

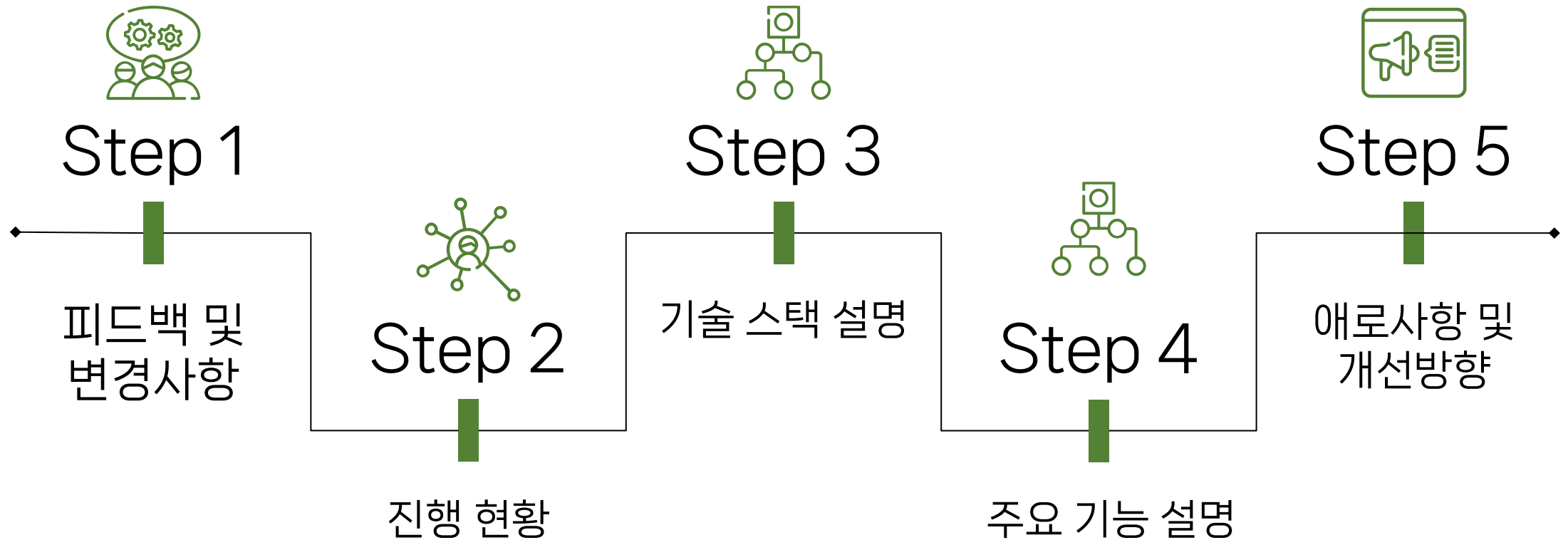


전북대학교

# 북대미문

201911827 백승원  
201911858 유상욱  
202011622 김동주  
202212903 고진영

# 목차





# 1. 피드백 및 변경사항

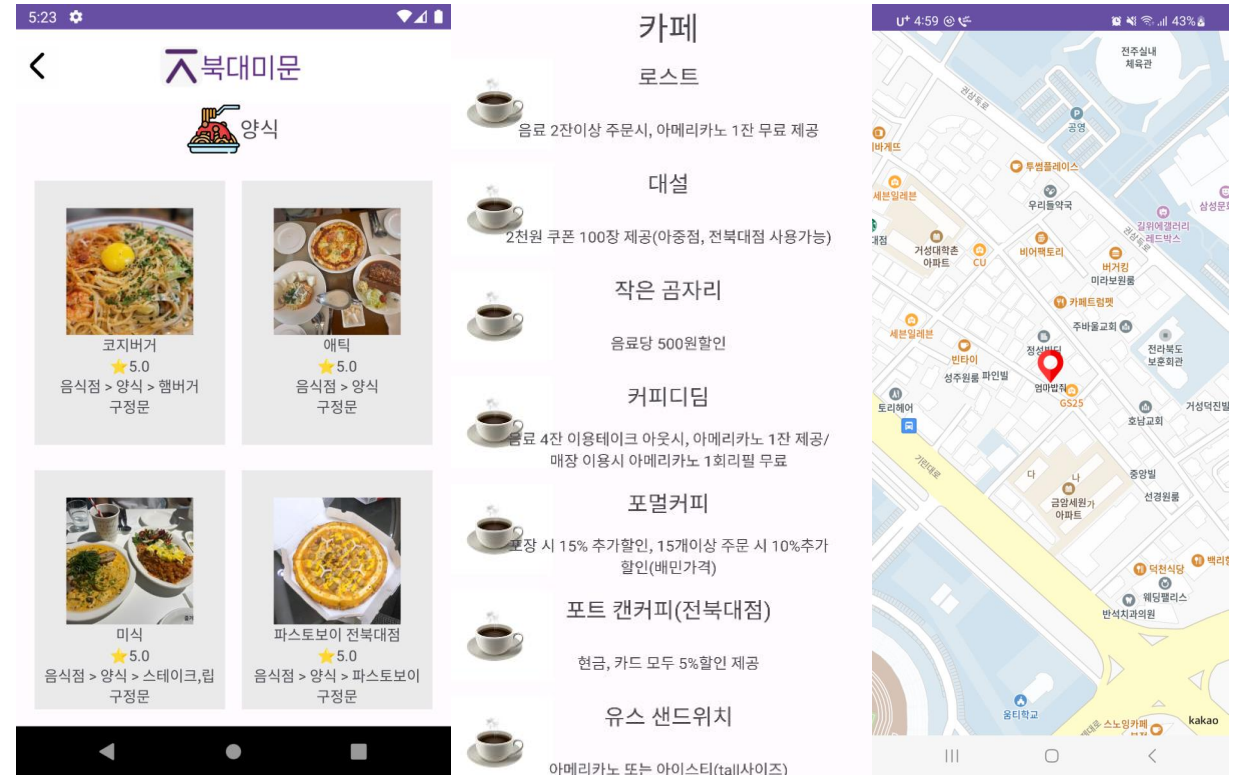
# 주제확립

- 주제 구체화

1차 점검 피드백에 따라 주요 기능에 일관성을 부여하기 위해 기존의 전북대 앱 주제를 전북대 맛집 앱으로 구체화

- 기존 맛집 어플과의 차별성

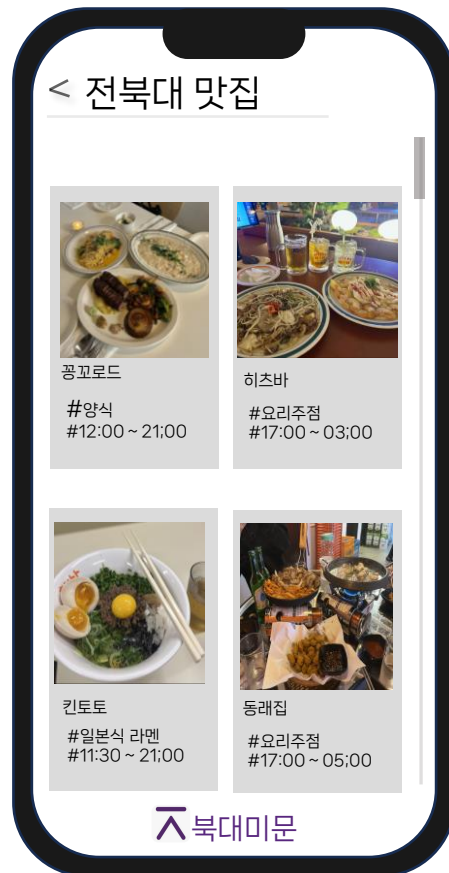
어플의 기능들을 일관성 있게 만들어 앱내 기능이 잘 맞물리게 만들었고 전북대 제휴업체, 음식점 랜덤 추천기능, 위시리스트와 친구추가의 기능을 활용한 공유시스템으로 다른 맛집 어플과 차별성을 만들 계획



