졸업작품(종합설계)

개인 포트폴리오

학 과 : 컴퓨터정보공학과

학 번 : 20200889

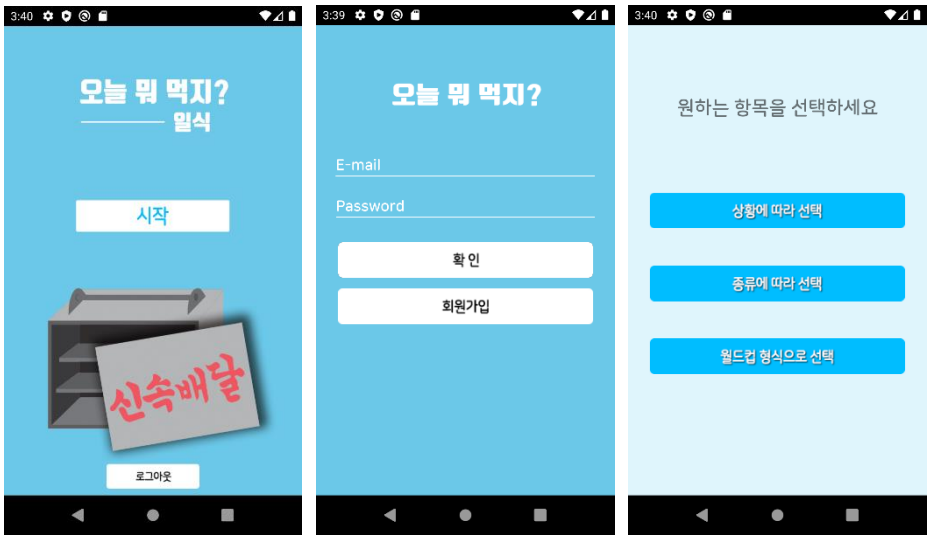
이 름 : 유재은

제출일 : 2021-11-29

목차

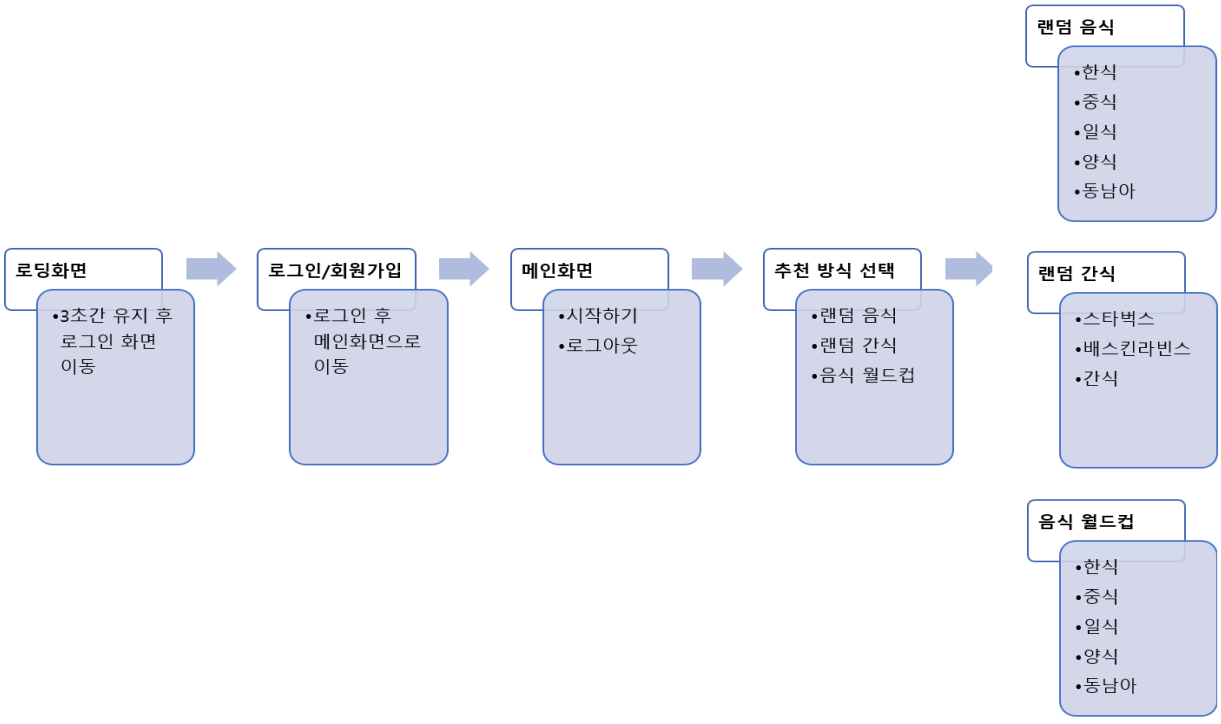
오늘 뭐 먹지?

