기술 문서

작성일시	2023년 12월 20일			작성자	송득모, 임준혁
팀원	송득모, 최효진, 임준혁, 김동현(팀장)				

핵심 기술	1. 카카오 지도 API 2. 카카오 로그인 API 3. 사용자들의 실시간 정보 공유를 위한 Firebase DB사용
세부 구현	1. LoginActivity - 로그인 액티비티 2. MainActivity - 메인(홈) 지도 액티비티 3. SetPlaceActivity - 장소 등록 액티비티 4. ShowPlaceActivity - 장소 상세 정보 액티비티 5. MyPageActivity - 마이 페이지 액티비티

	내용	비고
핵심 기술	**1. 카카오 지도 API** 지도외전 어플리케이션의 가장 기본이 되었던 것이 카카오 지도 API이다. 카카오 지도 API를 통해 지도를 앱에 구현하여 원하는 지도에 추가적인 위치 정보들을 추가할 수 있다. 기본적으로 카카오 지도 API를 사용하기 위해서는 카카오 개발자 포털에서 앱을 생성하고 API 키를 발급받아야 한다. 발급받은 키는 앱에서 지도 서비스에 접근하기 위한 인증 수단으로 사용된다. 카카오 지도 API 기능 중에서도 주요하게 사용된 두 가지 기능은 다음과 같다. 첫 번째는 MapView 클래스이다. 지도를 불러오고 화면을 표시하는 제일 기본적인 기능이 MapView 클래스를 활용하여 제작된다. 지도외전은 이것을 바탕으로 현재 위치로 돌아오는 메소드, 등록하고자 하는 장소의 위도, 경도 등을 받아오는 메소드, 위치의 도로명 주소를 검색하는 메소드 등을 활용한다. 두 번째는 MapPOIItem 클래스이다. MapPOIItem 클래스는 흔히 말하는 지도의 마커(핀)의 기능을 제공한다. 마커를 추가하고 읽고, 삭제하고 수정하는 기본적인 기능을 활용한다. 지도외전이 제공하는 마커의 종류는 현재 여섯 가지가 있다. 쓰레기통, 자판기, 붕어빵 가게, 의류 수거함, 철봉, 흡연장으로, 이것들은 기존 지도 앱에서는 위치를 제공하지 않고 있다. 지도외전은 MapPOIItem 클래스를 활용하여, 앱 이용자들이 이 장소들의 사진을 직접 찍어 등록하고, 평점과 리뷰, 즐겨찾기 기능을 추가하여 기존 지도 앱이 제공하지 않는 장소들을 쉽게 탐색하고 이용할 수 있도록 한다. **2. 카카오 로그인 API** 지도외전 어플리케이션을 이용하기 위해서 개인 정보를 식별한다. 카카오 로그인 API를 통해 자체적으로 회원 정보를 받지 않고 이미 구현되어 있는 기능을 이용함으로써 시간을 절약하고 실질적인 기능 구현에 집중할 수 있었다. 카카오(다음)에 회원 가입이 되어 있는 사용자라면 누구나 쉽고 빠르게 앱을	

별도의 다른 절차 없이 로그인만 하여 이용할 수 있다. 한번 로그인한 사용자라면 토큰 정보를 인식하여 번거롭게 계속 로그인 할 필요도 없다.

3. 사용자들의 실시간 정보 공유를 위한 Firebase DB 사용

Firebase는 실시간 데이터 베이스로 실시간 업데이트와 이벤트 처리가 가능한 구글 제공 플랫폼으로, Android 스튜디오와 연동하여 사용할 수 있다. Firebase에 저장한 컬렉션에는 reviews, sampleMarker, users 컬렉션이 있다.

Firebase에 저장되어 있는 컬렉션들은 애플리케이션을 종료하고 다시 실행해도 사라지지 않고, 저장되어 있다가 불러올 정보들이다.

reviews 컬렉션은 사용자 후기가 저장된다. reviews 컬렉션에는 리뷰가 등록된 장소의 markerID와 리뷰 목록을 저장하는 reviewList가 있다. reviewList에는 리뷰 내용인 content, 생성 날짜인 createdDate, 리뷰 작성자의 userId와 userNickName이 저장된다. 이 컬렉션에 담긴 정보를 이용하여 다른 이용자들이 올린 리뷰를 앱에서 확인할 수 있다.

