사용자 인터페이스 및 실습

최종 보고서

프로젝트 - HairMall

**사용자인터페이스실습-나반**

컴퓨터학부 20173605 김민정

컴퓨터학부 20180748 박소현

기계공학부 20141682 고동환

기계공학부 20131767 이한목

목차

1. 보고서 합본
2. 사용한 라이브러리
3. 구현여부 설명
4. Activity 설명
5. 힘들여 구현한 부분

프로젝트동영상링크: <https://www.youtube.com/watch?v=LjdC5RwH8Hk&feature=youtu.be>

1. **보고서 합본**

**사용자 인터페이스 실습(나반)**

**HairMall 2019.11.09 1주차 보고서**



**이한목 기계공학부 20131767**

-수정 내역

act11\_stylesearchingm (java) 내에서 작업 한 내용

package com.example.hairmall2.ui;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;

import android.Manifest;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.net.Uri;

import android.os.Bundle;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.LinearLayout;

import com.example.hairmall2.R;

import java.util.List;

public class act11\_stylesearchingm extends AppCompatActivity implements View.OnClickListener{

Button btn1;

ImageView imageView;

LinearLayout linearLayout;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.act11\_stylesearchingm);

btn1=findViewById(R.id.btnm); // 스타일 체험해보기 버튼의 주소.

linearLayout=findViewById(R.id.act11\_stylesearchingm); //// 가장 큰 레이아웃의 이름

btn1.setOnClickListener(this);

if(ContextCompat.checkSelfPermission(this, Manifest.permission.READ\_EXTERNAL\_STORAGE)!= PackageManager.PERMISSION\_GRANTED ||

ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE)!= PackageManager.PERMISSION\_GRANTED)

{

ActivityCompat.requestPermissions(this,new String[]{Manifest.permission.READ\_EXTERNAL\_STORAGE,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE}, 200);

}

}

@Override

public void onClick(View v) {

Matisse.from(this)

.choose(MimeType.of(MimeType.JPEG))

.countable(true)

.maxSelectable(1)

.spanCount(3)

.imageEngine(new GlideEngine())

.forResult(100);

}

@Override

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

super.onActivityResult(requestCode, resultCode, data);

if(requestCode==100 && resultCode == RESULT\_OK){

List<Uri> mSelected = Matisse.obtainResult(data);

for(Uri uri : mSelected){

ImageView imageView = new ImageView(this);

LinearLayout.LayoutParams params =

new LinearLayout.LayoutParams(ViewGroup.LayoutParams.WRAP\_CONTENT,

ViewGroup.LayoutParams.WRAP\_CONTENT);

linearLayout.addView(imageView,params);

Glide.with(this).load(uri).override(400,400).into(imageView);

}

}

}

}

act11\_stylesearchingm.xml 에서 작업한 내용

최상단 레이아웃에 아이디 추가

android:id="@+id/act11\_stylesearchingm"

버튼에 아이디 추가

android:id="@+id/btnm"

build.gradle(Modeule : hairmall2 에 아래 내용 추가

implementation 'com.github.bumptech.glide:glide:3.8.0'

implementation 'com.zhihu.android:matisse:0.5.2-beta4'

