


RE: 수행보고서 및 졸업논문 확인 부탁드립니다. ☞

보낸사람  이보경 <bkiee@pu.ac.kr>

받는사람 이경은 <tm0114@naver.com>

참조 <bkiee@pu.ac.kr>

경은,

나의 영문이름은 Bo Kyung Lee 로 해주세요.

보고서 및 논문을 확인하였음

이보경교수

20201201

종합설계 프로젝트 수행 보고서

프로젝트명	교통 약자를 위한 이동 지원 시스템
팀번호	S5-7
문서제목	수행계획서() 2차발표 중간보고서() 3차발표 중간보고서() 4차발표 중간보고서() 최종결과보고서(O)

2020.12.04

팀원 : 이경은 (팀장)

이자훈

전현정

지도교수 : 이보경

문서 수정 내역

작성일	대표작성자	버전(Revision)	수정내용	
2020.01.18	전현정	1.0	수행계획서	최초작성
2020.02.29	이경은	2.0	2차 발표자료	설계서 추가
2020.04.27	이자흔	3.0	UI 설계	프로토타입 추가
2020.06.03	이경은	3.1	UI 설계 수정	프로토타입 수정
2020.06.27	이자흔	4.0	4차 수행보고서	시험/테스트 결과, 코딩 및 데모 추가
2020.11.16	이자흔	4.1	4차 수행보고서	시험/테스트 결과, 코딩 및 데모 수정
2020.12.04	전현정	5.0	최종수행보고서	결론 추가

문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (2월)	중간발표2 (4월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	I. 서론 (1~6) II. 본론 (1~3) 참고자료	I. 서론 (1~6) II. 본론 (1~4) 참고자료	I. 서론 (1~6) II. 본론 (1~5) 참고자료	I. 서론 (1~6) II. 본론 (1~7) 참고자료	I II III

이 문서는 한국산업기술대학교 컴퓨터공학부의
“종합설계”교과목에서 프로젝트
“교통 약자를 위한 이동 지원 시스템”을 수행하는
(S5-7, 이경은, 이자흔, 전현정)들이 작성한 것으로
사용하기 위해서는 팀원들의 허락이 필요합니다.

목 차

I. 서론

1. 작품선정 배경 및 필요성	4p
2. 기존 연구/기술동향 분석	5~6p
3. 개발 목표	6p
4. 팀 역할 분담	7p
5. 개발 일정	7p
6. 개발 환경	7p

II. 본론

1. 개발 내용	8p
2. 문제 및 해결방안	8p
3. 시험시나리오	9~11p
4. 상세 설계	11~25p
5. Prototype 구현	26~29p
6. 시험/ 테스트 결과	30~33p
7. Coding & DEMO	34~58p

III. 결론

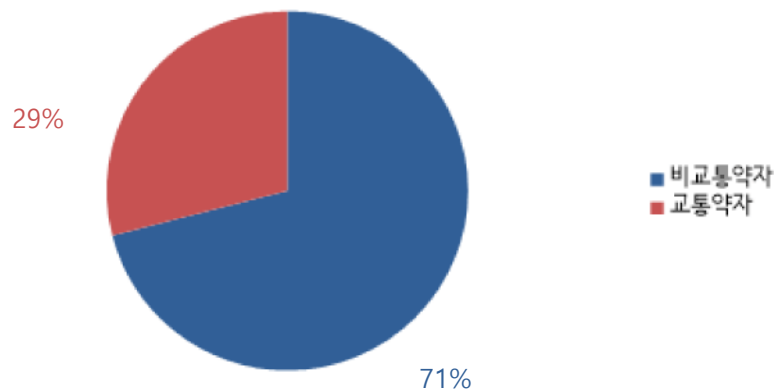
1. 연구 결과	59p
----------------	-----

❖ 참고자료	60p
--------------	-----

I. 서론

1. 작품선정 배경 및 필요성

'교통 약자'란 장애인, 고령자, 임산부, 영유아를 동반한 사람, 어린이 등 일상생활에서 이동에 불편을 느끼는 사람을 말함.¹⁾ 국토교통부, 한국교통안전공단에서 실시한 2018년도 교통 약자 실태조사에 따르면 국내 교통 약자 수는 지난해 대비 26만 명 증가한 1509만 명, 이는 4명 중 1명 이상으로 전체 인구의 29%에 달함.²⁾ 또한, 그 인구는 점차 증가하고 있음.



<그림 1> 교통 약자 비율 (2018년 기준)

그리고 도로 곳곳의 장애물, 도로 구조 문제 등으로 이동권을 상실한 교통 약자들이 불편함을 겪고 있음. 최근 3년간(2016년도~2018년) 대부분의 9대 광역 도 단위에서 영유아와 어린이보행자 교통사고 사상자 수는 감소하는 것으로 나타났으나, 고령자 보행자 교통사고는 증가하고 있는 것으로 나타남.

구분	2016년			2017년			2018년		
	영유아	어린이	고령자	영유아	어린이	고령자	영유아	어린이	고령자
전국	5,014	9,272	40,145	4,633	8,854	42,346	4,468	8,109	43,515
9개 도 단위	3,225	5,973	26,335	3,017	5,608	27,975	2,888	5,256	28,270

<표 1> 최근 3년간 교통 약자 보행교통사고 사상자 수 추이

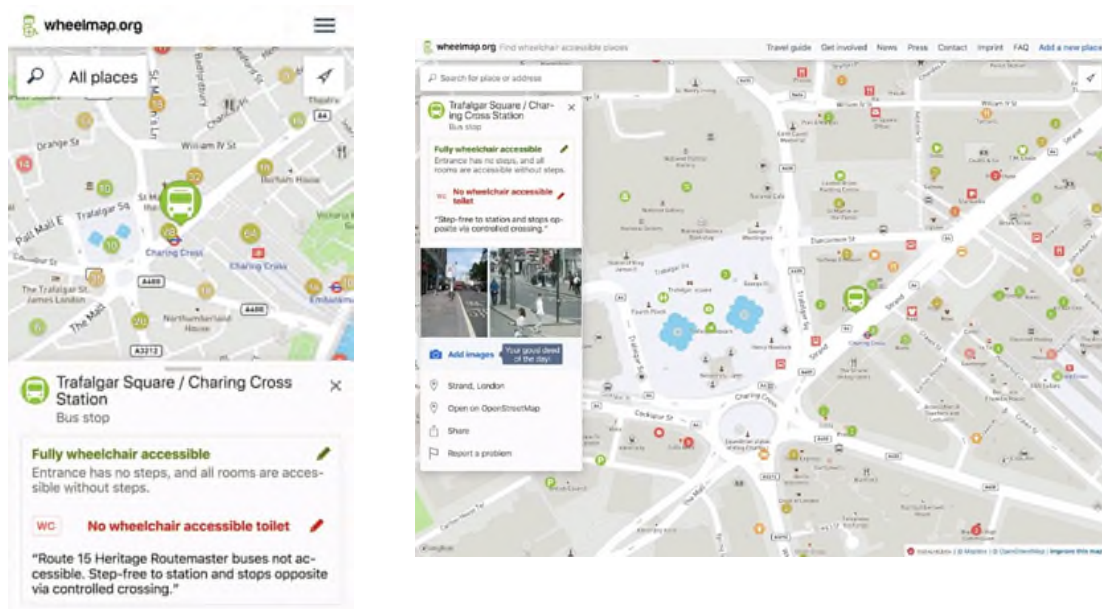
교통 약자가 이용하기 불편한 시설의 위치를 공유하고, 해당 시설을 우회하여 이동할 수 있도록 도움을 주어 교통 약자의 이동 편의성 증진, 정비 필요한 도로 및 장소의 정보 수집, 교통 약자의 사고 발생률 감소 등의 효과가 기대됨.

1) 교통약자의 이동편의 증진법 제2조(정의)

2) 국토부(2018), "2018년 교통약자 이동 편의 실태조사"

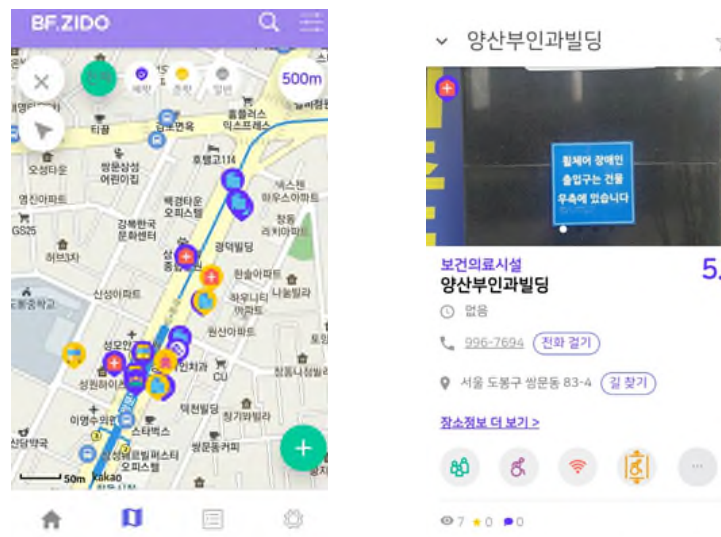
2. 기존 연구/기술 동향 분석

1) WheelMap : 휠체어가 접근 가능한 장소를 제공함.



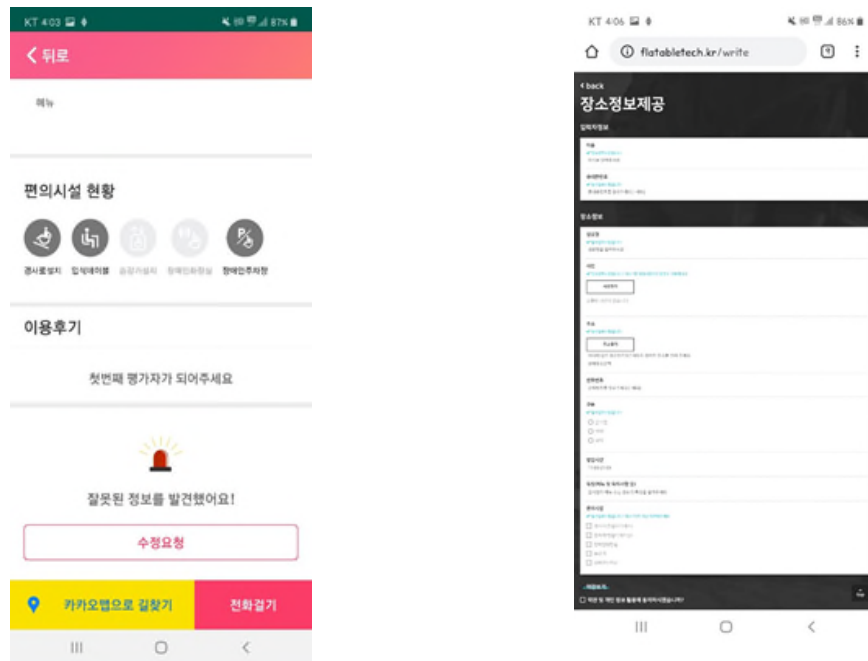
<그림 2> WheelMap 앱, 웹 실행 화면

2) 베프 지도 : 교통 약자가 접근 가능한 장소에 대한 정보 제공 및 소통 지원



<그림 3> 베프 지도 앱 실행 화면

3) 플랫 : 교통 약자가 접근 가능한 장소에 대한 정보 제공 및 소통 지원



<그림 4> 플랫 앱 실행 화면

유사 사례	특징
WheelMap	1) 앱과 웹 둘 다 지원 2) 휠체어만 취급 3) 로그인 없이 사용 가능
베프 지도	1) 앱만 지원 2) 교통 약자의 유형을 다양하게 구분 3) 다른 앱과 연동하여 길찾기 기능 제공
플랫	1) 앱과 웹 둘 다 지원 2) 편의시설 현황 표시 3) 이용 후기 공유 기능 제공 4) 사용자 유형에 따라 다른 지도 제공

3. 개발 목표

사용자가 이용하기 불편한 시설의 정보를 공유해 교통 약자의 이동 편의를 증진 시키는 시스템 개발을 목표로 함.

4. 팀 역할 분담

이경은	이자흔	전현정
교통 약자를 고려한 앱 UI 설계 및 구현	교통 약자를 고려한 웹 UI 설계 및 구현	DB 스키마 설계
서버 구현	웹 기본 기능 구현	앱 기본 기능 구현
길찾기 기능 구현	길찾기 기능 구현	길찾기 기능 구현

5. 개발 일정

항목	12월	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월
요구사항 정의 및 분석											
시스템 설계 및 상세 설계											
구현											
시험 및 데모											
문서화 및 발표											
산업기술대전											
졸업작품 최종 보고서 작성 및 패키징											

6. 개발 환경

OS	Windows 10
개발 언어	Java, CSS, HTML, Javascript
애플리케이션	Android Studio – Node.js – MySQL
지도 및 길찾기 기능	T map API

II. 본론

1. 개발 내용

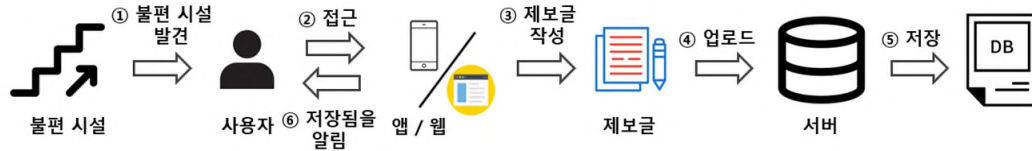
- 1) 불편 시설의 제보 : 사용자가 불편한 시설(길, 장소)에 대한 정보를 제보함.
**사용자가 제보할 때 교통 약자가 이용 가능한 정도와 정보가 필요한 사용자를 구분할 수 있는 태그를 표시하는 것을 기본으로 하며, 사진의 첨부과 도로의 경우 경사도, 계단 유무, 도로 파손 정도를 장소의 경우 엘리베이터 유무, 휠체어 경사로 등을 선택 사항으로 함.
- 2) 불편 시설의 정보 제공 : 사용자에게 제보 받은 정보를 제공함.
- 3) 불편 시설 알림 : 사용자의 위치 기준 100m 내 불편 시설이 존재할 경우 알림.
- 4) 길찾기 기능 : 사용자가 이용 가능한 경로를 제공함.
- 5) 네비게이션 기능 : 길찾기의 부가적인 기능으로 사용자의 보행 방향에 따라 안내 기능을 제공함.

2. 문제 및 해결방안

문제	해결방안
팀원들의 전반적인 프로젝트에 대한 이해 부족	WBS를 이용하여 요구사항 정리
목표까지 최단 경로 제공 API 확인 필요	T Map API를 이용해 목표 경로 제공 가능함을 확인함.
약자 타입에 따른 시나리오 구체화 필요	태그를 사용해 약자 유형에 따라 제보 글을 분류해서 약자 유형에 따라 시나리오를 분리할 필요가 없음.
대안 경로 제시 방안 구체화 필요	도보 DB와 A* 알고리즘을 이용하여 대안 경로를 제시함.