# 주요기능 변경

기존에 계획한 기능 중 일부 제거

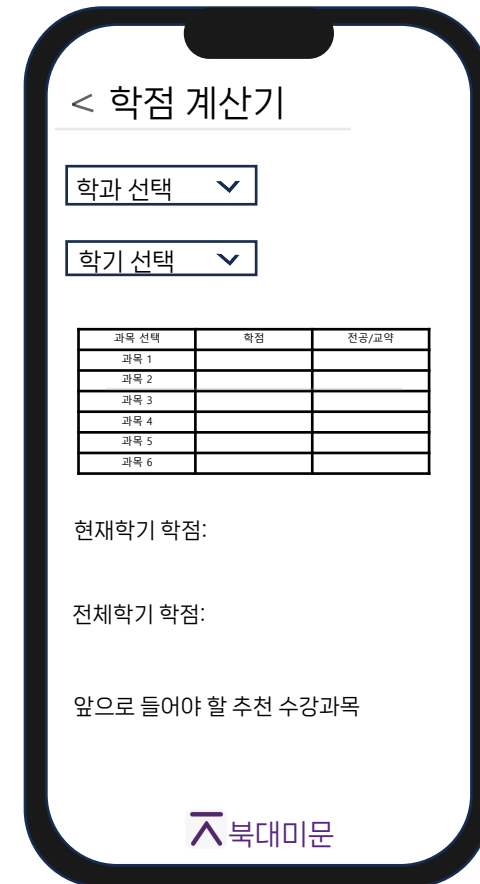
전북대 맛집/제휴업체



공지사항 키워드 알림



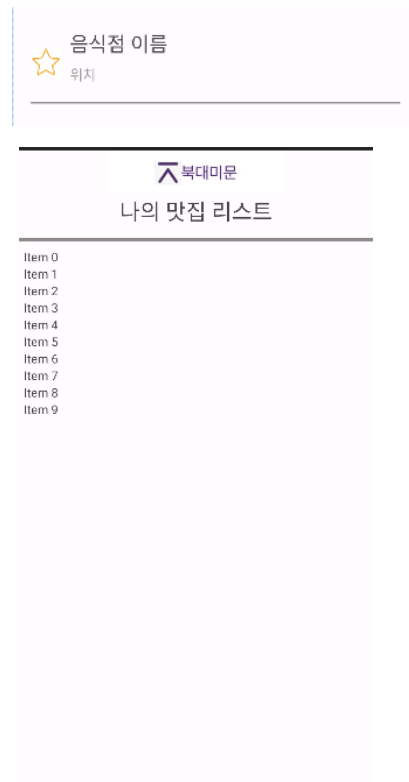
이수학점 계산기



# 주요기능 변경

일관성 있는 새로운 기능

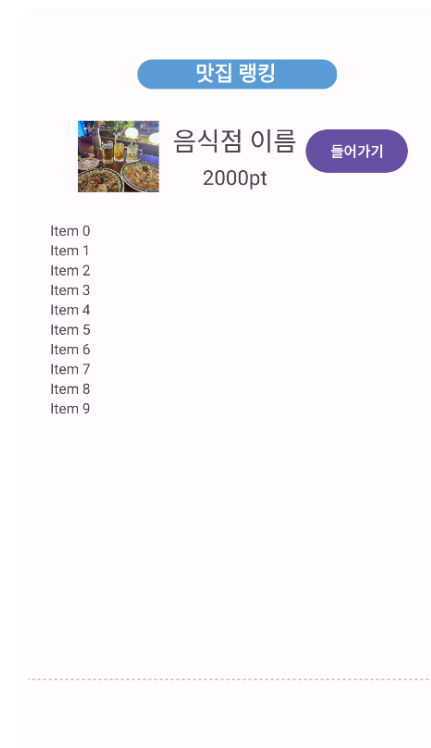
## 위시리스트



## 음식 필터 기능



## 맛집 랭킹



# DB 및 API 변경



강의내용과 관련 있는 DB로 변경



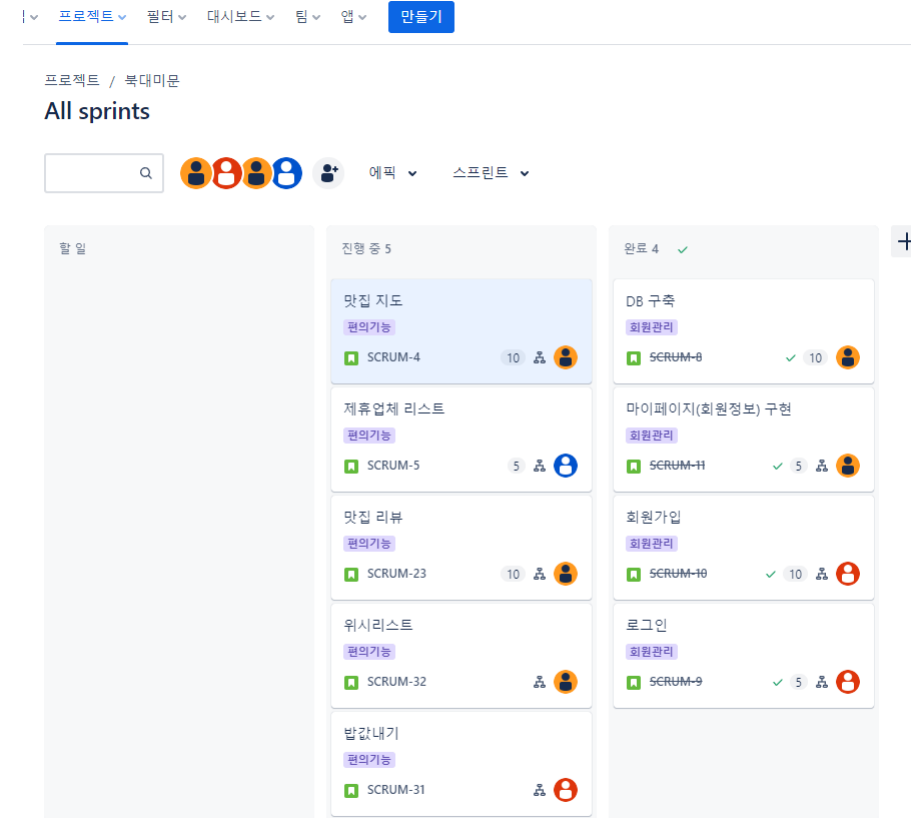
주제 확립에 따라 공공데이터 API는 사용하지 않는 것으로 결정

# 첫 스프린트 문제점 및 교훈

첫 스프린트는 구체적인 역할 분배 없이 유동적으로 프로젝트에 기여하는 방식으로 진행

이에 따라 팀원 별 코딩 역할이 모호해져 같은 기능에 대한 중복 작업 발생

이후 진행 방식을 변경해 팀원 별 역할을 명확하게 분배하여 진행중





# JIRA 변동 사항

- 기존 파이어베이스를 사용한 DB 구축을 계획하였으나 수업 내용과의 연관성을 위해 MySQL 서버 구축으로 변경
- 이에 따라 DB 구축 및 연동 담당자를 김동주->유상욱 으로 변경
- 김동주 팀원이 이미 구현한 파이어베이스 기반 로그인/회원가입 코드를 바탕으로 MySQL 기반 회원관리 구축

SCRUM-43	회원가입 기능구현	 김동주
SCRUM-42	회원가입 레이아웃	 김동주
	SCRUM-29 로그인 기능 구현	 김동주
	SCRUM-28 로그인 레이아웃	 김동주
	SCRUM-49 랭킹시스템 DB연동	 유상욱
	SCRUM-47 위시리스트 DB연동	 유상욱
	SCRUM-46 맛집 리뷰 DB연동	 유상욱
	SCRUM-45 로그인 DB연동	 유상욱
	SCRUM-44 회원가입 DB연동	 유상욱
	SCRUM-27 마이페이지와 DB 연동	 유상욱
	SCRUM-16 지도 표시 및 연동	 유상욱

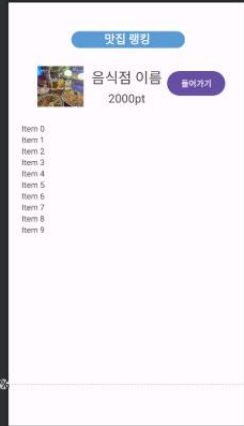


## 2. 진행 현황

# 데일리스크럼

- 매일 11시30분 디스코드 음성 채팅을 통해 15분 정도 스크럼 진행중
- 화면 공유 기능을 활용하여 안드로이드 스튜디오 화면을 띄워 코드 리뷰
- 조사한 자료를 공유하며 의견 교환
- 스크럼 종료 후 내용 기록

음식점 별점은 volley 통신함수 통해



11/28 데일리스크럼

- 안드로이드 스튜디오와 MySQL 연동 완료(한글 인코딩 문제만 해결하면됨)
- 추가기능(밥값내기, 맛집 위시리스트) 지라에 추가
- 기존 사례에서 기능을 찾아보는 것으로 방향잡음
- 빠른 시일 내에 주요기능 확정 후 개발 시작 시급(이번주 안으로 시작)

2023년 11월 29일

유상욱 2023.11.29. 오후 11:46

메인화면, 리뷰(단순 XML) - 백승원  
카카오API(XML에 표시, 코틀린 코드) - 유상욱  
제휴업체, 위시리스트, 랭킹(단순 XML 및 자료조사) - 고진영  
밥값내기(롤렛 사다리 XML 및 코틀린 코드) - 김동주 (수정됨)

유상욱 2023.11.29. 오후 11:56

11/29 데일리스크럼

- 추가기능 확정(카테고라이징, 롤렛/사다리, 랭킹시스템)
- 역할 분담(12/6 까지)
- 12/6 데일리 스크럼때 각자 준비해온 결과물 리뷰
- 내일 혹은 모레 스크럼때 프로젝트와 수업내용 연관성에 대해 간단히 논의

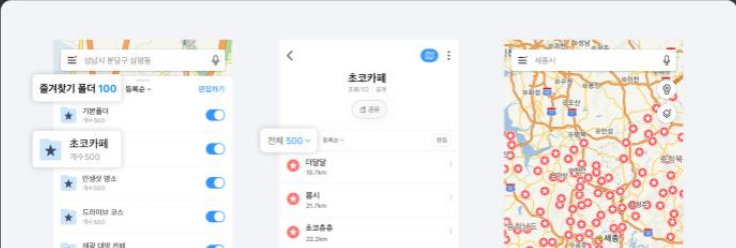
2023년 11월 30일

유상욱 2023.11.30. 오후 11:37

11/30 데일리스크럼

- 2차 중간발표 역할 유지
- 내일(12/1) 데일리스크럼은 생략
- 12/6 스크럼 이후 본격적으로 PPT 제작
- PPT 제작 전까지는 코드 작성에 집중
- 주요 기능과 수업내용 아직까지는 어느정도 밀접해보임

유상욱 2023.12.05. 오후 11:42



# JIRA 로드맵

- 회원관리 기능에 필요한 작업 전반을 완료하였음
- 마이페이지 구현은 우선순위가 낮다고 판단하여 천천히 진행중
- 주요 기능 스프린트 진행중



# 역할 분배 현황

백승원

메인 화면 구성  
맛집 리스트 및  
상세페이지 구현  
리뷰 작성 구현

유상욱

데이터베이스  
카카오 API  
지도 화면 구성  
HTTP 통신

김동주

로그인/회원가입  
레이아웃 및 로직  
작성  
롤렛 동작 및 애  
니메이션 구현

고진영

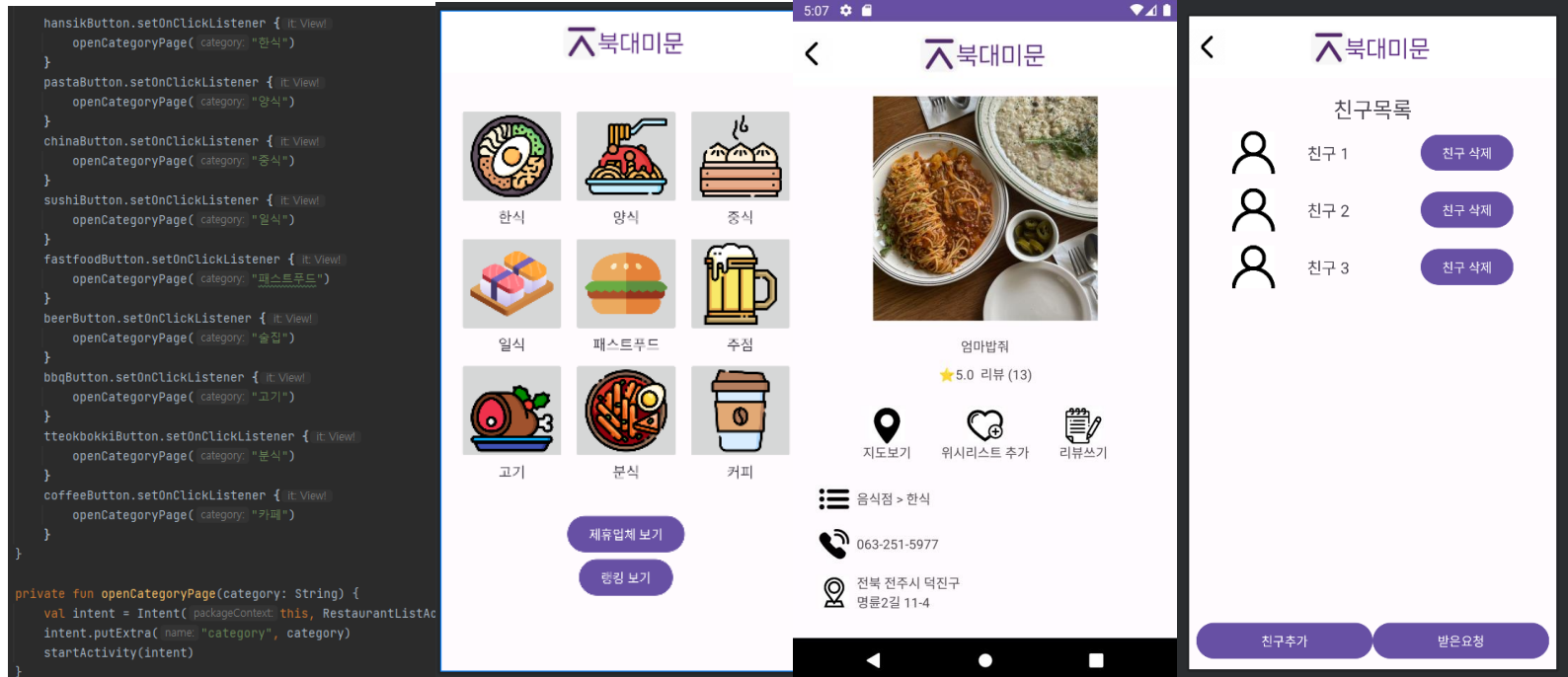
제휴업체 조사 및  
리스트 제작  
위시리스트 구현  
맛집 랭킹 구현

# 담당 테스트

백승원

맛집 분류 및 상세 설명  
위주/친구목록 위주로  
작업

SCRUM-56	맛집 상세 페이지 레이아웃 작성	백승원
SCRUM-55	맛집 리스트 레이아웃 작성	백승원
SCRUM-51	메인 화면에 카테고리 구성	백승원
SCRUM-36	맛집 리뷰 기능구현	백승원
SCRUM-35	맛집 리뷰 레이아웃	백승원