AndroidManifest.xml에 아래 내용 추가

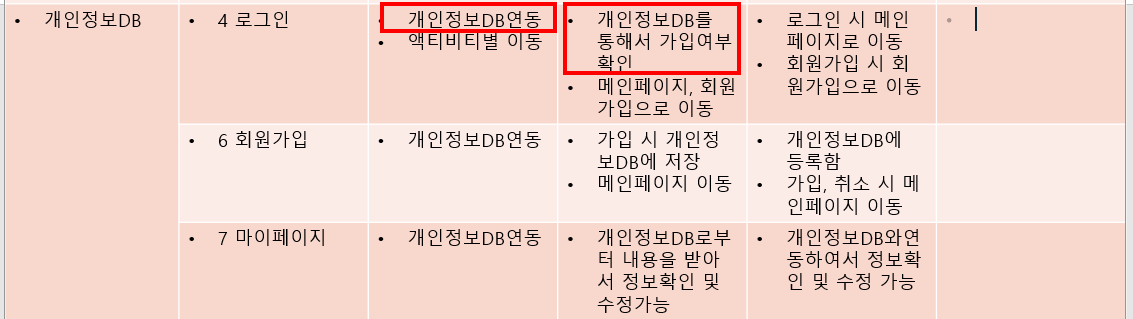
<!— hanmok added —>

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"/>

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

<!— hanmok added—>

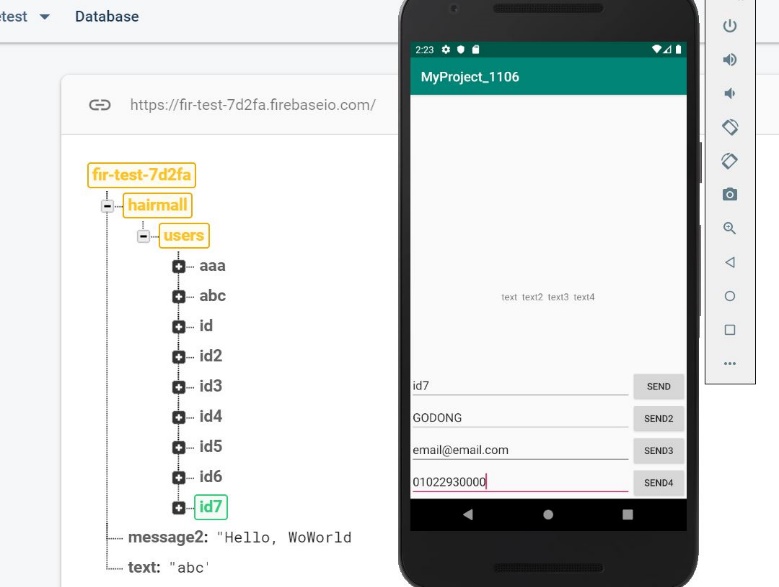
**고동환 기계공학부 20141682**



고동환 1주차 그림 1 역할 분담

(tab)개인정보DB - (depth1) 4 로그인 - (depth2) 개인정보DB 연동 진행중

(tab)개인정보DB - (depth1) 4 로그인 - (depth2) 개인정보DB - (depth3) 개인정보DB를 통해서 가입여부확인 진행중



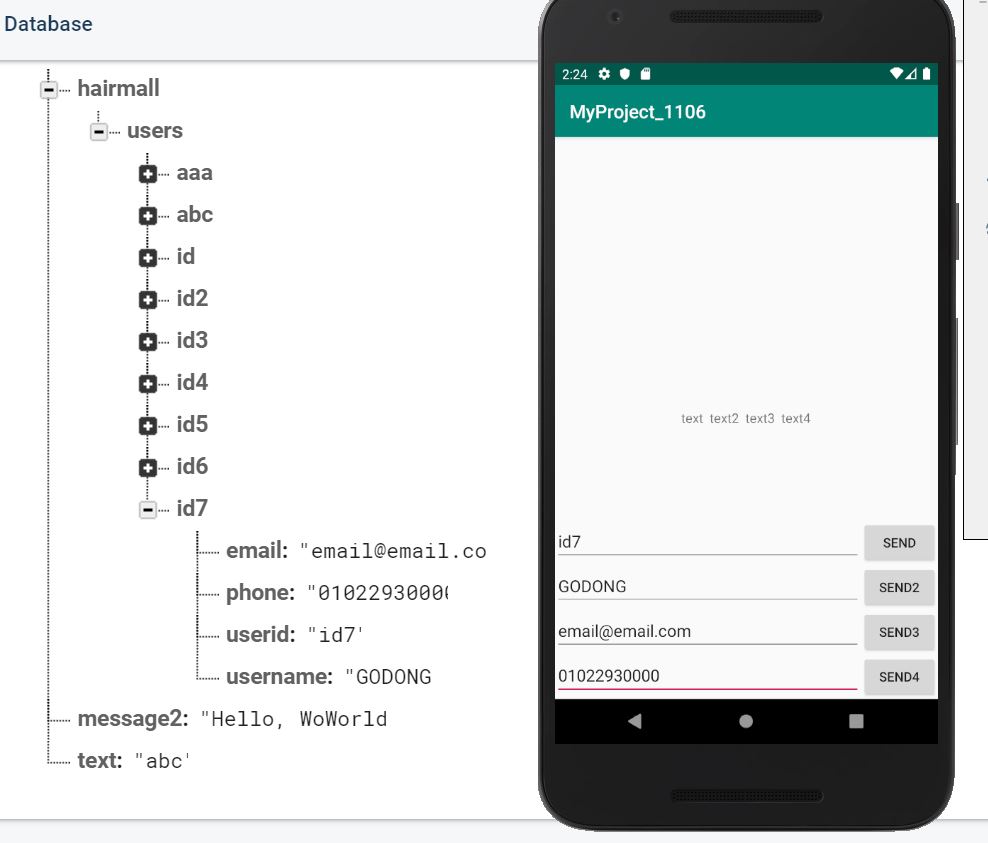
 그림2 firebase DB연동 샘플1

그림3 firebase DB연동 샘플2

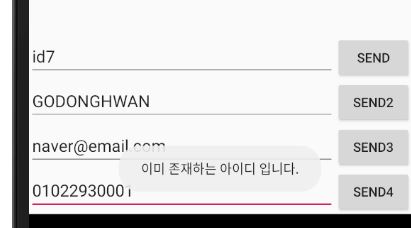


그림4 firebase DB연동 이미 존재하는 아이디 불가

그림 4에서는 firebase realtime database에 샘플로 만든 프로젝트로 회원정보를 입력하는 화면이다. 노란색 범례는 데이터가 변경됨을 의미하고 초록색 범례는 데이터가 추가됨을 의미한다. 그림 3 에서는 id7에 데이터가 제대로 들어가 있음이 보인다. 또한, 그림 4에서는 이미 존재하는 아이디는 다시 등록이 안되는 것을 볼 수 있다.

**김민정 컴퓨터학부 20173605**

직접 구현하려는 리뷰DB는 파이어베이스를 이용해서 구현하려한다.

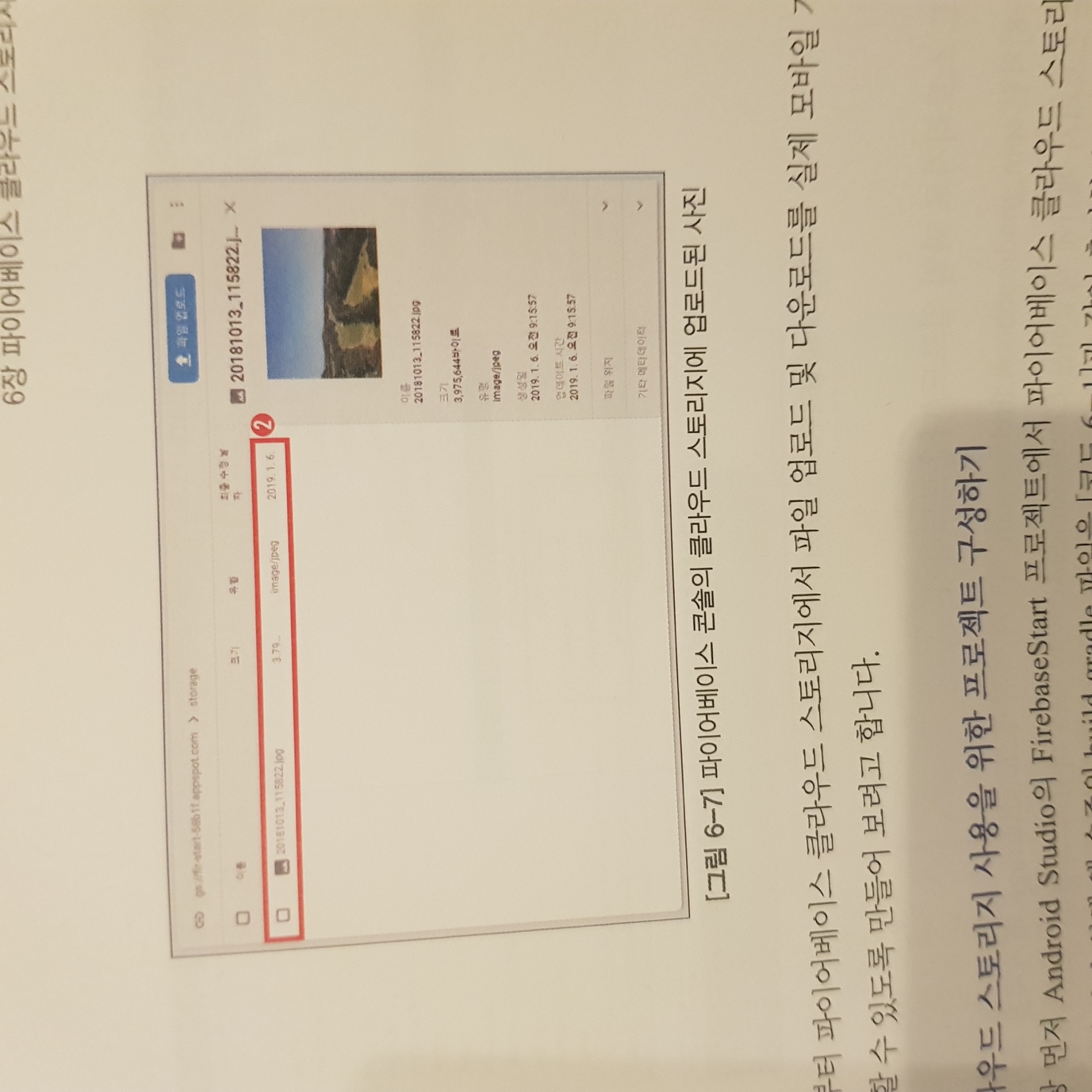
구글 파이어베이스서비스는 구글 애널리틱스와 구글 패브릭에서 제공하는 기능들을 포함한 다양한 기능들을 제공합니다. 파이어베이슨 개발자가 안드로이드,IOS등의 운영체제에 상관없이 앱을 만들 수 있도록 한 툴입니다. 따라서 저는 이번 앱을 만들때 파이어베이스를 이용하여 구현해보려 한다.

