이 서비스가 가능해진다면 다양한 산업과 연결되면 좋을 것 같다. 그 중 VR산업과 연결되면 좋은 서비스를 제공할 수 있다고 생각한다. 또한 전 세계적으로 제공하기 때문에 Update 문제가 있다. 이는 유저들이 많아져 사진을 Upload 한다면 인기있는 지역은 Update가 편리할 것으로 예상된다. 또한 AR 산업과 연결되면 자신의 모습과 배경이 같이 찍힌 사진을 Street View에서 자신의 모습이 담겨있는 서비스를 제공하면 자신의 주변 사람들, 친구, 가족과 재미있는 요소 중 하나가 될 것 이다.