1. 앱 개요



안드로이드 스튜디오를 이용한 음식 결정 앱으로, 메뉴 결정에 도움을 주는 앱이다.

2. 앱 흐름도



랜덤 선택 - 버튼 누를 때마다 랜덤으로 선택

음식 월드컵 – 16강, 8강, 4강, 최종, 결과 화면으로 구성

- 로딩화면

SplashActivity

```
Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        Intent main = new Intent(SplashActivity.this, LoginActivity.class);
        startActivity(main);
        finish();
    }
}, 2000);
```

Handler 사용하여 2초 후 로그인 화면으로 이동하도록 설정

```
import com.bumptech.glide.Glide;

ImageView gif_image = (ImageView) findViewById(R.id.gif_image);
Glide.with(this).load(R.drawable.loading).into(gif_image);
```

Gif 파일을 액티비티 이미지 뷰에 띄우기 위해 Glide를 import하여 로딩화면 구성

Manifest

```
<activity
    android:name=".SplashActivity"
    android:theme="@style/SplashTheme">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
```

Intent-filter 설정 시, 설정한 액티비티가 앱 실행 처음에 나타남.

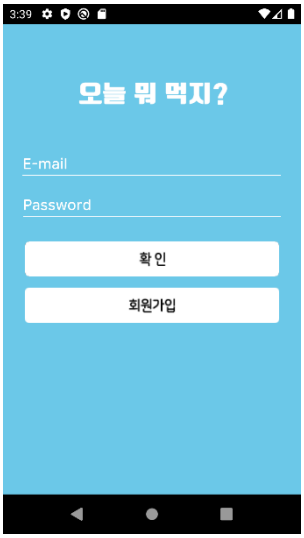
android:theme 속성을 통해 style을 적용할 수 있음.

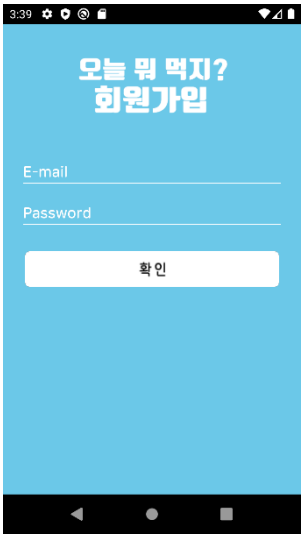
Style

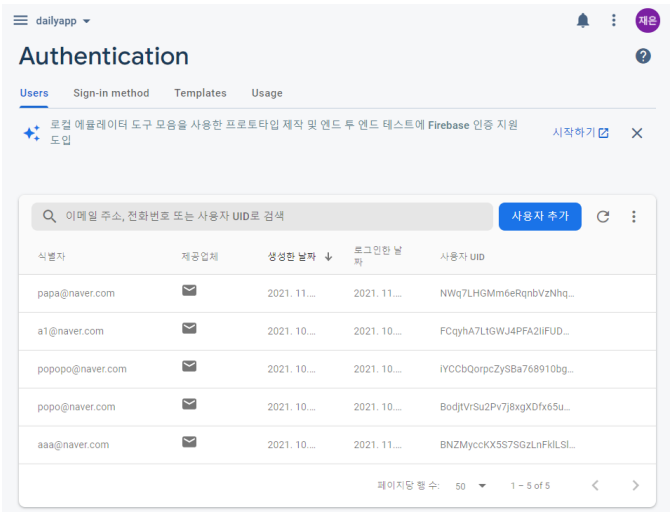
```
<style name="SplashTheme" parent="Theme.AppCompat.NoActionBar">
</style>
```