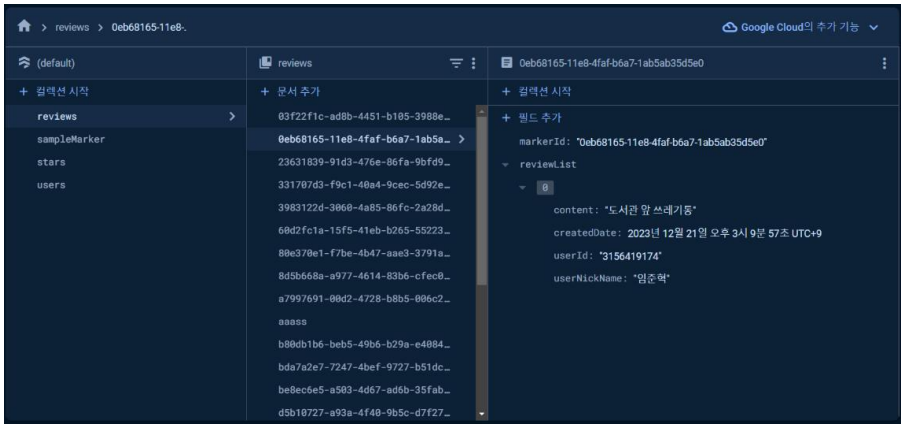

그림 1 - [reviews 컬렉션]

sampleMarker 컬렉션은 장소의 정보가 저장된다. sampleMarker 컬렉션에는 카테고리 분류를 저장하는 정수 category, 위치의 위도와 경도를 저장하는 gps, 고유 식별 번호인 id, 이미지 파일을 저장하기 위한 imageString, imageUri, 장소의 이름인 name, 장소의 평점인 star가 저장된다.

그림 2 - [sampleMarker 컬렉션]

users 컬렉션은 사용자의 정보가 저장된다. users 컬렉션에는 카카오 로그인 API를 받아온 정보들이 저장된다. email, id, nickname, token이 카카오 로그인 API에서 받아오는 정보들이다. places는 List 형태로 사용자가 즐겨 찾기로 등록한 장소들의 식별 번호가 저장된다. 이를 바탕으로 즐겨 찾기를 확인할 수 있는 기능이 구현된다.

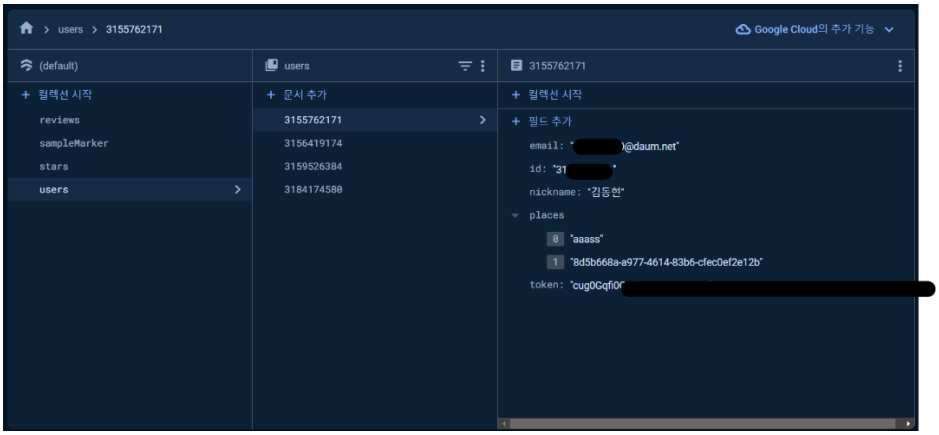

그림 3 - [users 컬렉션]