3. 시험시나리오

1) 불편 시설의 제보



<그림 5> 불편 시설 제보 시나리오

- ① 사용자가 불편 시설 발견
- ② 사용자가 앱에 접근해 제보 글을 작성
- ③ 제보 글을 서버를 통해 DB에 저장
- ④ 사용자에게 저장되었음을 알림

2) 불편 시설의 정보 제공

a. 현재 위치를 기준으로 불편 시설 정보 제공



<그림 6> 현재 위치에 대한 불편 시설 정보 제공 시나리오

- ① 사용자가 앱에 접근
- ② 사용자의 현재 위치 전달
- ③ DB에서 현재 위치에 대한 불편 시설 정보 취득
- ④ 지도에 표시하여 불편 시설 정보 제공

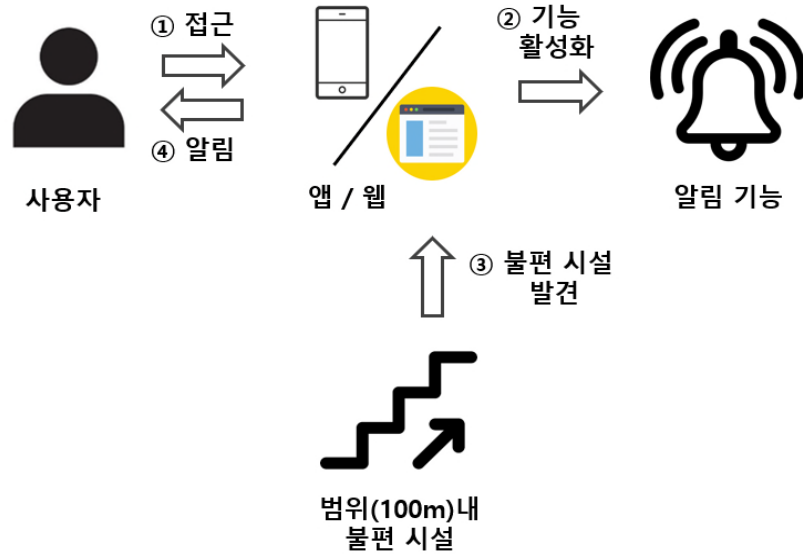
b. 장소 검색을 통한 정보 제공



<그림 7> 장소 검색을 통한 정보 제공 시나리오

- ① 사용자가 앱에 접근
- ② 사용자가 특정 장소를 검색
- ③ 검색 단어와 관련된 장소들에 대한 목록 출력
- ④ 사용자가 원하는 장소 선택
- ⑤ 선택한 장소에 대한 정보와 제보 글 출력

3) 불편 시설 알림



<그림 8> 불편 시설 알림 시나리오

- ① 사용자가 앱에 접근 후 알림 기능 활성화
- ② 사용자의 현 위치를 기준으로 100m 내 불편 시설이 존재할 경우 알림

4) 길찾기 기능



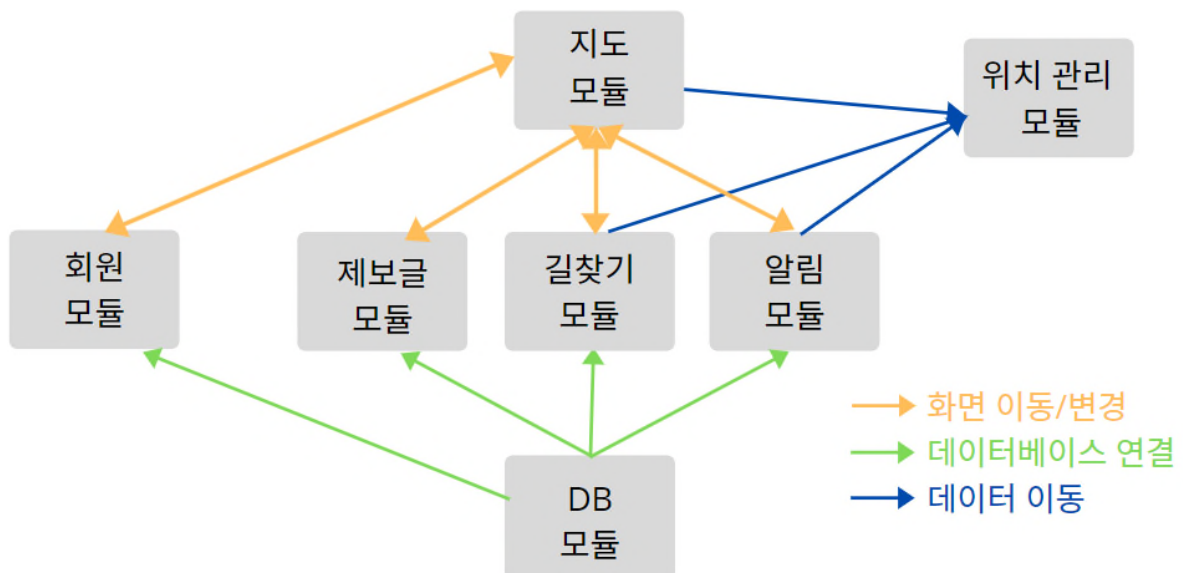
<그림 9> 길찾기 시나리오

- ① 사용자가 앱에 접근
- ② 사용자가 시작지점과 도착지점 선택
- ③ 설정된 사용자의 유형에 맞게 경로 제공
- ④ 네비게이션 기능을 선택할 경우 길 안내 제공

4. 상세 설계

1) 모듈 간 동작

모듈 간 상호 동작은 다음과 같다.



2) DB 모듈

a. 테이블

① 회원정보 : 아이디 비밀번호 이메일 교통약자유형

UserInformation			
이름	타입	NULL 여부	설명
<u>userID</u>	VARCHAR(12)	NO	아이디
userPW	VARCHAR(15)	NO	비밀번호
userMail	VARCHAR(30)	NO	등록 이메일
userType	VARCHAR(50)	NO	교통약자 유형

② 제보글 : 제보글 번호, 제보글 작성자의 아이디, 내용(이용 가능한 정도, 태그, 경사 각도, 계단 유무, 도로 파손 정도, 엘리베이터 유무, 휠체어 경사로, 제보글 사진), 작성 날짜, 해당 시설 ID

PostInformation			
이름	타입	NULL 여부	설명
<u>postNum</u>	INT	NO	제보글 번호
writerID	VARCHAR(12)	NO	작성자 아이디
postDate	DATE	NO	작성 날짜
poiID	VARCHAR(20)	NO	시설 POI ID
availability	INT	NO	이용 가능한 정도
tag	VARCHAR(50)	NO	불편함을 겪는 교통약자 유형
slope	INT	YES	경사 각도
stair	BOOLEAN	YES	계단 유무
roadBreakage	BOOLEAN	YES	도로 파손 여부
elevator	BOOLEAN	YES	엘리베이터 유무
wheelchairSlope	BOOLEAN	YES	휠체어 경사로 유무
postImage	MEDIUMBLOB	YES	제보글 사진

3) 제보글 모듈

- a. 기능 : 제보글 확인, 제보글 작성, 제보글 수정, 제보글 삭제, 제보글 신고
- b. 다루는 정보 : 제보글 작성자, 내용(이용 가능한 정도, 태그, 경사 각도, 계단 유무, 도로 파손 정도, 엘리베이터 유무, 휠체어 경사로), 작성 날짜, 해당 장소 또는 길

c. API

① 제보글 확인

- (a) 형식 : `PostVO readPost(int postNum)`
- (b) 리턴값 : `PostVO` 클래스 객체
- (c) 설명 : DB로부터 제보글의 내용을 읽음

② 제보글 작성

- (a) 형식 : `boolean writePost(PostVO post)`
- (b) 리턴값 : 등록이 성공했을 경우 `true`, 실패했을 경우 `false` 리턴
- (c) 설명 : DB에 제보글의 내용을 등록함.

③ 제보글 수정

- (a) 형식 : `boolean updatePost(PostVO post)`
- (b) 리턴값 : 수정이 성공했을 경우 `true`, 실패했을 경우 `false` 리턴
- (c) 설명 : DB에 등록된 제보글의 내용을 수정함

④ 제보글 삭제

- (a) 형식 : `boolean deletePost(int postNum)`
- (b) 리턴값 : 삭제가 성공했을 경우 `true`, 실패했을 경우 `false` 리턴
- (c) 설명 : 선택된 제보글을 DB에서 삭제함

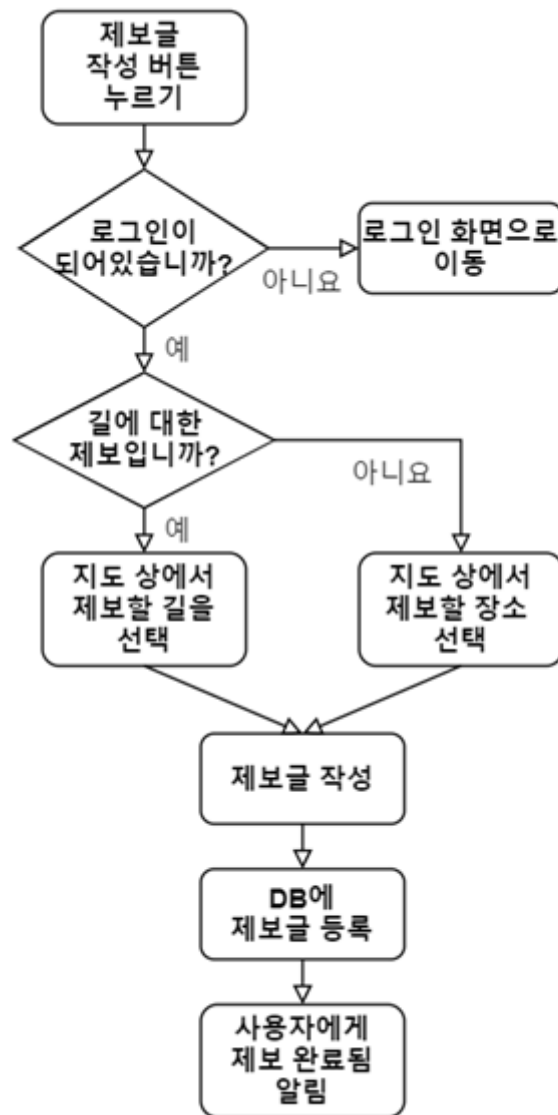
⑤ 제보글 목록 확인

- (a) 형식 : `ArrayList<PostVO> getPostList(String poild)`
- (b) 리턴값 : `ArrayList<PostVO>`
- (c) 설명 : DB로부터 선택된 시설의 제보글 목록을 읽음.

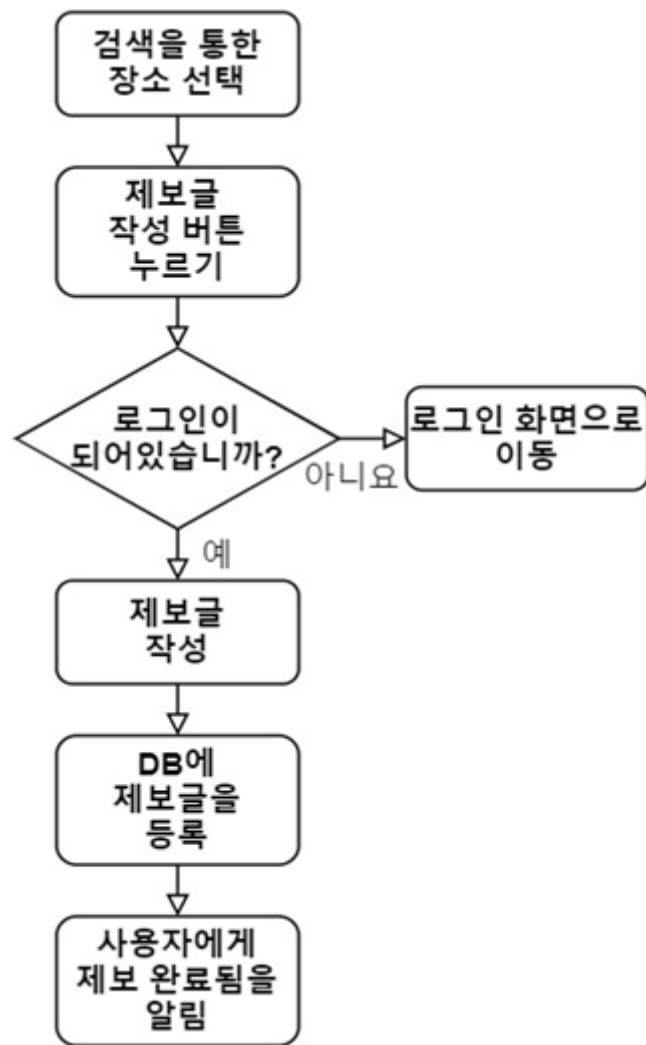
⑥ 제보글 신고

- (a) 형식 : `boolean declarePost(int postNum)`
- (b) 리턴값 : 신고가 성공했을 경우 `true`, 실패했을 경우 `false` 리턴
- (c) 설명 : DB에 선택된 제보글의 신고 여부를 갱신함.

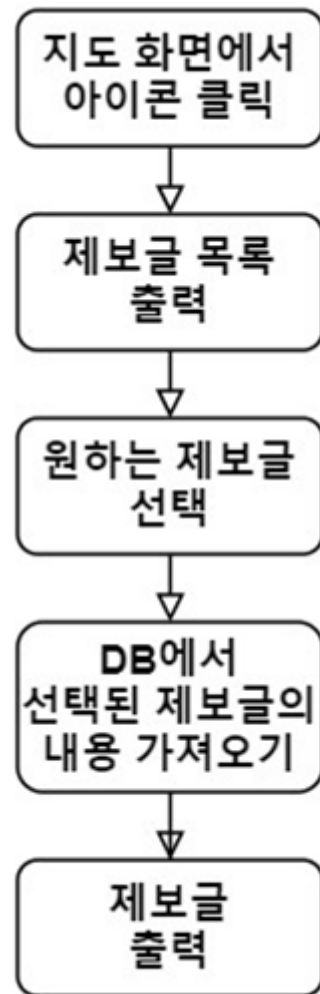
d. 적용 알고리즘 :



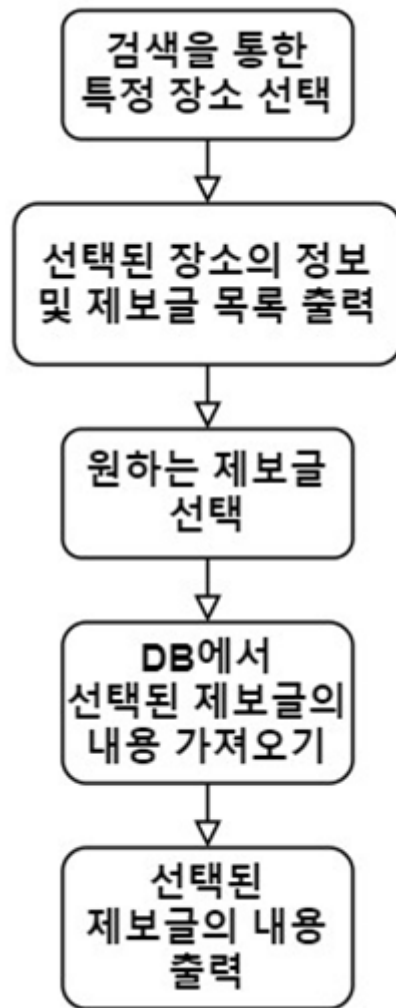
<그림 10> 제보글 작성 알고리즘의 순서도



<그림 11> 장소 검색을 통해 제보글 작성 알고리즘의 순서도



<그림 12> 제보글 확인 알고리즘의 순서도



<그림 13> 장소 검색 후 제보글 확인 알고리즘의 순서도

4) 알림 모듈

- a. 기능 : 알림 활성화, 알림 제공
- b. 다루는 정보 : 사용자의 위치 정보, 제보글이 있는 장소와의 거리 정보, 사용자와 불편시설까지의 범위
- c. API :
 - ① 알림 활성화
 - (a) 형식: void activateAlarm()
 - (b) 리턴값: 없음
 - (c) 설명: 알림을 활성화/비활성화함.
 - ② 알림 제공
 - (a) 형식 : void supplyAlarm()
 - (b) 리턴값 : 없음
 - (c) 설명 : 사용자에게 알림을 제공함.
- d. 적용 알고리즘 :



<그림 14> 불편 시설 알림 알고리즘 순서도

5) 길찾기 모듈

a. 기능 : 최단 경로 도출, 회피 경로 도출, 경로 그리기, 길안내

b. 다루는 정보 : 출발지, 도착지, 제보된 장소

c. API :

① 회피 경로 도출

(a) 형식 : void getAvoidRoute (Tmapv2.LatLng start, Tmap2.LatLng end)

(b) 리턴값 : 없음

(c) 설명 : 불편시설을 회피하는 경로를 탐색함.