기존에 mysql을 다뤄봤기에 처음다뤄보는 파이어베이스라 여러 책을 참고하려 구현하려고 이번 1주차때는 다음사진에보이는 “Firebase로 안드로이드 SNS앱 만들기”를 참고하여 공부 하였다. 

직접 리뷰쓰고 사진올리고 하는 부분들이 sns앱만드는 과정과 비슷해서 참고하면 좋을 것 같았다.

먼저 Firebase를 사용하기에 앞서 파이어베이스 콘솔에 접근하여 firebase콘솔에서 제공하는 프로젝트 설정을하여 직접 안드로이드 프로젝트에 추가 해주었다.

파이어 베이스 실시간 테이터베이스는 클라우드 호스팅 데이터 베이스이다. 데이터는 JSON으로 저장되며 연결된 모든 클라이언트에 실시간으로 동기화한다. 따라서 기기가 오프라인 상태일 때도 로컬에 데이터를 저장하며, 기기가 온라인 상태가 되면 오프라인일 때도 로컬에 데이터를 저장하며, 기기가 온라인 상태가 되면 오프라인일 때 발생한 로컬데이터와 원격 데이터를 동기화하여 충돌이 발생되지 않고 안정적으로 데이터를 사용할 수 있도록한다. 이를 위해 파어베이스 콘솔에서 데이터베이스를 만들어 잠금 모드 등 설정하여 파이어베이스에 접근가능하도록 하였다.

아직 SQL 문에 더 익숙해서 firebase에 빨리 친숙해질 수 있도록 이번주말까지 튜토리얼을 통해 익힌 뒤 2주차부터 바로 구현해보려한다.

🡪간단히 사진업로드하는 것부터 따라해보았다.

**박소현 컴퓨터학부 20180748**

<act6\_shopsearching>

import androidx.fragment.app.Fragment;

import androidx.lifecycle.Observer;

import androidx.lifecycle.ViewModelProviders;

import com.example.hairmall2.R;

<act1\_mainpage>

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.Nullable;

import androidx.annotation.NonNull;

import androidx.fragment.app.Fragment;

import androidx.lifecycle.Observer;

import androidx.lifecycle.ViewModelProviders;

import com.example.hairmall2.R;

<act11\_stylesearching>

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.Nullable;

import androidx.annotation.NonNull;

import androidx.fragment.app.Fragment;

import androidx.lifecycle.Observer;

import androidx.lifecycle.ViewModelProviders;

import com.example.hairmall2.R;

<act13\_opentalk>

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.Nullable;

import androidx.annotation.NonNull;

import androidx.fragment.app.Fragment;

import androidx.lifecycle.Observer;

import androidx.lifecycle.ViewModelProviders;

import com.example.hairmall2.R;

<**헤어몰2~3주차보고서>**

**20141682 기계공학부 고동환**

스크린샷이(가) 표시된 사진

자동 생성된 설명

고동환 2주차 그림 1 역할분담

 (tab)개인정보DB - (depth1) 3 로그인 - (depth2) 개인정보DB 연동, 액티비티별 이동 - (depth3) 개인정보DB를 통해서 가입여부확인, 메인페이지, 회원가입으로 이동

스크린샷이(가) 표시된 사진

자동 생성된 설명 (tab)개인정보DB - (depth1) 4 회원가입 - (depth2) 개인정보DB 연동 - (depth3) 가입 시 개인정보DB에 저장, 메인페이지 이동

그림 2 이미 존재하는 id

 2주차에는 act4\_join(회원가입)에서 firebase가 가능하도록 연동하였다. 그림 2에서 가입 시 이미 존재하는 id가 있을 경우 가입이 되지 않는다.

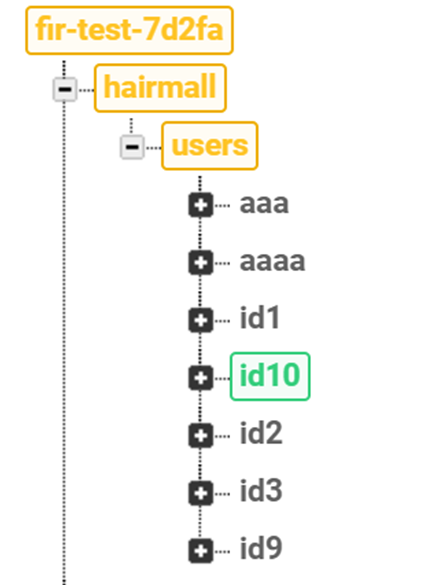


그림 3 id 추가

  그림 3은 act4\_join(4 회원가입)을 통해서 id를 추가하는 순간이다. 초록색으로 추가되고 있음을 보인다.