타이틀바 제거를 위한 style 정의

- 로그인/회원가입







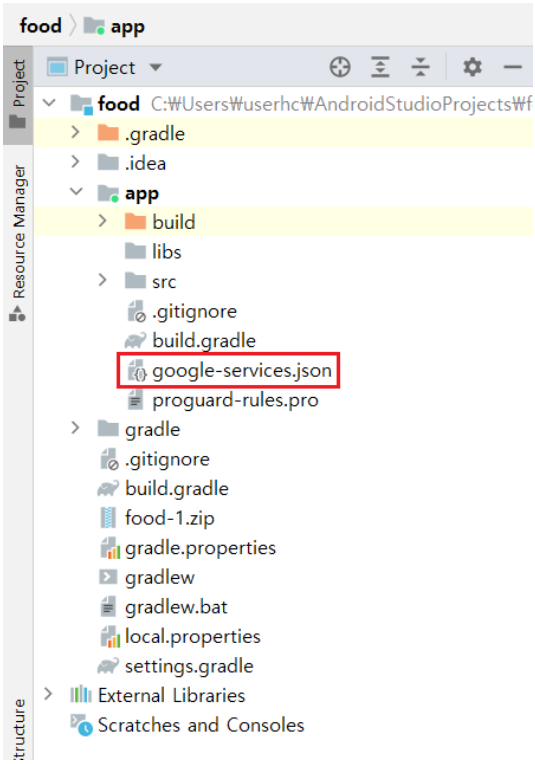
Firebase 연동

1. Firebase 콘솔 페이지(https://console.firebase.google.com/)에 접속하여 프로젝트 생성

2. google-service.json 파일 생성

   안드로이드 프로젝트 패키지 이름/앱 닉네임/SHA-1 입력

   Google-services.json 다운로드하여 Project – app 경로에 넣기



3. Firebase와 SDK 연결

build.gradle(Project : appname)

- repositories {
    google() *// Google's Maven repository*
  } 추가
- dependencies {
    classpath "com.android.tools.build:gradle:4.0.1"
    classpath 'com.google.gms:google-services:4.3.10'
    // NOTE: Do not place your application dependencies here; they belong
    // in the individual module build.gradle files
  } 추가
- allprojects {
    repositories {
      google()
    }
  } 추가

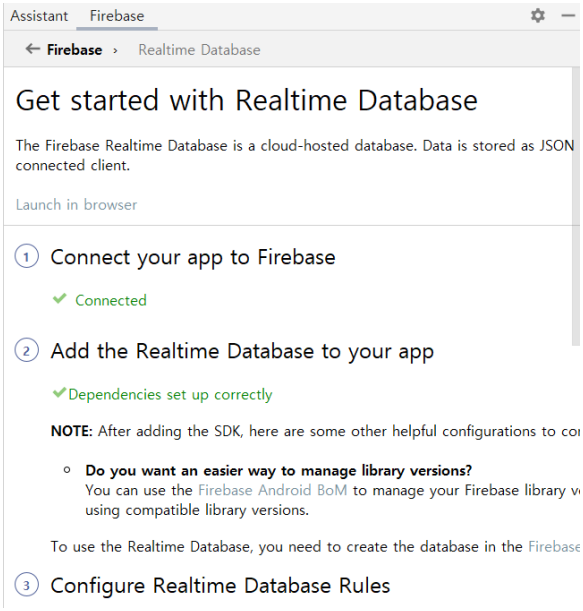
build.gradle(Module.app)

- apply plugin: 'com.android.application'
  apply plugin: 'com.google.gms.google-services'
  추가
- dependencies {
    implementation platform('com.google.firebase:firebase-bom:28.4.2')
    implementation 'com.google.firebase:firebase-analytics'
    implementation 'com.google.firebase:firebase-database:20.0.2'
    implementation 'com.google.firebase:firebase-auth'
  } 추가