② 최단 경로 도출

(a) 형식 : void getShortestRoute(Tmapv2.LatLng start, Tmap2.LatLng end)

(b) 리턴값 : 없음

(c) 설명 : 최단 경로를 탐색함.

③ 경로그리기

(a) 형식 : void drawRoute(var routeResult)

(b) 리턴값 : 없음

(c) 설명 : 도출된 경로를 지도 상에 표시함.

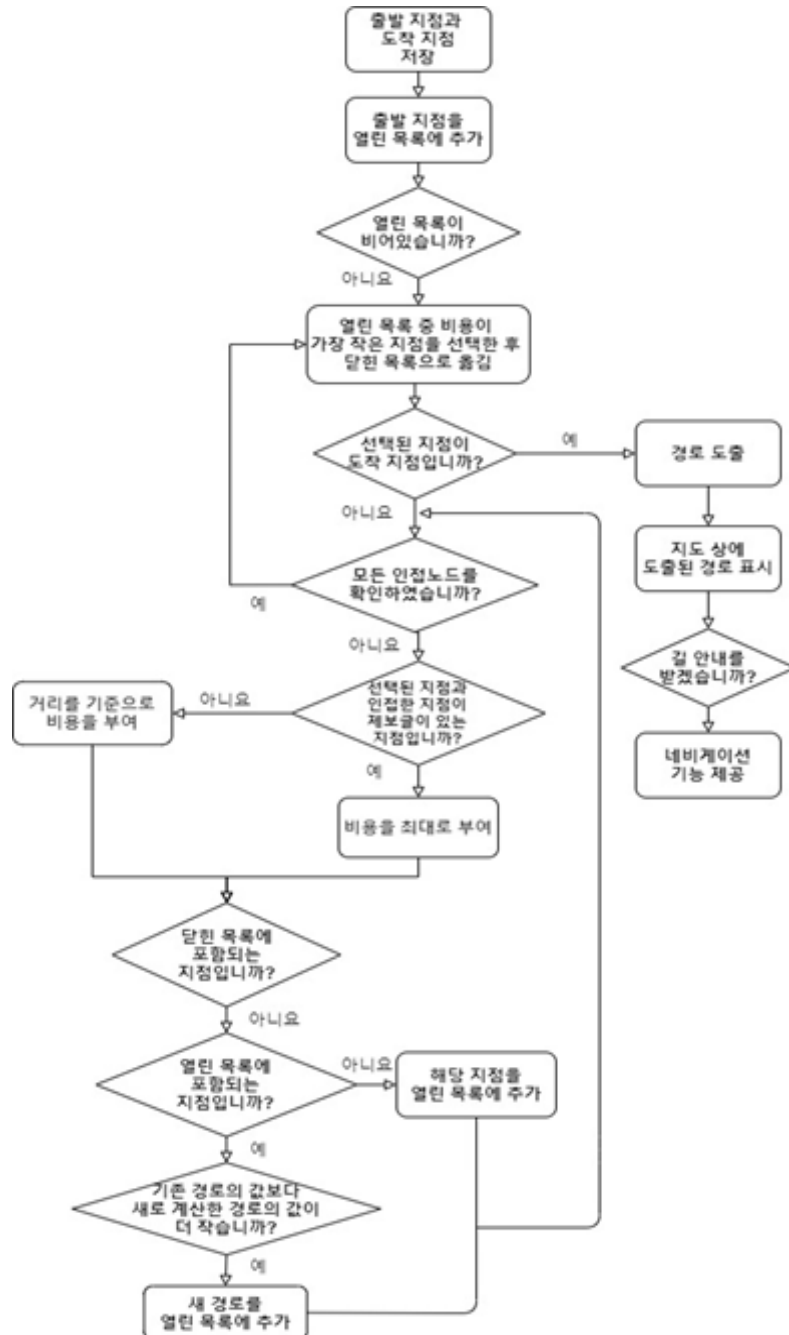
④ 길 안내

(a) 형식 : void guideRoute(var routeResult)

(b) 리턴값 : 없음

(c) 설명 : 도출된 경로를 바탕으로 사용자에게 길을 안내함.

d. 적용 알고리즘 :



<그림 15> 불편 시설을 회피하는 경로 찾기 알고리즘 순서도

6) 지도 모듈

a. 기능 : 지도 화면 제공, 아이콘 표시

b. 다루는 정보 : 없음

c. API :

① 지도 화면 제공

(a) 형식 : void initTmap()

(b) 리턴값 : 없음

(c) 설명 : 지도 화면을 생성하여 제공함.

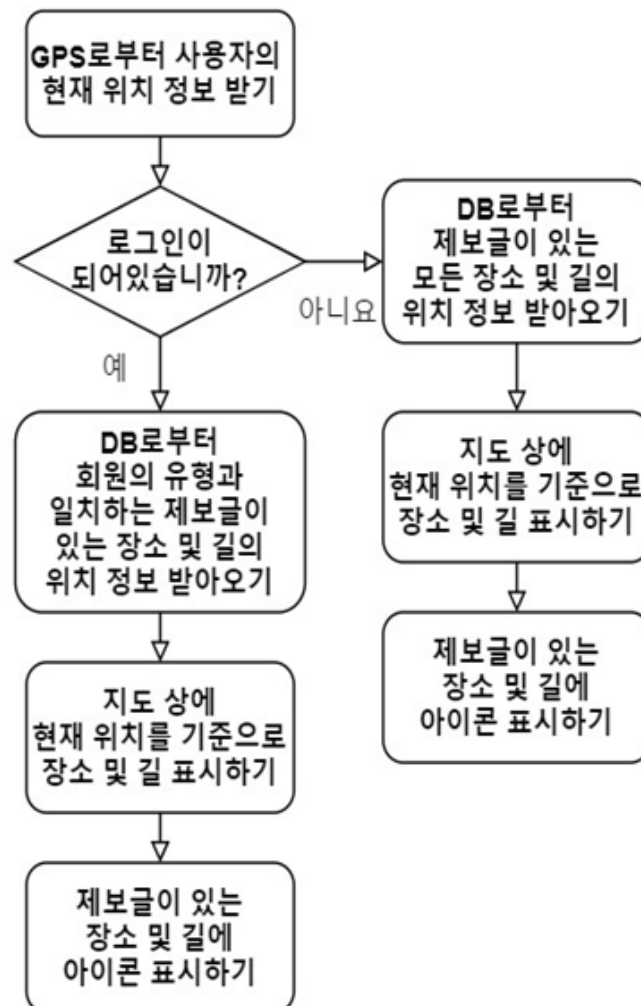
② 아이콘 표시

(a) 형식 : void addMarkers()

(b) 리턴값 : 없음

(c) 설명 : 지도 상에 제보글이 있는 지점에 아이콘을 표시함.

d. 적용 알고리즘 :



<그림 16> 지도 화면 제공 알고리즘 순서도

7) 위치 관리 모듈

a. 기능 : GPS, 주소 변환, 좌표 변환

b. 다루는 정보 : 현재 위치 좌표

c. API :

① GPS

(a) 형식 : TMapPoint getLocation()

(b) 리턴값 : TMapPoint 클래스의 객체로 현재 위치의 위도와 경도를 속성으로 가짐.

(c) 설명 : 사용자의 현재 위치를 받음.

② 주소 변환

(a) 형식 : String convertLonLatToAddress(var lonLat)

(b) 리턴값 : 주소

(c) 설명 : 좌표를 주소로 변환함.

③ 좌표 변환


(a) 형식: var convertAddressToLonLat(String address)

(b) 리턴값: Tmapv2.LatLng 클래스의 객체로 위도와 경도를 속성으로 가짐.

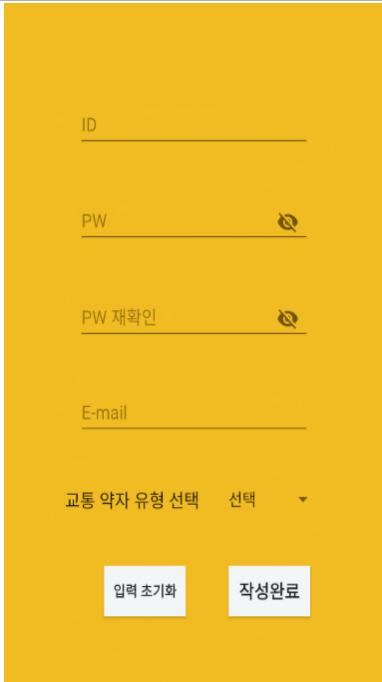
(c) 설명: 주소를 좌표로 변환함.

4-1. UI 설계

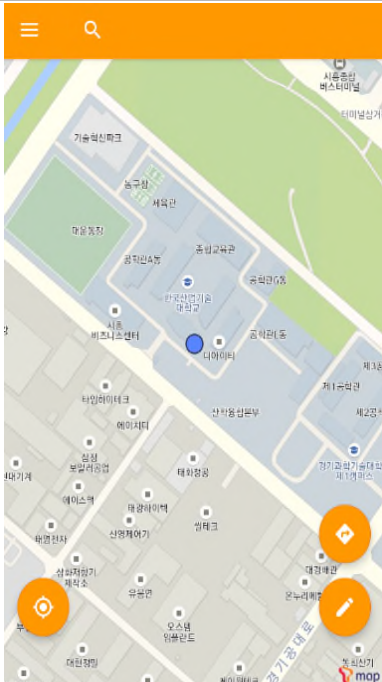
1) 로그인 화면

앱 화면	설명
 <p>The login screen has a solid orange background. It features a white 'ID' input field, a white 'PW' input field, a large white '로그인' (Login) button, a white '회원가입' (Sign Up) button, a white 'ID/PW 찾기' (Find ID/PW) button, and a small white '비회원 이용' (Non-member use) button at the bottom right.</p>	<ol style="list-style-type: none"> 1. 아이디, 비밀번호 입력 후 로그인 2. id/pw 찾기 클릭 시 아이디, 비밀번호 찾기 화면으로 이동함 3. 회원가입 버튼 클릭 시 회원가입 화면으로 이동함 4. 비회원 이용 버튼 클릭 시 비회원용 지도 화면으로 이동함 5. 아이디, 비밀번호 불일치 시 알림창 생성, 일치 시 회원용 지도 화면으로 이동함

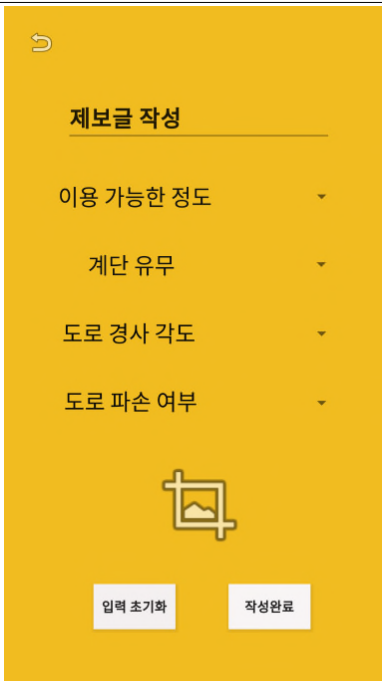
2) 회원가입 화면

앱 화면	설명
 <p>The sign-up screen has a solid orange background. It features white input fields for 'ID', 'PW', 'PW 재확인' (Confirm Password), and 'E-mail'. There are eye icons for toggling password visibility. Below the inputs is a '교통 약자 유형 선택' (Select vulnerable user type) dropdown menu. At the bottom are two white buttons: '입력 초기화' (Reset input) and '작성완료' (Finish writing).</p>	<ol style="list-style-type: none"> 1. 아이디, 비밀번호, 이메일 입력 후 교통 약자 유형 선택 2. 아이디, 이메일 입력 시 이미 존재하는 아이디, 이메일이면 아래에 오류 메시지를 띄움 3. PW, PW 재확인 텍스트 입력 시 패스워드 보기 버튼을 클릭하면 입력한 비밀번호를 텍스트로 표시함 4. 입력 초기화 버튼 클릭 시 입력한 내용이 모두 삭제됨 5. 작성완료 버튼 클릭 시 입력한 항목이 회원가입 조건과 불일치하면 알림창 생성, 일치하면 로그인 화면으로 이동함

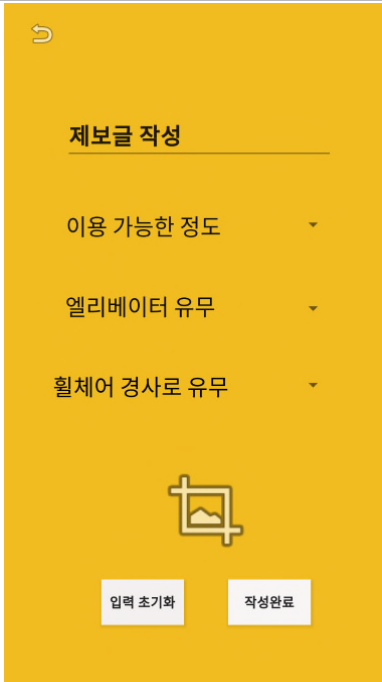
3) 지도 화면

앱 화면	설명
	<ol style="list-style-type: none"> 1. 지도 화면은 사용자의 현재 위치를 기준으로 표시함 2. 메뉴 버튼 클릭 시 메뉴 항목(마이 페이지, 길찾기, 제보글 작성, 로그아웃, 앱 정보 항목)을 표시함. 3. 각각의 메뉴 항목은 동일한 이름의 페이지로 이동함 4. 검색 버튼 클릭 시 장소나 도로명을 검색함 5. 현재 위치 버튼 클릭 시 사용자의 현재 위치를 지도상에 표시함 6. 길찾기 버튼 클릭 시 길찾기 화면으로 이동함 7. 제보글 작성 버튼 클릭 시 제보글 작성 화면으로 이동함


4) 도로 제보글 작성 화면

앱 화면	설명
	<ol style="list-style-type: none"> 1. 취소 버튼 클릭 시 지도 화면으로 이동함 2. 이용 가능한 정도와 위치는 필수로 작성하며, 사진 첨부과 경사도, 계단 유무와 도로 파손 여부는 선택사항으로 작성함 3. 사진 버튼 클릭 시 기기에 저장된 사진을 선택해 첨부함 4. 항목 작성 후 작성 완료 버튼 클릭 시 지도 화면으로 이동함 5. 입력 초기화 버튼 클릭 시 입력한 항목들이 모두 초기화됨

5) 장소 제보글 작성 화면

앱 화면	설명
	<ol style="list-style-type: none"> 1. 취소 버튼 클릭 시 지도 화면으로 이동함 2. 이용 가능한 정도는 필수로 작성하며 엘리베이터 유무, 휠체어 경사로 유무와 사진 첨부는 선택사항으로 작성함 3. 사진 버튼 클릭 시 기기에 저장된 사진을 선택해 첨부함 4. 항목 작성 후 작성 완료 버튼 클릭 시 지도 화면으로 이동함 5. 입력 초기화 버튼 클릭 시 입력한 항목들이 모두 초기화됨

6) 길찾기 화면

앱 화면	설명
	<ol style="list-style-type: none"> 1. 창 닫기 버튼 클릭 시 지도 화면으로 이동함 2. 출발지, 도착지 작성 후 최단 경로 또는 회피 경로 길찾기 버튼 클릭 3. 최단 경로 길찾기(걷는 아이콘) 버튼 클릭 시 결과 화면에 최단 경로 길찾기의 결과 목록을 표시함 4. 회피 경로 길찾기(휠체어 아이콘) 버튼 클릭 시 결과 화면에 회피 경로 길찾기의 결과 목록을 표시함 5. 길찾기 결과 화면은 사용자가 경로를 검색한 결과를 표시함 6. 출발지/도착지 전환 버튼 클릭 시 출발지에는 도착지, 도착지에는 출발지에 작성한 값이 표시됨