## DB와 API 위주로 작업

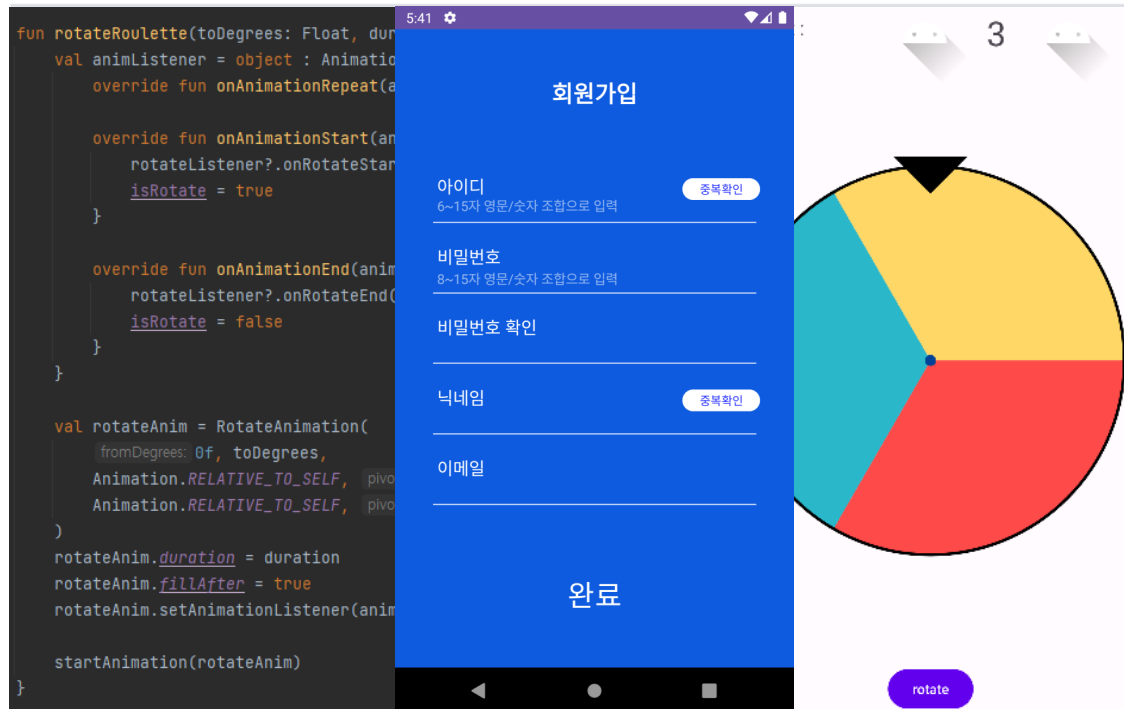
SCRUM-54	카카오 API로부터 리스트 불러오기	<pre> &lt;?php header("Content-Type: application/json");  // POST 요청으로 받은 데이터 \$query = \$_POST['query'];  // MySQL 연결 \$conn = mysqli_connect('localhost', 'root', '1234', 'bdmm');  // 연결 확인 if (\$conn-&gt;connect_error) {     die("Connection failed: " . \$conn-&gt;connect_error); }  // 쿼리 전송 \$result = mysqli_query(\$conn, \$query);  // 쿼리 결과를 배열로 저장 \$rows = mysqli_fetch_array(\$result);  if(\$conn-&gt;error){ // 에러 발생시     echo json_encode(array("result" =&gt; "error")); } else if (\$row == null) { // 결과가 빈 배열     echo json_encode(array("result" =&gt; "no result")); } else { // 그 밖의 경우 쿼리 결과 응답     echo json_encode(array("result" =&gt; \$rows)); }  // MySQL 연결 종료 \$conn-&gt;close(); ?&gt; </pre>	<pre> // POST 통신 함수 class HttpUtil {      fun post(page: String, query: String, extra: String = "", extra2: Int = 0): JSONObject {         // 통신할 php 서버 주소 및 페이지         val url = "https://bdmm-uqvpb.run.goorm.site/BDMM/\$page.php"          // 전송할 쿼리문         var postData = "query=\$query"          // 불필요한 오버로딩을 피하기 위해 추가 매개변수 설정         // 추가 매개변수(카카오맵 검색 범위) 설정시 쿼리 추가         if(extra == "구정문"){             postData += "&amp;rect=127.1318,35.840,127.1181,35.8475"         }         else if(extra == "사대부고"){             postData += "&amp;rect=127.1327,35.8408,127.1408,35.8450"         }          // 추가 매개변수2(카카오맵 검색 결과 페이지) 설정시 쿼리 추가         if(extra2 != 0)             postData += "&amp;page=\$extra2"          // 전송할 데이터를 바이트 단위로 변환         val data = postData.toByteArray(Charsets.UTF_8)     } } </pre>
SCRUM-49	행정시스템 DB연동		
SCRUM-47	위시리스트 DB연동		
SCRUM-46	맛집 리뷰 DB연동		
SCRUM-45	로그인 DB연동		
SCRUM-44	회원가입 DB연동		
SCRUM-27	마이페이지와 DB 연동		
SCRUM-16	지도 표시 및 연동		
SCRUM-15	카카오맵 API		

# 담당 테스트

김동주

회원관리 및 고급 뷰 구현  
위주로 작업

SCRUM-43	회원가입 기능구현	김동주
SCRUM-42	회원가입 레이아웃	김동주
SCRUM-38	룰렛 기능구현	김동주
SCRUM-37	룰렛 레이아웃	김동주
SCRUM-29	로그인 기능 구현	김동주
SCRUM-28	로그인 레이아웃	김동주



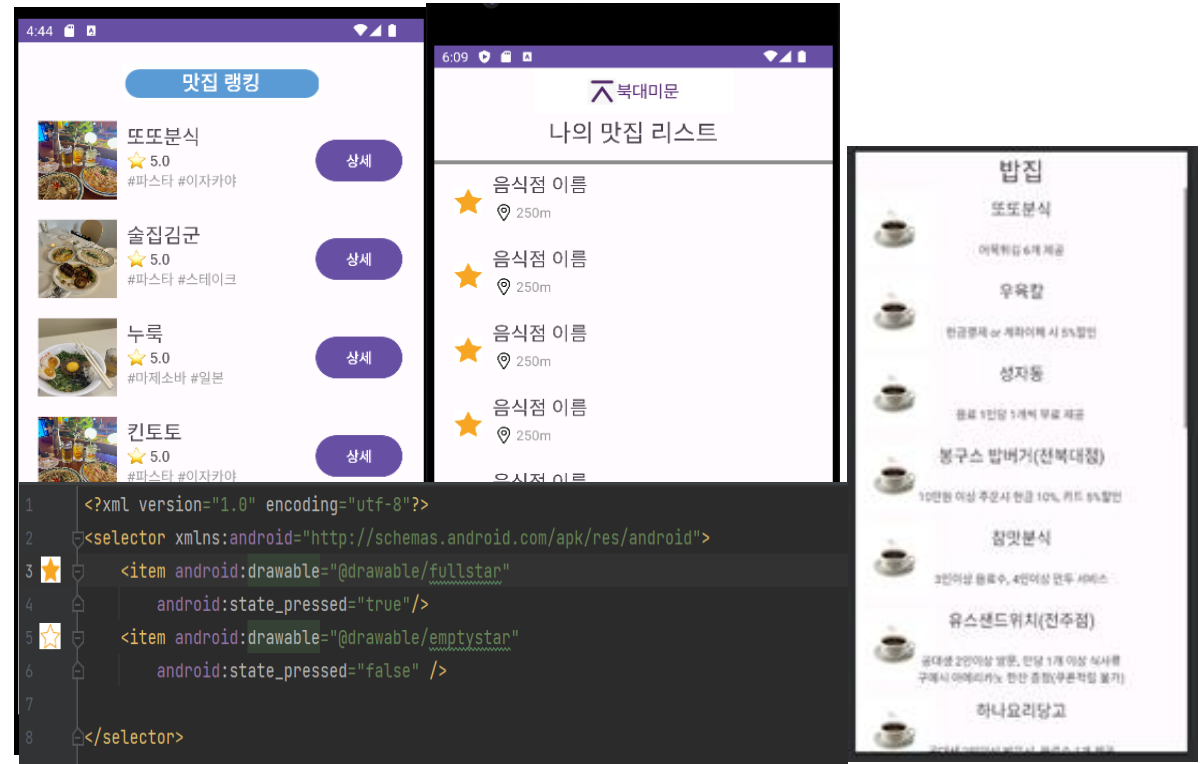


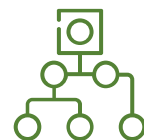
# 담당 테스트

고진영

맛집 기능 확장 위주로 작업

SCRUM-52	랭킹 레이아웃 작성	고진영
SCRUM-50	랭킹 기능구현	고진영
SCRUM-41	위시리스트 기능구현	고진영
SCRUM-40	위시리스트 레이아웃	고진영
SCRUM-39	제휴업체 리스트 레이아웃	고진영
SCRUM-30	제휴업체 리스트 기능구현	고진영
SCRUM-17	학생회 제휴업체 조사	고진영





# 3. 기술 스택 설명

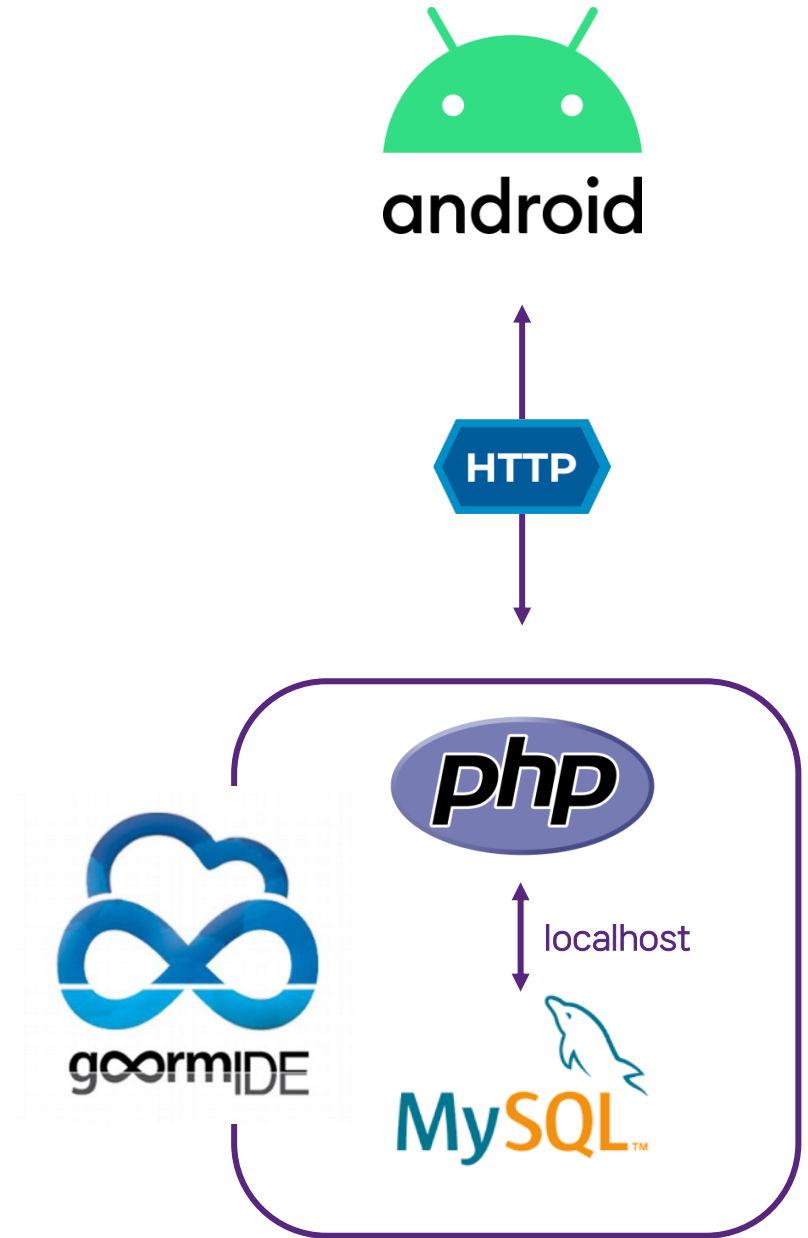
# 데이터베이스 구축

- 회원관리와 리뷰, 랭킹 등의 기능을 구현하기 위해 MySQL 서버가 필요
- 안드로이드의 경우 보안 문제로 인해 MySQL서버와 직접적으로 통신할 수 없음
- 이 때문에 MySQL서버에 접근하려면 HTTP통신이 가능한 별개의 서버가 필요함
- 구현 난이도, 호환성 등을 고려해서 서버 스택으로 PHP를 선택
- 위 조건에 가장 적합한 구름ide의 클라우드 서버를 이용하였음



# 데이터베이스 접근 과정

- 안드로이드에서 PHP서버에 POST 요청 전송
  - > PHP 서버가 요청을 수신 받아 로컬 포트를 통해 MySQL 서버에 쿼리 전송
  - > MySQL 서버에서 쿼리 수행 후 결과를 PHP 서버로 전송
  - > PHP 서버가 쿼리 수행 결과를 안드로이드에 응답으로 전송



# MySQL

- 클라우드 우분투 서버에 설치된 MySQL 서버
- 동일하게 로컬상에 설치된 PHP 서버와 통신
- 외부 접근 허용을 통해 워크벤치에서도 접속 가능
- 예시로 유저 테이블에 아이디, 닉네임, 비밀번호, 이메일 등 저장