그림 4 id추가2

 그림 4는 추가된 내용을 보여주고 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림5 존재하지 않는 아이디

스크린샷이(가) 표시된 사진

자동 생성된 설명 그림 5에서는 act3\_login(로그인)에서 존재하지 않는 아이디로 로그인 할 경우에 뜨는 메시지이다.

그림 6 비밀번호 불일치

 그림 6에서는 act3\_login(로그인)에서 아이디는 존재하나 비밀번호가 일치하지 않을 경우에 뜨는 메시지이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

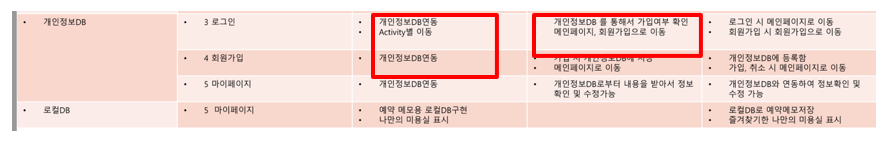
그림7 메인페이지로 이동

 act3\_login(3 로그인)에서 로그인을 성공하거나 act4\_join(6 회원가입)에서 가입이 성공할 경우에 메인페이지로 이동한다.

 \*맨 위에 textView, editText, Button은 간단하게 왼쪽 텍스트는 db를 경유해서 입력한 텍스트가 바뀌게 되는 것이고 오른쪽 버튼은 id를 액티비티끼리 intent로 보내주어서 현재id를 출력하도록 하는 디버그용이다.

 \*login을 Manifest.xml에서 MAIN으로 지정해서 실행 시 가장 먼저 켜지도록 하였다.

이유는 로그인이나 회원가입 후에 id를 각 액티비티에게 전달해주면서 시작하는데 그렇지 못할 경우에는 에러가 발생한다.

20141682 고동환 3주차 보고서

 고동환 3주차 그림 1 역할분담

 (tab)개인정보DB - (depth1) 5 마이페이지 - (depth2) 개인정보DB연동 - (depth3) 개인정보DB로부터 내용을 받아서 정보 확인 및 수정가능

스크린샷이(가) 표시된 사진

자동 생성된 설명 (tab)로컬DB - (depth1) 5 마이페이지 - (depth2) 예약 메모용 로컬DB구현

고동환 3주차 그림 2 mypage firebase 연동

 그림 2는 3주차에서 act5\_mypage(5 회원가입)에서 firebase가 가능하도록 연동한 모습이다. 로그인을 하고 들어오면 기존 정보가 나타난다. 가입 시에는 예약에 대한 메모정보가 있지 않아서 공백이다. firebase에도 나타나지 않는다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

고동환 3주차 그림 3 mypage firebase 연동\_시간 수정연동

그림 3은 시간정보를 수정하였을 때 firebase에 연동되어서 입력되는 모습이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

고동환 3주차 그림 4 mypage firebase 연동\_개인정보 수정연동

 그림 4는 개인정보를 수정하였을 때 firebase에 연동되어서 입력되는 모습이다.

 \*password변경 시 act3\_login의 ‘비밀번호가 일치하지 않습니다’하는 토스트 메세지가 뜨는 버그가 있다. 비밀번호는 정상적으로 변경되어서 저장된다.

**20173605 컴퓨터학부 김민정**

1. 헤어몰 어플아이콘변경하기
2. 엑티비티들 정리해서 메인만 뜨게 만들어주기
3. Firebase환경설정
4. <3주차>
5. 메인페이지에서 최근리뷰에 맞는 이미지 클릭시 그와 연관된 리뷰텍스트로 전환되게 설정해주기
6. Firebase환경설정 해주기2
7. 연습용 DB구축해서 흐름 파악하기(firestore사용)

**20180748 컴퓨터학부 박소현**

1. 메뉴바의 사람이미지 누르면 로그인 화면으로 이동하도록 구현

2. 메뉴바의 마이페이지 버튼 누르면 마이페이지 화면으로 이동 구현

3. 샵둘러보기에 지노헤어 이미지 누르면 지노헤어 가게 상세페이지로 이동하도록 구현

**20131767 기계공학부 이한목**

1. 헤어스타일 검색 액티비티에서 스타일 착용해보기 버튼 클릭시 갤러리와 연동기능 .

2. 가게 둘러보기에서 지도 이미지를 누를 시 지도 연동.

**20141682 고동환 보고서 4주차**



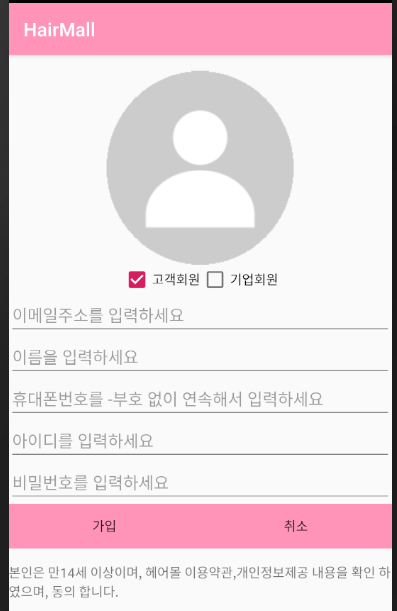
(tab)개인정보DB - (depth1) 4 회원가입 - (depth2) 개인정보DB연동 - (depth3) 개인정보DB로부터 내용을 받아서 정보 확인 및 수정가능

(tab)로컬DB - (depth1) 5 마이페이지 - (depth2) 예약 메모용 로컬DB구현

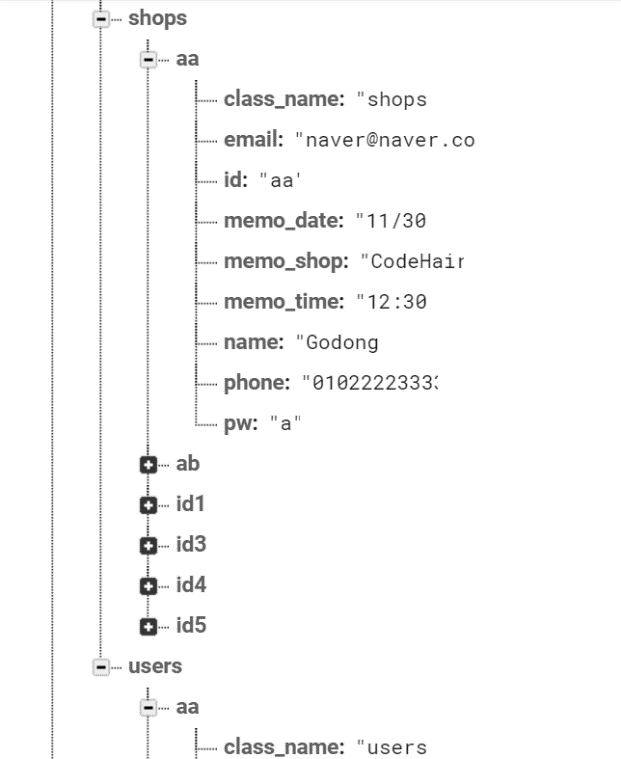
(tab)로컬DB - (depth1) 5 마이페이지 - (depth2) ‘나만의 미용실 표시’를 위해서 기업회원가입 추가(추후 진행 필요)