세부 구현

1. LoginActivity

지도외전을 실행하면 가장 처음으로 나오는 Activity이다. 카카오 로그인 API를 활용하여, 기존 토큰 존재 여부를 확인한 후, 존재한다면 자동 로그인을 시도한다. 존재하지 않으면 새로 로그인을 시도한다. 로그인 시, FirebaseDB에 사용자 정보가 없으면 로그인한 유저 정보를 저장한다.

```
75                  //----------로그인시, DB에 사용자 정보 없으면 로그인한 유저 정보 저장----------
76                  val userInfo = User()
77                  userInfo.id = user.id.toString()
78                  userInfo.email = user.kakaoAccount?.email.toString()
79                  userInfo.token = token.toString()
80                  userInfo.nickname = user.kakaoAccount?.profile?.nickname.toString()
81
82                  var firestore: FirebaseFirestore? = null
83                  firestore = FirebaseFirestore.getInstance()
84                  val userDocRef = firestore?.collection( collectionPath: "users")?.document(userInfo.id)
```

그림 4 - [LoginActivity]

지도외전을 실행하면 자동 로그인을 시도하기 때문에, 기존 토큰 정보가 있으면 자동 로그인을 시도하고, 기존 토큰 정보가 없으면 사용자에게 로그인을 하도록 요청한다. 따라서 LoginActivity의 layout은 일반적으로 볼 수 없다. 하지만 이후에 MyPageActivity에서 로그아웃할 수 있고, 이때 layout을 확인할 수 있다. 로그인을 성공하면 MainActivity로 넘어간다.

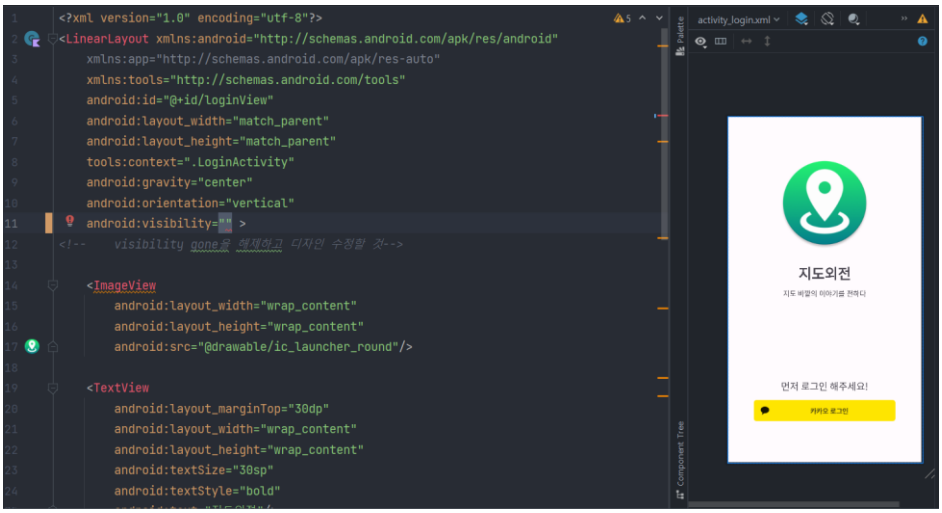

그림 5 - [activity_login.xml]

2. MainActivity

그림 6 - [MainActivity 실행 화면]

MainActivity는 카카오 지도 API를 바탕으로 지도외전의 핵심적인 기능을 제공하는 Activity이다. MainActivity에서 주요 기능은 다음과 같다.

1. FirebaseDB에서 마커 정보 불러오기, 새로 고침하기
2. SetPlaceActivity에서 전달받은 변수들로 마커 생성하기
3. ShowPlaceActivity를 실행하여 마커의 상세 정보(리뷰, 평점 등) 확인하기
4. 장소 검색하기
5. 카테고리 별로 장소 보기, 즐겨찾기 장소 보기