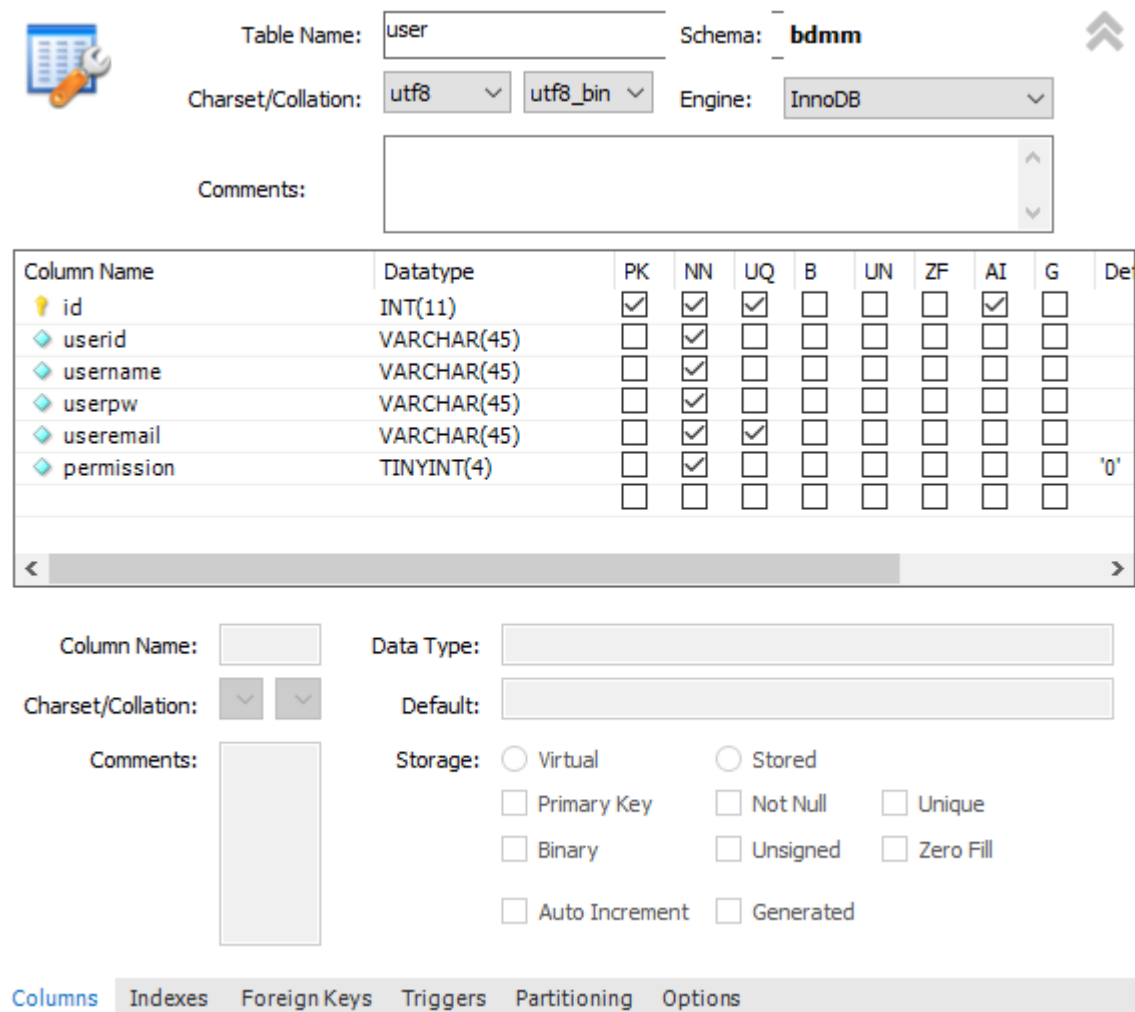


Table Name: user Schema: bdmm

Charset/Collation: utf8 utf8\_bin Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	De
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
userid	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
username	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
userpw	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
useremail	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
permission	TINYINT(4)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'

Column Name: Data Type: Charset/Collation: Default: Comments: Storage: ☐ Virtual ☐ Stored ☐ Primary Key ☐ Not Null ☐ Unique ☐ Binary ☐ Unsigned ☐ Zero Fill ☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

# PHP

- PHP서버에서 로컬 MySQL 서버와 통신하는 코드
- POST 요청으로 받은 데이터를 MySQL 서버에 쿼리 전송
- 쿼리 수행 결과는 JSON 형식으로 인코딩하여 출력

```
1 <?php
2 header("Content-Type: application/json; charset=UTF-8");
3
4 // POST 요청으로 받은 데이터
5 $query = $_POST['query'];
6
7 // MySQL 연결
8 $conn = mysqli_connect('localhost', 'root', '1234', 'bdmm', 3306);
9
10 // 연결 확인
11 if ($conn->connect_error) {
12     die("Connection failed: " . $conn->connect_error);
13 }
14
15 // 쿼리 전송
16 $result = mysqli_query($conn, $query);
17
18 // 쿼리 결과를 배열로 저장
19 $row = mysqli_fetch_array($result);
20
21 if($conn->error){ // 에러 발생시
22     echo json_encode(array("result" => "error", "message" => $conn->error, "query" => $query));
23 }
24 else if ($row == null) { // 결과가 빈 배열일때
25     echo json_encode(array("result" => "success", "message" => "empty"));
26 }
27 else { // 그 밖의 경우 쿼리 결과 응답
28     echo json_encode(array("result" => "success", "message" => $row));
29 }
30
31 // MySQL 연결 종료
32 $conn->close();
33 ?>
```

# HttpUtil.kt

- PHP서버와 통신하기 위한 코틀린 코드
- 필요한 요청에 따른 페이지 이름과 전송할 쿼리를 매개변수로 받는 post() 메소드 선언
- 추가 매개변수는 카카오맵 API에 접근할 때 사용

```
// POST 통신 함수
class HttpUtil {
    fun post(page: String, query: String, extra: String = "", extra2: Int = 0): JSONObject {
        // 통신할 php 서버 주소 및 페이지
        val url = "https://bdmm-uqypb.run.goorm.site/BDMM/$page.php"

        // 전송할 쿼리문
        var postData = "query=$query"

        // 불필요한 오버로딩을 피하기 위해 추가 매개변수 설정
        // 추가 매개변수(카카오맵 검색 범위) 설정시 쿼리 추가
        if(extra == "구정문"){
            postData += "&rect=127.1318,35.840,127.1181,35.8475"
        }
        else if(extra == "사대부고"){
            postData += "&rect=127.1327,35.8408,127.1408,35.8450"
        }

        // 추가 매개변수2(카카오맵 검색 결과 페이지) 설정시 쿼리 추가
        if(extra2 != 0)
            postData += "&page=${extra2}"

        // 전송할 데이터를 바이트 단위로 변환
        val data = postData.toByteArray(Charsets.UTF_8)
    }
}
```

- Try/catch 문 안에서 HTTP 통신 수행
- HTTP 응답 코드와 예외 발생여부에 따라 통신 결과 처리
- PHP서버로 부터 받은 JSON 데이터를 JSONObject 클래스로 파싱 하여 반환

```
// POST 요청
try {
    val connection = URL(url).openConnection() as HttpURLConnection
    connection.doOutput = true
    connection.requestMethod = "POST"
    connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded")
    connection.outputStream.write(data)

    val responseCode = connection.responseCode // 서버 응답 코드
    if (responseCode == HttpURLConnection.HTTP_OK) { // HTTP 연결 성공시
        val inputStream = connection.inputStream
        // POST 통신으로 받은 결과값 반환
        return JSONObject(inputStream.bufferedReader().use { it.readText() })
    } else { // HTTP 연결 실패시(서버 문제) 에러 코드 반환
        return JSONObject(
            json: "{ 'result': error, 'message': http error ${connection.responseCode} }"
        )
    }
} catch (e: Exception) { // 기타 통신 문제로 예외 발생시 에러 반환
    Log.v( tag: "HttpException", e.toString()) // 로그캣에도 출력
    return JSONObject( json: "{ 'result': error, 'message': $e }" )
}
```



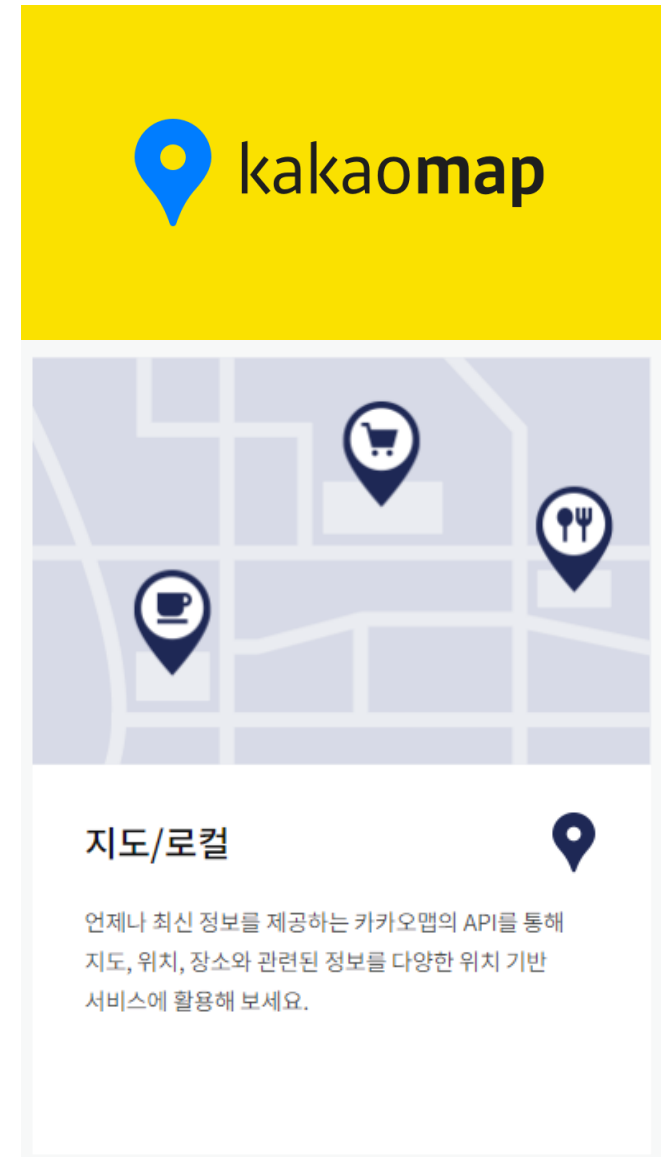
# 쿼리 결과 활용 예시

- HttpUtil 클래스를 http 변수에 선언하여 사용
- 통신 과정에서 UI 스레드를 방해하지 않기 위해 별개의 스레드를 생성하여 백그라운드 통신 수행
- httpResult 변수에 JSON 형태의 통신 결과를 담고 getString() 메소드를 통해 원하는 값을 꺼내 사용
- Toast 와 같이 UI 스레드에서 수행되어야 하는 작업은 runOnUiThread 블록에 작성

```
// 백그라운드 통신을 위한 스레드
v.isEnabled = false
var thread = Thread {
    val query = "SELECT userid FROM user WHERE userid='$user' AND userpw='$pass'"
    val httpResult = http.post( page: "sql", query)
    Log.v( tag: "Login", httpResult.getString( name: "result")) // 로그인 결과 로그
    runOnUiThread { // UI 작업은 메인 스레드에서만 가능하므로 따로 작업
        // id 와 password 일치시
        if (httpResult.getString( name: "result") == "success") {
            if(httpResult.getString( name: "message") == "empty"){
                Toast.makeText( context: this@LoginActivity,
                    text: "아이디와 비밀번호를 확인해 주세요.",
                    , Toast.LENGTH_SHORT).show()
            }
        } else {
            Toast.makeText( context: this@LoginActivity,
                text: "로그인 되었습니다."
                , Toast.LENGTH_SHORT).show()
            val intent = Intent( packageContext: this, MainActivity::class.java)
            startActivity(intent)
        }
    }
}
// 에러 발생시
else{
    Toast.makeText( context: this@LoginActivity,
        httpResult.getString( name: "message")
        , Toast.LENGTH_SHORT).show()
    Log.v( tag: "Login", httpResult.getString( name: "message")) // 에러 로그
}
v.isEnabled = true
}
thread.start()
```

# 카카오 API

- 전북대 근처 맛집의 리스트를 얻어오고 지도에 위치를 표시하기 위해 카카오의 Maps API와 Local API를 사용함
  - Maps API : 특정 위치 중심의 지도를 화면에 그리고 텍스트와 라벨을 표시하는 기능
  - Local API : 특정 키워드로 주어진 구역 내의 장소 정보를 검색하는 기능
  - Local API의 경우 REST API 이기 때문에 HTTP 통신을 이용해야 함
- > 미리 구현한 PHP 서버를 동일하게 활용



# PHP

- PHP서버에서 Local API에 HTTP 요청을 전송하는 코드
- PHP 내장 라이브러리인 cURL 사용
- 검색어와 검색 구역 범위, 페이지를 전달받아 요청 URL에 삽입(나머지 파라미터는 기본값 사용)
- Local API로부터 JSON 형태의 응답이 전달됨
- 결과를 파싱 없이 그대로 출력

```
1 <?php
2 // REST API 키
3 $key = '';
4 // 요청 URL 및 파라미터
5 $query = urlencode($_POST['query']); // 검색어
6 $rect = $_POST['rect']; // 검색 구역 범위(좌표)
7 $page = $_POST['page']; // 검색 결과 페이지
8 $url = "https://dapi.kakao.com/v2/local/search/keyword.json
9     ?page=$page
10     &size=15
11     &sort=accuracy
12     &query=$query
13     &rect=$rect";
14
15 // cURL 초기화
16 $ch = curl_init();
17
18 // cURL 옵션 설정
19 curl_setopt($ch, CURLOPT_URL, $url);
20 curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
21 curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
22 curl_setopt($ch, CURLOPT_HTTPHEADER, array(
23     'Authorization: KakaoAK ' . $key
24 ));
25
26 // cURL 실행 및 응답 얻기
27 $response = curl_exec($ch);
28
29 // cURL 세션 종료
30 curl_close($ch);
31
32 // 응답 출력
33 echo $response;
34 ?>
```