이번주에 작업한 내용은 3주차와 크게 다르지 않다. 나만의 미용실 표시를 하기 위해서 기업회원의 id와 정보가 필요하다고 판단되어서 기존의 개인회원과 더불어 기업회원의 회원가입을 추가하였다.

고동환 4주차 그림 1 로그인 고객회원 기업회원 선택

그림 1에서는 act3\_login에 개인회원과 기업회원 체크박스가 생겼음을 볼 수 있다.

고동환 4주차 그림 2 회원가입 고객회원 기업회원 선택

그림 2에서는 act4\_join에 고객회원과 기업회원 체크박스가 생겼음을 볼 수 있다.

고동환 4주차 그림 3 Firebase Database 고객회원 기업회원 분류저장

그림 3에서는 Firebase realtime Database에 shops와 users로 기업회원과 고객회원이 나뉘어서 저장되고 있음을 보여주고 있다.

3주차와 같이 개인정보를 수정할 수 있는 act5\_mypage는 생략한다. 기업회원에 대한 추가적인 기능은 아직 없으며, 나만의 미용실을 표시하기에 앞서 기업id를 만들 수 있도록 하였다.

**20131767 기계공학부 이한목**

헤어스타일 검색 액티비티에서 각 머리스타일 버튼 클릭시 하단 사진들 변경

**20173605 컴퓨터학부 김민정**

메인페이지에서 오늘의 미용실 클릭시 해당 텍스트 나오고 테두리처리 해주기

**20180748 컴퓨터학부 박소현**

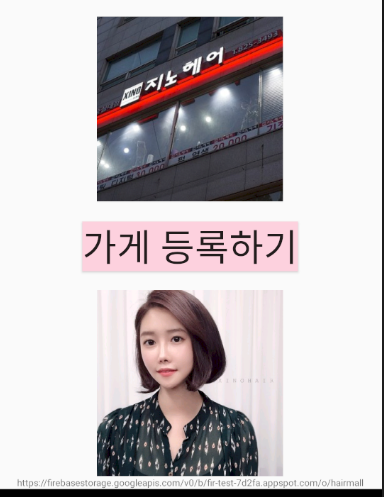
리안헤어, 이철헤어커커, 박승철 헤어스튜디오 가게 상세페이지 제작

**<헤어몰 5주차 보고서>**

**20141682 고동환**

\*시간 관계상 merge를 못해서 소스코드를 따로 올림

고동환 5주차 그림 1

 그림 1에서는 기업회원 가게 작성 가능

고동환 5주차 그림 2

그림 2에서는 밑에 사진이 firestorage에 연동되었고 이를 glide로 보여주는 사진이고 그 밑에 다운로드 uri가 표시됨. Uri를 이제 firebase realtime database에 연동해서 즐겨찾기 리뷰를 위한 사진을 glide로 mypage불러올 예정인데 아직 구현 못함.

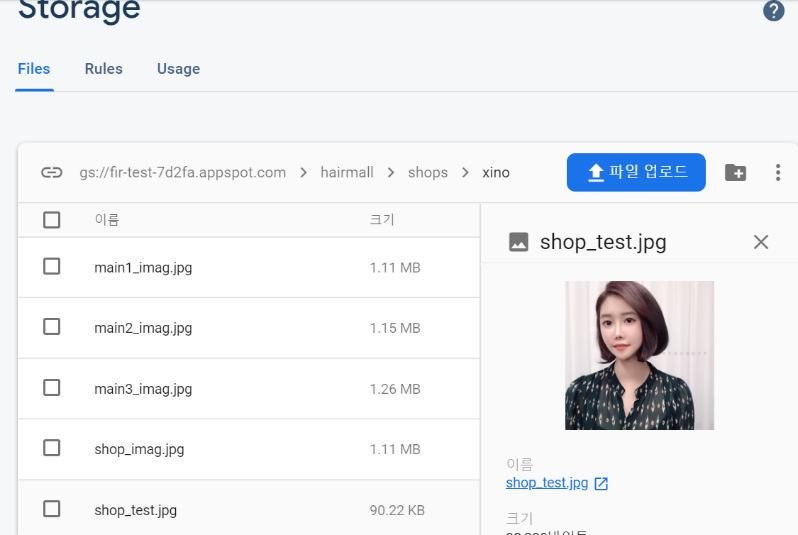
고동환 5주차 그림3

그림 3은 실제 storage에 가게 id별로 저장되고 있음을 보여준다. 현재 storage에 저장하기 uri값 가져오기 glide활용해서 보여주기가 가능하게 되었고 realtime database는 기존에 로그인에서 이용하였듯이 이용하면 되기 때문에 기술적인 부분은 어느정도 진행되었다.



