2023년 충북대학교 소프트웨어학과 졸업작품연구과제

[실시간 영상 분석을 통한 인원 밀집도 공유 시스템]

요구사항 명세서

2023년 10월 30일

문서번호 : 2023-요구사항_명세서_001

소　　속 : 충북대학교 소프트웨어학과

팀　　명 : 시나브로

팀　　원 : 김윤희, 성열암, 임수연

교　　수 : Aziz Nasridinov

(평 가 지 표)

평가항목	점수
개발목표 및 내용 (3점)	
업무분석표 (3점)	
사용자 요구사항 (3점)	
비기능 요구사항 (3점)	
Use-Case Diagram (3점)	
Class Diagram (6점)	
CRC cards 및 Class 명세 (3점)	
Interface 및 Abstract 클래스 명세 (3점)	
User Interface 명세 (3점)	
Sequence Diagram (3점)	
클래스의 메소드 알고리즘 명세 (3점)	
총 점	

제/개정 이력

버전	날짜	작성자 성명	제/개정사항	비 고
ver1.0	03/17	임수연, 김윤희, 성열암	초안 작성	
ver1.1	03/24	임수연, 김윤희, 성열암	Use case Diagram 수정	
ver1.2	04/01	임수연, 김윤희, 성열암	Sequence Diagram 작성 및 클래스의 메소드 알고리즘 명세 작성	
ver1.3	04/07	임수연, 김윤희, 성열암	최종안 작성	
ver1.4	10/15	임수연, 김윤희, 성열암	오탈자 수정	
ver1.4	10/30	임수연, 김윤희, 성열암	Use-case Diagram 수정	

목 차

1. 서 론

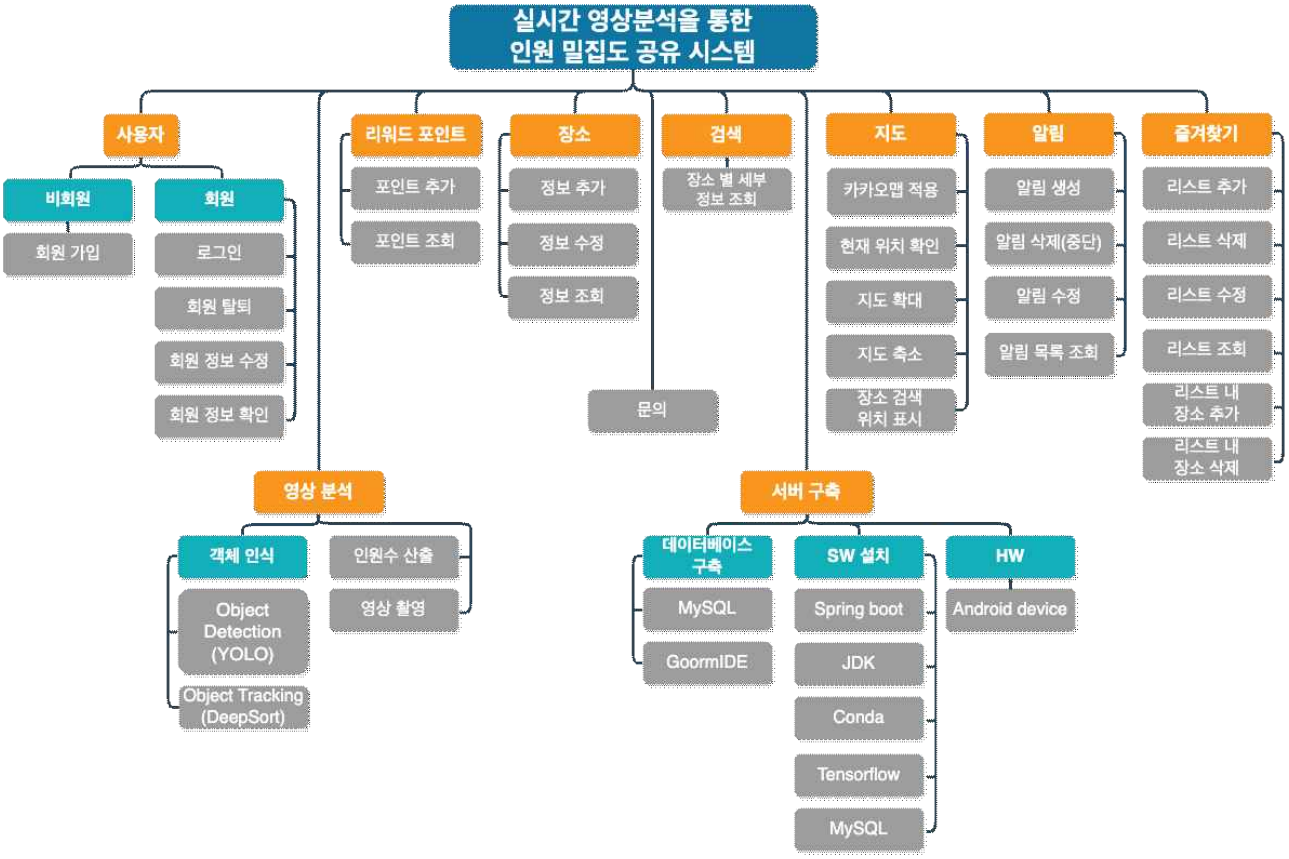
1.1 개발 목표

실시간 영상 분석을 통한 인원 밀집도 공유 시스템 개발을 목표로 함.

1.2 개발 범위

- 로그인 기능 개발
- 회원가입 기능 개발
- 지도 UI 개발
- 알림 기능 개발
- 즐겨찾기 기능 개발
- 장소 정보관리 기능 개발
- 리워드 시스템 개발
- 인원 밀집도 분석 모델 개발

1.3 업무 분석표(WBS)



[그림 1.1 – 업무분석표]

1.4 용어 정의

용어	설명
객체 탐지 (Object detection)	객체 감지(object detection)는 컴퓨터 비전 및 이미지 처리와 관련된 컴퓨터 기술로, 디지털 이미지 및 비디오에서 특정 계층(사람, 건물, 자동차 등)의 의미론적 객체(semantic object)의 인스턴스를 감지하는 것을 의미한다.
스캔	애플리케이션이 현재 실행되고 있는 모바일 디바이스 내에 탑재된 카메라 기능을 이용하여 현재 위치의 주변 환경을 촬영하는 것을 의미한다.
시스템	실시간 영상 분석을 통한 인원 밀집도 공유 시스템을 의미한다.
인원수 공유	애플리케이션 내에서 스캔 기능을 통하여 얻은 시각 데이터를 객체 탐지 영상 처리 모델을 이용하여 파악된 인원수 데이터는 애플리케이션 내의 장소 정보 다이얼로그 요소 내에서 사용자에게 정보로써 공유됨을 의미한다.
즐겨찾기	특정 위치에 대한 정보를 사용자가 지도 내에서 직접 찾지 않고 리스트로써 사용자가 등록한 장소들에 한해서 정보에 빠르게 접근하여 해당 장소에 등록된 정보를 조회할 수 있도록 운영하는 기능을 의미한다.
사용자	애플리케이션 내에서 회원/비회원/관리자 이용자를 아울러 총칭한다.
관리자	시스템에 가입한 사용자들의 정보를 조회 및 제어하는 사용자를 의미한다.
회원	시스템의 회원등록 인증 절차를 모두 거쳐 가입한 사용자를 의미한다.
비회원	시스템의 회원등록 인증 절차를 거치지 않고 시스템을 이용하는 사용자를 의미한다.

[표 1.2 – 용어 정의]

2. 사용자별 요구사항 정의

No.	ID	Name	Description
1	-	회원	시스템에 가입한 회원 사용자이다.
1.1	UR-001	로그인	회원가입 시 사용하였던 이메일과 비밀번호를 이용하여 시스템에 로그인한다.
1.2	UR-002	로그아웃	애플리케이션에 로그인된 상태에서 로그아웃하여 비회원 상태로 전환한다.
1.3	UR-003	정보 수정	로그인 이후에 설정 메뉴 내에서 본인의 회원 정보(프로필 이미지, 이름, 이메일, 비밀번호)를 수정한다.
1.4	UR-004	탈퇴	(미사용 애플리케이션, 보안 문제, 기타 등)의 이유로 시스템으로부터 개인의 모든 정보들을 제거하여 비회원으로 전환한다.
1.5	UR-005	비밀번호 찾기	회원가입 당시 사용하였던 비밀번호를 분실하여 로그인이 불가한 경우 사용자는 가입 시 사용하였던 이메일을 통하여 새로운 비밀번호를 발급받는다.
1.6	UR-006	알림 설정	설정 메뉴 내 알림 설정 창에서 (공지사항, 추천/혜택, 야간 광고성, 즐겨찾기 장소 인원수 변동)에 대한 알림(notification)을 선택적으로 활성화한 사용자의 경우 해당하는 알림을 수신한다.
1.7	UR-007	즐겨찾기	장소 정보가 등록되어 있는 특정 위치에 대하여 리스트 단위로 해당 정보를 추가/제거/조회하고, 리스트는 추가/제거/수정한다.
1.8	UR-008	장소 정보 제어	특정 위치에 대한 장소 정보를 추가/삭제/수정한다.
1.9	UR-009	장소 정보 확인	특정 위치에 대해 등록/미등록된 장소 정보를 확인한다.
1.10	UR-010	장소 검색	지도 상단의 검색란을 이용하여 관련된 장소를 조회하고 클릭 시 해당 위치로 지도의 시점을 이동한다.