# Local API 활용

- Local API를 활용하고 있는 RestaurantListActivity.kt
- SQL 쿼리를 전송할 때와 동일한 방식으로 통신
- 장소의 이름, 주소, 카테고리, 전화번호, 좌표 제공
- 한식, 일식, 중식 등 특정 카테고리  
의 맛집 리스트를 불러와  
화면에 나열 가능

```
// 음식점 리스트 불러오기
var places = JSONArray() // 장소 정보를 저장할 변수
var thread = Thread() {
    val http = HttpUtil()
    var pageNum = 1 // 불러올 결과 페이지 번호
    while(true){ // 결과 페이지가 더이상 없을때까지 반복
        val httpResult = http.post(
            page: "search_place",
            selectedGenre.toString(),
            extra: "구정문",
            pageNum)
        places = httpResult.getJSONArray( name: "documents") // 검색 결과 배열
        runOnUiThread {
            for (i in 0 ≤ ..<places.length()) { // 페이지 내 결과 안에서 반복
                // 여기에 동적 뷰 생성 구현
                val place = places.getJSONObject(i) // i 번째 장소 객체

                // 테스트 로그
                Log.v( tag: "place", place.getString( name: "place_name"))
            }
        }
    }
}
```

752 place	com.mop.bdm	V	대성갈비
752 place	com.mop.bdm	V	동보집 전복대점
752 place	com.mop.bdm	V	대성갈비
752 place	com.mop.bdm	V	호구곰창 본점
752 place	com.mop.bdm	V	통달이곰창
752 place	com.mop.bdm	V	뉴욕닭갈비
752 place	com.mop.bdm	V	나리곰창
752 place	com.mop.bdm	V	문짜갈비
752 place	com.mop.bdm	V	오복보쌈
752 place	com.mop.bdm	V	구가네닭발 전복대점
752 place	com.mop.bdm	V	냉돈창고 전주덕진점
752 place	com.mop.bdm	V	곱닭

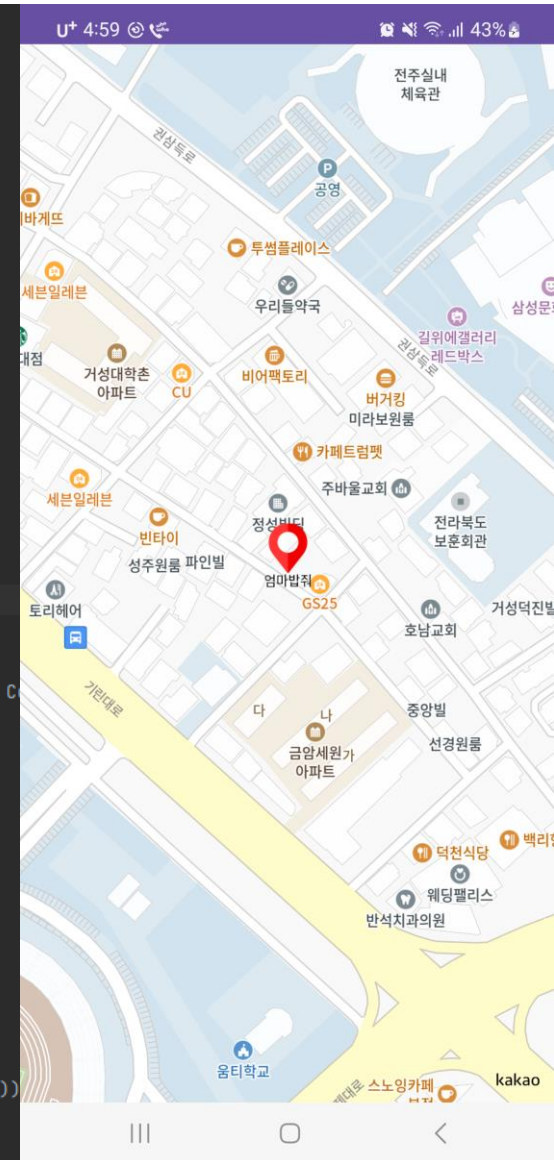
# Maps API 활용

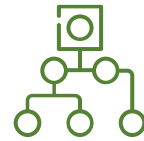
- Maps API를 활용하고 있는 MapActivity.kt
- MapActivity를 호출한 액티비티로부터 좌표를 전달받음
- mapView.start() 메소드로 화면에 지도를 그림
- 해당 좌표에 텍스트와 라벨을 표시
- 해당 좌표로 카메라 이동

```
val mapView = findViewById<MapView>(R.id.map_view)
// 카카오맵 그리기
mapView.start(object : MapLifecycleCallback() {
    override fun onMapDestroy() {
        // 지도 API가 정상적으로 종료될 때 호출됨
    }

    override fun onMapError(error: Exception) {
        // 인증 실패 및 지도 사용 중 에러가 발생할 때 호출됨
        Log.v( tag: "MapView", msg: "OnMapError() : $error")
    }
}, object : KakaoMapReadyCallback() {
    override fun onMapReady(kakaoMap: KakaoMap) {
        // 인증 후 API가 정상적으로 실행될 때 호출됨
        // 표시할 장소 좌표
        val position = LatLng.from(
            intent.getStringExtra( name: "y")!!.toDouble(),
            intent.getStringExtra( name: "x")!!.toDouble()
        )
        // 라벨 스타일
        val styles = kakaoMap.labelManager!!.addLabelStyles(
            LabelStyles.from(
                LabelStyle.from(R.drawable.pin_loc).setTextStyles( size: 20, C
            )
        )
        // 라벨 옵션
        val options = LabelOptions.from(position).setStyles(styles)
        // 라벨 레이어
        val layer = kakaoMap.labelManager!!.layer
        // 라벨 표시
        val label = layer!!.addLabel(options)
        // 장소 이름 표시
        label.changeText(intent.getStringExtra( name: "name"))

        // 지도 화면 이동 및 확대 수준 설정
        kakaoMap.moveCamera(CameraUpdateFactory.newCenterPosition(position))
        kakaoMap.moveCamera(CameraUpdateFactory.zoomTo( zoomLevel: 17))
    }
})
```

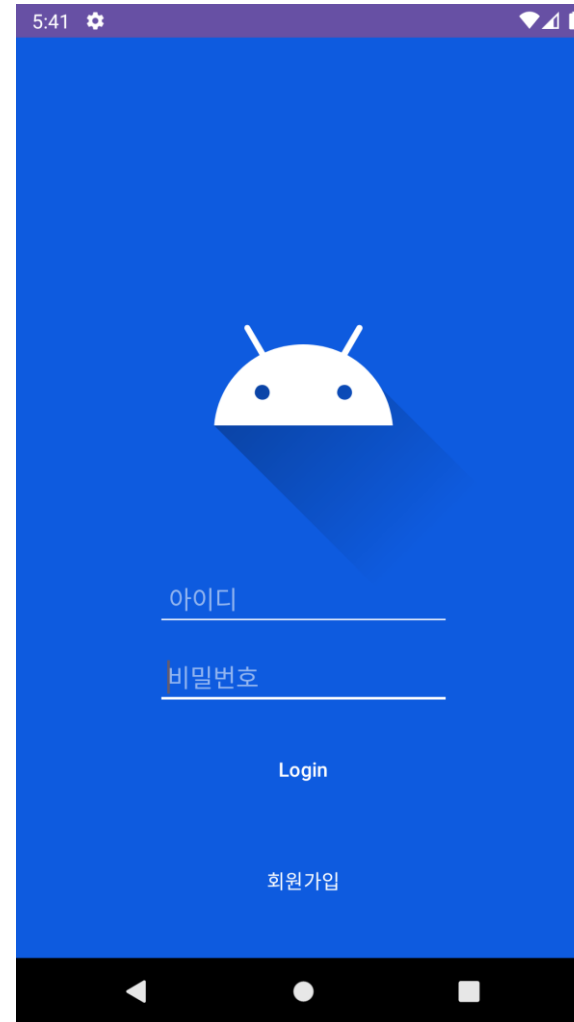




### 3. 주요 기능 설명

# 로그인 및 회원가입

- 초기 버전으로 현재는 기능 위주의 구현만 진행된 상태
- HTTP 통신을 통해 MySQL에 접근하여 아이디와 비밀번호 일치 시에만 로그인 가능하게 구현
- 회원가입 페이지의 아이디, 닉네임의 중복확인 기능 구현
- 빈 필드 값 혹은 사전 정의한 정규식과 불일치하는 값이 존재할 경우 등 여러 케이스에 대응



5:41

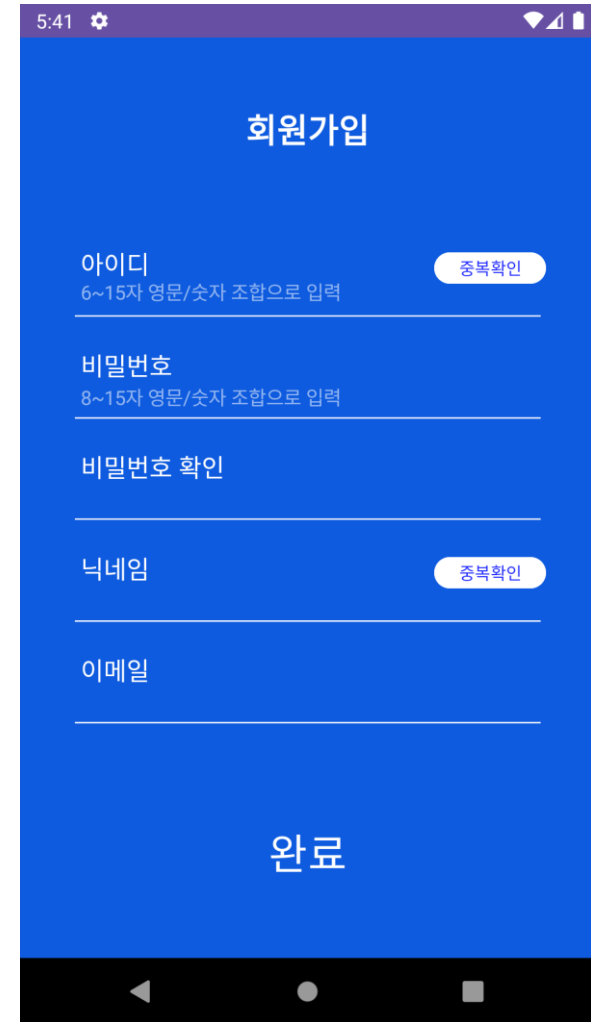
아이디

비밀번호

Login

회원가입

The login screen features a blue background with a white Android robot icon in the center. Below the icon are two input fields labeled '아이디' (ID) and '비밀번호' (Password). At the bottom, there are two buttons: 'Login' and '회원가입' (Sign Up).



5:41

회원가입

아이디  
6~15자 영문/숫자 조합으로 입력

중복확인

비밀번호  
8~15자 영문/숫자 조합으로 입력

비밀번호 확인

닉네임

중복확인

이메일

완료

The sign-up screen has a blue background. At the top, it says '회원가입' (Sign Up). Below this are four input fields: '아이디' (ID) with a '중복확인' (Check for duplicates) button, '비밀번호' (Password) with a '중복확인' button, '비밀번호 확인' (Confirm password), and '이메일' (Email). At the bottom, there is a '완료' (Done) button.