첫 번째, ‘FirebaseDB에서 마커 정보 불러오기, 새로 고침하기’ 기능은 FirebaseDB에 저장된 sampleMarker 컬렉션에 접근하여 장소에 관한 정보들을 전부 불러온다. 컬렉션에 있는 모든 문서의 필드를 MapPOIItem으로 저장하고, 전체를 currentPOIItems 배열에 저장한다. currentPOIItems에는 DB에 있는 모든 장소 정보들이 마커로 저장된다. 이를 통해 5번 ‘카테고리 별로 장소 보기, 즐겨찾기 장소 보기’ 기능을 제공할 수 있다. 새로고침 기능은 이 절차를 fun loadDB() 형태의 함수로 선언하여 필요한 곳에서 사용할 수 있게 한다.

```
// DB에 저장된 데이터 불러오기
fun loadDB() {
    db.collection( collectionPath: "sampleMarker") CollectionReference
        .get() Task<QuerySnapshot!>
        .addOnSuccessListener { result ->
            for (document in result) {
                val name = document.getString( field: "name") ?: ""
                val latitude = (document["gps"] as GeoPoint).latitude
                val longitude = (document["gps"] as GeoPoint).longitude
                val category = document.getLong( field: "category")?.toInt() ?: 0
                val imageUri: Uri? = null
                val imageString: String? = null
                val id: String = document.id
                val star: Int = document.getLong( field: "star")?.toInt() ?: 0
                mapView.addPOIItem(createMarker(name, latitude, longitude, imageUri, category, star, id))
                Log.d( tag: "test", id)
            }
            currentPOIItems = mapView.poiItems
            // Now you have the most up-to-date list of items
        }
        .addOnFailureListener { exception ->
            Log.w( tag: "kim",  msg: "Error getting documents.", exception)
        }
}
```

그림 7 - [MainActivity - FirebaseDB 불러오기]

두 번째, ‘SetPlaceActivity를 실행하여 마커의 상세 정보(리뷰, 평점 등) 확인하기’ 기능은 마커를 새로 생성하는 데 이용된다. MainActivity에서 마커를 새로 생성하기는 비효율적이기에, 위도와 경도 같은 핵심적인 정보만 Intent를 이용하여 SetPlaceActivity로 전달한다. 그 다음 SetPlaceActivity에서 이용자가 세부 정보들을 입력하게 하고 전달받도록 한다. SetPlaceActivity에서 전달받는 인자들은 이름, 카테고리, 이미지, 평점, id 등이 있다.

```
// 다른 엑티비티로부터 결과를 받아오는 코드 ***등록할 좌표 정보, 이름, 카테고리 등을 받아와서 mapView에 좌표로 등록**
resultLauncher = registerForActivityResult(
    ActivityResultContracts.StartActivityForResult()){ result ->
    // SetPlaceActivity에서 등록 성공하여 돌아온다면
    if (result.resultCode == RESULT_OK) {

        val name = result.data?.getStringExtra( name: "set_name") ?: ""
        val latitude = result.data?.getDoubleExtra( name: "set_latitude",  defaultValue: 0.0)
        val longitude = result.data?.getDoubleExtra( name: "set_longitude",  defaultValue: 0.0)
        val category = result.data?.getIntExtra( name: "categoryNum",  defaultValue: 0)
        val imageString = result.data?.getStringExtra( name: "image")
        val imageUri = Uri.parse(imageString)
        var nullableStar = result.data?.getIntExtra( name: "star",  defaultValue: 0)
        var star = nullableStar ?: 0

        val id = result.data?.getStringExtra( name: "id") ?: ""
        lastSetTime = result.data?.getLongExtra( name: "last_set_time",  defaultValue: 0)

        Log.d( tag: "kim",  msg: "got name : ${name}, got lat :${latitude}, got lon : ${longitude}")
        mapView.addPOIItem(createMarker(name, latitude!!, longitude!!, imageUri, category, star, id))
    }
```

그림 8 - [MainActivity - SetPlaceActivity에서 인자 전달받기]

세 번째, ‘ShowPlaceActivity를 실행하여 마커의 상세 정보(리뷰, 평점 등) 확인하기’ 기능은 생성된 마커의 상세 정보를 보는데 이용한다. 두 번째 기능과 유사하게, MainActivity에서 장소 정보를 처리하지 않고 ShowPlaceActivity로 넘기는 역할을 한다. ShowPlaceActivity에서는 특정한 인자를 전달받지 않으므로 구체적인 기능은 ShowPlaceActivity에 기술한다.