1.11	UR-011	스캔	현재 위치에서 반경 50m 이하인 장소를 애플리케이션 내 스캔 버튼을 이용하여 카메라로 촬영한 후 객체 탐지 영상 처리 모델을 거쳐 해당 장소 내에 현재 위치한 인원 수 데이터를 구한다.
2	-	비회원	시스템에 가입하지 않은 사용자이다.
2.1	UR-012	장소 정보 확인	특정 위치에 대해 등록/미등록된 장소 정보를 확인한다.
2.2	UR-013	장소 검색	지도 상단의 검색란을 이용하여 관련된 장소를 조회하고 클릭 시 해당 위치로 지도의 시점을 이동한다.
3	-	관리자	시스템에 가입한 회원들의 정보를 관리하는 사용자이다.
3.1	UR-014	회원 정보 제어	시스템에 가입된 회원들의 정보를 조회 및 제거한다.

[표 2.1 – 사용자별 요구사항 정의]

3. 업무별 요구사항 정의

No.	ID	Name	Description
1	-	회원관리	시스템에 가입하고자 하는 비회원 사용자 및 시스템에 이미 가입한 회원 사용자의 정보를 조회, 수정해야 한다.
1.1	WR-001	로그인 액티비티 설계	회원 사용자가 시스템에 로그인할 수 있도록 하는 UI를 설계한다.
1.2	WR-002	비밀번호 초기화 액티비티 설계	회원 사용자가 새로운 비밀번호를 발급받을 수 있도록 하는 UI를 설계한다.
1.3	WR-003	서비스 이용약관 동의 액티비티 설계	사용자의 서비스 이용약관에 대한 UI를 설계한다.
1.4	WR-004	개인정보 처리 방침 액티비티 설계	사용자의 개인정보 처리 방침에 대한 UI를 설계한다.
1.5	WR-005	개인정보 수집 및 이용 동의 액티비티 설계	사용자의 개인정보 수집 및 이용 동의에 대한 UI를 설계한다.
1.6	WR-006	개인정보 제 3자 제공 동의 액티비티 설계	사용자의 개인정보 제 3자 제공 동의에 대한 UI를 설계한다.
1.7	WR-007	회원가입 단계 액티비티 설계	사용자가 시스템에 회원가입을 할 수 있도록 하는 단계 UI를 설계한다.
1.8	WR-008	설정 fragment 설계	사용자가 설정 기능을 이용할 수 있도록 하는 UI를 설계한다.
1.9	WR-009	약관 및 개인정보 처리 동의 여부 액티비티 설계	사용자의 약관 및 개인정보 처리 동의 여부에 대한 UI를 설계한다.
1.10	WR-010	회원정보 확인 액티비티 설계	시스템을 이용하는 사용자가 가입한 회원 본인임을 검증하기 위한 UI를 설계한다.
1.11	WR-011	회원정보 액티비티 설계	회원 사용자가 본인의 회원정보를 조회할 수 있도록 하는 UI를 설계한다.
1.12	WR-012	회원정보 수정 액티비티 설계	회원 사용자가 본인의 정보를 수정할 수 있도록 하는 UI를 설계한다.

1.13	WR-013	비밀번호 변경 액티비티 설계	회원 사용자가 본인의 비밀번호를 수정할 수 있도록 하는 UI를 설계한다.
1.14	WR-014	회원 탈퇴 단계 액티비티 설계	회원 사용자가 시스템에서 탈퇴할 수 있도록 하는 UI를 설계한다.
2	-	스캔	애플리케이션 내 카메라 기능을 이용하여 특정 위치의 인원수 정보를 추가, 갱신해야 한다.
3	-	알림	사용자에게 공지사항, 추천/혜택, 야간 광고성, 즐겨찾기 장소 인원수 변동에 대한 알림을 발송해야 한다.
3.1	WR-015	알림 설정 액티비티 설계	사용자가 공지사항, 추천/혜택, 야간 광고성, 즐겨찾기 장소 인원수 변동 알림을 설정할 수 있도록 하는 UI를 설계한다.
4	-	지도	사용자는 카카오맵을 이용하여 지도 기능을 사용해야 한다.
4.1	WR-016	카카오맵 API 적용	사용자가 지도 기능을 사용할 수 있도록 Kakao Maps API를 적용한다.
4.2	WR-017	카카오맵 마커 표시	사용자가 메인 액티비티 내에 있는 카카오맵 클릭 시 해당 위치에 장소명 캡션과 마커 아이콘이 함께 나타나도록 한다.
5	-	즐겨찾기	회원 사용자는 특정 위치에 대한 장소 정보를 별도로 저장하여 관리할 수 있도록 한다.
5.1	WR-018	즐겨찾기 fragment 설계	회원 사용자가 즐겨찾기 기능을 이용할 수 있도록 하는 UI를 설계한다.
5.2	WR-019	리스트 내부 정보 확인 액티비티 설계	회원 사용자가 특정 리스트 내 등록하였던 즐겨찾기 정보를 확인하는 UI를 설계한다.
5.3	WR-020	리스트 삭제 액티비티 설계	회원 사용자가 시스템에 추가하였던 리스트 목록을 삭제할 수 있도록 하는 UI를 설계한다.
5.4	WR-021	즐겨찾기 리스트 내 장소 추가 액티비티 설계	회원 사용자가 특정 장소에 대하여 즐겨찾기 리스트에 추가할 수 있도록 하는 UI를 설계한다.
6	–	장소 관리	사용자는 특정 위치에 대한 정보의 추가, 조회, 수정을 할 수 있도록 한다.

6.1	WR-022	장소 정보 list 액티비티 설계	회원 사용자가 특정 리스트를 클릭했을 때 그 안에 저장되어 있는 즐겨찾기 정보를 확인할 수 있도록 하는 UI를 설계한다.
6.2	WR-023	신규 장소 정보 등록 안내 액티비티 설계	신규 장소 정보 등록에 대한 안내가 담긴 UI를 설계한다.
6.3	WR-024	장소 정보 등록하기 액티비티 설계	회원 사용자가 장소 정보를 등록할 수 있도록 하는 UI를 설계한다.
6.4	WR-025	장소 목록 액티비티 설계	사용자가 특정 위치에 등록된 장소 목록을 확인할 수 있도록 하는 UI를 설계한다.
7	-	포인트 리워드	회원은 시스템의 특정 요청 사항에 대하여 수행을 완료한 후 포인트를 지급받는다.
7.1	WR-026	포인트 지급	시스템의 회원 사용자는 특정 위치에 대한 인원수 정보 등록 시 시스템으로부터 일정 수준의 포인트를 지급받도록 한다.
8	-	튜토리얼	시스템을 처음 이용하는 사용자로 하여금 시스템 애플리케이션의 사용 방법을 익힐 수 있도록 별도의 시스템 튜토리얼 페이지를 구성한다.
8.1	WR-027	인원수 스캔 튜토리얼 액티비티 설계	인원수 스캔 기능 사용에 대한 내용이 담긴 UI를 설계한다.
8.2	WR-028	장소 등록 튜토리얼 액티비티 설계	장소 등록 기능 사용에 대한 내용이 담긴 UI를 설계한다.
8.3	WR-029	장소 정보 확인 튜토리얼 액티비티 설계	장소 정보 확인에 대한 내용이 담긴 UI를 설계한다.
9	-	검색	사용자가 특정 위치에 대한 장소 정보를 확인하고자 할 때에 검색 기능을 이용하여 해당 장소에 빠르게 접근할 수 있도록 하여야 한다.
10	-	서버 개발	시스템이 운영되도록 통신 서버를 구성한다.
10.1	WR-030	Database 구축	애플리케이션과 서버 간의 데이터 통신을 위해 데이터베이스를 구축한 후 스키마를 추가한다.
10.2	WR-031	배포	작업된 서버를 시스템을 이용하는 사용자가 개별적으로 빌드하지 않고 범용적으로 이용할 수 있도록 Koyeb 플랫폼을 이용하여 배포한다.