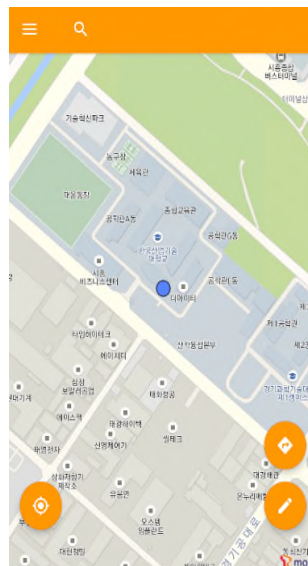
5. Prototype 구현

1) 로그인 화면



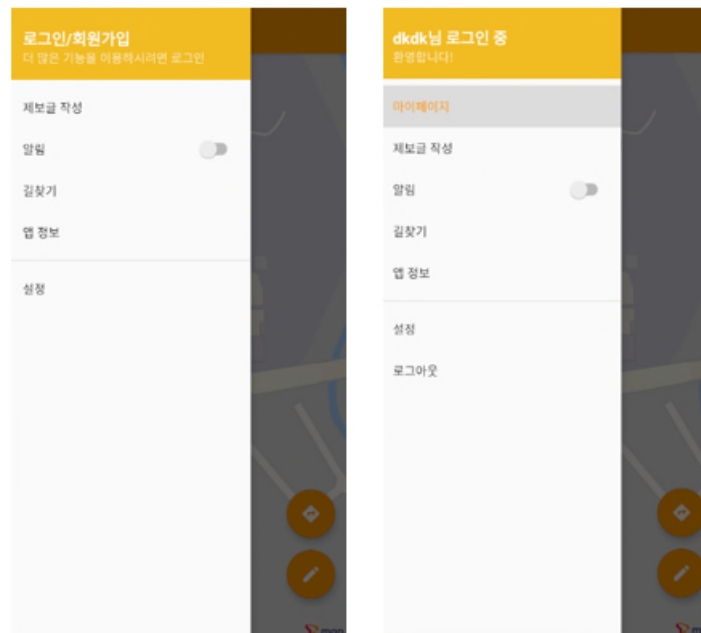
로그인 화면은 첫 실행 화면으로, 회원가입 시 작성한 아이디, 비밀번호를 입력 후 로그인 버튼을 선택하면 회원용 지도 화면으로 이동한다. 비회원 이용 버튼을 선택하면 비회원용 지도 화면으로 이동하여, 회원가입을 하지 않고도 지도 화면에서 일부 기능을 사용할 수 있다.

2) 메인 화면



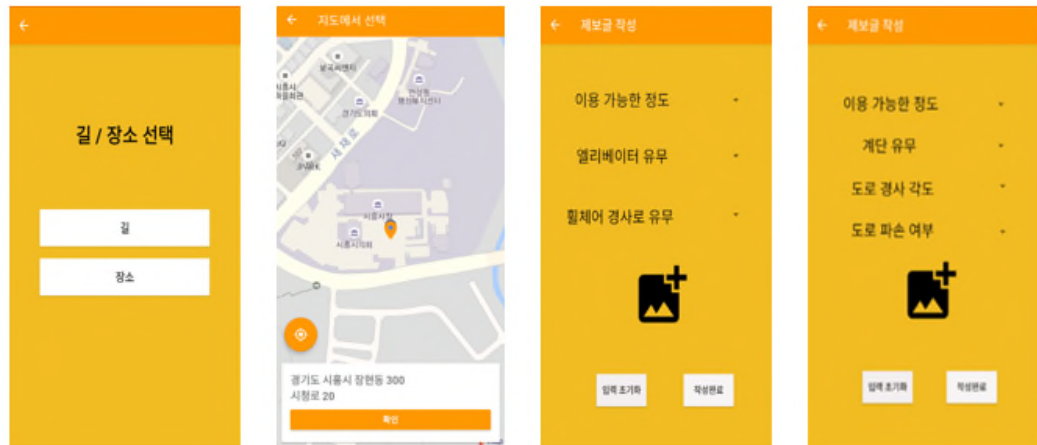
로그인 후 이동하는 화면으로 비회원의 경우 유형 맞춤 길찾기 기능과 제보글 작성 이 불가능하다. 앱 화면의 경우 하단에 지도 화면을 배치하고 좌측 상단에 메뉴 버튼과 검색 버튼, 하단에 사용자의 현재 위치 찾기 버튼과 길찾기, 제보글 작성 버튼을 배치하였다.

3) 메뉴 화면



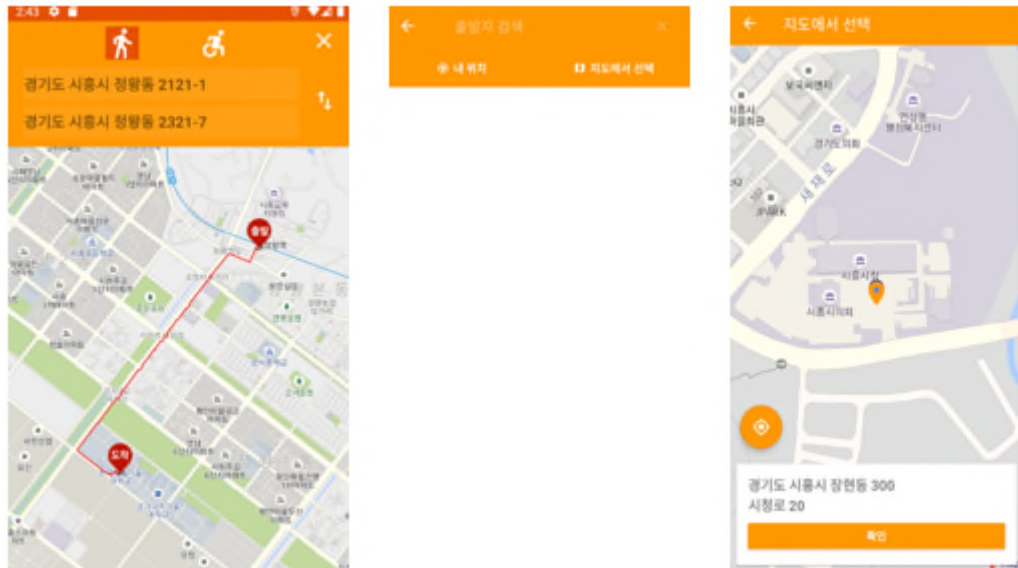
지도 화면에서 표시되는 메뉴 화면으로, 왼쪽이 비회원 지도 화면의 메뉴, 오른쪽은 회원 지도 화면의 메뉴이다. 회원의 경우 마이페이지, 길찾기, 제보글 작성, 로그아웃, 앱 정보 기능을 사용할 수 있으며 각 버튼을 선택 시 각 기능에 해당하는 화면으로 화면을 이동한다. 비회원의 경우 회원가입과 앱 정보 기능만을 사용할 수 있다. 메뉴 중 제보글 작성 버튼을 선택하면 도로와 장소 제보글 중 작성할 유형을 선택한 뒤 제보글 작성 화면으로 이동한다.

4) 제보글 작성 화면



제보하는 대상이 도로인지 장소인지에 따라 제보할 내용이 달라지기 때문에 대상에 따라 제보글 작성 화면이 다르다. 왼쪽부터 차례대로 제보글 유형 선택 화면, 제보글 위치 선택 화면, 장소 제보글 작성 화면, 도로 제보글 작성 화면이다. 제보글 작성 화면은 회원용 지도 화면에서 제보글 작성 버튼 클릭 시 표시되는 화면이다. 이용 가능한 정도, 위치와 같은 필수 작성 항목과 사진 첨부 등의 선택 항목으로 구성되어 있다. 제보글의 항목을 작성한 뒤 작성완료 또는 작성하기 버튼을 선택하면 지도 화면으로 화면이 이동한다.

5) 길찾기 화면



길찾기는 두 가지 종류를 제공한다. 다른 앱에서도 많이 제공하는 가장 빠른 경로를 도출하는 일반적인 경로 찾기와 교통 약자 유형에 맞게 불편 시설을 분류하여 그것을 회피한 경로를 도출하는 회피 경로 찾기이다. 제일 왼쪽 화면의 맨 위에 배치된 버튼 두 개를 통해 길찾기 서비스를 선택할 수 있다. 왼쪽 버튼을 누른다면 일반 경로, 오른쪽 버튼을 누르면 회피 경로를 제공한다. 출발 및 도착 부분을 누르면 가운데 화면으로 이동하여 현재 위치를 출발지/도착지로 등록하거나 지도에서 직접 선택한 위치를 출발지/도착지로 등록하거나 검색한 장소를 출발지/도착지로 등록할 수 있다. 세 번째 화면은 지도에서 직접 선택할 시의 화면이다. 그 후 도출된 경로를 왼쪽 화면의 하단에 보여준다.

6. 시험/ 테스트 결과

1) 회원가입 및 로그인

a. 회원가입 화면

Registration screen (a) showing fields for ID, Password, Password Confirmation, Email, and a dropdown for vehicle type selection. The ID field contains 'qwerty001', the Password field contains '*****', the Password Confirmation field contains '*****', and the Email field contains 'qwerty001@naver.com'. The vehicle type selection dropdown is set to '휠체..'. There are buttons for '입력 초기화' (Reset Input) and '작성완료' (Complete Writing).

b. 로그인 화면

Login screen (b) showing fields for ID and Password. The ID field contains 'dkdk' and the Password field contains '*****'. There is a '로그인' (Login) button, a '회원가입' (Sign Up) button, and an 'ID/PW 찾기' (Find ID/PW) button. There is also a '비회원 이용' (Non-member Use) button at the bottom right.

2) 제보글 작성, 확인

a. 도로 제보글 작성 시 화면

Road report writing screen (a) showing a form with fields for '이용 가능한 정도' (Degree of Use) set to '2', '교통약자 유형 선택' (Select Vulnerable User Type) set to '보조..', '도로 경사 각도' (Road Slope Angle) set to '낮음' (Low), '계단 유무' (Stairs) set to '없음' (None), and '도로 파손 여부' (Road Damage) set to '있음' (Yes). There is a button for '입력 초기화' (Reset Input) and a button for '작성완료' (Complete Writing).

b. 장소 제보글 작성 시 화면

Location report writing screen (b) showing a form with fields for '이용 가능한 정도' (Degree of Use) set to '2', '교통약자 유형 선택' (Select Vulnerable User Type) set to '휠체..' (Wheelchair), '엘리베이터 유무' (Elevator) set to '없음' (None), and '휠체어 경사로 유무' (Wheelchair Ramp) set to '있음' (Yes). There is a button for '입력 초기화' (Reset Input) and a button for '작성완료' (Complete Writing).

필수 항목 *	
* 이용 가능한 정도	2
* 교통약자 유형 선택	보조기구 이용 자
<hr/>	
도로 경사 각도	낮음
계단 유무	없음
도로 파손 여부	있음
<div>수정</div> <div>삭제</div>	

* 필수 항목

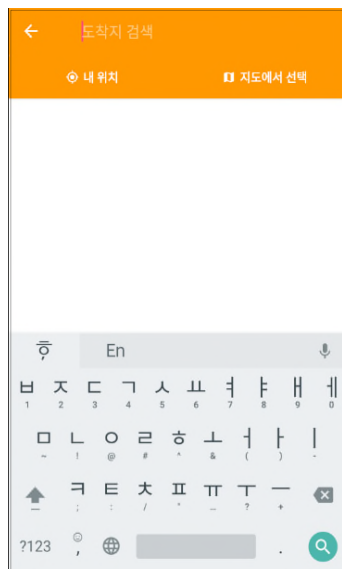
* 이용 가능한 정도	2
* 교통약자 유형 선택	휠체어 이용자

엘리베이터 유무	없음
휠체어 경사로 유무	있음

수정

삭제

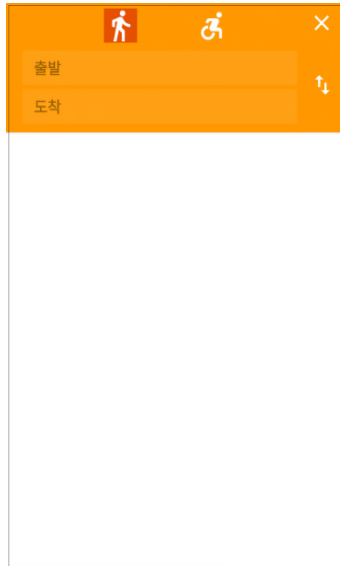
a. 길찾기 시 장소 검색 화면



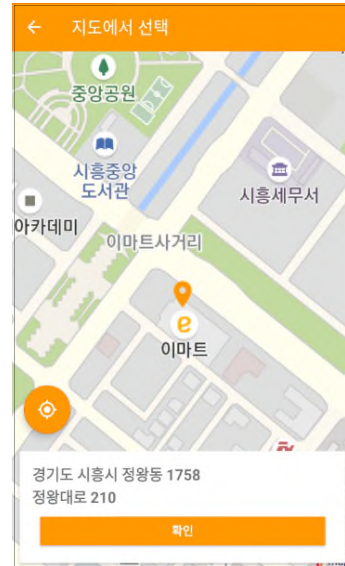
← 이마트	
뜨끈이감자탕 시화이마트점	349원
경기 시흥시 정왕동 중심상가1길 18	
무한리필칼집생고기 시화이마트점	362원
경기 시흥시 정왕동 중심상가1길 14	
무한리필칼집생고기 시화이마트점 주차장	362원
경기 시흥시 정왕동 중심상가1길 14	
날리맘보 시화이마트점	456원
경기 시흥시 정왕동 중심상가3길 25	
이마트24 시화중심상가점	467원
경기 시흥시 정왕동 중심상가3길 26	
굽참고 시화이마트점	483원
경기 시흥시 정왕동 중심상가3길 24	
굽참고 시화이마트점 주차장	483원
경기 시흥시 정왕동 중심상가3길 24	
굽참고 시화이마트점 정문	483원
경기 시흥시 정왕동 중심상가3길 24	
가꾸는동안 시화이마트점	561원
경기 시흥시 정왕동 중심상가로 228	
뽕스밥버거시화이마트점	566원
경기 시흥시 정왕동 중심상가3길 24	