네 번째, ‘장소 검색하기 기능’이다. MainActivity layout에 상단에 주소 검색바를 이용하여 도로명 주소 등을 검색하면 해당 장소로 이동한다.

```
// 주소 검색 이동
binding.submitButton.setOnClickListener { it: View!
    val address = binding.searchBar.text.toString()
    Log.d( tag: "location_map",  msg: "address: $address")
    val geocoder = Geocoder( context: this)
    val addresses = geocoder.getFromLocationName(address,  maxResults: 1)

    if (!addresses.isNullOrEmpty()) {
        val latitude = addresses[0].latitude
        val longitude = addresses[0].longitude

        Log.d( tag: "location_map",  msg: "location success")
        Log.d( tag: "location_map",  msg: "$latitude , $longitude")
        val mapPoint = MapPoint.mapPointWithGeoCoord(latitude, longitude)
        mapView.setMapCenterPoint(mapPoint,  animated: true)
    } else {
        Log.d( tag: "location_map",  msg: "location fail")
    }
}
```

그림 9 - [MainActivity - 주소 검색 이동하기]

다섯 번째, '카테고리 별로 장소 보기, 즐겨찾기 장소 보기' 기능이다. 1번 기능 'FirebaseDB에서 마커 정보 불러오기, 새로 고침하기'에서 저장한 모든 마커들이 저장된 currentPOIItems 배열에서, 특정한 조건에 해당하는 카테고리 혹은 즐겨찾기 장소만 볼 수 있도록 하는 기능이다. 이 역시 카카오 맵 API를 통해 구현한다.

주요 기능 다섯 가지 외에도 마이 페이지로 이동하는 기능, 현재 위치로 돌아오는 기능 등이 존재한다.

3. SetPlaceActivity

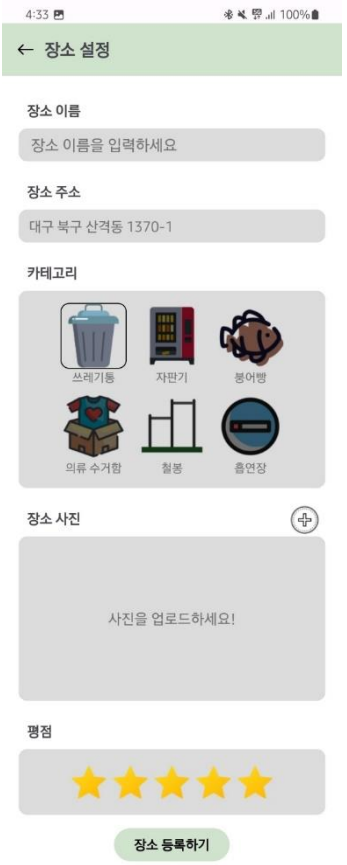

그림 10 - [SetPlaceActivity 실행 화면]

SetPlaceActivity는 MainActivity로부터 위도와 경도, 이전 마커 마지막 저장 시간을 받고 이외 마커 등록에 필요한 모든 정보를 다시 MainActivity로 넘기는 Activity이다.

SetPlaceActivity의 주요 기능은 장소 이름, 카테고리, 사진, 평점을 저장하여 MainActivity로 넘기는 것이다. 다음과 같은 세부적인 구현 기능이 있다.

첫 번째, MainActivity로 넘겨받은 위도 경도를 바탕으로 도로명 주소를 알려준다.

두 번째, 입력한 정보를 바탕으로 '장소 등록하기' 버튼을 클릭하면 FirebaseDB의 sampleMarker 컬렉션에 새로운 문서를 만든다. sampleMarker 컬렉션에는 카테고리 분류를 저장하는 정수 category, 위치의 위도와 경도를 저장하는 gps, 고유 식별 번호인 id, 이미지 파일을 저장하기 위한 imageString, imageUri, 장소의 이름인 name, 장소의 평점인 star가 저장된다.