이후 Sync Now 버튼을 클릭하여 동기화

4. Firebase 추가되었는지 확인

Tools – Firebase – Realtime Database > 1번, 2번 Connected



5. Firebase 데이터베이스 생성

SignupActivity

```
private FirebaseAuth mFirebaseAuth; // 파이어베이스 인증
private DatabaseReference mDatabaseRef; // 실시간 데이터베이스

//파이어베이스 접근 설정
mFirebaseAuth = FirebaseAuth.getInstance();
mDatabaseRef = FirebaseDatabase.getInstance().getReference("food");

String email = editEmail_su.getText().toString().trim();
String pwd = editPassword_su.getText().toString().trim();

if (!email.equals("") && !pwd.equals("")) {
    // 이메일과 비밀번호가 공백이 아닌 경우
    mFirebaseAuth.createUserWithEmailAndPassword(email,
pwd).addOnCompleteListener(SignupActivity.this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
```

```
                // 회원가입이 이루어졌을 때의 처리
                if (task.isSuccessful()) {
                    FirebaseUser firebaseUser = mFirebaseAuth.getCurrentUser();
                    UserAccount account = new UserAccount();
                    account.setIdToken(firebaseUser.getUid());
                    account.setEmailId(firebaseUser.getEmail());
                    account.setPassword(pwd);

                    // setValue : database 에 insert

mDatabaseRef.child("UserAccount").child(firebaseUser.getUid()).setValue(account);

                    Toast.makeText(SignupActivity.this, "회원가입 성공",
Toast.LENGTH_SHORT).show();

                    // 로그인 액티비티로 이동
                    Intent intent = new Intent(SignupActivity.this, LoginActivity.class);
                    startActivity(intent);
                    finish(); // 현재 액티비티 파괴
                } else {
                    // 회원가입 실패
                    Toast.makeText(SignupActivity.this, "회원가입 실패",
Toast.LENGTH_SHORT).show();
                }
            }
        });
} else {
    // 이메일과 비밀번호가 공백인 경우
    Toast.makeText(SignupActivity.this, "계정과 비밀번호를 입력하세요.",
Toast.LENGTH_LONG).show();
}
```

회원가입을 시도했을 때 아이디토큰, 이메일, 비밀번호를 저장

이메일과 비밀번호 입력하지 않았으면 입력하라는 토스트 창을,
잘못된 이메일과 비밀번호를 입력하면 회원가입 실패 토스트 창을 띄우도록 설정
회원가입 성공 시, 로그인 액티비티로 이동

LoginActivity

```
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

private FirebaseAuth mFirebaseAuth; // 파이어베이스 인증
private DatabaseReference mDatabaseRef; // 실시간 데이터베이스

//파이어베이스 접근 설정
mFirebaseAuth = FirebaseAuth.getInstance();
mDatabaseRef = FirebaseDatabase.getInstance().getReference("food");

// 로그인 요청
String email = editEmail_lg.getText().toString().trim();
String pwd = editPassword_lg.getText().toString().trim();

if (!editEmail_lg.getText().toString().equals("") && !editPassword_lg.getText().toString().equals("")) {
    //String 형 변수 email.pwd(edittext 에서 받아오는 값)으로 로그인하는것
    mFirebaseAuth.signInWithEmailAndPassword(email, pwd)
            .addOnCompleteListener(LoginActivity.this, new OnCompleteListener<AuthResult>()
{
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        // 로그인 성공 -> 메인 액티비티로 이동
                        Intent intent = new Intent(LoginActivity.this, MainActivity.class);
                        startActivity(intent);
                        finish(); // 현재 액티비티 종료
                    } else {
                        Toast.makeText(LoginActivity.this, "로그인 실패",
Toast.LENGTH_SHORT).show();
                    }
                }
            });
```

```
} else { // 빈칸이면
    Toast.makeText(LoginActivity.this, "계정과 비밀번호를 입력하세요.",
Toast.LENGTH_LONG).show();
}
```