# LoginActivity.kt

- PHP서버와 통신하며 로그인 기능을 구현한 코드
- SQL문을 PHP서버에 직접 전송
- 쿼리 결과가 "success"이고 메시지가 "empty"가 아닐 때 로그인 성공 처리
- 통신은 성공했지만 내용이 "empty"일 때에는 아이디/패스워드의 불일치로 처리됨
- 연속 클릭으로 중복적인 통신 동작을 하지 않도록 통신 시작 전 버튼을 비활성화 함

```
// 빈칸 제출시 Toast
if (user == "" || pass == "") {
    Toast.makeText( context: this@LoginActivity, text: "아이디와 비밀번호를 모두 입력해주세요.", Toast.LENGTH_SHORT).show()
}
else {
    // 백그라운드 통신을 위한 스레드
    v.isEnabled = false
    var thread = Thread {
        val query = "SELECT userid FROM user WHERE userid='$user' AND userpw='$pass'"
        val httpResult = http.post( page: "sql", query)
        Log.v( tag: "Login", httpResult.getString( name: "result")) // 로그인 결과 로그
        runOnUiThread { // UI 작업은 메인 스레드에서만 가능하므로 따로 작업
            // id 와 password 일치시
            if (httpResult.getString( name: "result") == "success") {
                if(httpResult.getString( name: "message") == "empty"){
                    Toast.makeText( context: this@LoginActivity,
                        text: "아이디와 비밀번호를 확인해 주세요.",
                        , Toast.LENGTH_SHORT).show()
                }
            }
            else {
                Toast.makeText( context: this@LoginActivity,
                    text: "로그인 되었습니다."
                    , Toast.LENGTH_SHORT).show()
                val intent = Intent( packageContext: this, MainActivity::class.java)
                startActivity(intent)
            }
        }
    }
    // 에러 발생시
    else{
        Toast.makeText( context: this@LoginActivity,
            httpResult.getString( name: "message")
            , Toast.LENGTH_SHORT).show()
        Log.v( tag: "Login", httpResult.getString( name: "message")) // 에러 로그
    }
    v.isEnabled = true
}
}
thread.start()
}
```



# RegisterActivity.kt

```
// 완료 버튼 클릭 시
btnRegister.setOnClickListener { v ->
    val user = editTextId.text.toString()
    val pass = editTextPassword.text.toString()
    val repass = editTextRePassword.text.toString()
    val nick = editTextNick.text.toString()
    val email = editTextEmail.text.toString()
    val pwPattern = "^(?=.*[A-Za-z])(?=.*[0-9])[A-Za-z0-9]{8,15}$"
    val emailPattern = "^[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*@[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*.[a-zA-Z]{2,3}$"

    // 사용자 입력이 비었을 때
    if (user == "" || pass == "" || repass == "" || nick == "" || email == "") {
        Toast.makeText(
            context, this@RegisterActivity,
            text: "회원정보를 모두 입력해주세요.",
            Toast.LENGTH_SHORT
        ).show()
        return@setOnClickListener
    }

    // 아이디 중복
    if (!checkId) {
        Toast.makeText(context, this@RegisterActivity, text: "아이디 중복확인을 해주세요.", Toast.LENGTH_SHORT).show()
        return@setOnClickListener
    }

    // 비밀번호 형식 불일치
    if (!Pattern.matches(pwPattern, pass)) {
        Toast.makeText(
            context, this@RegisterActivity,
            text: "비밀번호 형식이 옳지 않습니다.",
            Toast.LENGTH_SHORT
        ).show()
        return@setOnClickListener
    }

    // 비밀번호 재확인 불일치
    if (pass != repass) {
        Toast.makeText(
            context, this@RegisterActivity,
            text: "비밀번호가 일치하지 않습니다.",
            Toast.LENGTH_SHORT
        ).show()
        return@setOnClickListener
    }
}
```

```
// 모든 케이스 만족시
v.isEnabled = false
var thread = Thread{
    val query = "INSERT INTO user (username, userid, userpw, useremail)" +
        "VALUES ('$nick', '$user', '$pass', '$email')"
    val httpResult = http.post(page: "sql", query) // 파싱 오류는 HttpUtil.kt 에서 사전에
    runOnUiThread {
        // 가입 성공 시 Toast를 띄우고 메인 화면으로 전환
        if (httpResult.get("result") == "success") {
            Toast.makeText(
                context, this@RegisterActivity,
                text: "가입되었습니다.",
                Toast.LENGTH_SHORT
            ).show()
            val intent =
                Intent(applicationContext, LoginActivity::class.java)
            startActivity(intent)
        }
        // 가입 실패 시
        else {
            Toast.makeText(
                context, this@RegisterActivity,
                text: "가입 실패하였습니다.",
                Toast.LENGTH_SHORT
            ).show()
            Log.v(tag: "Login", httpResult.getString(name: "message")) // 가입 실패 로그
            Log.v(tag: "Login", httpResult.getString(name: "query")) // 가입 실패 로그
        }
        v.isEnabled = true
    }
}
thread.start()
```

# 메인 화면

- GridLayout 을 이용한 메인 메뉴 구현
- 각 메뉴 버튼 클릭 시 openCategoryPage() 를 통해 리스트 액티비티로 전환
- 해당하는 카테고리의 이름을 putExtra() 메소드를 이용해 리스트 액티비티에 전달

```
hansikButton.setOnClickListener { it: View!
    openCategoryPage( category: "한식")
}
pastaButton.setOnClickListener { it: View!
    openCategoryPage( category: "양식")
}
chinaButton.setOnClickListener { it: View!
    openCategoryPage( category: "중식")
}
sushiButton.setOnClickListener { it: View!
    openCategoryPage( category: "일식")
}
fastfoodButton.setOnClickListener { it: View!
    openCategoryPage( category: "패스트푸드")
}
beerButton.setOnClickListener { it: View!
    openCategoryPage( category: "술집")
}
bbqButton.setOnClickListener { it: View!
    openCategoryPage( category: "고기")
}
tteokbokkiButton.setOnClickListener { it: View!
    openCategoryPage( category: "분식")
}
coffeeButton.setOnClickListener { it: View!
    openCategoryPage( category: "카페")
}
}

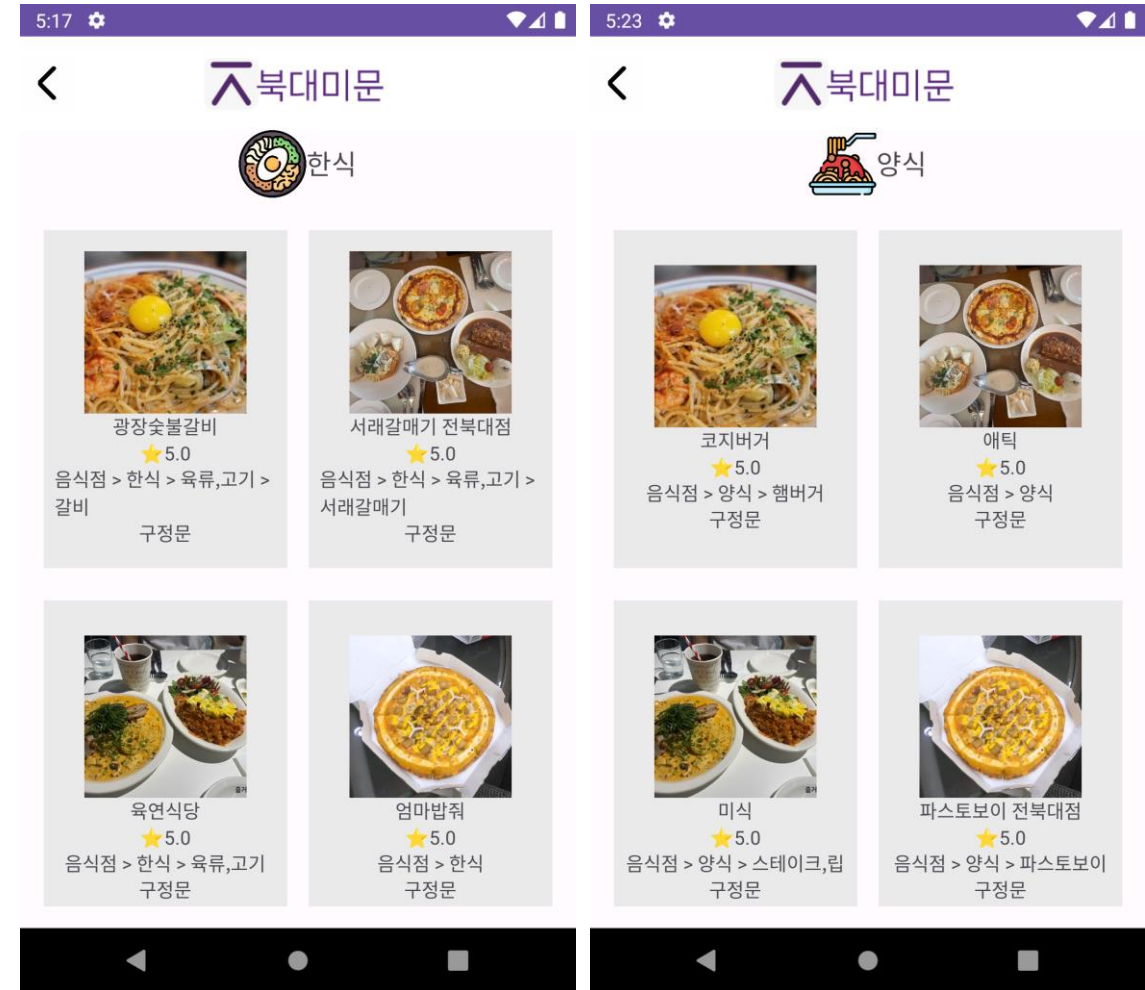
private fun openCategoryPage(category: String) {
    val intent = Intent( packageContext, RestaurantListActivity::class )
    intent.putExtra( name: "category", category)
    startActivity(intent)
}
```



# 맛집 리스트

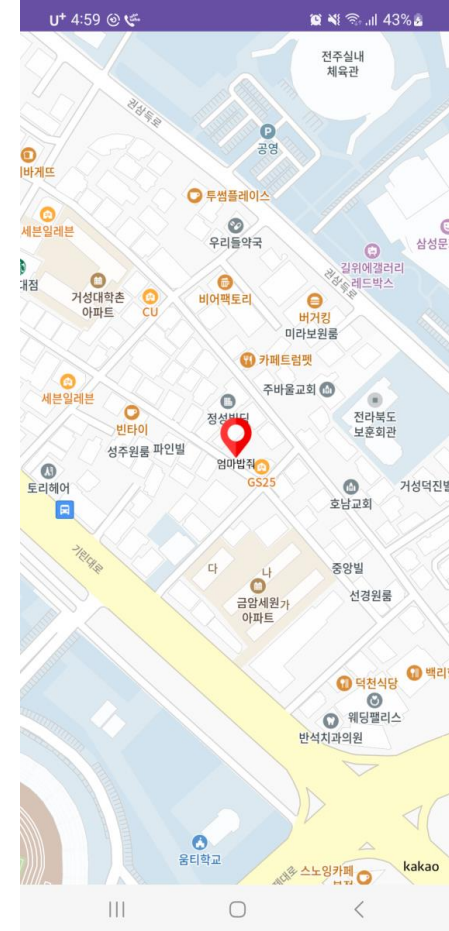
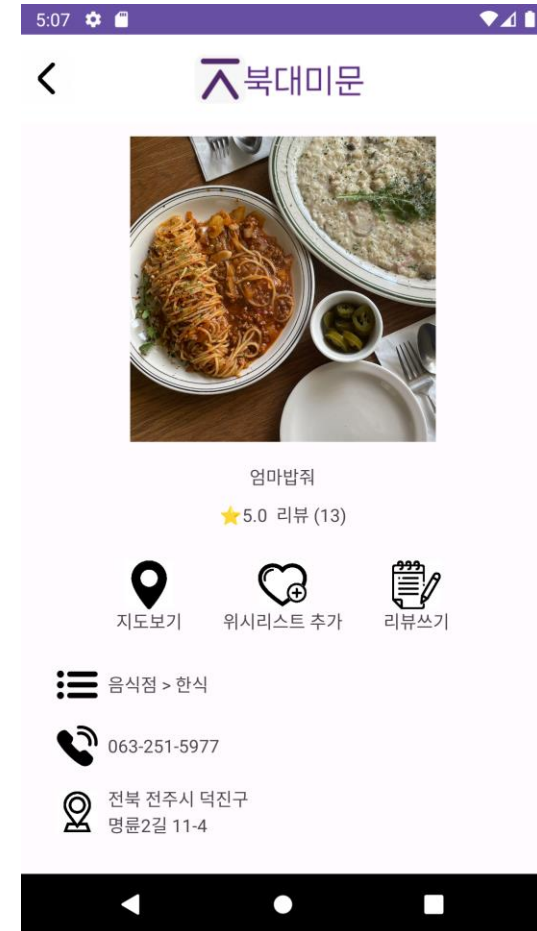
- 메인 메뉴로부터 전달받은 카테고리 이름으로 Local API에 검색
- 검색 결과를 정적 뷰에 표시(임시)
- 각 맛집을 클릭할 경우 상세페이지로 전환
- 상세페이지 전환 시 클릭한 장소 정보를 액티비티에 전달

```
private fun navigateToRestaurantDetail(placeInfo: JSONObject) {  
    val intent = Intent(packageContext, RestaurantDetailActivity::class.java)  
    // 상세페이지 액티비티에 장소 정보 전달  
    intent.putExtra(name: "name", placeInfo.getString(name: "place_name"))  
    intent.putExtra(name: "category", placeInfo.getString(name: "category_name"))  
    intent.putExtra(name: "address", placeInfo.getString(name: "road_address_name"))  
    intent.putExtra(name: "phone", placeInfo.getString(name: "phone"))  
    intent.putExtra(name: "x", placeInfo.getString(name: "x"))  
    intent.putExtra(name: "y", placeInfo.getString(name: "y"))  
    startActivity(intent)  
}
```