15	-	Service 정의	시스템과 서버의 데이터 통신을 위하여 다양한 Service클래스를 정의하여 API를 개발한다.
15.1	WR-032	자기 자신의 회원 정보 조회 Service 정의	회원 사용자 정보를 조회할 수 있도록 하는 Service 클래스를 정의한다.
15.2	WR-033	회원 전체의 정보 조회 Service 정의	회원 사용자들 전체의 정보를 조회할 수 있도록 하는 Service 클래스를 정의한다.
15.3	WR-034	회원 추방 Service 정의	회원 사용자의 모든 정보를 제거할 수 있도록 하는 Service 클래스를 정의한다.
15.4	WR-035	프로필 이미지 변경 Service 정의	회원 사용자의 프로필 이미지를 변경할 수 있도록 하는 Service 클래스를 정의한다.
15.5	WR-036	이름/이메일 변경 Service 정의	회원 사용자의 이름/이메일을 변경할 수 있도록 하는 Service 클래스를 정의한다.
15.6	WR-037	비밀번호 변경 Service 정의	회원 사용자의 비밀번호를 변경할 수 있도록 하는 Service 클래스를 정의한다.
15.7	WR-038	포인트 조회 Service 정의	회원 사용자의 포인트를 조회할 수 있도록 하는 Service 클래스를 정의한다.
15.8	WR-039	특정 유저의 포인트 증가 Service 정의	회원 사용자의 포인트를 증가시킬 수 있도록 하는 Service 클래스를 정의한다.
15.9	WR-040	특정 유저의 포인트 감소 Service 정의	회원 사용자의 포인트를 감소시킬 수 있도록 하는 Service 클래스를 정의한다.
15.10	WR-041	로그인 Service 정의	회원 사용자가 본인의 정보(이메일, 비밀번호)를 통하여 시스템에 로그인할 수 있도록 하는 Service 클래스를 정의한다.
15.11	WR-042	로그아웃 Service정의	회원 사용자가 로그아웃할 수 있도록 하는 Service 클래스를 정의한다.
15.12	WR-043	회원가입 Service 정의	비회원 사용자가 시스템에 회원가입이 가능하도록 하는 Service 클래스를 정의한다.
15.13	WR-044	임시 비밀번호 생성 Service 정의	회원 사용자가 신규 비밀번호를 메일 주소로 발급받을 수 있도록 하는 Service 클래스를 정의한다.
15.14	WR-045	이메일 검증 Service 정의	이메일을 변경 및 회원가입 시 새로 입력한 이메일이 이미 시스템 데이터베이스 상에 존재하는지 검증하는 Service 클래스를 정의한다.

15.15	WR-046	리스트 생성 Service 정의	회원 사용자가 리스트 데이터를 생성하도록 하는 Service 클래스를 정의한다.
15.16	WR-047	리스트 삭제 Service 정의	서버는 회원 사용자가 추가하였던 리스트 데이터를 제거하도록 하는 Service 클래스를 정의한다.
15.17	WR-048	리스트 정보 수정 Service 정의	회원 사용자가 추가하였던 리스트 데이터를 수정하도록 하는 Service 클래스를 정의한다.
15.18	WR-049	리스트 내 장소 정보 추가 Service 정의	회원 사용자가 본인이 생성하였던 리스트에 특정 장소 정보를 추가할 수 있도록 하는 Service 클래스를 정의한다.
15.19	WR-050	리스트 내 장소 정보 삭제 Service 정의	회원 사용자가 본인이 생성하였던 리스트로부터 특정 장소 정보를 제거할 수 있도록 하는 Service 클래스를 정의한다.
15.20	WR-051	장소 정보 등록 Service 정의	회원 사용자가 특정 위치에 대한 장소 정보를 추가하도록 하는 Service 클래스를 정의한다.
15.21	WR-052	장소 정보 삭제 Service 정의	회원 사용자가 특정 위치에 대한 장소 정보를 제거하도록 하는 Service 클래스를 정의한다.
15.22	WR-053	장소 정보 수정 Service 정의	회원 사용자가 특정 위치에 대한 장소 정보를 수정하도록 하는 Service 클래스를 정의한다.
15.23	WR-054	장소 정보 조회 Service 정의	사용자가 특정 위치에 대한 장소 정보를 확인할 수 있도록 하는 Service 클래스를 정의한다.
15.24	WR-055	인원수 조회 Service 정의	사용자가 특정 위치에 등록된 장소에 대한 인원수 정보를 조회할 수 있도록 하는 Service 클래스를 정의한다.
15.25	WR-056	특정 장소 내 인원수 추가/갱신 Service 정의	회원 사용자가 특정 장소에 대한 인원수 데이터를 추가/갱신할 수 있도록 하는 Service 클래스를 정의한다.
15.26	WR-057	인원수 반환 Service 정의	시각 데이터에서 객체 탐지 모델을 통해 인원수 데이터를 응답으로 반환하는 Service 클래스를 정의한다.

[표 3.1 – 업무별 요구사항 정의]

4. 비기능적 요구사항

4.1 성능 요구사항

No.	ID	Name	Description
1	PR-001	성능 요구사항	시스템은 사용자의 스캔 완료 표시 후 최대 2초 이내에 인원수 계산을 마무리해야 한다.
2	PR-002	성능 요구사항	시스템의 응답시간은 최대 3초를 초과하지 않아야 한다.
3	PR-003	성능 요구사항	데이터베이스는 최대 2초 이내에 데이터의 모든 CRUD 동작을 완료할 수 있어야 한다.
4	PR-004	성능 요구사항	객체 탐지 모델은 사람 객체를 실시간으로 탐지해야 한다.
5	PR-005	성능 요구사항	객체 탐지 모델은 사람 객체를 실시간으로 구별(인식)해야 한다.
6	PR-006	성능 요구사항	객체 탐지 모델은 사람 객체에 실시간으로 고유 번호를 부여해야 한다.
7	PR-007	성능 요구사항	본 애플리케이션은 최소 4GB 이상의 메모리 성능을 가진 모바일 디바이스 내에서 반드시 실행되어야 한다.
8	PR-008	성능 요구사항	시스템은 애플리케이션 내 카메라 기능을 통하여 스캔 시 사람 객체를 85% 이상의 정확도로 탐지해야 한다.
9	PR-009	성능 요구사항	시스템은 애플리케이션 내 카메라 기능을 통하여 스캔 시 사람 객체를 85% 이상의 성공률로 구별(인식)해야 한다.

[표 4.1 – 성능 요구사항]

4.2 보안 요구사항

No.	ID	Name	Description
1	SR-001	보안 요구사항	관리자를 제외한 일반 회원 사용자는 본인 이외의 다른 사용자의 개인정보를 조회할 수 없어야 한다.
2	SR-002	보안 요구사항	회원가입 시 입력된 비밀번호는 SHA-256 알고리즘 및 Salt 문자열을 적용하여 암호화된 형태로 데이터베이스에 저장되어야 한다.
4	SR-004	보안 요구사항	시스템은 악의적인 사용자에 의하여 데이터베이스 정보의 추가/삭제/수정이 이뤄지지 않도록 제한한다.
5	SR-004	보안 요구사항	시스템은 의도치 않은 데이터의 손실 및 변경이 발생하지 않도록 서비스를 제공해야 한다.
6	SR-006	보안 요구사항	관리자 회원은 데이터베이스의 접근/관리 권한을 부여받아 시스템 데이터베이스를 제어할 수 있어야 한다.
7	SR-007	보안 요구사항	애플리케이션을 통한 회원가입 시 이메일 입력란에 추가되는 이메일은 반드시 이메일 인증을 통하여 확인 검증이 완료되어야 한다.
8	SR-008	보안 요구사항	애플리케이션 내에서 로그인 시 입력되는 비밀번호는 회원가입 시 사용되었던 SHA-256 알고리즘 및 Salt 문자열을 적용하여 암호화된 형태로 Request Body에 담겨야 한다.
9	SR-009	보안 요구사항	시스템은 애플리케이션 내에서 관리자 계정으로 로그인한 사용자의 경우 관리자 액티비티에 대한 접근 권한을 부여해야 한다.

[표 4.2 – 보안 요구사항]

4.3 신뢰성 요구사항

No.	ID	Name	Description
1	RR-001	신뢰성 요구사항	시각 데이터 처리 모델은 80% 이상의 정확도를 지녀야 한다.
2	RR-002	신뢰성 요구사항	네트워크가 정상적으로 연결된 경우 서버와의 통신이 100% 이상 없이 실행되어야 한다.

[표 4.3 – 신뢰성 요구사항]

4.4 데이터베이스 요구사항

No.	ID	Name	Description
1	DR-001	데이터베이스 요구사항	시스템에서는 MySQL DBMS를 사용하여 데이터를 관리되어야 한다.
2	DR-002	데이터베이스 요구사항	시스템 데이터베이스는 점검 상태를 제외하고 항시 Running 상태를 유지해야 한다.
3	DR-003	데이터베이스 요구사항	서버에서 데이터베이스 접속을 시도할 때에는 반드시 user 사용자로 접속해야 한다.
4	DR-004	데이터베이스 요구사항	시스템에 회원가입 된 개별 사용자의 정보를 조회할 수 있어야 한다.
5	DR-005	데이터베이스 요구사항	시스템에 회원가입 된 전체 사용자의 정보를 조회할 수 있어야 한다.
6	DR-006	데이터베이스 요구사항	시스템에 회원가입 된 개별 사용자의 정보를 일괄적으로 제거할 수 있어야 한다.
7	DR-007	데이터베이스 요구사항	시스템에 회원가입 된 개별 사용자의 프로필 이미지를 추가/변경할 수 있어야 한다.
8	DR-008	데이터베이스 요구사항	시스템에 회원가입 된 개별 사용자의 이름/이메일을 추가/변경할 수 있어야 한다.
9	DR-009	데이터베이스 요구사항	시스템에 회원가입 된 개별 사용자의 비밀번호를 변경할 수 있어야 한다.