이 외에도 카테고리나 평점의 별을 클릭했을 때, 시각적으로 무엇이 눌려졌는지 강조하는 디자인이 구현되어 있다.

```
if (leftTime in 1 ≤ .. ≤ limitTime) {
    val sec = (leftTime) / 1000
    Toast.makeText( context: this, text: "${limitTime / 1000 - sec}초 후에 등록이 가능해요.", Toast.LENGTH_SHORT).show()
}
else {
    val name = binding.placeName.text.toString()
    val uniqueId = UUID.randomUUID()
        .toString()//랜덤한 아이디(키값) 생성, DB저장용, but MainActivity에서도 사용해야 하므로 intent로 넘겨줌

    // 등록 시간
    intent.putExtra( name: "last_set_time", System.currentTimeMillis())

    intent.putExtra( name: "set_latitude", latitude)
    intent.putExtra( name: "set_longitude", longitude)
    intent.putExtra( name: "set_name", name)

    intent.putExtra( name: "isSet", value: true)
    intent.putExtra( name: "categoryNum", currentSelectedNum)
    intent.putExtra( name: "image", uri.toString())
    intent.putExtra( name: "star", currentScore)
    intent.putExtra( name: "id", uniqueId)
    intent.putExtra( name: "author", MainActivity.staticUserNickname)
    // uri는 String으로 변환해서 intent로 넘기고, 받을 때 다시 parse 해야 함
```

그림 11 - [SetPlaceActivity - MainActivity로 인자 전달]

4. ShowPlaceActivity

showPlaceActivity는 장소의 상세 정보를 보여주는 Activity 이다. MainActivity의 지도 화면에서 마커를 클릭하고 표시되는 이미지를 클릭하면 상세 정보 페이지로 들어간다. 이때 MainActivity에서 showPlaceActivity로 전환된다.

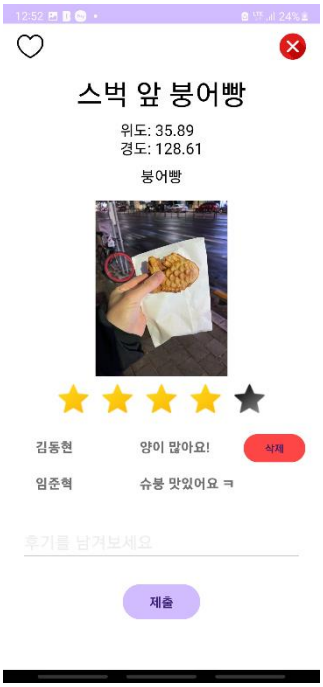

그림 12 - [ShowPlaceActivity 실행 화면]

showPlaceActivity의 기능은 다음과 같다

1. 해당 마커 정보를 화면에 표시
2. 리뷰 작성 및 삭제
3. 즐겨찾기 추가 및 제거

showPlaceActivity의 주요 기능의 세부설명은 다음과 같다..

1. 해당 마커의 정보 표시: MainActivity 에서 넘겨받은 intent를 통해 해당 마커의 내용을 화면에 표시한다. 그리고 marker id를 통해 유의 reviews 테이블에서 해당 마커에 등록된 댓글들을 불러와 recycler view에 담아서 화면에 보여준다
2. 리뷰 작성 및 삭제: 리뷰를 작성할 수 있고 삭제할 수 있다.
   리뷰 작성 -> EditText 태그에서 리뷰 작성 후 제출 버튼을 누르면 현재 마커의 id를 이용해 DB의 reviews 테이블에 속한 해당 마커의 review list에 리뷰 내용 및 작성자(유저) 정보가 추가되게 된다.