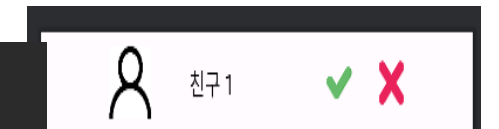
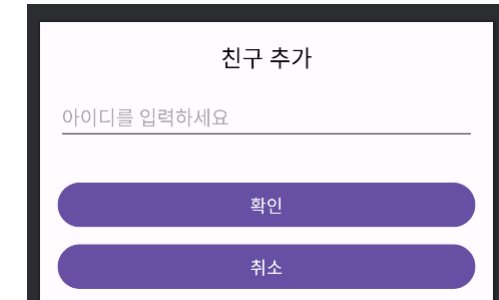
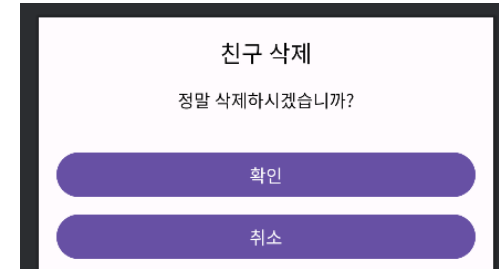
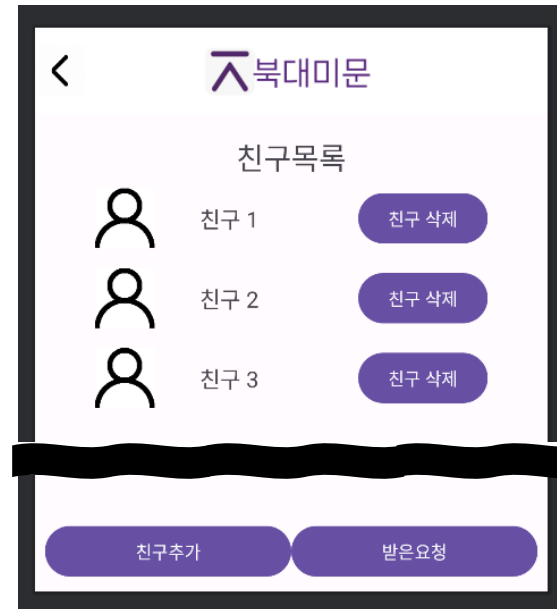
# 맛집 상세 페이지

- 특정 맛집의 상세 페이지를 ScrollView 로 구현
- RestaurantList 액티비티 뿐만 아니라 맛집 랭킹, 위시리스트, 제휴업체 리스트 등 여러 액티비티로부터 접근 가능하도록 구현
- Intent로 전달받은 장소 정보를 레이아웃에 표시
- 지도보기 버튼 클릭 시 해당 장소를 지도에 표시



# 친구목록

- 친구삭제 버튼 클릭 시 친구삭제 대화상자 열림. 확인 및 취소버튼에 대한 클릭 리스너 설정  
확인버튼 클릭 시 친구 항목 뷰를 레이아웃에서 삭제
- createFriendRequestItem으로 친구 요청 항목 동적으로 생성후 레이아웃 추가
- 수락 또는 거부버튼으로 친구추가 및 추가요청 삭제를 위한 API 호출, 친구요청 항목 레이아웃에서 제거



```
private fun createFriendRequestItem(request: FriendRequest) {  
    val inflater = LayoutInflater.from(context, this)  
    val friendRequestItem = inflater.inflate(R.layout.friend_request_item, root, null)  
  
    val nameTextView = friendRequestItem.findViewById<TextView>(R.id.friendRequestName)  
    val acceptButton = friendRequestItem.findViewById<ImageView>(R.id.acceptButton)  
    val declineButton = friendRequestItem.findViewById<ImageView>(R.id.declineButton)  
  
    val friendRequestsLayout = findViewById<LinearLayout>(R.id.friendRequestsLayout)  
    nameTextView.text = request.name
```

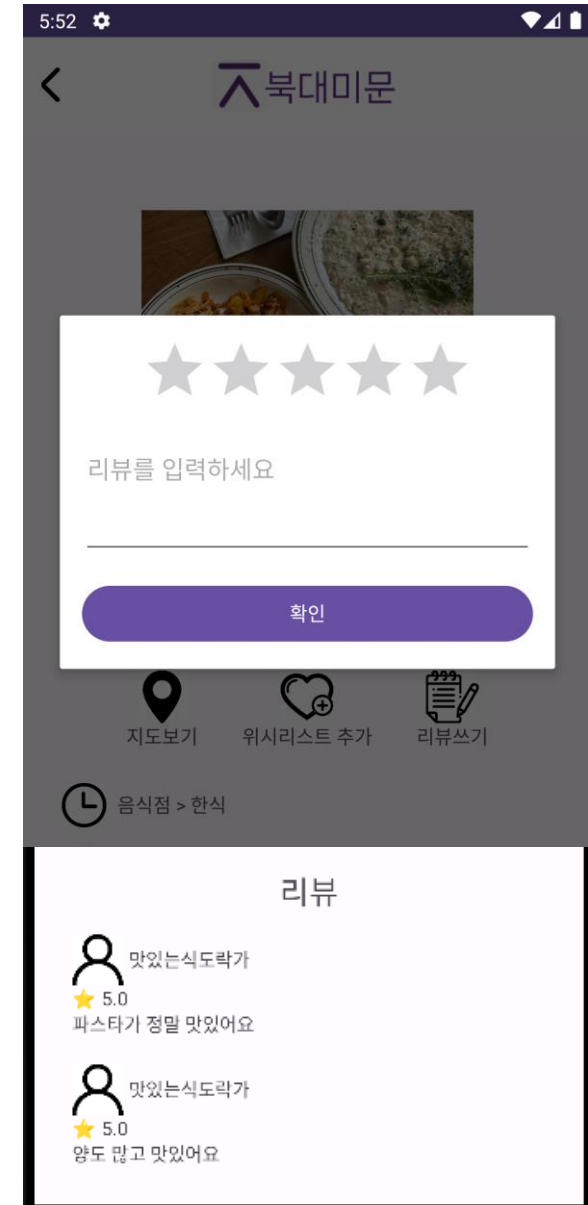
```
deleteFriendButton.setOnClickListener { it: View? -> {  
    val inflater = LayoutInflater.from(context, this)  
    val view = inflater.inflate(R.layout.friend_deletion_modal, root, null)  
  
    val confirmButton = view.findViewById<Button>(R.id.buttonConfirmDelete)  
    val cancelButton = view.findViewById<Button>(R.id.buttonCancelDelete)  
  
    val dialog = AlertDialog.Builder(context, this).AlertDialogBuilder()  
        .setView(view) .AlertDialog.Builder()  
        .create()  
  
    confirmButton.setOnClickListener { it: View? -> {  
        // 친구 삭제 API  
        friendsLayout.removeView(friendItemView) // 친구 삭제  
        dialog.dismiss()  
    }  
  
    cancelButton.setOnClickListener { it: View? -> {  
        dialog.dismiss()  
    }  
  
    dialog.show()  
}
```

# 리뷰 작성

- 리뷰 쓰기 버튼 클릭 시 showReviewPopup() 메소드를 통해 리뷰 팝업 표시
- createReviewLayout() 메소드를 통해 항목을 인플레이트 하여 리뷰 내용을 표시
- 현재까지는 DB와 연동되지 않은 상태

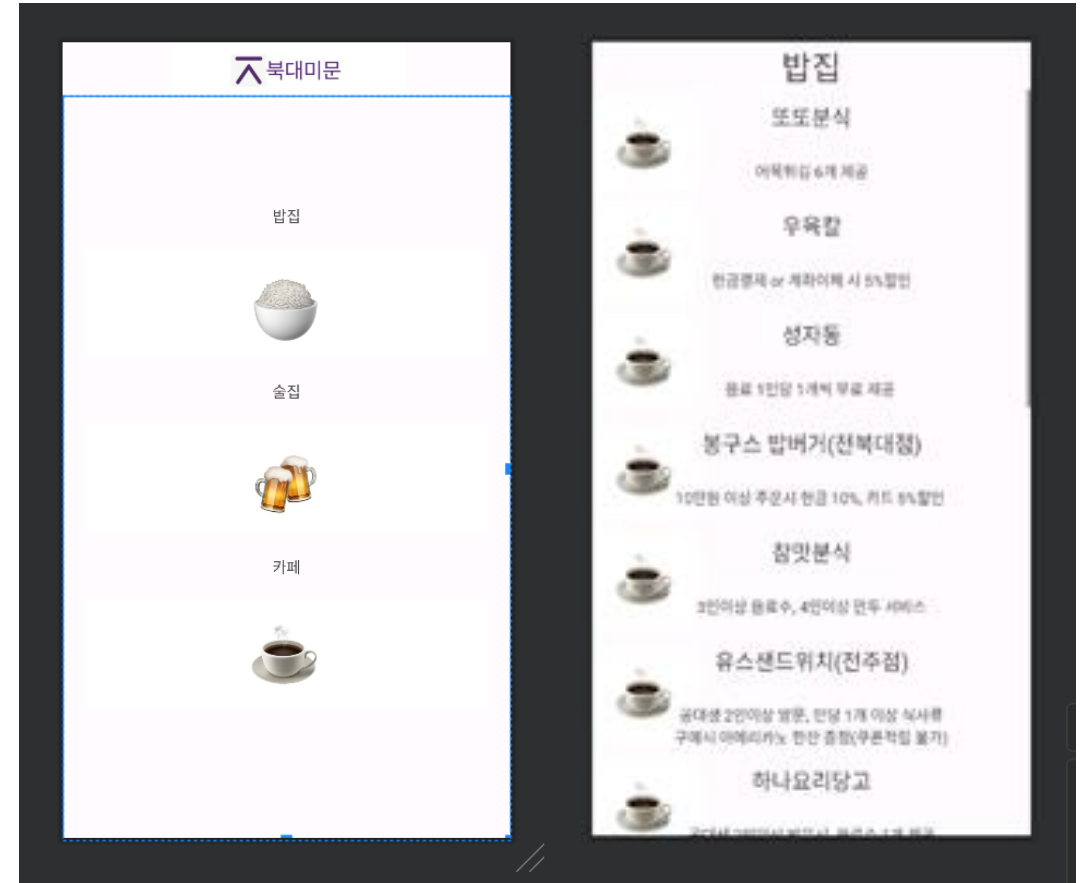
```
private fun showReviewPopup() {  
    val inflater = LayoutInflater.from(context) { this }  
    val view = inflater.inflate(R.layout.popup_write_review, root = null)  
  
    val editTextReview = view.findViewById<EditText>(R.id.editTextReview)  
    val buttonSubmitReview = view.findViewById<Button>(R.id.buttonSubmitReview)  
    val ratingBar = view.findViewById<RatingBar>(R.id.ratingBar) // Add this line  
  
    val dialog = AlertDialog.Builder(context) { this } AlertDialog.Builder()  
        .setView(view) AlertDialog.Builder()  
        .create()  
  
    buttonSubmitReview.setOnClickListener { it: View? -> {  
        val newReviewText = editTextReview.text.toString()  
  
        val userRating = ratingBar.rating  
  
        val newReviewLayout = createReviewLayout(newReviewText, userRating)  
  
        reviewsLayout.addView(newReviewLayout)  
  
        dialog.dismiss()  
    }  
  
    dialog.show()  
}
```

```
private fun createReviewLayout(reviewText: String, userRating: Float): View {  
    val inflater = LayoutInflater.from(context) { this }  
    val reviewLayout = inflater.inflate(R.layout.review_item, root = null)  
  
    val ratingTextView = reviewLayout.findViewById<TextView>(R.id.starRating3)  
    val ratingValueTextView = reviewLayout.findViewById<TextView>(R.id.ratingValue2)  
    val reviewTextView = reviewLayout.findViewById<TextView>(R.id.ratingText1)  
    val deleteButton = reviewLayout.findViewById<Button>(R.id.deleteButton) // Add delete  
  
    val uniqueTag = "review_${System.currentTimeMillis()}"  
    reviewLayout.tag = uniqueTag  
  
    ratingTextView.text = "★"  
    ratingValueTextView.text = userRating.toString()  
    reviewTextView.text = reviewText  
  
    deleteButton.setOnClickListener { it: View? -> {  
        removeReviewByTag(uniqueTag) // Remove the review item based on its unique tag  
    }  
  
    return reviewLayout  
}
```



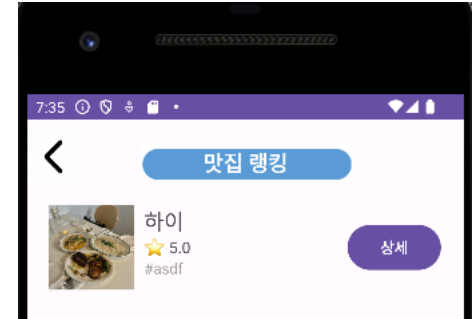
# 제휴업체리스트

- ScrollView를 사용한 정적 리스트
- 사전에 조사한 제휴 업체들을 3가지로 분류
- 사진, 이름, 제휴내용 표시
- 클릭 시 상세 페이지로 이동할 수 있도록 연결 중