10	DR-010	데이터베이스 요구사항	리스트 관련 데이터를 지정 테이블에 추가/제거/수정할 수 있어야 한다.
11	DR-011	데이터베이스 요구사항	개별 리스트를 기준으로 해당 리스트와 연관된 장소 정보 데이터는 지정 테이블에 추가/제거할 수 있어야 한다.
12	DR-012	데이터베이스 요구사항	특정 위치에 대한 장소 정보 데이터를 지정 테이블에 추가/제거/삭제할 수 있어야 한다.
13	DR-013	데이터베이스 요구사항	특정 위치에 대한 장소 데이터를 조회할 수 있어야 한다.
14	DR-014	데이터베이스 요구사항	특정 위치에 대한 장소 정보 내 인원수 데이터를 추가/수정할 수 있어야 한다.

[표 4.4 – 데이터베이스 요구사항]

4.5 SW 이식성 요구사항

No.	ID	Name	Description
1	STR-001	SW 이식성 요구사항	애플리케이션은 Android 운영체제가 탑재되어 있으며, SDK 버전 21 이상의 모든 모바일 디바이스 환경에서 실행되어야 한다.

[표 4.5 – SW 이식성 요구사항]

4.6 유지보수 요구사항

No.	ID	Name	Description
1	MR-001	유지보수 요구사항	시스템 내에서 발견되는 버그들은 각각 우선순위를 부여하여 관리한다.
2	MR-002	유지보수 요구사항	빌드 중인 서버가 중단되어 시스템 이용에 장애가 발생하지 않도록 정기적으로 배포 환경을 점검한다.
3	MR-002	유지보수 요구사항	시스템에 개선사항이 발생 시 시스템에 반영될 수 있도록 지속적인 관리 및 운영이 이뤄져야 한다.

[표 4.6 – 유지보수 요구사항]

4.7 HW, SW 및 통신 요구사항

No.	ID	Name	Description
1	HSCR-001	HW, SW 및 통신 요구사항	시스템은 카카오맵 API 및 서버와의 데이터 통신을 기반으로 운영된다.
2	HSCR-002	HW, SW 및 통신 요구사항	시스템은 반드시 안정적인 네트워크 상태가 보장된 환경에서 시스템을 이용할 수 있도록 하여야 한다.
3	HSCR-003	HW, SW 및 통신 요구사항	시스템은 정상적인 네트워크 환경에서 사용자의 권한별 수행하는 데이터 통신은 반드시 이상 없이 수행된다.

[표 4.7 – HW, SW 및 통신 요구사항]

4.8 동시성 요구사항

No.	ID	Name	Description
1	CR-001	동시성 요구사항	서버는 초당 최대 100개까지의 데이터 요청에 대하여 정상적으로 작업을 수행하여야 한다.

[표 4.8 – 동시성 요구사항]

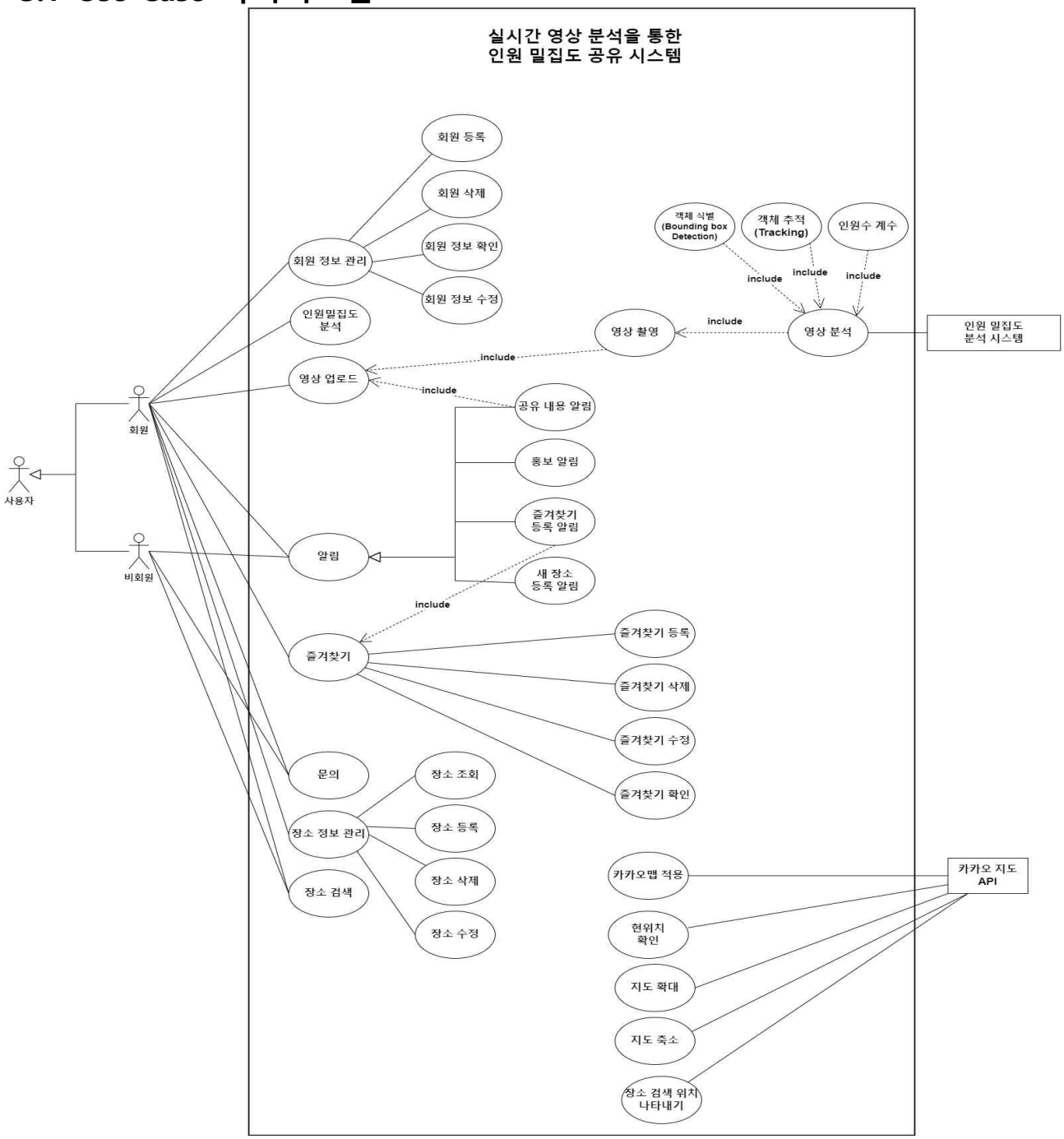
4.9 사용자 인터페이스 요구사항

No.	ID	Name	Description
1	UIR-001	사용자 인터페이스 요구사항	사용자는 모바일 디바이스 및 키보드와 같은 외부 입력 장치를 통하여 애플리케이션 내에 문자를 입력할 수 있어야 한다.
2	UIR-002	사용자 인터페이스 요구사항	애플리케이션 내에서 사용되는 지도는 카카오 오픈 API에서 지원하는 카카오맵을 이용한다.
3	UIR-003	사용자 인터페이스 요구사항	즐겨찾기, 설정, 문의, 개발 정보 관련 UI 화면은 모두 Sidebar Navigation Layout을 통해 접근할 수 있도록 한다.
4	UIR-004	사용자 인터페이스 요구사항	지도의 시점 및 위치를 신속히 제어할 수 있도록 카카오맵 상단에 현재 위치 버튼, 지도 확대/축소 버튼, 검색창을 배치해야 한다.
5	UIR-005	사용자 인터페이스 요구사항	카카오맵 상단의 특정 위치를 클릭하였을 때 해당 장소에 등록/미등록된 정보 데이터를 Bottom Sheet를 통해 나타나도록 해야 한다.
6	UIR-006	사용자 인터페이스 요구사항	애플리케이션을 실행하였을 때 프로젝트 대표 이미지가 화면 중심에 오도록 배치 후 사용자에게 표시되어야 한다.
7	UIR-007	사용자 인터페이스 요구사항	데이터베이스 내에 정보가 변경될 때 Dialog창을 화면 중앙에 생성하여 의도하지 않은 데이터의 조작을 방지한다.
8	UIR-008	사용자 인터페이스 요구사항	애플리케이션 내에서 데이터의 변경이 발생하였을 때 사용자에게 수행했던 작업에 대한 피드백을 실시간으로 전달할 수 있도록 확인 관련 안내 문구와 함께 화면의 하단에 Toast Message를 표시하여야 한다.