이메일과 비밀번호 입력하지 않았으면 입력하라는 토스트 창을,
잘못된 이메일과 비밀번호를 입력하면 로그인 실패 토스트 창을 띄우도록 설정
로그인 성공 시, 메인 액티비티로 이동

로그아웃

```
mFirebaseAuth.signOut();

Intent intent = new Intent(MainActivity.this, LoginActivity.class);
startActivity(intent);
finish();
```

로그아웃 버튼 클릭 시, 로그아웃되고 로그인 액티비티로 이동

- 액티비티 이동

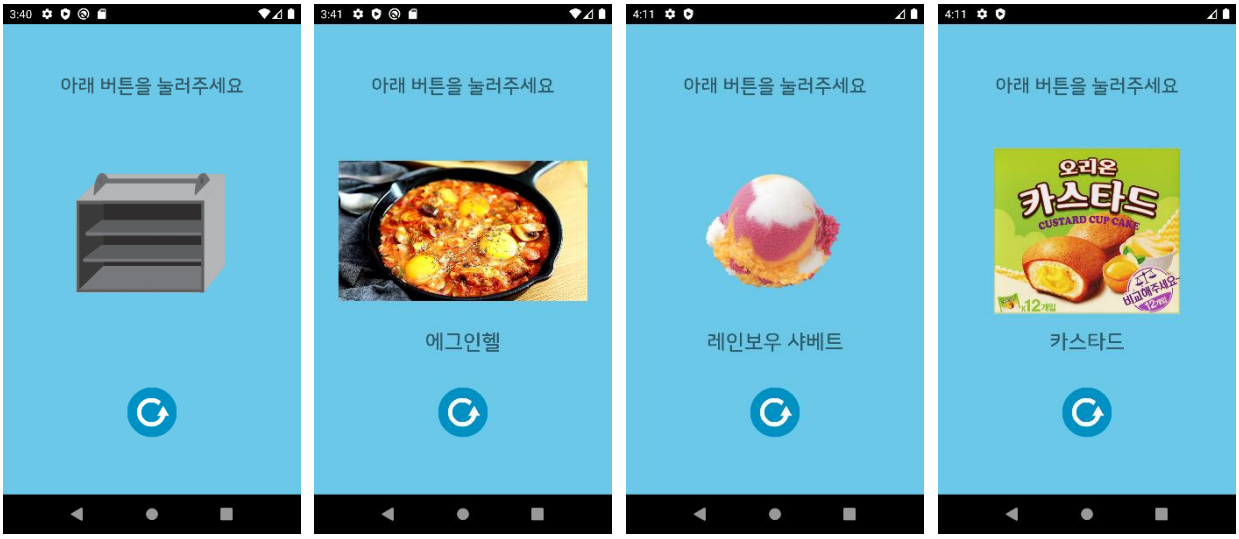
액티비티와 액티비티 사이를 이동할 땐 Intent 사용

```
Intent intent = new Intent(this, Activity.class);
// 첫번째 인자=자신, 두번째 인자=이동할 액티비티
startActivity(intent);
Finish(); // 액티비티 종료
```

```
btn_start.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this, Menus.class);
        startActivity(intent);
    }
});
```

메인 화면의 시작 버튼을 누르면 결정 방법을 선택할 수 있는 화면으로 이동되도록 함

- 랜덤 선택



버튼 누를 때마다 랜덤으로 설정한 이미지와 텍스트 보여줌

activity_korean

```
<ImageButton
    android:id="@+id/click"
    android:layout_width="70dp"
    android:layout_height="60dp"
    android:layout_gravity="center"
    android:background="@android:color/transparent"
    android:scaleType="centerInside"
    android:src="@drawable/replay"
    tools:ignore="SpeakableTextPresentCheck" />
```