# 맛집 랭킹

- AddDynamicData 메서드를 통해 동적으로 데이터 추가
- 상세 버튼(클릭 리스너를 통해 RestaurantDetailActivity로 이동)



```
// 동적으로 데이터를 추가하는 메서드
private fun addDynamicData() {
    val dynamicItem = Rankingitem(
        ContextCompat.getDrawable(context: this, R.drawable.star)!!,
        name: "동적 추가",
        getDrawable(R.drawable.position)!!,
        grade: "4.5",
        explain: "#동적데이터"
    )
}
```

```
btn.setOnClickListener { it: View!
    val intent = Intent(it.context, RestaurantDetailActivity::class.java)
    intent.putExtra(name: "name", item.name)
    it.context.startActivity(intent)
}
```





# 위시리스트

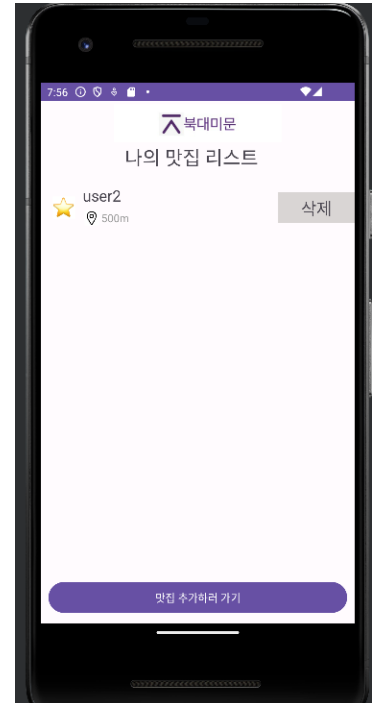
- addNewItem메서드를 통해 동적으로 아이템 추가
- 리사이클러뷰 삭제 버튼 구현(클릭리스너)
- 하단 맛집 추가하러 가기 버튼을 통해 메인액티비티로 전환

```
removeTextView.setOnClickListener { it: View!
    val list = arrayListOf<WishListViewitem>()
    list.addAll(differ.currentList)
    list.remove(wish)

    // 해당 아이템 삭제 adapter에 알리기
    differ.submitList(list)
}
```

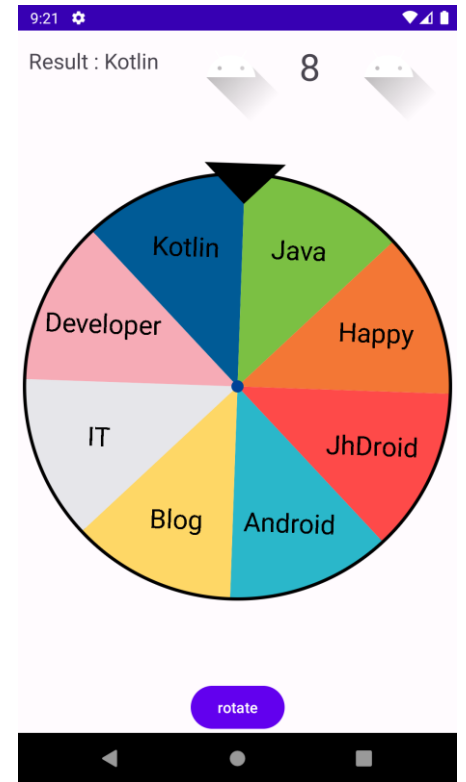
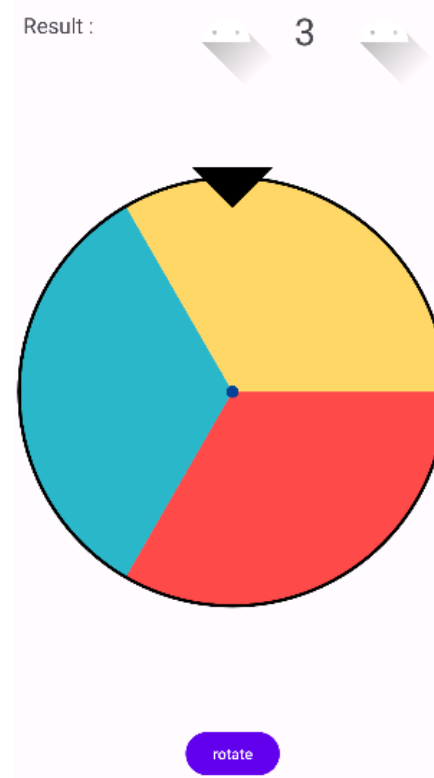
```
private fun setupEvents() {
    binding.addButton.setOnClickListener { it: View!
        val intent = Intent( packageContext: this, MainActivity::class.java)
        startActivity(intent)
    }
}
```

```
val mWish = WishListViewitem( id: 1, ContextCompat.getDrawable( context: this, R.drawable.star)!!,
    title: "user2",
    drawable(R.drawable.position)!!,
    subTitle: "500m")
wishList.add(mWish)
```



# 룰렛 스피너

- 데이터 바인딩을 이용한 커스텀 뷰로 구현
- 별도의 인터페이스를 구성하여 회전 이벤트 처리
- 룰렛의 구체적인 동작은 Roulette.kt 에서 정의
- 캔버스 안에서 그래픽 요소를 정의하여 룰렛을 그림
- 밥값내기, 맛집 랜덤 고르기, 리뷰 작성 포인트 지급 기능 등에 활용할 예정



# Roulette.kt

- onDraw() : 룰렛의 그래픽 요소를 그리기 위해 호출
- drawRoulett() : 룰렛의 외곽선, 항목 수와 색상, 텍스트 결정
- drawTopMarker() : 룰렛 상단 마커의 모양 결정하고 화면에 그림

```
@SuppressLint("DrawAllocation")
override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    if (canvas == null) return

    val shorterLength = min(right, height)

    val rectLeft = (width / 2f) - (shorterLength / 2f) + paddingLeft + Constant.
    val rectRight = (width / 2f) + (shorterLength / 2f) - paddingRight - Constar
    val rectTop = (height / 2f) - (shorterLength / 2f) + paddingTop + Constant.f
    val rectBottom = (height / 2f) + (shorterLength / 2f) - paddingBottom - Cons

    rectF.set(rectLeft, rectTop, rectRight, rectBottom)

    centerX = (rectF.left + rectF.right) / 2f
    centerY = (rectF.top + rectF.bottom) / 2f

    drawRoulette(canvas, rectF)

    if (centerPointVisibility == VISIBLE) {
        canvas.drawCircle(centerX, centerY, centerPointRadius, centerPointPaint)
    }

    if (topMarkerVisibility == VISIBLE) {
        drawTopMarker(canvas, rectF)
    }
}
```

```
private fun drawRoulette(canvas: Canvas, rectF: RectF) {
    // draw roulette border line
    canvas.drawArc(rectF, startAngle: 0f, sweepAngle: 360f, useCenter: false, strokePaint)

    if (rouletteSize in 2..8) {
        val sweepAngle = 360f / rouletteSize.toFloat()
        val radius = (rectF.right - rectF.left) / 2 * 0.7

        for (i in 0..until < rouletteSize) {
            fillPaint.color = Color.parseColor(shapeColors[i])

            // draw roulette arc
            val startAngle = if (i == 0) 0f else sweepAngle * i
            canvas.drawArc(rectF, startAngle, sweepAngle, useCenter: true, fillPaint)

            // draw roulette text
            val medianAngle = (startAngle + sweepAngle / 2f) * Math.PI / 180f
            val x = (centerX + (radius * cos(medianAngle))).toFloat()
            val y = (centerY + (radius * sin(medianAngle))).toFloat() + Constant.DE
            val text = if (i > rouletteDataList.size - 1) emptyMessage else roulette

            canvas.drawText(text, x, y, textPaint)
        }
    } else {
        throw IndexOutOfBoundsException("size out of roulette")
    }
}
```

```
private fun drawTopMarker(canvas: Canvas, rectF: RectF) {
    val path = Path()
    val y = rectF.top - 30f

    val point1 = PointF(x: centerX - Constant.DEFAULT_TOP_MARKER_LENGTH, y) // 왼쪽 포인트
    val point2 = PointF(x: centerX + Constant.DEFAULT_TOP_MARKER_LENGTH, y) // 오른쪽 포인트
    val point3 = PointF(centerX, y: rectF.top - 30f + Constant.DEFAULT_TOP_MARKER_LENGTH) /

    path.reset()

    path.moveTo(point1.x, point1.y)
    path.lineTo(point2.x, point2.y)
    path.lineTo(point3.x, point3.y)
    path.lineTo(point1.x, point1.y)

    path.close()

    canvas.drawPath(path, topMarkerPaint)
}
```

# Roulette.kt

- rotateRoulett() : 룰렛을 회전시키고 애니메이션 적용, rotateListener 에 회전 결과 전달
- getRouletteRotateResult() : 룰렛의 회전 결과를 계산, 회전 각도에 따라 선택된 항목을 결정하여 반환

```
fun rotateRoulette(toDegrees: Float, duration: Long, rotateListener: RotateListener?) {  
    val animListener = object : Animation.AnimationListener {  
        override fun onAnimationRepeat(animation: Animation?) {}  
  
        override fun onAnimationStart(animation: Animation?) {  
            rotateListener?.onRotateStart()  
            isRotate = true  
        }  
  
        override fun onAnimationEnd(animation: Animation?) {  
            rotateListener?.onRotateEnd(getRouletteRotateResult(toDegrees))  
            isRotate = false  
        }  
    }  
  
    val rotateAnim = RotateAnimation(  
        fromDegrees: 0f, toDegrees,  
        Animation.RELATIVE_TO_SELF, pivotXValue: 0.5f,  
        Animation.RELATIVE_TO_SELF, pivotYValue: 0.5f  
    )  
    rotateAnim.duration = duration  
    rotateAnim.fillAfter = true  
    rotateAnim.setAnimationListener(animListener)  
  
    startAnimation(rotateAnim)  
}
```

```
private fun getRouletteRotateResult(toDegrees: Float): String {  
    val moveDegrees = toDegrees % 360  
    val resultAngle = if (moveDegrees > 270) 360 - moveDegrees + 270 else 270 - moveDegrees  
    for (i in 1..rouletteSize) {  
        if (resultAngle < (360 / rouletteSize) * i) {  
            if (i - 1 >= rouletteDataList.size) {  
                return emptyMessage  
            }  
  
            return rouletteDataList[i - 1]  
        }  
    }  
  
    return ""  
}
```

# 애로사항

1. 카카오맵 API가 공식적으로 코틀린을 지원하지 않아서 학습에 어려움 겪음
2. HTTP 통신과 JSON 파싱에서의 빈번한 오류 발생
3. 맛집 리스트 제작 과정 중 DB에서 정보를 받아오는 과정
4. DB에 리뷰를 저장하는 과정
5. 여전히 더 많은 기능 구상이 필요함

# 개선 방향 및 향후 계획

- 맛집 리스트, 위시리스트, 맛집 랭킹 등에 동적 뷰 생성 적용
- 친구 추가 기능 구상
  - > 서로 맛집 위시리스트를 공유하며 친분을 쌓는데 도움
- 리뷰 작성을 장려하기 위한 포인트 시스템 구상
  - > 유저 랭킹으로 연결 혹은 **광고 수익 분배**

맛집 지도 플랫폼 어디까지 의 또하나 중요한 특징이자 장점은 맛집 지도를 제작하는 유저들에게는 광고 수익이 리워드로 제공된다는 점입니다. 맛집 지도가 생성이 되면 트래픽에 따라 구글 애드센스 수익이 발생하게 되는데요, 어디까지 수수료를 제외한 광고 수익은 매달 맛집 지도를 생성해 맛집 추천 한 유저에 리워드로 **ABL** 토큰 을 지급을 해 줍니다. 어디까지 가 지급한 ABL 토큰은 카카오펀드와 연동이 되어 있어서 KLIP 이라는 카카오 가상자산 지갑에서 확인할 수 있습니다.

**ABL** 토큰은 이더리움 기반 유틸리티 토큰으로 한마디로 **가상자산** 입니다. ABL 토큰은 이미 암호화폐 거래소에서 거래가 가능한 거래형 토큰입니다. 기호에 따라 바로 현금화를 해서 사용을 해도 되고 이번 기회에 가상자산을 보유하는 경험을 하고 싶다면 장기보유 등을 통한 투자도 가능합니다.



A decorative graphic consisting of a thin black horizontal line and a thin black vertical line intersecting at the top left. A small green horizontal bar is at the top right, and a small green vertical bar is at the bottom left.

# 감사합니다