4) 최단 경로 길찾기

a. 길찾기 버튼 클릭 시 화면



b. 출발지/목적지 선택 후 지도에서 찾기 버튼 클릭 시 화면

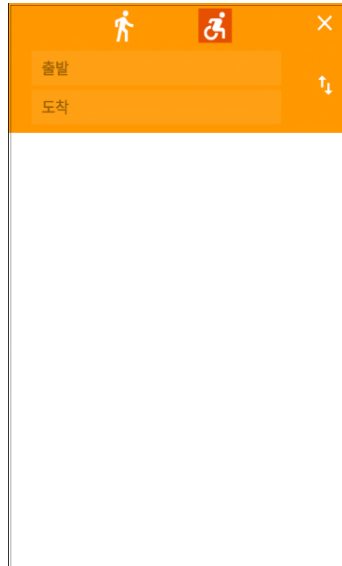


c. 출발지, 목적지를 모두 작성하면 출력되는 최단 경로 화면

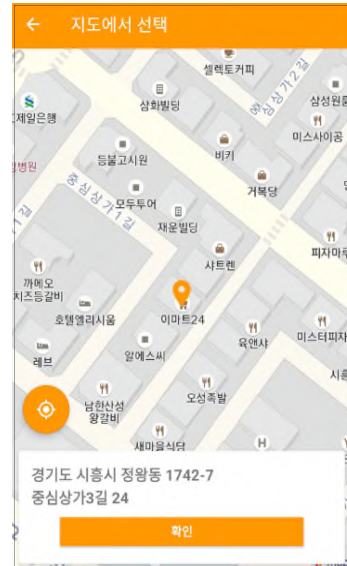


5) 회피 경로 길찾기

a. 회피 경로 길찾기 선택 시 화면



b. 출발지/목적지 선택 후 지도에서 찾기 버튼 클릭 시 화면



c. 출발지, 목적지를 모두 작성하면 출력되는 회피 경로 화면



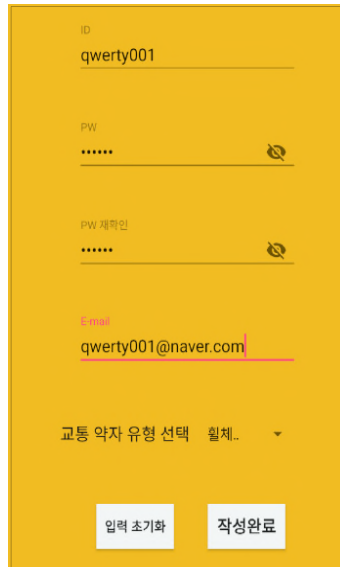
d. 최단거리와 비교 시, 불편시설을 지나지 않음을 확인 가능



7. Coding & Demo

1) 회원가입

a. 회원가입 화면



b. 기능 구현 코드

- ① 작성 완료 버튼 클릭 시, 필요한 정보(아이디, 패스워드, 패스워드 확인, 이메일, 교통 약자 유형)를 모두 작성하면 회원가입이 완료된다.

```
// 작성완료
public void writeFin(View v) {
    textManage();
    if(!idCheck || !passwdCheck || !checkCheck || !mailCheck ||
        spinner.getSelectedItem().toString().equals("선택"))
        return;

    String userID = id.getText().toString();
    String userPassword = pw.getText().toString();
    String userMail = email.getText().toString();
    String userType = spinner.getSelectedItem().toString();

    Response.Listener<String> responseListener = new Response.Listener<String> (){
        @Override
        public void onResponse(String response) {
            try{
                JSONObject jsonResponse = new JSONObject(response);
                boolean success = jsonResponse.getBoolean( "name": "success");
                if(success){
                    AlertDialog.Builder builder = new AlertDialog.Builder( context: JoinActivity.this);
                    builder.setMessage("회원가입이 완료되었습니다!");
                    builder.setNegativeButton( text: "확인", (dialog, which) → {
                        Intent intent = new Intent( packageContext: JoinActivity.this, LoginActivity.class);
                        JoinActivity.this.startActivity(intent);
                        finish();
                    })
                    .create()
                    .show();
                }
            }
        }
    };
}
```

```

        else{
            AlertDialog.Builder builder = new AlertDialog.Builder(context JoinActivity.this);
            builder.setMessage("회원가입 실패 : 입력한 정보를 다시 확인해주세요.")
                .setNegativeButton(text: "다시 시도", listener: null)
                .create()
                .show();
        }
    } catch (Exception e){
        e.printStackTrace();
    }
}

JoinRequest JoinRequest = new JoinRequest(userID, userPassword, userMail, userType, responseListener);
RequestQueue queue = Volley.newRequestQueue(context JoinActivity.this);
queue.add(JoinRequest);
}

```

② 회원가입 시 작성한 정보(아이디, 비밀번호, 이메일, 교통 약자 유형)가 서버로 전송된다.

```

public class JoinRequest extends StringRequest {

    final static private String URL = "http://121.168.1.81/register.php";
    private Map<String, String> parameters;

    public JoinRequest(String userID, String userPassword, String userMail, String userType, Response.Listener<String> listener){
        super(Method.POST, URL, listener, errorListener: null); //해당 URL에 POST방식으로 파마미터들을 전송함
        parameters = new HashMap<>();
        parameters.put("userID", userID);
        parameters.put("userPassword", userPassword);
        parameters.put("userMail", userMail);
        parameters.put("userType", userType);
        //parameters.put("userType", userType);
    }

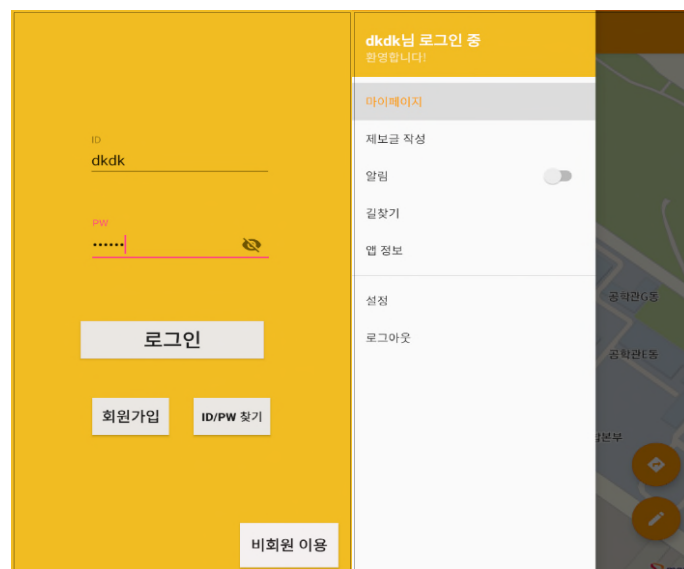
    protected Map<String, String> getParams() { return parameters; }
}

```

2) 로그인

a. 출력 화면

왼쪽은 로그인 화면, 오른쪽은 로그인 후 이동한 메인 화면의 메뉴이다.



b. 기능 구현 코드

- ① 아이디, 패스워드 입력 후 로그인 버튼 클릭 시 회원가입이 되어있으면 메인 화면으로 이동한다.

```
public void login(final View v) {
    manageInputError();
    if(!idCheck || !pwCheck)
        return;

    EditText idText = (EditText) findViewById(R.id.insert_ID);
    EditText pwText = (EditText) findViewById(R.id.insert_PW);

    userID = idText.getText().toString();
    userPassword = pwText.getText().toString();

    Response.Listener<String> responseListener = (response) -> {
        try{
            JSONObject jsonResponse = new JSONObject(response);
            boolean success = jsonResponse.getBoolean( name: "success");
            if(success){
                String id = jsonResponse.getString( name: "userID");
                save(id);
                nonMember(v);
                finish();
            }
            else{
                AlertDialog.Builder builder = new AlertDialog.Builder( context: LoginActivity.this);
                builder.setMessage("아이디 혹은 비밀번호가 일치하지 않습니다. "+"\\n"+"입력한 내용을 다시 확인해주세요.")
                    .setNegativeButton( text: "확인", listener: null)
                    .create()
                    .show();
            }
        } catch (Exception e){
            e.printStackTrace();
        }
    };

    LoginRequest loginRequest = new LoginRequest(userID, userPassword, responseListener);
    RequestQueue queue = Volley.newRequestQueue( context: LoginActivity.this);
    queue.add(loginRequest);
}
```

- ② 로그인 시 해당 사용자의 아이디, 비밀번호가 서버로 전송된다.

```
public class LoginRequest extends StringRequest {

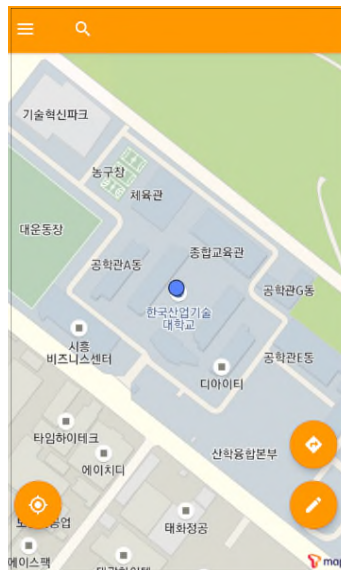
    final static private String URL = "http://121.168.1.81/login.php";
    private Map<String, String> parameters;

    public LoginRequest(String userID, String userPassword, Response.Listener<String> listener){
        super(Method.POST, URL, listener, errorListener: null); //해당 URL에 POST방식으로 파라미터들을 전송함
        parameters = new HashMap<>();
        parameters.put("userID", userID);
        parameters.put("userPassword", userPassword);
    }

    protected Map<String, String> getParams() { return parameters; }
}
```

3) 메인 화면

a. 출력 화면



b. 기능 구현 코드

① 지도 프래그먼트를 생성한다.

```
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

    appData = getActivity().getSharedPreferences( name: "appData", Context.MODE_PRIVATE);
    id = appData.getString( key: "ID", defValue: ""); // 로그인 정보

    ViewGroup view = (ViewGroup) inflater.inflate(R.layout.fragment_map, container, attachToRoot: false);
    linearLayout = view.findViewById(R.id.linearLayoutTmap);

    FloatingActionButton currentLocation = (FloatingActionButton)activity.findViewById(R.id.fab_currentLocation);
    currentLocation.setVisibility(View.VISIBLE);
}
```

② 생성한 지도 프래그먼트를 메인 화면에 표시한다.

```
//메인 화면(app_bar_main)의 지도 부분 레이아웃을 가져와 대입
linearLayout = (LinearLayout) findViewById(R.id.linearLayoutTmap);

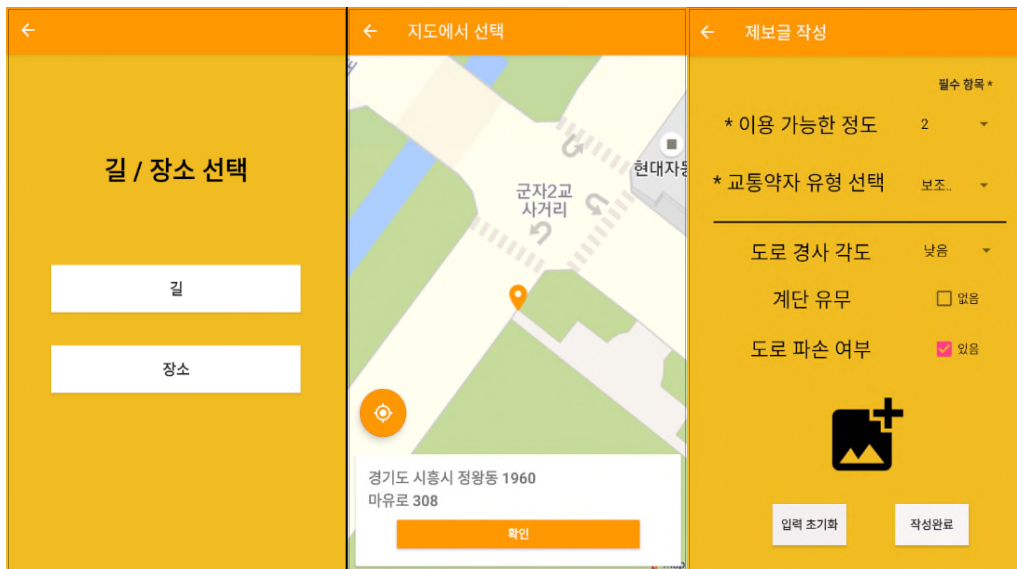
//지도 생성
tMapView = new TMapView( context: this);
tMapView.setSKTMapApiKey( API 인증키 );
tMapView.setLanguage(TMapView.LANGUAGE_KOREAN);
tMapView.setIconVisibility(true); //현재위치로 표시될 아이콘을 표시할지 여부
//지도 띄울 때 확대 정도
tMapView.setZoomLevel(17);
tMapView.setMapType(TMapView.MAPTYPE_STANDARD);

//화면에 지도 표시
linearLayout.addView(tMapView);
setGPS();
```


4) 도로 제보글 작성

a. 출력 화면

왼쪽부터 차례대로 제보글 유형 선택 화면, 제보글 위치 선택 화면, 도로 제보글 작성 화면이다.



b. 기능 구현 코드

- ① 제보글 유형 선택 프래그먼트에서 제보글의 유형을 선택한다. 도로(길)과 장소 중에서 도로를 선택한 경우 type을 road로 전송한다.

```
buttonRoad.setOnClickListener(new Button.OnClickListener(){
    @Override
    public void onClick(View v) { // 버튼 클릭시 다음 프래그먼트에 선택 타입 전달
        ChoosePostTypeFragmentDirections.ActionFragmentWritePostToFragmentPoint action
            = ChoosePostTypeFragmentDirections.actionFragmentWritePostToFragmentPoint( type: "road");
        Navigation.findNavController(v).navigate(action);
    }
});
```

- ② 앞에서 전송한 type값으로 도로 제보글인지 장소 제보글인지를 구분한다. type 값이 road면 지도에서 선택한 위치의 좌표를 전송하고, 도로 제보글 작성 화면으로 이동한다.

```
choosePoint.setOnClickListener((v) -> {
    String postType = ChoosePointFragmentArgs.fromBundle(getArguments()).getType();
    TMapPoint center = activity.getCenter();

    appData = getActivity().getSharedPreferences( name: "appData", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = appData.edit();

    Double latitude = center.getLatitude(); Double longitude = center.getLongitude();
    if(postType.equals("road")) { // 전 프래그먼트에서 길 선택했을 경우
        ChoosePointFragmentDirections.ActionFragmentPointToFragmentWriteRoad roadPoint =
            ChoosePointFragmentDirections.actionFragmentPointToFragmentWriteRoad(latitude.toString(), longitude.toString());
        Navigation.findNavController(v).navigate(roadPoint);
    }
});
```

- ③ 작성 완료 버튼 클릭 시, 필수 항목(이용 가능한 정도, 교통약자 유형 선택)을 모두 작성하면 제보글 작성이 완료된다.

```

@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.imagechoose:
            Intent intent = new Intent(Intent.ACTION_PICK);
            intent.setDataAndType(android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI, type: "image/*");
            startActivityForResult(intent, GET_GALLERY_IMAGE);
            break;
        case R.id.reset:
            spinner_availability.setAdapter(adapter_availability);
            spinner_angle.setAdapter(adapter_angle);
            spinner_type.setAdapter(adapter_type);
            stairs.setChecked(false);
            breakage.setChecked(false);
            // 이미지뷰 초기화
            break;
        case R.id.writeFin:
            String lat = RoadFragmentArgs.fromBundle(getArguments()).getLatitude();
            String lon = RoadFragmentArgs.fromBundle(getArguments()).getLongitude();
            id = appData.getString( key: "ID", defValue: ""); // 로그인 정보

            if(!spinner_availability.getSelectedItem().equals("선택") &&
                !spinner_type.getSelectedItem().equals("선택")) {
                PostRequest postRequest = new PostRequest(getContext());
                postRequest.writeRoad(id, lat, lon, spinner_availability.getSelectedItem().toString(),
                    spinner_type.getSelectedItem().toString(), stairs.getText().toString(),
                    breakage.getText().toString(), spinner_angle.getSelectedItem().toString(),
                    v, url: "postRoadRegister.php");
            }
            else {
                AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
                builder.setMessage("필수 항목을 전부 입력해주시시오.")
                    .setNegativeButton( text: "확인", listener: null)
                    .create().show();
            }
            break;
    }
}