background를 투명으로 설정하여 깔끔하게 보이도록 함

이미지를 중앙에 가득차게 보이기 위해 scaleType을 “centerInside”로 설정

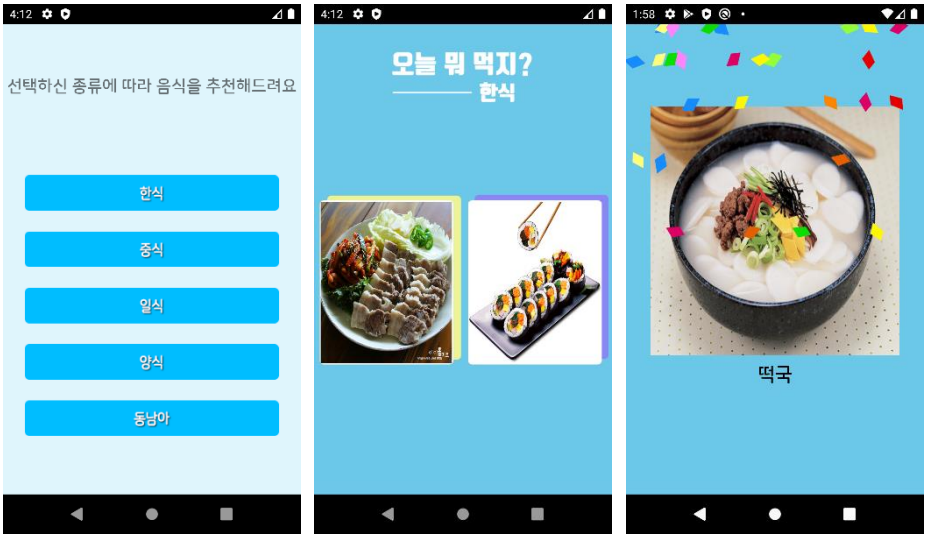
korean

```
import java.util.Random;

Random random = new Random();

btn_click.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        int count = random.nextInt(8)+1;
        switch(count){
            case 1 :
                imageView.setImageResource(R.drawable.k_pic1);
                txtResult.setText("김치찌개");
                break;
                 ...
                   }
            }
        });
```

이미지 버튼을 선택하면 랜덤으로 보여주기 위하여 Random을 import 하여 구성

Switch를 이용하여 ImageView와 TextView에 보여줄 내용을 설정

- 음식 월드컵



GameManager

```
public ArrayList<Player> fdwc = new ArrayList<Player>();

public void addFood(String name, int img, int id) {
    Player tmpFood = new Player();
    tmpFood.setName(name);
    tmpFood.setFoodIndex(id);
    tmpFood.setImg(img);
    fdwc.add(tmpFood);
}
```

ArrayList에 이름, 아이디, 이미지 저장

GameActivity

```
new Handler().postDelayed(new Runnable() {

    @Override
    public void run() {
        // TODO Auto-generated method stub

        GameManager.getInstance().fdwc
                .remove(nowPlay2);
        // GameManager.getInstance().randomList();
        nowPlay = 0;
        nowPlay2 = 1;
        foodImg1.setImageResource(GameManager
                .getInstance().fdwc.get(
                nowPlay).getImg());
        foodImg2.setImageResource(GameManager
                .getInstance().fdwc.get(
                nowPlay2).getImg());
    }
}, 1500);
```