[표 4.9 – 사용자 인터페이스 요구사항]

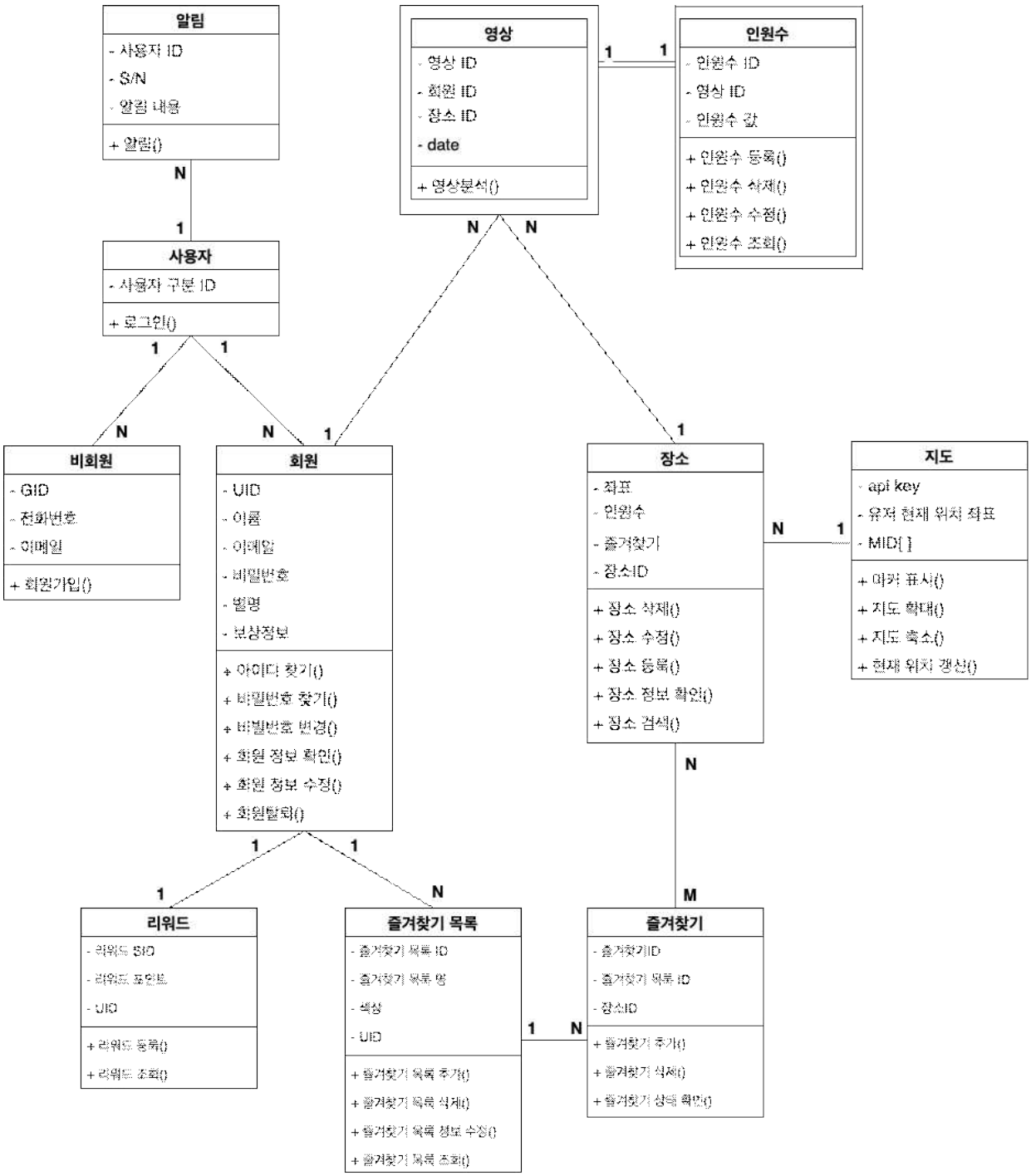
5. 데이터 명세

5.1 Use-Case 다이어그램



[그림 5.1 – Use-Case 다이어그램]

5.2 클래스 다이어그램



[그림 5.2 – 클래스 다이어그램]

5.3 CRC Card 및 클래스 명세

5.3.1 알림

속성	속성명	자료형	크기	제약조건	Collaborating Classes
UID	UID	char	20	FK	회원
S/N	SerialNo	Integer	10	PK	
알림 내용	Noti_contents	String	200		
Operations:	alarm();				

[표 5.1 – "알림" 클래스 명세]

5.3.2 사용자

속성	속성명	자료형	크기	제약조건	Collaborating Classes
사용자 구분 ID	ID	char	20		
Operations:	login();				

[표 5.2 – "사용자" 클래스 명세]

5.3.3 회원

속성	속성명	자료형	크기	제약조건	Collaborating Classes
UID	UID	char	20	PK	리워드
이름	Uname	char	20		
이메일	Uemail	char	20		
비밀번호	PassWd	char	20		
별명	NickName	char	20		
보상정보	reward	Integer	10	FK	

Operations:	find_id();
	find_pw();
	change_pw();
	user_info();
	modify_user_info();
	withdrawal();				

[표 5.3 – "회원" 클래스 명세]

5.3.4 비회원

속성	속성명	자료형	크기	제약조건	Collaborating Classes
비회원 ID	GID	char	20	PK	
전화번호	GphoneNum	char	20		
이메일	Gemail	char	20		
Operations:	join();				

[표 5.4 – "비회원" 클래스 명세]

5.3.5 리워드

속성	속성명	자료형	크기	제약조건	Collaborating Classes
리워드 SID	RSID	char	20	PK	회원
리워드 포인트	reward	Integer	10		
UID	UID	char	20	FK	
Operations:	register_rewards(); reward_lookup();				

[표 5.5 – "리워드" 클래스 명세]

5.3.6 즐겨찾기

속성	속성명	자료형	크기	제약조건	Collaborating Classes
즐겨찾기 ID	FID	char	20	PK	즐겨찾기 목록, 장소
즐겨찾기 목록 ID	Flist_ID	char	10	FK	
장소 ID	PID	char	20	FK	

Operations:	add_bookmark();
	delete_bookmark();
	show_bookmark();				

[표 5.6 – "즐겨찾기" 클래스 명세]

5.3.7 즐겨찾기 목록

속성	속성명	자료형	크기	제약조건	Collaborating Classes
즐겨찾기 목록 ID	Flist_ID	char	10	PK	회원
즐겨찾기 목록 명	Flist_name	char	20		
색상	color	char	32		
UID	UID	char	20	FK	

Operations:	add_bookmark_list();
	delete_bookmark_list();
	modify_bookmark_info_list();
	show_bookmark_list();				

[표 5.7 – "즐겨찾기 목록" 클래스 명세]

5.3.8 장소

속성	속성명	자료형	크기	제약조건	Collaborating Classes
좌표	location	String	100		즐겨찾기, 장소, 영상
인원수	PeopleNum	Integer	20		
즐겨찾기 ID	FID	char	20	FK	
장소 ID	PID	char	20	PK	
장소 명	PlaceName	String	30		

Operations:	delete_place();
	create_place();
	edit_place();
	lookup_place();
	search_place();				

[표 5.8 – "장소" 클래스 명세]

5.3.9 지도

속성	속성명	자료형	크기	제약조건	Collaborating Classes
api key	ApiKey	String	150	PK	장소
현재위치	Nowlocation	String	150		
MID[ ]	Array_MID	char	999	FK	

Operations:	show_marker();
	map_zoom_in();
	map_zoom_out();
	renew_current_location();				

[표 5.9 – "지도" 클래스 명세]

5.3.10 영상

속성	속성명	자료형	크기	제약조건	Collaborating Classes
영상분석 ID	Analysis_ID	char	20	FK	장소, 인원수
장소 ID	PID	char	20	FK	
영상 ID	VID	char	20	PK	
Operations:	image_analysis();				

[표 5.10 – "영상" 클래스 명세]

5.3.11 인원수

속성	속성명	자료형	크기	제약조건	Collaborating Classes
인원수 ID	PnumID	char	20	PK	영상
영상 분석 ID	Analysis_ID	char	20	FK	
인원수 값	Analysis_Num	Integer	30	FK	

Operations:	add_people_num();
	delete_people_num();
	modify_people_num();
	lookup_people_num();				

[표 5.11 – "인원수" 클래스 명세]

5.4 데이터베이스 스키마 테이블 명세

5.4.1 User

데이터 항목	변수명	자료형	길이	제약사항
사용자 번호	id	integer	4	PK
사용자 아이디	uid	char	20	NN
사용자 이름	uname	char	20	NN
비밀번호	pwd	varchar	20	NN
이메일	email	varchar	30	NN
포인트 총점	pointscore	integer	4	default=0
프로필 사진	file_name	blob	20	

[표 5.12 – "User" 스키마 테이블 명세]

5.4.2 Bookmark

데이터 항목	변수명	자료형	길이	제약사항
즐겨찾기 번호	bid	integer	4	PK
장소 번호	pid	integer	20	FK
라벨 아이디	lid	char	20	FK

[표 5.13 – "Bookmark" 스키마 테이블 명세]

5.4.3 Label

데이터 항목	변수명	자료형	길이	제약사항
라벨 번호	lid	integer	4	PK
아이디	uid	char	20	FK
라벨 이름	labelname	varchar	20	NN
라벨 색상	labelcolor	varchar		

[표 5.14 – "Label" 스키마 테이블 명세]

5.4.4 Places

데이터 항목	변수명	자료형	길이	제약사항
장소번호	pid	integer	4	PK
장소 이름	pname	varchar	20	NN
주소	addr	varchar	30	NN
위도	lat	float	20	
경도	longt	float	20	
상세 정보	detail	varchar	30	

[표 5.15 – "Places" 스키마 테이블 명세]

5.4.5 Headcount

데이터 항목	변수명	자료형	길이	제약사항
인원수 번호	hid	integer	4	PK
장소 번호	pid	integer	20	FK
인원수	count	integer	4	NN
시간	timestamp	char	20	NN
사용자 번호	writer	char	20	NN