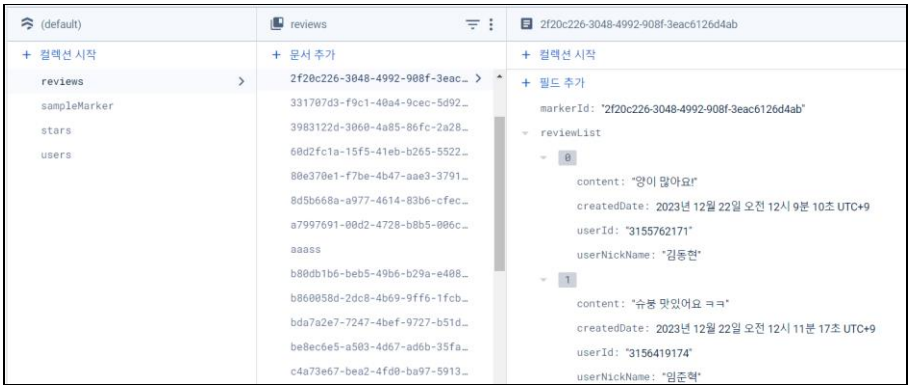

그림 13 - [Firebase의 reviews 테이블 화면]

리뷰 삭제 -> 리뷰 삭제는 해당 리뷰를 작성한 유저만이 삭제가 가능하도록 하였다. 유저가 삭제 버튼을 누르면 recycler view에 속한 해당 리뷰의 position 을 이용해 DB의 해당 마커의 review list에서 해당 인덱스(position과 동일)에 위치한 리뷰를 삭제한 후,

recycler view를 갱신한다.

[리뷰 작성]

```
//*********************리뷰 등록 버튼*********************//
binding.submitCommentButton.setOnClickListener { it: View!
    //입력창 내용 가져오기
    val content : String = binding.commentEditText.text.toString()
    val review = Review()
    review.markerId = id
    review.addReview(MainActivity.staticUserId.toString(), MainActivity.staticUserNickname, content, Date())
    //리뷰 등록
    val db : FirebaseFirestore = FirebaseFirestore.getInstance()
    val docRef : DocumentReference = db.collection( collectionPath: "reviews").document(id)
    docRef.get()
        .addOnSuccessListener { document: DocumentSnapshot ->
            if(document != null && document.exists()){
                //기존에 문서가 존재하는 경우, 기존의 reviewList 가져옴
                val existingReviewList : MutableList<KeyValueElement> = document.data?.get("reviewList") as MutableList<KeyValueElement>
                //새 리뷰를 기존 reviewList에 추가
                existingReviewList.addAll(review.reviewList)
                //firebase 문서 업데이트
                docRef.update( field: "reviewList", existingReviewList)
                    .addOnSuccessListener { it: Void!
                        Log.d( tag: "DB",  msg: "reviewList successfully updated in existing document!")
                        binding.commentEditText.text.clear()
                        Toast.makeText( context: this,  text: "리뷰가 등록되었습니다.", Toast.LENGTH_SHORT).show()
```

그림 14 - [제출 버튼을 눌렀을 때 리뷰를 등록한다]

[리뷰 삭제]

```
binding.commentDeleteBtn.setOnClickListener{ it: View!
    Log.d( tag: "review",  msg: "MyAdapter: onBindViewHolder: commentDeleteBtn: ${reviewList[position]}")
    Log.d( tag: "review",  msg: "MyAdapter: onBindViewHolder: commentDeleteBtn: ${position}")
    //리뷰 등록
    val db : FirebaseFirestore = FirebaseFirestore.getInstance()
    val docRef : DocumentReference = db.collection( collectionPath: "reviews").document(MainActivity.currentMarkerId)
    docRef.get()
        .addOnSuccessListener { document: DocumentSnapshot ->
            if(document != null && document.exists()){
                //기존에 문서가 존재하는 경우, 기존의 reviewList 가져옴
                val existingReviewList : MutableList<KeyValueElement> = document.data?.get("reviewList") as MutableList<KeyValueElement>
                //해당 position 의 review를 지움
                existingReviewList.removeAt(position)
                //firebase 문서 업데이트
                docRef.update( field: "reviewList", existingReviewList)
                    .addOnSuccessListener { it: Void!
                        Log.d( tag: "DB",  msg: "reviewList successfully updated in existing document!")
                        Toast.makeText(context,  text: "리뷰가 삭제되었습니다.", Toast.LENGTH_SHORT).show()
                    }
                    .addOnFailureListener { e ->
                        Log.d( tag: "DB",  msg: "Fail, can not updated exsisting reviewList", e)
```