표 1 일정표 (한목씨 최종 일정표 ppt 공유바람니다! 그걸로 합쳐주세요)

약식으로 서술

개인정보DB – 3 로그인 – 개인정보 DB연동 Activity별 이동 – 개인정보Db를 통해서 가입여부 확인 메인페이지 회원가입으로 이동 구현

개인정보DB - 4 회원가입 개인정보 DB연동 가입 시 개인정보 DB에 저장 메인페이지로 이동 구현

로컬DB – 마이페이지 – 예약 메모용 로컬 DB구현 – firebase로 구현

로컬DB – 마이페이지 – 나만의 미용실 표시 – 미구현이고 구현중이고 기술적인 부분이 어느정도 진행된 상태임

**20173605 김민정**

최근 리뷰와 최근 미용실 한번더 클릭시 해당페이지로 이동 구현

\*\*\*\*\*\*\*\*깃정리가 안되서 아직 마스터에 푸쉬못함 따라서 보고서에 간략하게 추가코드부분을 올림\*\*\*\*\*\*다음주중으로 모든DB구축 완성예정!!구현중에 있음

리뷰작성액티비티에서 취소시 shopsearching액티비티로 이동구현

//리뷰 텍스트

final TextView reviewText = root.findViewById(R.id.review\_text);

//오늘의 미용실 텍스트

final TextView shopText = root.findViewById(R.id.shop\_text);

final ImageView[] reviewImage = new ImageView[3];

final ImageView[] shopImage = new ImageView[3];

reviewImage[0] = root.findViewById(R.id.recent\_review1);

reviewImage[1] = root.findViewById(R.id.recent\_review2);

reviewImage[2] = root.findViewById(R.id.recent\_review3);

shopImage[0] = root.findViewById(R.id.today\_shop1);

shopImage[1] = root.findViewById(R.id.today\_shop2);

shopImage[2] = root.findViewById(R.id.today\_shop3);

testText[0]="1번 리뷰";

testText[1]="2번 리뷰";

}

}

});

shopImage[i].setOnClickListener(new KnowIndexOnClickListener(i) {

public void onClick(View v) {

shopText.setText(testText[index]);

shopImage[index].setPadding(4, 4, 4, 4);

shopImage[index].setBackground(ContextCompat.getDrawable(context, R.drawable.borderline));

if (index == 0) {

shopImage[1].setPadding(0, 0, 0, 0);

shopImage[2].setPadding(0, 0, 0, 0);

} else if (index == 1) {

shopImage[0].setPadding(0, 0, 0, 0);

shopImage[2].setPadding(0, 0, 0, 0);

} else if (index == 2) {

shopImage[0].setPadding(0, 0, 0, 0);

shopImage[1].setPadding(0, 0, 0, 0);

}

}

});

}

return root;

}

**20180748 박소현**

1. act11에서 man 버튼 누르면 act11\_stylesearchingm으로 이동,

woman 버튼 누르면 act11\_stylesearching으로 이동하도록 구현

2. shopsearching activity에서 리안헤어, 이철헤어커커, 박승철헤어스튜디오 사진 누르면 각각 act8\_shopdetail\_2,3,4로 이동하도록 구현

**20131767 이한목**

Act 7 지도 미용실 여러개 넣는것과 Act 11 의 카메라 연동 진행중. 지도는 끝났지만 카메라는 아직 작업중.

2. 사용한 외부 라이브러리 목록

2.1 import

**import** android.content.ComponentName;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.Button;  
**import** android.widget.ImageView;

**import** android.widget.TextView;

**import** androidx.annotation.Nullable;

**import** androidx.annotation.NonNull;

**import** androidx.fragment.app.Fragment;

**import** androidx.lifecycle.Observer;

**import** androidx.lifecycle.ViewModelProviders;

**import** com.example.hairmall2.MapsActivity;

**import** com.example.hairmall2.R;

**import** androidx.lifecycle.LiveData;

**import** androidx.lifecycle.MutableLiveData;

**import** androidx.lifecycle.ViewModel;

**import** android.content.Context;  
**import** android.util.Log;

**import** android.widget.EditText;   
**import** androidx.core.content.ContextCompat;  
**import** com.example.hairmall2.KnowIndexOnClickListener;

**import** com.google.android.gms.tasks.OnCompleteListener;

**import** com.google.android.gms.tasks.Task;

**import** com.google.firebase.database.DataSnapshot;

**import** com.google.firebase.database.DatabaseError;

**import** com.google.firebase.database.DatabaseReference;

**import** com.google.firebase.database.FirebaseDatabase;

**import** com.google.firebase.database.ValueEventListener;

**import** com.google.firebase.firestore.FirebaseFirestore;

**import** com.google.firebase.firestore.Query;  
**import** com.google.firebase.firestore.QueryDocumentSnapshot;  
**import** com.google.firebase.firestore.QuerySnapshot;

**import** java.util.ArrayList;

**import** androidx.appcompat.app.AppCompatActivity;

**import static** android.view.View.***VISIBLE***;

**import** android.net.Uri;  
**import** android.widget.CheckBox;

**import** android.widget.Toast;

**import** java.util.HashMap;

**import** java.util.Map;

**import** android.graphics.Bitmap;  
**import** android.graphics.BitmapFactory;  
**import** android.graphics.drawable.BitmapDrawable;  
**import** com.bumptech.glide.Glide;  
**import** com.example.hairmall2.shop;  
**import** com.google.android.gms.tasks.Continuation;

**import** com.google.android.gms.tasks.OnFailureListener;

**import** com.google.android.gms.tasks.OnSuccessListener;  
**import** com.google.firebase.storage.FirebaseStorage;  
**import** com.google.firebase.storage.StorageReference;  
**import** com.google.firebase.storage.UploadTask;  
**import** java.io.ByteArrayOutputStream;  
**import** java.io.FileNotFoundException;  
**import** java.io.InputStream;  
**import** android.content.ClipData;  
**import** com.google.firebase.firestore.DocumentReference;  
**import** java.text.SimpleDateFormat;  
**import** java.util.Date;  
**import** android.view.Menu;  
**import** androidx.appcompat.widget.Toolbar;  
**import** androidx.drawerlayout.widget.DrawerLayout;  
**import** androidx.navigation.NavController;  
**import** androidx.navigation.Navigation;  
**import** androidx.navigation.ui.AppBarConfiguration;  
**import** androidx.navigation.ui.NavigationUI;  
**import** com.google.android.material.floatingactionbutton.FloatingActionButton;  
**import** com.google.android.material.navigation.NavigationView;

**import** androidx.fragment.app.FragmentActivity;  
**import** com.google.android.gms.maps.CameraUpdateFactory;  
**import** com.google.android.gms.maps.GoogleMap;  
**import** com.google.android.gms.maps.OnMapReadyCallback;  
**import** com.google.android.gms.maps.SupportMapFragment;  
**import** com.google.android.gms.maps.model.LatLng;  
**import** com.google.android.gms.maps.model.Marker;  
**import** com.google.android.gms.maps.model.MarkerOptions;  
**import** android.provider.MediaStore;  
**import** android.view.GestureDetector;  
**import** android.view.MotionEvent;  
**import** android.view.ScaleGestureDetector;  
**import** android.widget.ImageButton;  
**import** android.widget.RelativeLayout;  
**import** java.io.IOException;