```

- ④ 도로 제보글 작성 시 사용자의 아이디, 해당 사용자가 작성한 필수 항목(이용 가능한 정도, 교통약자 유형 선택)과 지도에서 선택한 위치의 좌표가 서버로 전송된다. 선택 항목(도로 경사 각도, 계단 유무, 도로 파손 여부)을 작성한 경우, 필수 항목과 함께 전송된다.

```

//도로(길) 제보글 작성
public void writeRoad(String userID, String latitude, String longitude, String availability, String type,
    String stair, String roadBreakage, String slope, View v, String url) {
    parameters = new HashMap<>();
    parameters.put("userID", userID);
    parameters.put("latitude", latitude);
    parameters.put("longitude", longitude);
    parameters.put("availability", availability);
    parameters.put("type", type);
    parameters.put("stair", stair);
    parameters.put("roadBreakage", roadBreakage);
    parameters.put("slope", slope);

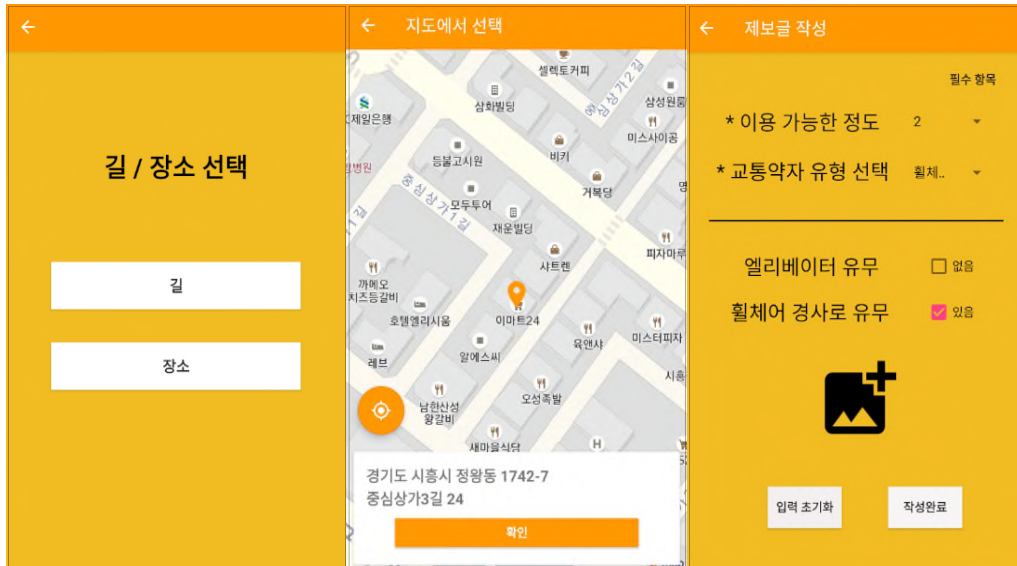
    sendRequest(v, url);
}

```

5) 장소 제보글 작성

a. 출력 화면

왼쪽부터 차례대로 제보글 유형 선택 화면, 제보글 위치 선택 화면, 장소 제보글 작성 화면이다.



b. 기능 구현 코드

- ① 제보글 유형 선택 프래그먼트에서 제보글의 유형을 선택한다. 도로(길)과 장소 중에서 장소를 선택한 경우 type을 place로 전송한다.

```
buttonPlace.setOnClickListener(new Button.OnClickListener(){
    @Override
    public void onClick(View v) { // 위와 동일
        ChoosePostTypeFragmentDirections.ActionFragmentWritePostToFragmentPoint action
            = ChoosePostTypeFragmentDirections.actionFragmentWritePostToFragmentPoint( type: "place");
        Navigation.findNavController(v).navigate(action);
    }
});
```

- ② 앞에서 전송한 type값으로 도로 제보글인지 장소 제보글인지를 구분한다. type 값이 place면 지도에서 선택한 위치의 좌표를 전송하고, 장소 제보글 작성 화면으로 이동한다.

```
activity.setPoint(choosePoint, address); // 중심 주소 표시
choosePoint.setOnClickListener((v) -> {
    String postType = ChoosePointFragmentArgs.fromBundle(getArguments()).getType();
    TMapPoint center = activity.getCenter();

    appData = getActivity().getSharedPreferences( name: "appData", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = appData.edit();

    Double latitude = center.getLatitude(); Double longitude = center.getLongitude();
    if(postType.equals("road")) { // 전 프래그먼트에서 길 선택했을 경우
        ChoosePointFragmentDirections.ActionFragmentPointToFragmentWriteRoad roadPoint =
            ChoosePointFragmentDirections.actionFragmentPointToFragmentWriteRoad(latitude.toString(), longitude.toString());
        Navigation.findNavController(v).navigate(roadPoint);
    }
    else if(postType.equals("place")){ // 장소 선택했을 경우
        ChoosePointFragmentDirections.ActionFragmentPointToFragmentWritePlace placePoint =
            ChoosePointFragmentDirections.actionFragmentPointToFragmentWritePlace(latitude.toString(), longitude.toString());
        Navigation.findNavController(v).navigate(placePoint);
    }
});
```


- ③ 작성 완료 버튼 클릭 시, 필수 항목(이용 가능한 정도, 교통약자 유형 선택)을 모두 작성하면 제보글 작성이 완료된다.

```

@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.imagechoose:
            Intent intent = new Intent(Intent.ACTION_PICK);
            intent.setDataAndType(android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI, type: "image/*");
            startActivityForResult(intent, GET_GALLERY_IMAGE);
            break;
        case R.id.reset:
            spinner_availability.setAdapter(adapter_availability);
            spinner_type.setAdapter(adapter_type);
            elevator.setChecked(false);
            wheel.setChecked(false);
            // 이미지 뷰 초기화 -> 아직 남음
            break;
        case R.id.writeFin:
            String lat = PlaceFragmentArgs.fromBundle(getArguments()).getLatitude();
            String lon = PlaceFragmentArgs.fromBundle(getArguments()).getLongitude();
            id = appData.getString( key: "ID", defValue: ""); // 로그인 정보

            if(!spinner_availability.getSelectedItem().equals("선택") &&
                !spinner_type.getSelectedItem().equals("선택")){
                PostRequest postRequest = new PostRequest(getContext());
                postRequest.writePlace(id, lat, lon, spinner_availability.getSelectedItem().toString(),
                    spinner_type.getSelectedItem().toString(), elevator.getText().toString(),
                    wheel.getText().toString(), v, url: "postPlaceRegister.php");
            }
            else {
                AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
                builder.setMessage("필수 항목을 전부 입력해주시시오.")
                    .setNegativeButton( text: "확인", listener: null)
                    .create()
                    .show();
            }
            break;
    }
}

```

- ④ 장소 제보글 작성 시 사용자의 아이디, 해당 사용자가 작성한 필수 항목(이용 가능한 정도, 교통약자 유형 선택)과 지도에서 선택한 위치의 좌표가 서버로 전송된다. 선택 항목(엘리베이터 유무, 휠체어 경사로 유무)을 작성한 경우, 필수 항목과 함께 전송된다.

```

// 장소 제보글 작성
public void writePlace(String userID, String latitude, String longitude, String availability, String type,
    String elevator, String wheel, View v, String url){
    parameters = new HashMap<>();
    parameters.put("userID", userID);
    parameters.put("latitude", latitude);
    parameters.put("longitude", longitude);
    parameters.put("availability", availability);
    parameters.put("type", type);
    parameters.put("elevator", elevator);
    parameters.put("wheel", wheel);

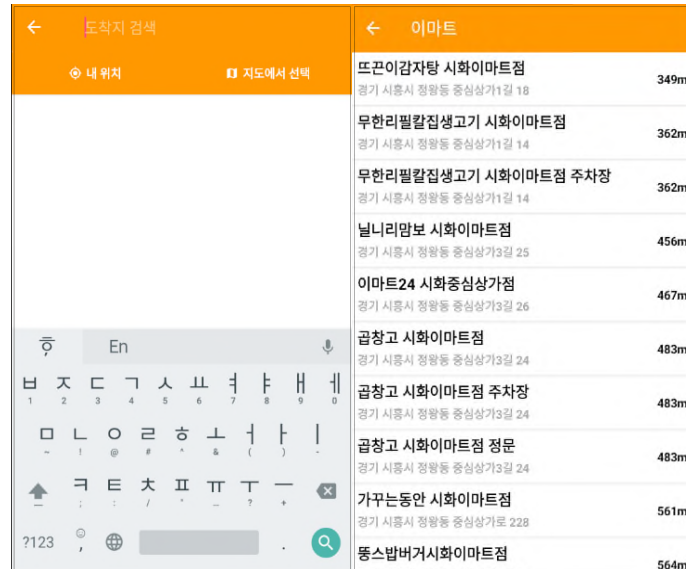
    sendRequest(v, url);
}

```

6) 장소 검색

a. 출력 화면

왼쪽은 길찾기 시 장소 검색 화면, 오른쪽은 검색 내용 입력 시 검색 결과 화면이다.



b. 기능 구현 코드

- 출발지 또는 도착지를 선택하고 검색어를 입력하면 검색 프래그먼트에서 입력 받은 검색어를 처리한다. 출발지와 도착지 중에서 검색 시 선택한 항목이 type 값으로 전송된다.

```

@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
    ViewGroup view = (ViewGroup) inflater.inflate(R.layout.fragment_search, container, attachToRoot: false);
    activity.findViewById(R.id.appBarLayout).setOutlineProvider(null);

    type = SearchFragmentArgs.fromBundle(getArguments()).getType(); // 검색 타입(출발지/도착지)

    if(type.equals("start"))
        searchView.setQueryHint("출발지 검색");
    else
        searchView.setQueryHint("도착지 검색");
}

```

- type값과 입력한 검색어를 검색 결과 프래그먼트에 전송하고, 검색 결과 화면으로 이동한다.

```

searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
        SearchFragmentDirections.actionFragmentSearchToFragmentSearchResult
            = SearchFragmentDirections.actionFragmentSearchToFragmentSearchResult(type, query);
        Navigation.findNavController(getView()).navigate(actionFragmentSearchToFragmentSearchResult);
        return false;
    }
}

```

- ①에서 전송한 type 값과 사용자가 입력한 검색어를 받아온다. type값으로 출발지와 목적지 중 어느 것을 검색하는지를 구분한다.

```

@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
    ViewGroup view = (ViewGroup) inflater.inflate(R.layout.fragment_search_result, container, attachToRoot: false);

    Toolbar toolbar = activity.findViewById(R.id.toolbar);
    type = SearchResultFragmentArgs.fromBundle(getArguments()).getType(); // 출발지/도착지 구분
    String searchWorld = SearchResultFragmentArgs.fromBundle(getArguments()).getSearchWord(); // 검색어

```

- ④ 사용자의 현재 위치를 가져오고, 사용자의 현재 위치와 입력한 검색어를 사용해 검색결과 목록을 생성한다. 검색어와 일치하는 장소 중 사용자의 현재 위치에서 반경 33km 이내에 있는 장소를 100개까지 출력한다.

```

// 검색어 이용 리스트 생성
public void createSearchResultList(String searchWorld) {

    TMapData tMapData = new TMapData();
    TMapPoint point = activity.getLocation(); // 현재 위치 받아오기

    // 33km 내, 100개 결과 출력
    tMapData.findAroundKeywordPOI(point, searchWorld, nRadius: 33, nSearchCount: 100, new TMapData.FindAroundKeywordPOIListenerCallback() {
        @Override
        public void onFindAroundKeywordPOI(ArrayList<TMapPOIItem> arrayList) {
            adapter.addItem(arrayList, activity, type);
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    adapter.notifyDataSetChanged();
                }
            });
        }
    });
}

```

- ⑤ 검색결과 목록의 각 항목은 장소명과 주소, 현재 위치와의 거리를 표시한다. 목록에서 항목을 선택하면 길찾기 화면으로 이동하고, 출발지 또는 목적지에 선택한 항목의 장소명이 표시된다.

```

public ItemViewHolder(@NonNull final View itemView) {
    super(itemView);
    name = itemView.findViewById(R.id.poiName);
    address = itemView.findViewById(R.id.poiAddress);
    distance = itemView.findViewById(R.id.distance);
    itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            appData = activity.getSharedPreferences( name: "appData", Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = appData.edit();
            String latitude = list.get(getAdapterPosition()).noorLat;
            String longitude = list.get(getAdapterPosition()).noorLon;

            if(type.equals("start")){ // 길찾기 지도 선택
                editor.putString("StartLat", latitude);
                editor.putString("StartLong", longitude);
            }
            else{
                editor.putString("EndLat", latitude);
                editor.putString("EndLong", longitude);
            }
            editor.apply(); editor.commit();
            Navigation.findNavController(v).navigate(R.id.action_fragment_search_result_to_fragment_findRoute);
        }
    });
}

```

- ⑥ 리스트에 출력되는 각 항목은 해당하는 위치의 위도, 경도가 주소로 변환되고, 사용자의 현재 위치와의 거리가 계산된다. 각각의 장소명과 주소, 현재 위치와의 거리가 화면에 표시된다.

```
void onBind(TMapPOIItem item){
    double radius = Double.parseDouble(item.radius);
    String firstAddress = item.upperAddrName + " " + item.middleAddrName + " " + item.lowerAddrName
        + " " + item.roadName + " " + item.buildingNo1;
    String secondAddress = item.buildingNo2;

    firstAddress = firstAddress.replaceAll( regex: "null", replacement: "");
    if(secondAddress == null)
        address.setText(firstAddress);
    else if(secondAddress.equals("0")){
        secondAddress = secondAddress.replaceAll( regex: "0", replacement: "");
        address.setText(firstAddress + secondAddress);
    }
    else
        address.setText(firstAddress + "-" + secondAddress);

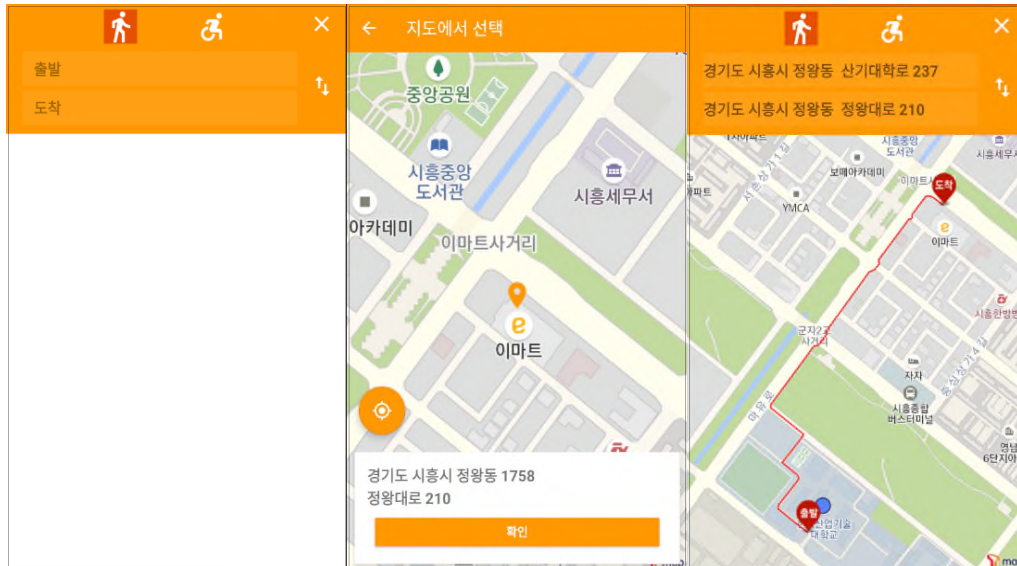
    if(radius < 1)
        distance.setText(Math.round(radius * 1000) + "m");
    else
        distance.setText(Math.round(radius * 100) / 100.0 + "km");

    name.setText(item.getPOIName());
}
```

7) 최단 경로 길찾기

a. 출력 화면

왼쪽부터 길찾기 버튼 클릭 시 화면, 출발지/목적지 선택 후 지도에서 찾기 버튼 클릭 시 화면, 출발지, 목적지를 모두 작성하면 출력되는 최단 경로 화면이다.



b. 기능 구현 코드

① 검색 프래그먼트에서 출발지, 목적지를 표시한다.

```
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
    ViewGroup view = (ViewGroup) inflater.inflate(R.layout.fragment_search, container, attachToRoot: false);
    activity.findViewById(R.id.appBarLayout).setOutlineProvider(null);

    type = SearchFragmentArgs.fromBundle(getArguments()).getType(); // 검색 타입(출발지/도착지)

    view.findViewById(R.id.chooseappoint).setOnClickListener(this);
    view.findViewById(R.id.mylocation).setOnClickListener(this);

    searchView = (SearchView)activity.findViewById(R.id.search_view);
    searchView.setVisibility(View.VISIBLE);
    searchView.setIconified(false);
    searchView.setOnClickListener(this);

    if(type.equals("start"))
        searchView.setQueryHint("출발지 검색");
    else
        searchView.setQueryHint("도착지 검색");

    closeButton = (ImageView) searchView.findViewById(androidx.appcompat.R.id.search_close_btn);
    closeButton.setVisibility(View.INVISIBLE);

    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) { return false; }

        @Override
        public boolean onQueryTextChange(String newText) {...}
    });
    return view;
}
```