[표 5.16 – "Headcount" 스키마 테이블 명세]

5.4.6 Reward

데이터 항목	변수명	자료형	길이	제약사항
보상 번호	rid	integer	4	PK
보상 포인트	point	integer	4	NN

[표 5.17 – "Reward" 스키마 테이블 명세]

5.5 데이터구조 명세

(해당 사항 없음)

5.6 Interface 및 Abstract 클래스 명세

```
abstract class Person {
    int num;
    String id;
    String pwd;
    insert();
    delete();
    update();
    retrieve();
};

class Memeber extends Person {}
```

6. 사용자 인터페이스 명세

6.1 관리자 사용자 인터페이스 명세

(해당 사항 없음)

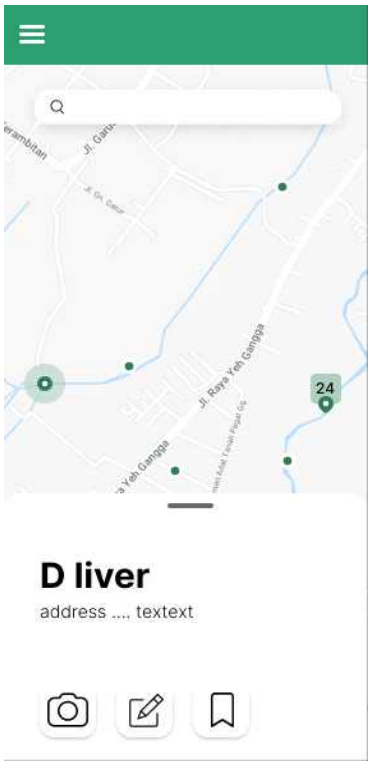
6.2 일반 사용자 인터페이스 명세

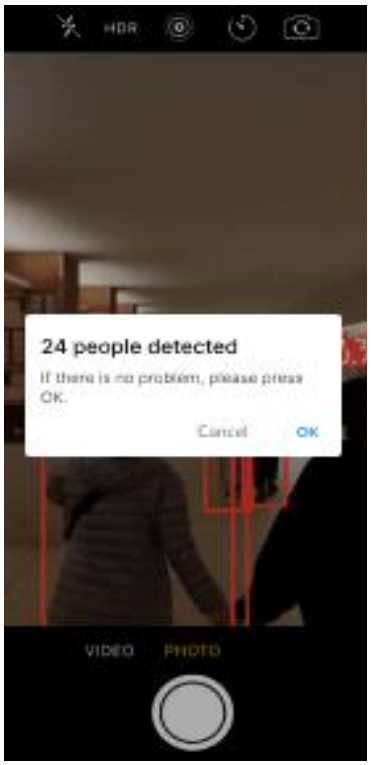
6.2.1 튜토리얼 화면

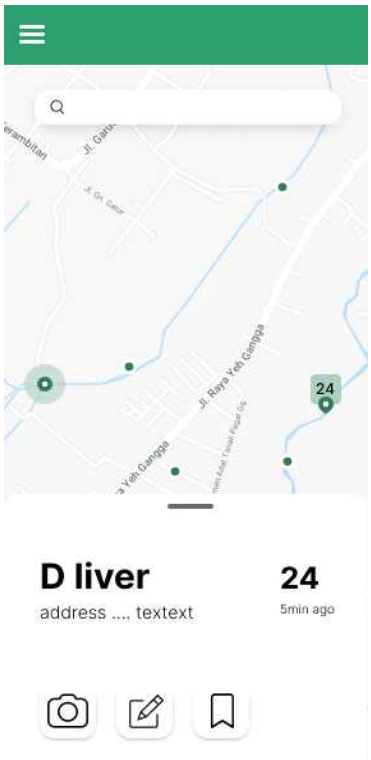
(개발 예정)