**import** com.google.firebase.database.Exclude;  
**import** com.google.firebase.database.IgnoreExtraProperties;

2.2 implemetation

implementation 'com.google.firebase:firebase-analytics:17.2.1'

implementation 'com.google.firebase:firebase-database:19.2.0' //실시간 데이터베이스

implementation 'com.google.firebase:firebase-core:17.2.1'

implementation 'com.google.firebase:firebase-firestore:21.3.1' // cloud 데이터베이스

implementation fileTree(dir: 'libs', include: ['\*.jar'])

implementation 'com.google.firebase:firebase-storage:19.1.0' // 가게 이미지 저장

implementation 'com.github.bumptech.glide:glide:4.10.0'

implementation 'com.android.support:appcompat-v7:28.0.0-rc03' // recyclerView 이용

implementation 'androidx.appcompat:appcompat:1.1.0'

implementation 'com.google.android.material:material:1.0.0'

implementation 'androidx.constraintlayout:constraintlayout:1.1.3'

implementation 'androidx.navigation:navigation-fragment:2.1.0' // 지도 네비게이션

implementation 'androidx.navigation:navigation-ui:2.1.0' // 지도 네비게이션

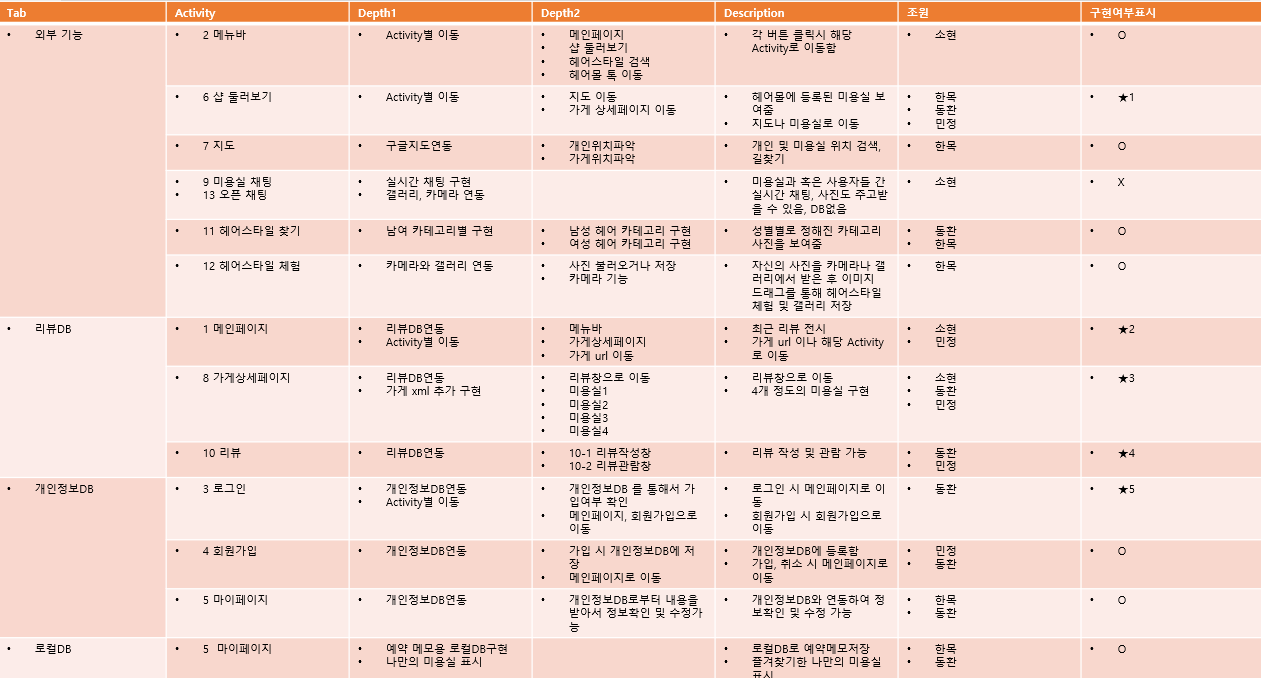
implementation 'androidx.lifecycle:lifecycle-extensions:2.1.0'

testImplementation 'junit:junit:4.12'

androidTestImplementation 'androidx.test.ext:junit:1.1.1'

androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'

3. 구현여부 설명



★1(샵 둘러보기):

완전하게 DB랑 연동은 되지 않고 기존의 drawable 있는 자원을 사용한다. 하지만 DB랑 부분 연결은 되어있는데 즐겨찾기 추가의 경우 Firebase realtime DB와 연동되어 있다.

★2(메인페이지):

레이아웃의 구성이 완전하게 DB랑 연동되지는 않고 기존의 drawable 있는 자원을 사용한다. 하지만 DB랑 부분 연결은 되어있는데 리뷰의 경우 FirebaseCloud 데이터베이스와 연동되어 있다.

★3(가게상세페이지):

완전하게 DB랑 연동은 되지 않고 기존의 drawable 있는 자원을 사용한다. 하지만 DB랑 부분 연결은 되어있는데 즐겨찾기 추가의 경우 Firebase realtime DB와 연동되어 있다.

★4(리뷰):

이미지 저장은 연동되지 않지만 텍스트는 FirebaseCloud 데이터베이스와 연동되어서 작성 시 메인화면에서 볼 수 있다.

★5(로그인):

일부 세세한 기능 등이 구현되지 않았지만 Firebase realtime DB통해 가게와 유저로 나누어 가입할 수 있으며 아이디 여부와 비밀번호 일치여부를 판단할 수 있는 등 계획서에 나와있는 대로 핵심적인 구현은 완료되었다.

4. Activity 설명

Act 1 메인페이지(MainPage)

새로운 리뷰와 추천미용실을 볼 수 있는 페이지이다. 리뷰와 미용실은 3개씩 화면에 출력되며 각 리뷰 또는 미용실 클릭 시 리뷰에 대한, 또는 미용실에 대한 정보가 이미지 아래에 나타난다.

리뷰와 미용실 각각 한번 터치시 해당이미지와 관련된 정보들이 텍스트뷰로 나타난다.

이때 리뷰는 최신작성한 리뷰순서별로 나오게 구현해두었다. 가게 정보는 원래 있던 데이터정보에서 가져오게 하였다.