- ② 길찾기 화면에서 출발지를 선택한 경우 type에 start를, 목적지를 선택한 경우 end를 전송한다. 출발지,목적지 바꾸기 버튼을 선택하면 출발지를 목적지로, 목적지를 출발지로 설정하고 경로를 표시한다.

```

@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.end:
            editor.remove("StartLat"); editor.remove("StartLong");
            editor.remove("EndLat"); editor.remove("EndLong");
            editor.commit(); // 길찾기 종료 시 저장된 좌표값 삭제

            // 메인화면으로 이동
            Navigation.findNavController(v).navigate(R.id.action_fragment_findRoute_to_fragment_map);
            break;
        case R.id.route:
            routeType = true;
            route.setSelected(true); avoidRoute.setSelected(false);
            break;
        case R.id.avoidRoute:
            routeType = false;
            route.setSelected(false); avoidRoute.setSelected(true);
            break;
        case R.id.start :
            FindRouteFragmentDirections.ActionFragmentFindRouteToFragmentSearch actionFragmentFindRouteToFragmentSearchStart
                = FindRouteFragmentDirections.actionFragmentFindRouteToFragmentSearch( type: "start");
            Navigation.findNavController(v).navigate(actionFragmentFindRouteToFragmentSearchStart);
            break;
        case R.id.arrive:
            FindRouteFragmentDirections.ActionFragmentFindRouteToFragmentSearch actionFragmentFindRouteToFragmentSearchArrive
                = FindRouteFragmentDirections.actionFragmentFindRouteToFragmentSearch( type: "arrive");
            Navigation.findNavController(v).navigate(actionFragmentFindRouteToFragmentSearchArrive);
            break;

        case R.id.arrive:
            FindRouteFragmentDirections.ActionFragmentFindRouteToFragmentSearch actionFragmentFindRouteToFragmentSearchArrive
                = FindRouteFragmentDirections.actionFragmentFindRouteToFragmentSearch( type: "arrive");
            Navigation.findNavController(v).navigate(actionFragmentFindRouteToFragmentSearchArrive);
            break;
        case R.id.change:
            editor.putString("StartLat", endLat);
            editor.putString("StartLong", endLong);
            editor.putString("EndLat", startLat);
            editor.putString("EndLong", startLong);
            editor.apply(); editor.commit();

            if(startLat.equals(""))
                end.setText("");
            if(endLat.equals(""))
                start.setText("");

            setFindRoute();
        default:
            break;
    }
}

```

- ③ 사용자의 현재 위치나 사용자가 지도에서 선택한 위치의 좌표를 받아와 주소로 변환한다. 앞에서 전송한 type값으로 출발지와 목적지를 구분한다.

```
@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.search_view:
            searchView.setIconified(false);
            searchView.onActionViewExpanded();
        case R.id.mylocation:
            TMapPoint point = activity.getLocation(); // 현재 위치 받아오기
            Double latitude = point.getLatitude(); Double longitude = point.getLongitude();
            appData = getActivity().getSharedPreferences( name: "appData", Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = appData.edit();
            if(type.equals("start")) {
                editor.putString("StartLat", latitude.toString());
                editor.putString("StartLong", longitude.toString());
            }
            else {
                editor.putString("EndLat", latitude.toString());
                editor.putString("EndLong", longitude.toString());
            }
            editor.apply(); editor.commit();
            Navigation.findNavController(v).navigate(R.id.action_fragment_search_to_fragment_findRoute);
            break;
        case R.id.choosemappoint:
            SearchFragmentDirections.ActionFragmentSearchToFragmentPoint action
                = SearchFragmentDirections.actionFragmentSearchToFragmentPoint(type);
            Navigation.findNavController(v).navigate(action);
    }
}
```

- ④ 지도에서 위치 선택 시 선택한 위치의 좌표를 주소로 변환해 지도에 표시한다.

```
//지도에서 위치 선택시 하단에 해당좌표의 주소 표시
public void setAddress(final TextView address, final TMapPoint center) {
    TMapData tMapData = new TMapData();
    tMapData.reverseGeocoding(center.getLatitude(), center.getLongitude(), addressType: "A10", (tMapAddressInfo) -> {
        String str = tMapAddressInfo.strCity_do+ " "+tMapAddressInfo.strGu_gun+ " "
            + tMapAddressInfo.strLegalDong+ " "+tMapAddressInfo.strBunji+ " "
            + tMapAddressInfo.strRi+ "\n"+tMapAddressInfo.strRoadName+ " "+
            tMapAddressInfo.strBuildingIndex;
        str = str.replaceAll( regex: "null", replacement: "");
        address.setText(str);
    });
}
```

- ⑤ 출발지나 목적지를 선택하면 선택한 좌표를 주소로 변환한다. 출발지, 목적지가 모두 입력된 경우 최단 경로를 구한다.

```
public void setFindRoute(){
    getPoint(); // 지정된 point 가져옴 (String 이기 때문에 Double 로 변환 필요)
    // 해당 프래그먼트 불러올 때 point 값 확인 하여 모든 값이 지정되어 있는 경우 길찾기 시작 -> 화면 나갈 때 저장된 값 삭제 필요
    // 둘 중 하나 이상 지정된 경우 주소 표시

    //출발지만 null이 아닌 경우
    if(!startLat.equals("")){
        //가져온 값을 double로 변환
        start_point = convDouble(startLat, startLong);
        activity.setRouteAddress(start, start_point);
    }

    //도착지만 null이 아닌 경우
    if(!endLat.equals("")) {
        //가져온 값을 double로 변환
        end_point = convDouble(endLat, endLong);
        activity.setRouteAddress(end, end_point);
    }

    //출발지, 도착지 모두 null이 아닌 경우
    if( (!startLat.equals("")) && (!endLat.equals("")) ){
        rootHistory.setVisibility(View.INVISIBLE);

        //최단거리 구함
        activity.getShortestPath(start_point, end_point);
    }
}
```

- ⑥ 선택한 위치의 좌표를 가져와 주소로 변환하고, 출발지와 목적지에 표시한다.

```
//출발지, 목적지의 좌표(위도, 경도)가져옴
public void getPoint(){
    startLat = appData.getString( key: "StartLat", defValue: "");
    startLong = appData.getString( key: "StartLong", defValue: "");
    endLat = appData.getString( key: "EndLat", defValue: "");
    endLong = appData.getString( key: "EndLong", defValue: "");
}
```

```
//문자열을 실수로 형변환
public TMapPoint convDouble(String sLat, String sLong){
    //double형으로 형변환
    double dLat = Double.parseDouble(sLat);
    double dLong = Double.parseDouble(sLong);
    TMapPoint point = new TMapPoint(dLat, dLong);
    return point;
}
```

```
//길찾기 시 선택한 위치를 주소로 변환, 출발지/목적지에 표시
public void setRouteAddress(final TextView address, final TMapPoint point) {
    TMapData tMapData = new TMapData();
    tMapData.reverseGeocoding(point.getLatitude(), point.getLongitude(), addressType: "A10", (tMapAddressInfo) -> {
        String str = tMapAddressInfo.strCity_do+ " "+tMapAddressInfo.strGu_gun+ " "
            + tMapAddressInfo.strLegalDong+ " "+ tMapAddressInfo.strRi+ " "
            +tMapAddressInfo.strRoadName+ " "+ tMapAddressInfo.strBuildingIndex;
        str = str.replaceAll( regex: "null", replacement: "");
        address.setText(str);
    });
}
```


⑦ 입력받은 출발지, 목적지를 가지고 최단 경로를 지도에 표시한다.

```
//출발지, 목적지를 받아 (최단)경로 표시
public void getShortestPath(TMapPoint start, TMapPoint arrive) {
    //출발지의 위도, 경도(start.latitude, start.longitude)
    start = new TMapPoint(start.getLatitude(), start.getLongitude());
    //목적지의 위도, 경도
    arrive = new TMapPoint(arrive.getLatitude(), arrive.getLongitude());

    //출발지, 목적지를 마커로 표시
    TMapMarkerItem startMark = new TMapMarkerItem();
    TMapMarkerItem endMark = new TMapMarkerItem();
    startMark.setTMapPoint(start);
    endMark.setTMapPoint(arrive);

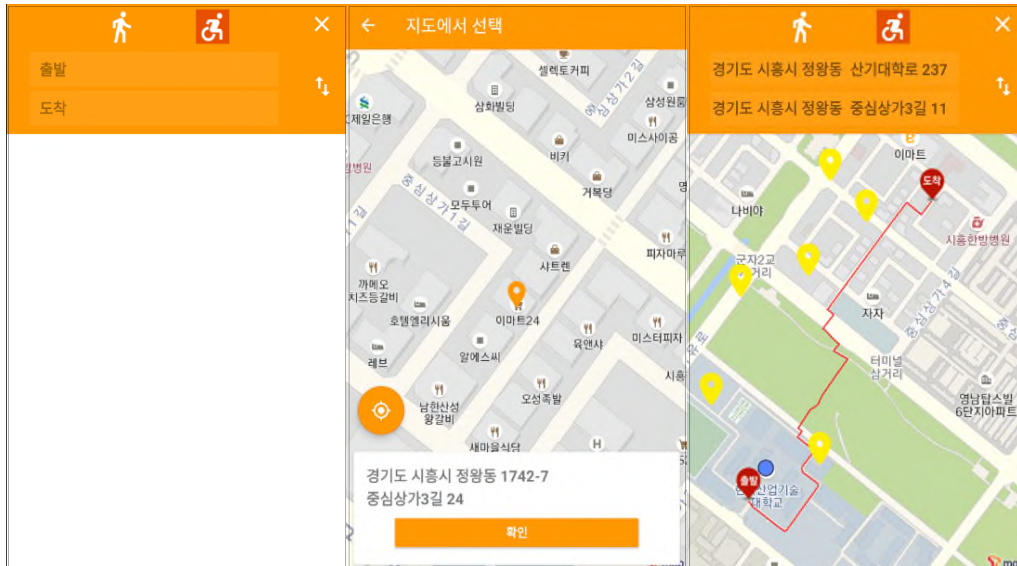
    //보행자 (최단)경로를 요청함(경로 종류, 출발지, 도착지, 검색결과를 받는 인터페이스 함수)
    TMapData data = new TMapData();
    final TMapPoint finalStart = start;
    final TMapPoint finalArrive = arrive;
    data.findPathDataWithType(TMapData.TMapPathType.PEDESTRIAN_PATH, start, arrive, (path) -> {
        //UiThread 사용
        runOnUiThread(() -> {
            //보여줄 경로(선)의 두께, 색상 설정
            path.setLineWidth(5);
            path.setLineColor(Color.RED);
            //경로를 지도에 표시함
            tMapView.addTMapPath(path);

            //결과에 맞춰 중심 좌표, 확대 정도 변경
            ArrayList<TMapPoint> point = new ArrayList<>();
            point.add(finalStart);
            point.add(finalArrive);
            TMapInfo info = tMapView.getDisplayTMapInfo(point);
            tMapView.setCenterPoint(info.getTMapPoint().getLongitude(), info.getTMapPoint().getLatitude());
            //+0.0020
            tMapView.setZoomLevel(info.getTMapZoomLevel()-1);
        });
    });
}
```

8) 회피 경로 길찾기

a. 출력 화면

왼쪽부터 회피 경로 길찾기 선택 시 화면, 출발지/목적지 선택 후 지도에서 찾기 버튼 클릭 시 화면, 출발지와 목적지를 모두 작성하면 출력되는 회피 경로 화면이다.



b. 기능 구현 코드

① 검색 프래그먼트에서 출발지, 목적지를 표시한다.

```
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
    ViewGroup view = (ViewGroup) inflater.inflate(R.layout.fragment_search, container, attachToRoot: false);
    activity.findViewById(R.id.appBarLayout).setOutlineProvider(null);

    type = SearchFragmentArgs.fromBundle(getArguments()).getType(); // 검색 타입(출발지/도착지)

    view.findViewById(R.id.chooseappoint).setOnClickListener(this);
    view.findViewById(R.id.mylocation).setOnClickListener(this);

    searchView = (SearchView)activity.findViewById(R.id.search_view);
    searchView.setVisibility(View.VISIBLE);
    searchView.setIconified(false);
    searchView.setOnClickListener(this);

    if(type.equals("start"))
        searchView.setQueryHint("출발지 검색");
    else
        searchView.setQueryHint("도착지 검색");

    closeButton = (ImageView) searchView.findViewById(androidx.appcompat.R.id.search_close_btn);
    closeButton.setVisibility(View.INVISIBLE);

    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) { return false; }

        @Override
        public boolean onQueryTextChange(String newText) {...}
    });
    return view;
}
```

- ② 길찾기 화면에서 출발지를 선택한 경우 type에 start를, 목적지를 선택한 경우 end를 전송한다. 출발지, 목적지 바꾸기 버튼을 선택하면 출발지를 목적지로, 목적지를 출발지로 설정하고 경로를 표시한다.

```
public void onClick(View v) {
    switch (v.getId()){
        case R.id.end:
            editor.remove("StartLat"); editor.remove("StartLong");
            editor.remove("EndLat"); editor.remove("EndLong");
            editor.commit(); // 길찾기 종료 시 저장된 좌표값 삭제
            Navigation.findNavController(v).navigate(R.id.action_fragment_findRoute_to_fragment_map);
            searchView.clearFocus(); searchView.setIconified(true);
            break;
        case R.id.route:
            checkButton( routeType: "", button: "route");
            setFindRoute();
            break;
        case R.id.avoidRoute:
            checkButton( routeType: "", button: "avoidRoute");
            setFindRoute();
            break;
        case R.id.start :
            FindRouteFragmentDirections.ActionFragmentFindRouteToFragmentSearch actionFragmentFindRouteToFragmentSearchStart
                = FindRouteFragmentDirections.actionFragmentFindRouteToFragmentSearch( type: "start");
            Navigation.findNavController(v).navigate(actionFragmentFindRouteToFragmentSearchStart);
            break;
        case R.id.arrive:
            FindRouteFragmentDirections.ActionFragmentFindRouteToFragmentSearch actionFragmentFindRouteToFragmentSearchArrive
                = FindRouteFragmentDirections.actionFragmentFindRouteToFragmentSearch( type: "arrive");
            Navigation.findNavController(v).navigate(actionFragmentFindRouteToFragmentSearchArrive);
            break;
        case R.id.change:
            editor.putString("StartLat", endLat);
            editor.putString("StartLong", endLong);
            editor.putString("EndLat", startLat);
            editor.putString("EndLong", startLong);
            editor.apply(); editor.commit();

            if(startLat.equals(""))
                end.setText("");
            if(endLat.equals(""))
                start.setText("");
            setFindRoute();
            break;
    }
}
```

- ③ 사용자의 현재 위치나 사용자가 지도에서 선택한 위치의 좌표를 받아와 주소로 변환한다. 앞에서 전송한 type값으로 출발지와 목적지를 구분한다.

```
@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.search_view:
            searchView.setIconified(false);
            searchView.onActionViewExpanded();
        case R.id.mylocation:
            TMapPoint point = activity.getLocation(); // 현재 위치 받아오기
            Double latitude = point.getLatitude(); Double longitude = point.getLongitude();
            appData = getActivity().getSharedPreferences( name: "appData", Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = appData.edit();
            if(type.equals("start")) {
                editor.putString("StartLat", latitude.toString());
                editor.putString("StartLong", longitude.toString());
            }
            else {
                editor.putString("EndLat", latitude.toString());
                editor.putString("EndLong", longitude.toString());
            }
            editor.apply(); editor.commit();
            Navigation.findNavController(v).navigate(R.id.action_fragment_search_to_fragment_findRoute);
            break;
        case R.id.choosemappoint:
            SearchFragmentDirections.ActionFragmentSearchToFragmentPoint action
                = SearchFragmentDirections.actionFragmentSearchToFragmentPoint(type);
            Navigation.findNavController(v).navigate(action);
    }
}
```