GameManager에서 저장했던 이미지를 띄움

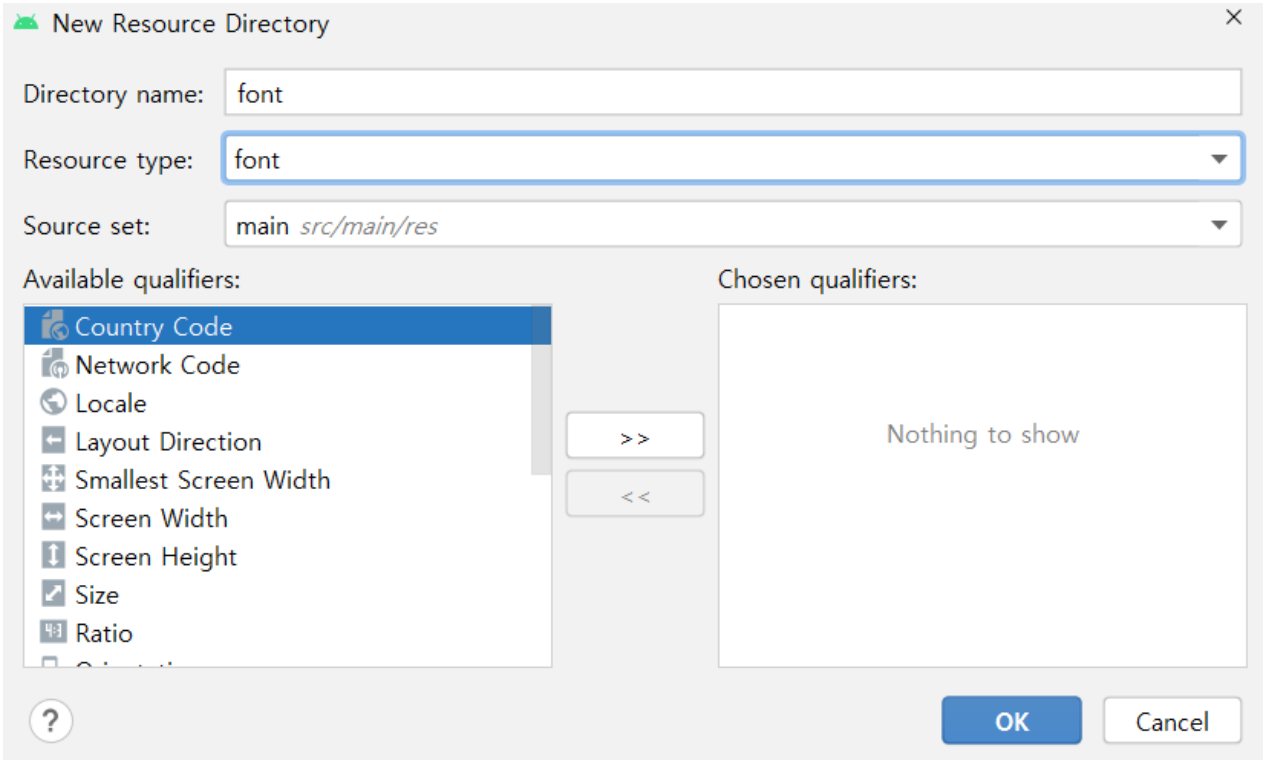
FinishActivity

```
String name=getIntent().getStringExtra("name");
int profile=getIntent().getIntExtra("img", 0);
imgProfile.setImageResource(profile);
tv_name.setText(name);
```