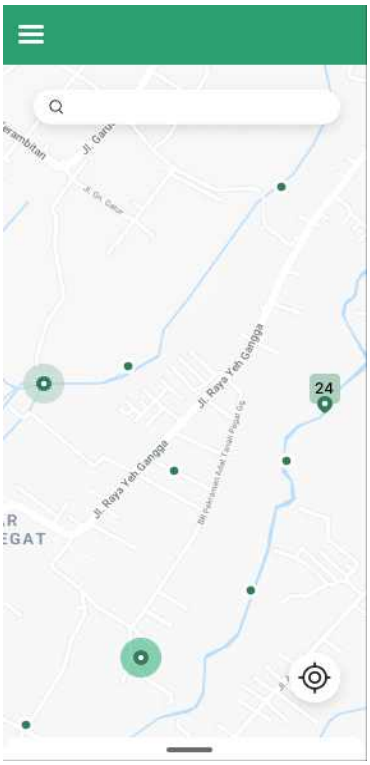
6.2.2 지도 화면

6.2.3 영상 공유 분석 화면

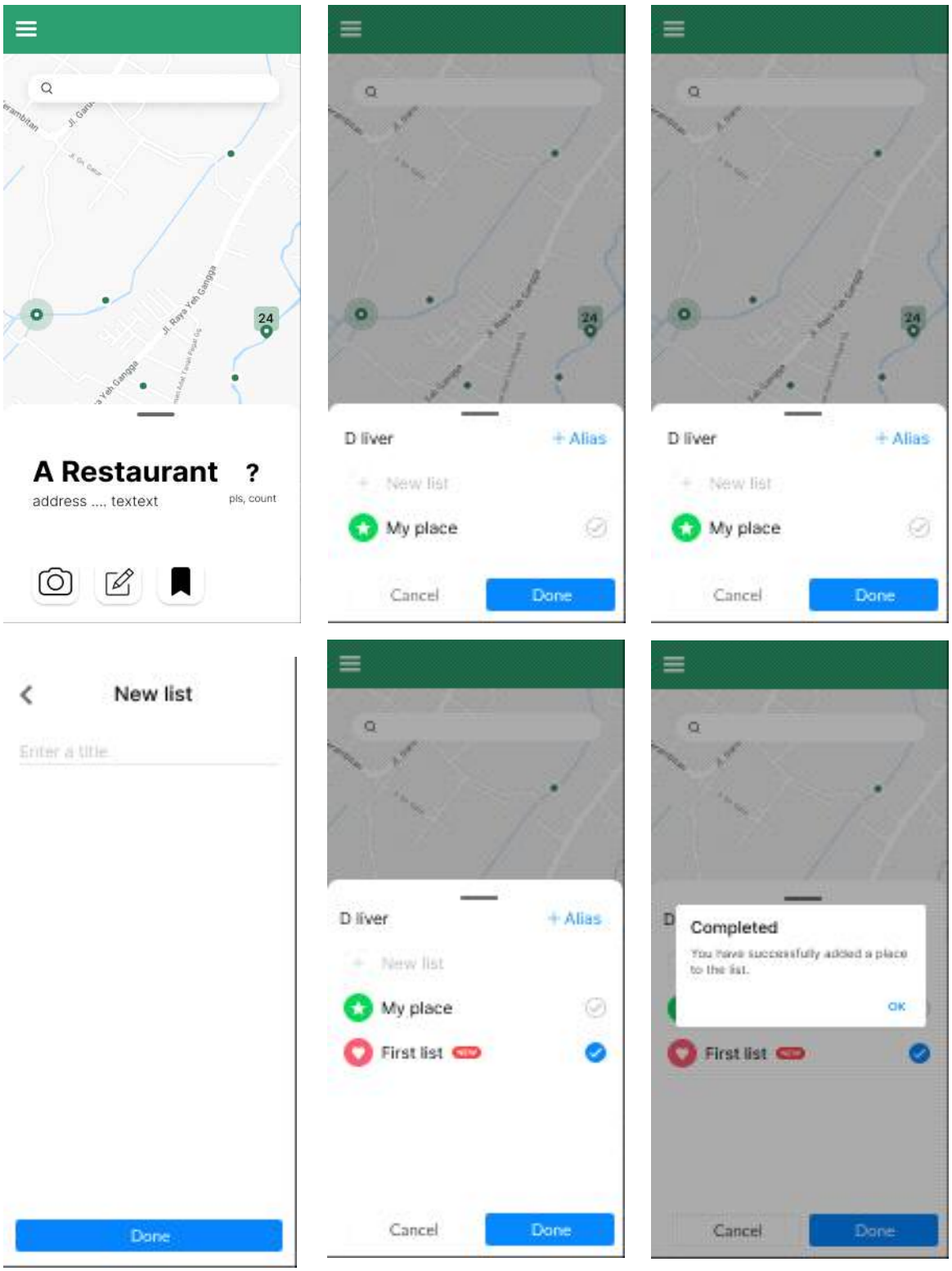


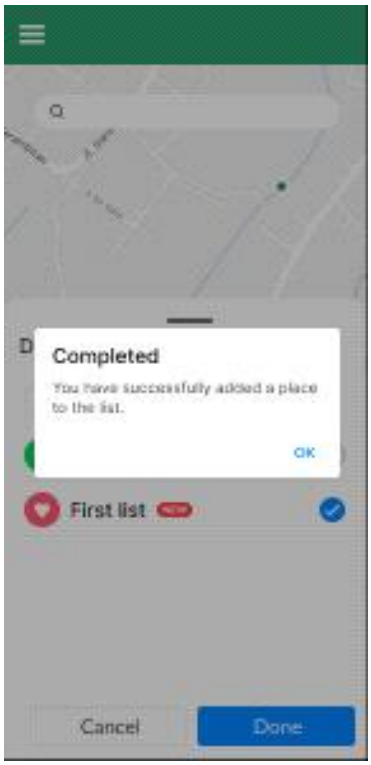


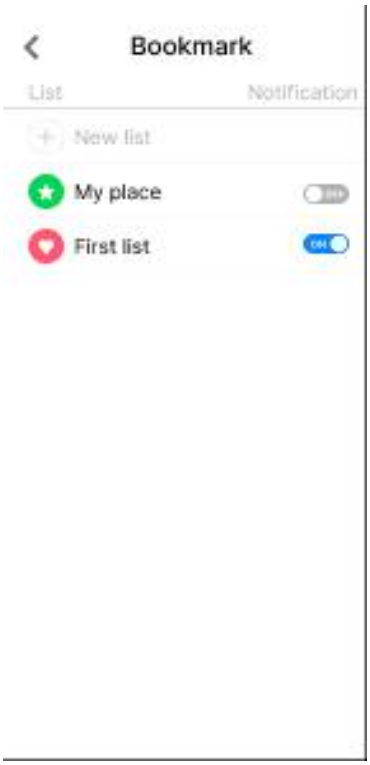




6.2.4 즐겨찾기 화면

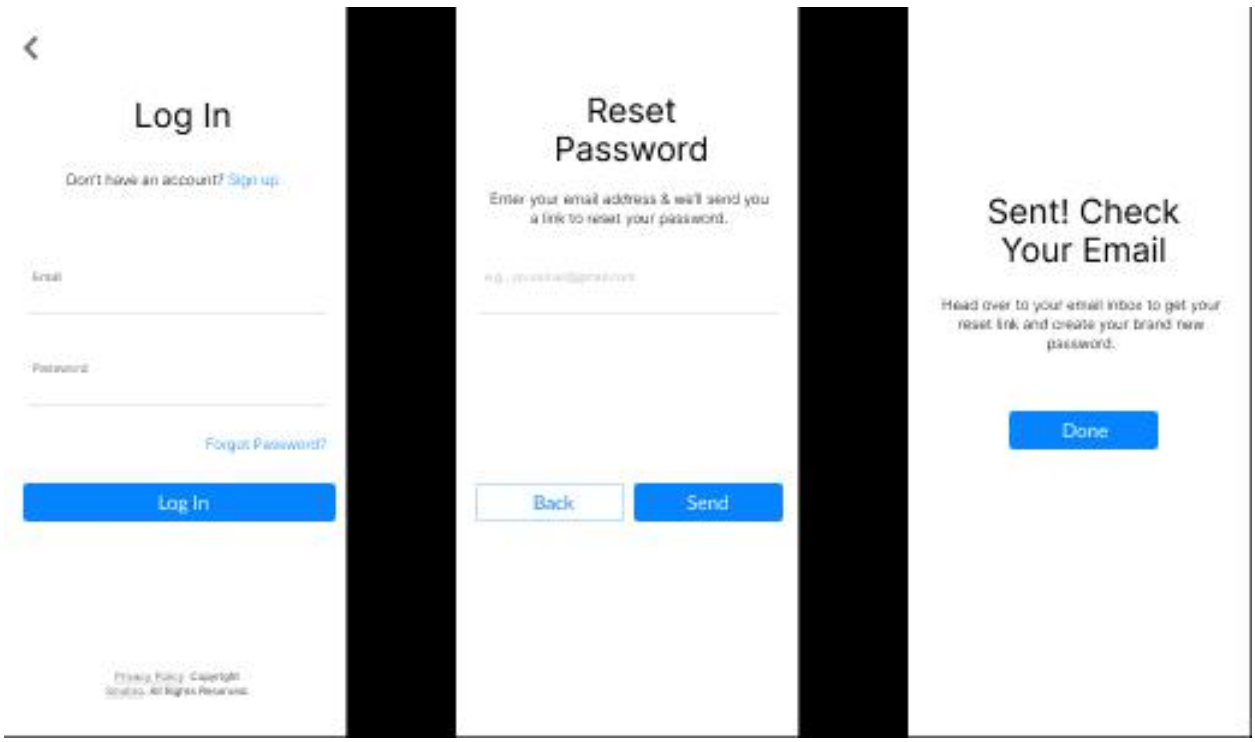

A Restaurant
address .... textext
pls, count
D liver
+ Alias
New list
My place
Cancel
Done
New list
Enter a title
Done
First list
Completed
You have successfully added a place to the list.
OK


Completed
You have successfully added a place to the list.
OK
First list
Cancel
Done

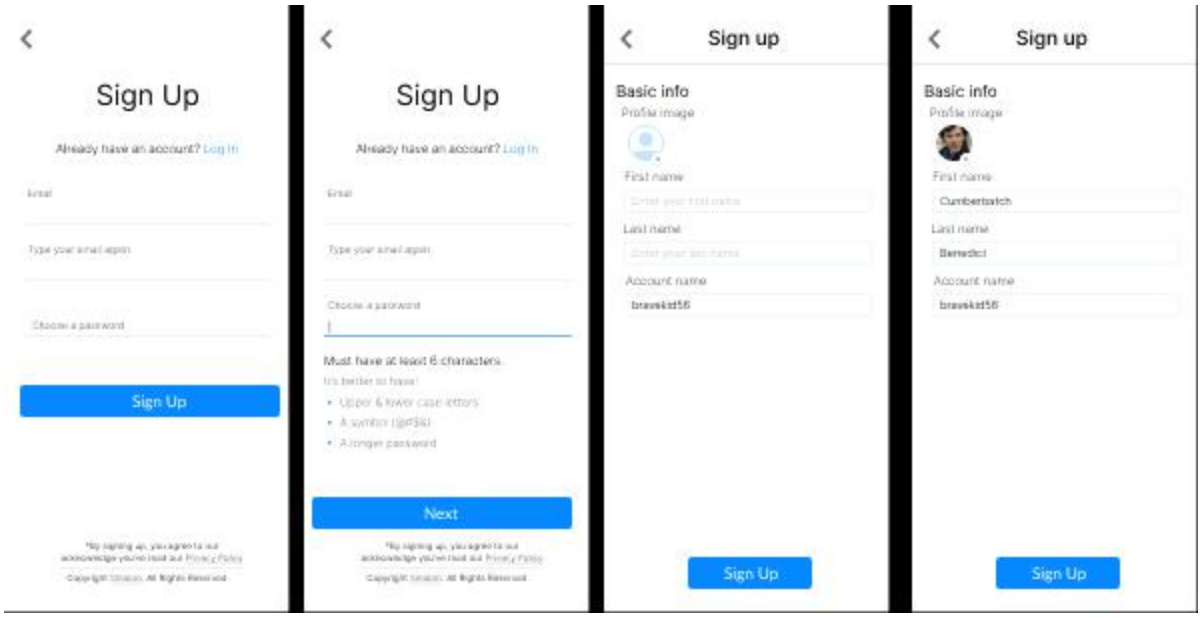

Bookmark
List
Notification
New list
My place
First list


D liver
address .... textext
24

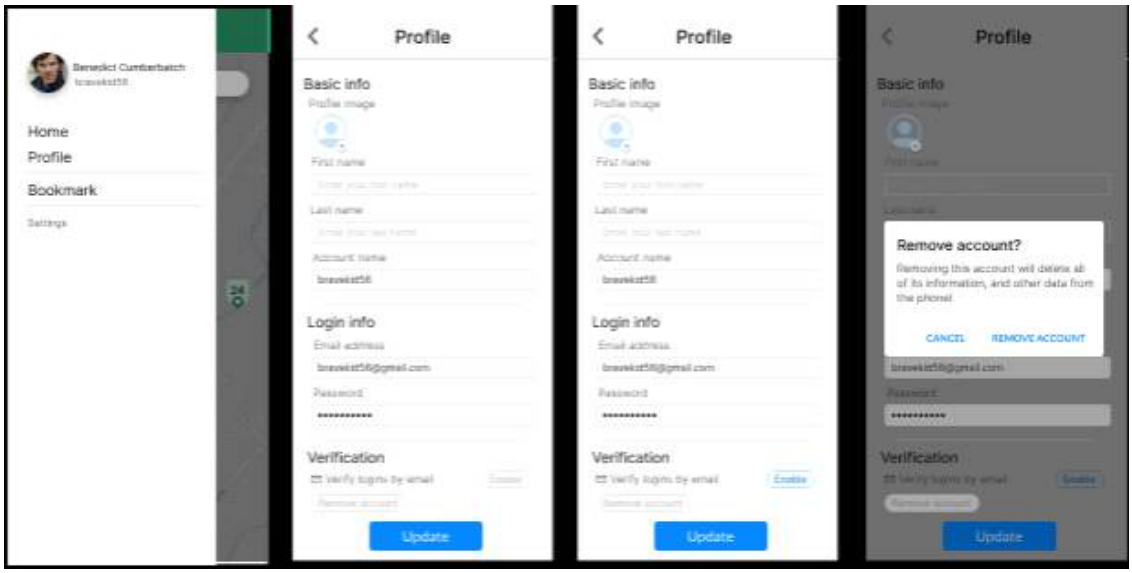
6.2.5 로그인 화면


Log In
Don't have an account? Sign up
Email
Password
Forgot Password?
Log In
Reset Password
Enter your email address & we'll send you a link to reset your password.
Back
Send
Sent! Check Your Email
Head over to your email inbox to get your reset link and create your brand new password.
Done