- ④ 지도에서 위치 선택 시 선택한 위치의 좌표를 주소로 변환해 지도에 표시한다.

```
//지도에서 위치 선택시 하단에 해당좌표의 주소 표시
public void setAddress(final TextView address, final TMapPoint center) {
    TMapData tMapData = new TMapData();
    tMapData.reverseGeocoding(center.getLatitude(), center.getLongitude(), addressType: "A10", (tMapAddressInfo) -> {
        String str = tMapAddressInfo.strCity_do+ " "+tMapAddressInfo.strGu_gun+ " "
            + tMapAddressInfo.strLegalDong+ " "+tMapAddressInfo.strBunji+ " "
            + tMapAddressInfo.strRi+ "\n"+tMapAddressInfo.strRoadName+ " "+
            tMapAddressInfo.strBuildingIndex;
        str = str.replaceAll( regex: "null", replacement: "");
        address.setText(str);
    });
}
```

- ⑤ 출발지나 목적지를 선택하면 선택한 좌표를 주소로 변환한다. 회피 경로 버튼을 클릭했으며, 출발지와 목적지가 모두 입력되고, 둘 사이의 거리가 2km 이하일 경우 회피 경로 요청 함수를 호출한다.

```
public void setFindRoute(){
    getPoint(); // 지정된 point 가져올 (String 이기 때문에 Double 로 변환 필요)
    // 해당 프래그먼트 불러올 때 point 값 확인 하여 모든 값이 지정되어 있는 경우 길찾기 시작 -> 화면 나갈 때 저장된 값 삭제 필요
    // 둘 중 하나 이상 지정된 경우 주소 표시

    //출발지만 null이 아닌 경우
    if(!(startLat.equals(""))){
        //가져온 값을 double로 변환
        start_point = convDouble(startLat, startLong);
        activity.setRouteAddress(start, start_point);
    }

    //도착지만 null이 아닌 경우
    if(!(endLat.equals(""))){
        //가져온 값을 double로 변환
        end_point = convDouble(endLat, endLong);
        activity.setRouteAddress(end, end_point);
    }
}
```

```
//출발지, 도착지 모두 null이 아닌 경우
if( !(startLat.equals("")) && !(endLat.equals("")) ) {
    String routeType = appData.getString( key: "routeType", defValue: "");
    //회피거리 구함
    if(routeType.equals("route")) {
        rootHistory.setVisibility(View.INVISIBLE);
        activity.getShortestPath(start_point, end_point, passList: null, searchOption: 0);
    }
    else{
        // 거리가 2km 이하일 경우
        // 회피 경로 찾기
        double distance = getDistance();
        rootHistory.setVisibility(View.VISIBLE);

        if(distance > 2) {
            Toast.makeText(activity.getApplicationContext(), text: "회피 경로 찾기는 두 지점 사이의 거리가 \n2km 이하인 경우에만 가능합니다.", Toast.LENGTH_LONG).show();
        }
        else {
            getPostLocation(distance);
        }
    }
}
}
```

```
//출발지, 목적지의 좌표(위도, 경도)가져옴
public void getPoint(){
    startLat = appData.getString( key: "StartLat", defValue: "");
    startLong = appData.getString( key: "StartLong", defValue: "");
    endLat = appData.getString( key: "EndLat", defValue: "");
    endLong = appData.getString( key: "EndLong", defValue: "");
}
```

```
//문자열을 실수로 형변환
public TMapPoint convDouble(String sLat, String sLong){
    //double형으로 형변환
    double dLat = Double.parseDouble(sLat);
    double dLong = Double.parseDouble(sLong);
    TMapPoint point = new TMapPoint(dLat, dLong);
    return point;
}
```

```
//길찾기 시 선택한 위치를 주소로 변환, 출발지/목적지에 표시
public void setRouteAddress(final TextView address, final TMapPoint point) {
    TMapData tMapData = new TMapData();
    tMapData.reverseGeocoding(point.getLatitude(), point.getLongitude(), addressType: "A10", (tMapAddressInfo) -> {
        String str = tMapAddressInfo.strCity_do+" "+tMapAddressInfo.strGu_gun+" "
            + tMapAddressInfo.strLegalDong+" " + tMapAddressInfo.strRi+" "
            +tMapAddressInfo.strRoadName+" "+ tMapAddressInfo.strBuildingIndex;
        str = str.replaceAll( regex: "null", replacement: "");
        address.setText(str);
    });
}
```


⑥ 서버에서 도출된 경로의 주변 제보글 목록을 받아온다.

```

public void getPostLocation(double distance){
    // 서버로부터 제보글 목록 받기
    progressBar.setVisibility(View.VISIBLE);
    PostLocationRequest request = new PostLocationRequest(getContext(), activity, start_point, end_point, distance, rootHistory, progressBar);
    request.sendPostLocation( requestUrl: "getLocation.php");
}

public PostLocationRequest(Context context, MainActivity activity, TMapPoint start_point, TMapPoint end_point, double distance,
    CardView rootHistory, ProgressBar progressBar) {
    this.context = context;
    this.activity = activity;
    this.start_point = start_point;
    this.end_point = end_point;
    this.distance = distance;
    this.rootHistory = rootHistory;
    this.progressBar = progressBar;
    Log.d( tag: "거리 test", String.valueOf(distance));
    parameters = new HashMap<>();
    String post = start_point.getLongitude() + "," + start_point.getLatitude()
        + "," + end_point.getLongitude() + "," + end_point.getLatitude() + "," + distance;
    parameters.put("postLocation", post);
}

public void sendPostLocation(String requestUrl) {
    String url = baseUrl + requestUrl;
    postID.clear();
    StringRequest request = new StringRequest(
        Request.Method.POST,
        url,
        new Response.Listener<String>() { // 정상 응답
            @Override
            public void onResponse(String mJsonString) {
                try {
                    JSONObject jsonObject = new JSONObject(mJsonString);
                    JSONArray jsonArray = jsonObject.getJSONArray(TAG_JSON);

                    for (int i = 0; i < jsonArray.length(); i++) {
                        JSONObject item = jsonArray.getJSONObject(i);
                        String longitude = item.getString(TAG_Longitude);
                        String latitude = item.getString(TAG_Latitude);

                        TMapPoint post = new TMapPoint(item.getDouble(TAG_Longitude), item.getDouble(TAG_Latitude));
                        postLocation.add(post);
                        postID.add(getLinkID(latitude, longitude));
                    }
                    getAvoidRoute();
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                // 에러 발생
            }
        }) {

        @Override
        protected Map<String, String> getParams() { return parameters; }
    };
    request.setShouldCache(false);
    Volley.newRequestQueue(context).add(request);
}

```

- ⑦ 도출 경로와 서버로부터 받아온 제보글 위치 정보를 비교하여 포함 여부를 확인한다.

```
public String getLinkID(final String lat, final String lon) {
    String linkID = null;
    Thread mThread = run() -> {
        try {
            URL url;
            String str = "";
            url = new URL( spec: "https://apis.openapi.sk.com/tmap/road/nearToRoad?version=1&format=json&opt=2" +
                "&appKey=" + "17>xf199d1a5d81142c4bcb217ace3818a7b" + "&lat=" + lat
                + "&lon=" + lon);

            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("POST");
            conn.setDoOutput(true);

            if (conn.getResponseCode() == conn.HTTP_OK) {
                InputStreamReader tmp = new InputStreamReader(conn.getInputStream(), charsetName: "UTF-8");
                BufferedReader reader = new BufferedReader(tmp);
                StringBuffer buffer = new StringBuffer();
                while ((str = reader.readLine()) != null) {
                    buffer.append(str);
                }
                receiveMsg = buffer.toString();
                reader.close();
            } else {
                Log.i( tag: "통신 결과", msg: conn.getResponseCode() + "에러");
            }
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    };
    mThread.start(); // Thread 실행
    try {
        mThread.join();
        JSONObject object = new JSONObject(receiveMsg).getJSONObject("resultData").getJSONObject("header");
        linkID = object.getString( name: "linkId") + object.getString( name: "idxName");
    } catch (InterruptedException | JSONException e) {
        e.printStackTrace();
    }
    return linkID;
}
```

- ⑧ 포함할 경우 다시 경로를 도출하고, 포함하지 않을 경우, 입력받은 출발지, 목적지를 가지고 회피 경로를 지도에 표시한다.

```
public void getAvoidRoute() {
    Boolean isInclude = false; // 재보글 포함 여부

    for (int i = 0; i < 4; i++) {
        makeRoute(searchOption[i], passList: "");
        isInclude = checkRoute();

        if (isInclude.equals(true)) {
            // 경로 출력 함수
            postID.clear();
            ArrayList<TMapPoint> startEnd = new ArrayList<>();
            startEnd.add(start_point); startEnd.add(end_point);

            rootHistory.setVisibility(View.INVISIBLE);
            progressBar.setVisibility(View.INVISIBLE);
            activity.getAvoidPath(list, startEnd);

            return;
        }
        list.clear();
    }

    if (isInclude.equals(false)) {
        progressBar.setVisibility(View.INVISIBLE);
        postID.clear();
        Toast.makeText(activity.getApplicationContext(), text: "회피 경로를 찾을 수 없습니다", Toast.LENGTH_LONG).show();
    }
}
```



```

public void makeRoute(final int searchOption, final String passList) {
    coordinateID = new ArrayList<>();
    Thread mThread = run() → {
        try {
            URL url;
            String str = "";

            if (passList.equals("")) {
                url = new URL( spec: "https://apis.openapi.sk.com/tmap/routes/pedestrian?version=1&format=json" +
                    "&appKey=" + "17xxcf199d1a5d81142c4bcb217ace3818a7b" + "&startX=" + Double.toString(start_point.getLongitude())
                    + "&startY=" + Double.toString(start_point.getLatitude()) + "&endX=" + Double.toString(end_point.getLongitude()) +
                    "&endY=" + Double.toString(end_point.getLatitude()) + "&startName=start&endName=end" +
                    "&searchOption=" + searchOption);
            } else
                url = new URL( spec: "https://apis.openapi.sk.com/tmap/routes/pedestrian?version=1&format=json" +
                    "&appKey=" + "17xxcf199d1a5d81142c4bcb217ace3818a7b" + "&startX=" + Double.toString(start_point.getLongitude())
                    + "&startY=" + Double.toString(start_point.getLatitude()) + "&endX=" + Double.toString(end_point.getLongitude()) +
                    "&endY=" + Double.toString(end_point.getLatitude()) + "&startName=start&endName=end" +
                    "&passList=" + passList + "&searchOption=" + searchOption);

            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("POST");

            conn.setDoOutput(true);

            if (conn.getResponseCode() == conn.HTTP_OK) {
                InputStreamReader tmp = new InputStreamReader(conn.getInputStream(), charsetName: "UTF-8");
                BufferedReader reader = new BufferedReader(tmp);
                StringBuffer buffer = new StringBuffer();
                while ((str = reader.readLine()) != null) {
                    buffer.append(str);
                }
                receiveMsg = buffer.toString();
                reader.close();
            } else {
                Log.i( tag: "통신 결과", msg: conn.getResponseCode() + "에러");
            }
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    };
    mThread.start(); // Thread 실행
}

```

```

try {
    mThread.join();

    JSONArray features = new JSONObject(receiveMsg).getJSONArray( name: "features");
    for (int i = 0; i < features.length(); i++) {
        JSONObject jsonObject = features.getJSONObject(i);
        jsonObject = jsonObject.getJSONObject("geometry");
        if (jsonObject.getString( name: "type").equals("LineString")) {
            coordinateArray = (JSONArray) jsonObject.get("coordinates");
            test += jsonObject.getString( name: "coordinates") + ",";
        }
    }

    test = test.replace( target: "[", replacement: "").replace( target: "]", replacement: "");
    StringTokenizer = new StringTokenizer(test, delim: ",");

    while (stringTokenizer.hasMoreTokens()) {
        if (i % 2 == 0) {
            x2 = stringTokenizer.nextToken();
            x1 = Double.parseDouble(x2);
        } else {
            y2 = stringTokenizer.nextToken();
            y1 = Double.parseDouble(y2);
            TMapPoint point = new TMapPoint(y1, x1);
            // 도로명으로 변경 array 저장
            list.add(point);
            coordinateID.add(getLinkID(y2, x2));
            Thread.sleep( millis: 50);
        }
        i++;
    }
} catch (InterruptedException | JSONException e) {
    e.printStackTrace();
}

```

```

public void getAvoidPath(final List<TMapPoint> list, final ArrayList<TMapPoint> point) {
    runOnUiThread(() -> {
        TMapPolyLine tMapPolyLine = new TMapPolyLine();
        tMapPolyLine.setLineColor(Color.RED);
        tMapPolyLine.setLineWidth(5);

        for(int i=0;i<list.size();i++){
            tMapPolyLine.addLinePoint(list.get(i));
        }

        tMapView.addTMapPath(tMapPolyLine);

        addPostMarker();

        TMapInfo info = tMapView.getDisplayTMapInfo(point);
        tMapView.setCenterPoint(info.getTMapPoint().getLongitude(),info.getTMapPoint().getLatitude());
        tMapView.setZoomLevel(info.getTMapZoomLevel()-1);
    });
}

```

III. 결론

1. 연구 결과

2018년도 교통약자 실태조사에 따르면 현재 교통약자 수는 총인구의 1/4 이상이며, 고령화 등으로 인해 연평균 2.4%씩 꾸준히 증가하고 있다. 또한, 교통약자들은 현재 도로 곳곳의 장애물, 도로 구조 문제 등으로 시설을 이용하는 데 불편함을 겪고 있고 이에 따라 일상 생활에서 이동과 접근을 위한 교통약자의 요구는 점차 증가하게 되었다. 기존에 존재하는 교통약자를 위한 애플리케이션의 경우 대부분 특정 교통약자만을 취급하거나, 교통약자가 접근 가능한 장소에 대한 정보만을 제공하였다. 따라서, 교통약자의 이동 편의성을 증진하기 위해 추가로 장소 외에 길도 제보할 수 있도록 하며, 제보된 길의 위치 정보를 이용하여 해당 시설을 우회하여 이동할 수 있도록 회피 경로 길찾기 기능을 제공하였다. 결론적으로, 이 보고서에서 다루는 교통약자를 위한 이동 지원 시스템은 사용자가 불편을 겪는 길이나 장소를 지도에서 선택하거나 장소 검색 기능을 이용하여 선택하고, 제보 내용을 입력받아 위치 정보와 함께 서버의 DB에 저장하는 길과 장소 제보 기능과 제보된 시설들을 마커를 통해 지도에 표시하여 현재 위치 주변에 있는 불편시설이나 원하는 장소 근처에 존재하는 불편시설을 확인할 수 있도록 한다. 또한, 출발지와 도착지를 사용자에게 입력받아 최단 경로나 회피 경로 길찾기 기능을 이용하여 지도에서 경로와 경로 주변 불편시설 정보를 확인할 수 있게 한다.

❖ 참고자료

1. 교통약자의 이동편의 증진법 제2조(정의)
2. 국토부(2018), "2018년 교통약자 이동 편의 실태조사"
3. 알고리즘 참고 논문 "이용후, 김상운. 안전운전을 위해 위험지역을 회피하는 내비게이션 경로탐색. 전자공학회논문지, 50(8), 171-179, 2013"
4. T map API "<http://tmapapi.sktelecom.com/main.html#webv2/guide/apiGuide.guide1>"