Act 2 메뉴바(MenuBar)

여러 액티비티로 이동하게 하는 액티비티이다. Act1 메인페이지, Act5 마이페이지, Act6 샵 둘러보기, Act11 헤어스타일 검색, 헤어몰 톡 액티비티로 이동가능하다.

Act 3 로그인(Login)

로그인, 아이디찾기, 비밀번호찾기 및 비회원 로그인으로 이동할 수 있는 액티비티이다. 이미 회원가입을 했을 시 아이디 및 패스워드를 클릭 후 로그인하면 메인페이지로 이동한다.

Act 4 회원가입(Join)

이메일주소, 이름, 휴대폰번호, 아이디, 비밀번호를 입력시 가입할 수 있다. 가입하면 자동으로 로그인되어 메인페이지로 이동하며, 자료들은 모두 Firebase에 저장된다.

Act 5 마이페이지(MyPage)

마이페이지에서는 이메일주소, 이름, 휴대폰번호, 비밀번호를 수정할 수 있으며 예약메모를 작성하여서 realtime DB에 저장할 수 있다. 또한 가게의 메인 이미지를 firestorage에 넣고 받아온 uri를 realtime DB의 샵의 위치에 넣어두고 recyclerView에 담아서 보여줄 수 있다.

Act 6 샵 둘러보기(shopsearching)

샵 둘러보기는 미리 설정된 4개의 가게를 보여준다. 여기서 즐겨찾기는 realtime DB에 연동되어 있으며 각 이미지를 클릭 시 해당 리뷰 창으로 들어간다. 여기서 구글지도로 이동할 수 도 있다.

Act 7 지도(MapsActivity)

어플에 등록된 헤어샾 4개에 대하여 그 미용실들의 좌표를 입력하였고, 전화번호는 설정, 미용실이름 클릭 시 인텐트를 이용하여 다이얼 앱으로 이동하게하였다. Zoom을 15.0f 로 설정하여 멀리서도 많은 미용실들을 한눈에 볼 수 있으며 구글 길찾기를 통해 쉽게 원하는 미용실로 이동 가능하다.

Act 8 가게 상세페이지(shopDetail)

해당 가게 페이지는 전화버튼시 전화 엑티비티가 뜨며 가게 번호를 인탠트로 전달하기때문에 바로 가게로 전화가능하다. 채팅문의버튼은 채팅액티비티로 이동하며 리뷰작성은 리뷰작성액티비티로 이동한다. 별표를 누르면 realtime DB에 즐겨찾기한 가게 이름이 저장되어 이로부터 가게의 이름을 알아낼 수 있다. 이로부터 해당 가게의 메인 이미지 uri를 받아와서 mypage에서 즐겨찾기로 볼 수 있게 된다.

Act 10 리뷰(Review)

리뷰작성 시 firebase Cloud에 정보가 올라가며 그에 맞게 평점과 헤어스타일 정보도 들어간다. 나중에 메인페이지에 리뷰내용이 DB를 통해 나타난다. 또한 메인페이지에서 최신 리뷰별로 리뷰를 나타내기 위해 리뷰작성시 시간데이터까지 저장해준다.

갤러리에서 최대3개의 사진을 다중 선택하여 가져오게 만들었다. 저장 버튼을 누르면 해당 DB로 가고(이미지 불러오기 까지 구현), 취소 버튼을 누르면 그전 가게 상세페이지로 이동한다.

Act 11 헤어스타일 찾기(stylesearching)

스타일 찾기 액티비티이다. 성별을 선택 가능하며 각 성별 당 6개의 헤어스타일, 그리고 각 헤어스타일당 4개의 사진을 보여준다. 성별과 헤어스타일은 버튼으로 설정하였고, 그리하여 총 12개의 화면을 제공한다. 또한, 이 액티비티에서 ‘스타일 체험해보기’ 버튼 클릭 시 Act12 헤어스타일 체험으로 이동한다.

Act 12 스타일 체험하기(puthairstyles)

스타일 체험하기 액티비티이다. 좌측 상단 버튼 클릭 시 사진을 갤러리로부터 가져올 수 있으며, 이 사진에 다양한 헤어스타일을 입혀볼 수 있다. Act 11 스타일 찾기 액티비티와 마찬가지로 성별을 선택 가능하며, 각 성별당 4개의 헤어스타일을 제공한다. 모든 헤어스타일은 하단에 목록으로 나와있으며, 각 목록은 버튼이다. 이 버튼들을 클릭 시 해당 헤어스타일이 출현하며 드래그하면 사진에 가상으로 착용이 가능하다. 버튼을 다시 누르면 해당 헤어스타일이 보이지 않게 된다.

헤어스타일을 착용중에 다른 헤어스타일을 클릭하면 기존 스타일이 있던 자리에 새로 누른 스타일이 그대로 위치하게 된다.

5. 힘들여 구현한 부분

1. 리뷰작성시 평점, 리뷰내용, 스타일정보등 저장하여 task성공시 firebase Cloud에서 shop\_name기준으로 정보를 가져온다. 또한, collection에서 리뷰작성할때의 시간을 넣어주는 것으로 date\_day변수에 저장하여 qurery문으로 내림차순정리하여 최신리뷰들이 메인페이지에서 해당 리뷰이미지 클릭시에 평점과 리뷰가 뜨게 구현하였다.

2. mypage의 즐겨찾기는 recyclerView를 이용하였으며 그 전에 가게작성하기로부터 메인 이미지를 받아와서 firestore에 저장한 뒤 uri를 즉시 샵DB에 저장하여서 shopdetail에서 유저의 별표 클릭을 감지하면 유저 DB에 가게이름을 저장한 뒤에 mypage에서 즐겨찾기를 보여주기 위해 유저 DB에서 순서대로 가게 이름을 가져오며 그 때에 가게 DB 함수 호출로 연속해서 들어가서 uri를 가져온 후에 recyclerView에 가게 이름+" "+가게 uri를 주면 쪼개서 가게 uri는 glide로 보여주고 가게 아이디는 밑에 보여준다.

3. 가게작성하기는 일정표에 없었으나 작성하였다. 가게 id로 들어가면 mypage에서 볼 수 있으며 가게 메인 image를 firebaseStore에 저장할 수 있으며 uri를 받아와서 가게DB에 저장할 수 있다. 다른 이미지들도 저장하여 glide로 즉시 보여줄 수도 있다. 2번을 하기 위한 전제조건이다.