그림 15 - [삭제 버튼을 눌렀을 때 리뷰를 삭제한다]

3. 즐겨 찾기 추가 및 제거

유저는 즐겨찾기 버튼을 통해 즐겨찾기 마커 리스트에 현재 마커를 등록하거나 제거할 수 있다. 즐겨찾기 리스트는 DB의 users 테이블에 들어있는 유저 정보의 하위 필드에 저장된다. 현재 페이지에 접속하게 되면 유저 정보를 이용해 DB의 users 테이블에서 해당 유저의 즐겨찾기 리스트를 불러와 현재 마커 id(documentId)와 비교 후, 리스트에 존재한다면 즐겨찾기 버튼을 눌려진 상태로 표시한다.

```
var usersDB : DocumentReference = db.collection( collectionPath: "users").document(MainActivity.staticUserId.toString())
usersDB.get().addOnSuccessListener() { document: DocumentSnapshot ->
    existingPlaceList = document.data?.get("places") as MutableList<String>

    // 즐겨찾기 places 에 속하면 하트 활성화
    for (item : String in existingPlaceList) {
        if (item == documentId) {
            binding.heartButton.setImageResource(R.drawable.red_heart)
            isFavorite = true
            break
        }
    }
}
```

그림 16 - [DB에서 해당 유저의 즐겨찾기 리스트를 가져온다]

눌려지지 않은 즐겨찾기 버튼을 누르면 동일하게 방식으로 DB에 해당 유저 정보로 접근하여 즐겨찾기 리스트에 추가하고, 이와 반대로 눌려진 즐겨찾기 버튼을 누르면 즐겨찾기 리스트에서 현재 마커 id(documentId)를 제거한다.

```
binding.heartButton.setOnClickListener { it: View!
    // 현재 이미지 리소스 가져오기
    val currentImageResource : Drawable! = binding.heartButton.drawable

    // 현재 이미지와 비교하여 변경
    if (!isFavorite) {
        // 현재 이미지가 blank_heart이면 filled_heart로 변경
        binding.heartButton.setImageResource(R.drawable.red_heart)
        isFavorite = true

        usersDB.get()
            .addOnSuccessListener() { document: DocumentSnapshot ->
                if(document != null && document.exists()){
                    //기존에 즐겨찾기가 존재하는 경우, 기존의 places 가져옴
                    //새 즐겨찾기를 기존 places에 추가
                    existingPlaceList.add(documentId)
                    //firebase 문서 업데이트
                    usersDB.update( field: "places", existingPlaceList) Task<Void!>
                        ?.addOnSuccessListener { it: Void!
                            Log.d( tag: "DB",  msg: "placeList successfully updated in existing document!")
                            Toast.makeText( context: this,  text: "즐겨찾기가 등록되었습니다.", Toast.LENGTH_SHORT).show()
                        } Task<Void!>?
                        ?.addOnFailureListener { e ->
                            Log.d( tag: "DB",  msg: "Fail, can not updated exsisting placeList", e)
                            Toast.makeText( context: this,  text: "Error, 즐겨찾기가 등록되지 않았습니다.", Toast.LENGTH_SHORT).show()
                        }
```

그림 17 – [즐겨찾기 버튼을 눌렀을 때 기존의 상태에 따라 즐겨찾기 추가or제거 기능 수행]

5. MyPageActivity

MyPageActivity는 사용자 정보를 간략하게 보여주는 Activity 이다. MyPageActivity에는 MainActivity에 있는 static 변수인 staticUserId, staticUserEmail, staticUserNickName을 가져와 보여주는 역할을 한다.

MyPageActivity에는 로그아웃 버튼이 존재하며 이를 통해 카카오 계정을 로그아웃하고 새로운 계정 혹은 기존 계정의 로그인을 새로 시도할 수 있다.

그림 18 - [MyPageActivity 실행 화면]

추가 기술 사항		

특이사항	