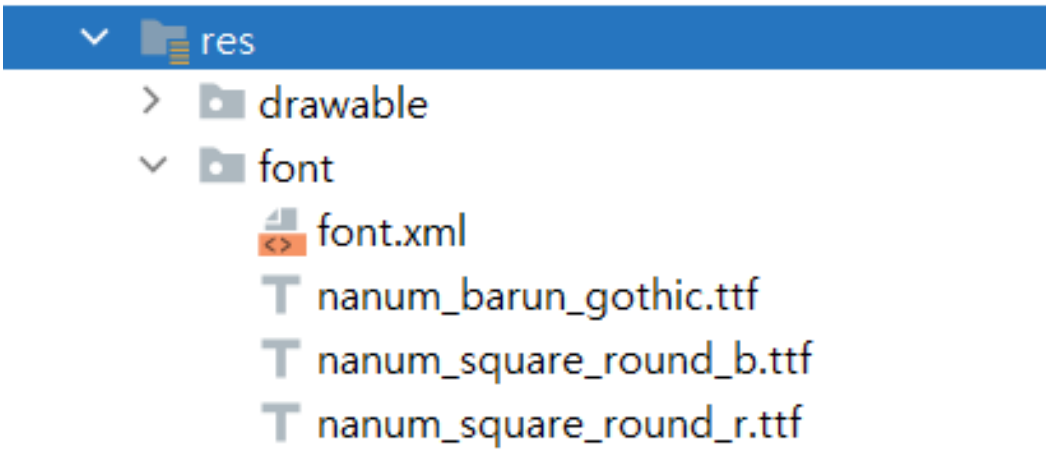
이미지 뷰와 텍스트 뷰에 받아온 결과를 출력

- 폰트 변경

res – New – Android Resource Directory – Resource type font로 설정 후 폴더 생성



Font 폴더에 원하는 폰트 파일 넣기



font

```
<font-family xmlns:android="http://schemas.android.com/apk/res/android">
    <font android:fontStyle="normal"
        android:fontWeight="400"
        android:font="@font/nanum_barun_gothic"/>
    <font android:fontStyle="normal"
        android:fontWeight="400"
        android:font="@font/nanum_square_round_r"/>
    <font android:fontStyle="normal"
        android:fontWeight="400"
        android:font="@font/nanum_square_round_b"/>
</font-family>
```

TextView 속성에 android:fontFamily="@font/nanum_square_round_b" 추가

```
<TextView
    android:id="@+id/select_kinds"
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:layout_gravity="center"
    android:gravity="center"
    android:text="아래 버튼을 눌러주세요"
    android:fontFamily="@font/nanum_square_round_b"
    android:textSize="23sp"
    android:textStyle="bold" />
```

그 외

- ListView

사용자가 정의한 데이터 목록을 아이템 단위로 구성하여 화면에 출력하는 ViewGroup의 한 종류로, 세로 방향으로 나열되며, 아이템의 개수가 많아짐에 따라 스크롤 기능을 사용해 ListView의 표시 기준 위치를 이동시킬 수 있다.

ListView에 사용자가 정의한 데이터를 표시하기 위해서는 Adapter를 사용해야 하는데, 어댑터(Adapter)는 사용자의 데이터를 받아 뷰(View)를 생성해주는 객체로 ListView와는 독립적으로 동작하는 객체이다.

- Fragment

액티비티 분할 가능, 다른 액티비티에서도 사용할 수 있어 재사용성이 뛰어남

반드시 하나의 액티비티 안에 소속되어야 하고, 독립적인 생명주기를 가짐.

onCreate() : 프래그먼트가 생성될 때 호출, 프래그먼트가 중지 혹은 정지된 후 재개될 때 보유하기 원하는 프래그먼트의 필수 컴포넌트을 초기화한다.

onCreateView() : Fragment에 실제 사용할 뷰를 만드는 작업을 하는 메소드로, LayoutInflater를 인자로 받아서 layout으로 설정한 XML을 연결하거나 bundle에 의한 작업을 하는 메소드이다.

onStart() : 호출되면 화면의 모든 UI가 만들어진 지고 호출이 된다.

onResume() : 호출되고 난 다음에 사용자와 Fragment와 상호작용이 가능하다. (사용자가 버튼을 누르거나 하는 이벤트를 받을 수 있게 됨)

onSaveInstanceState() : Activity와 동일하게 Fragment가 사라질때 현재의 상태를 저장하고 나중에 Fragment가 돌아오면 다시 저장한 내용을 사용할 수 있게 해주는 메소드이다.

onStop() : 호출되면 Fragment가 더이상 보이지 않는 상태이고 더이상 Activity에서 Fragment에게 오퍼레이션을 할 수 없게 된다.

- Kotlin

Java 문법과 약간의 차이가 있음

함수 선언 방법

1. fun sum(a: Int, b: Int): Int {
    return a + b
    }
2. fun sum2(a: Int, b: Int) = a + b

괄호 생략, return type 기재 생략, return문 생략

변수 선언 방법

1. 즉시 대입(타입선언)
   val a: Int = 1
2. 즉시 대입(타입체크)
   val b = 2
3. 선언 후 초기화
   val c: Int
   c = 3

조건식

1. 기본 조건식

```
fun maxOf(a: Int, b: Int): Int {
    if (a > b) {
        return a
    } else {
        return b
    }
}
```

2. return 생략 조건식

```
fun maxf2(a: Int, b: Int) = if(a > b) a else b
```

주석

// : 한 줄 주석

/* ~ */ : 여러 줄 주석