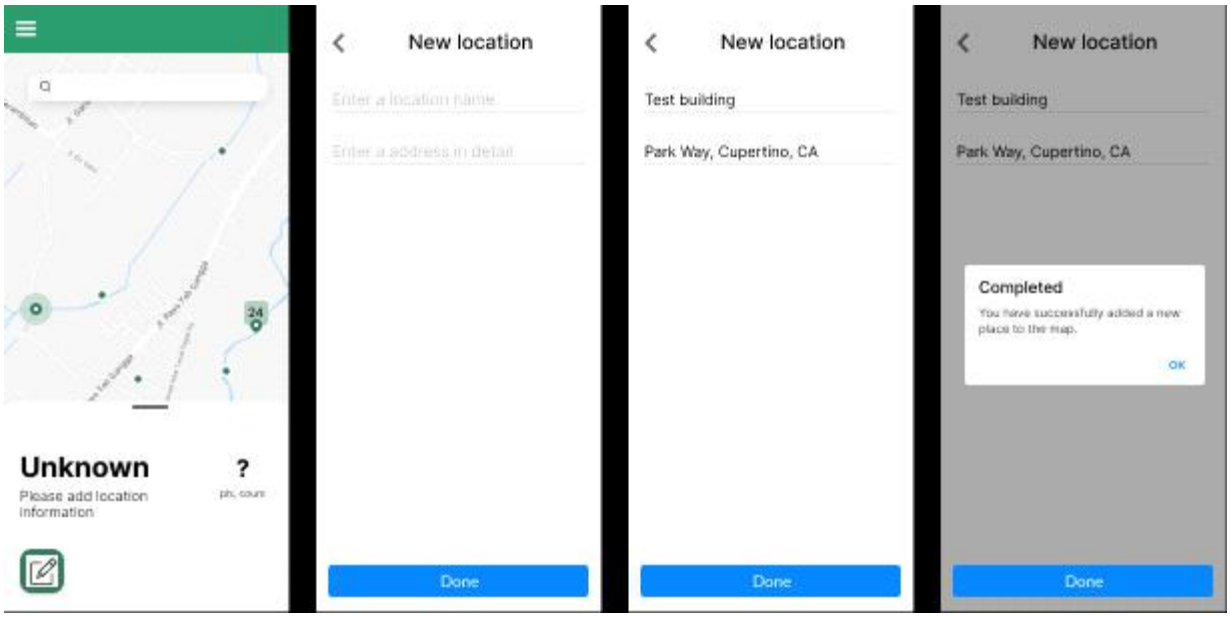
6.2.6 회원가입 화면



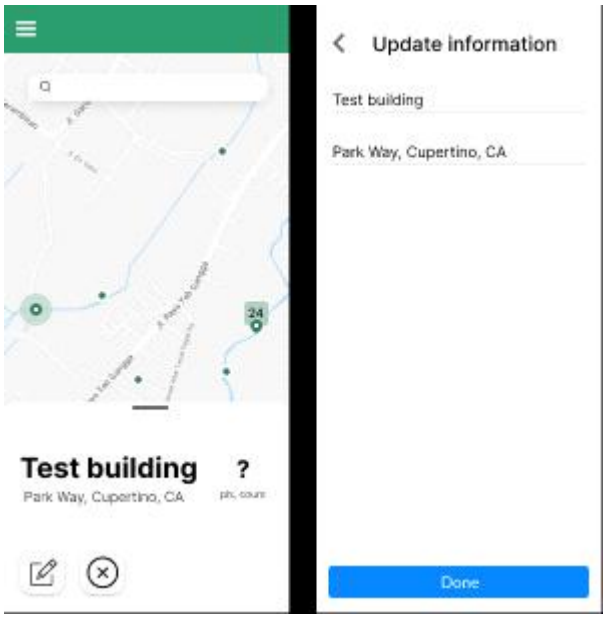
6.2.7 회원 정보 입력/수정/탈퇴 화면



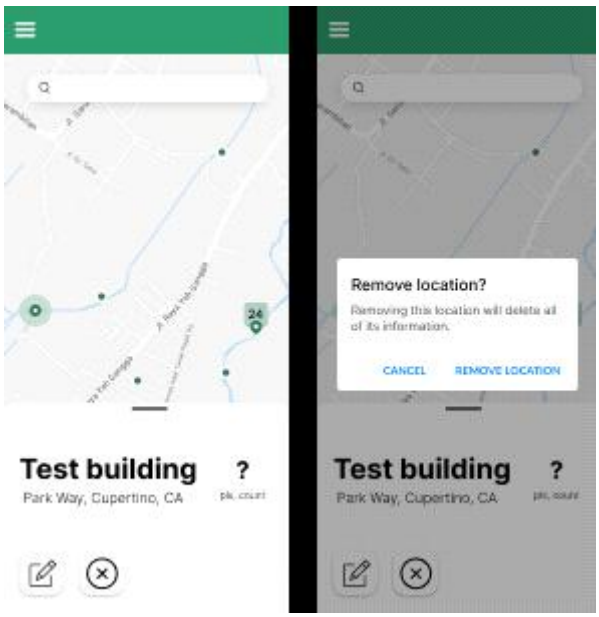
6.2.8 장소 등록 화면



6.2.9 장소 정보 수정 화면

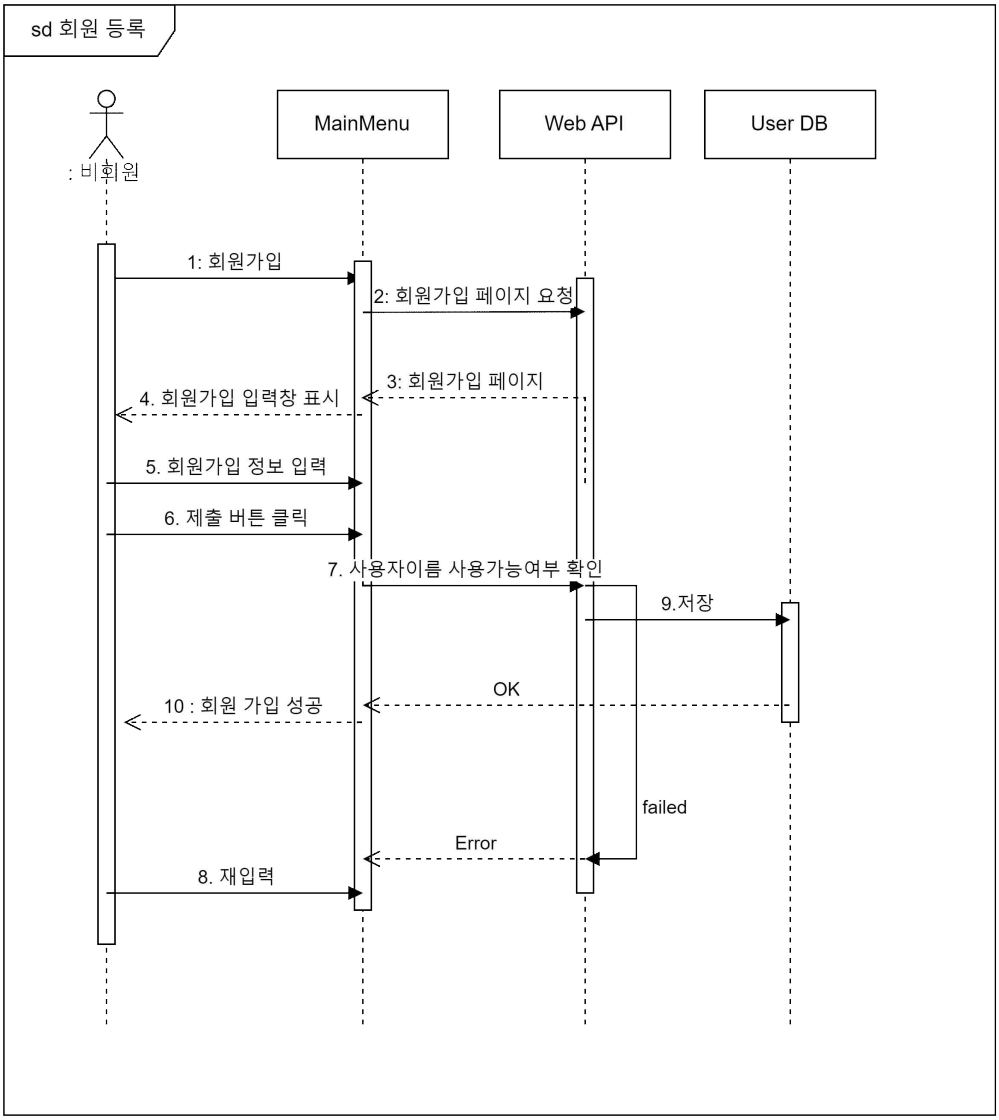


6.2.10 장소 정보 삭제 화면



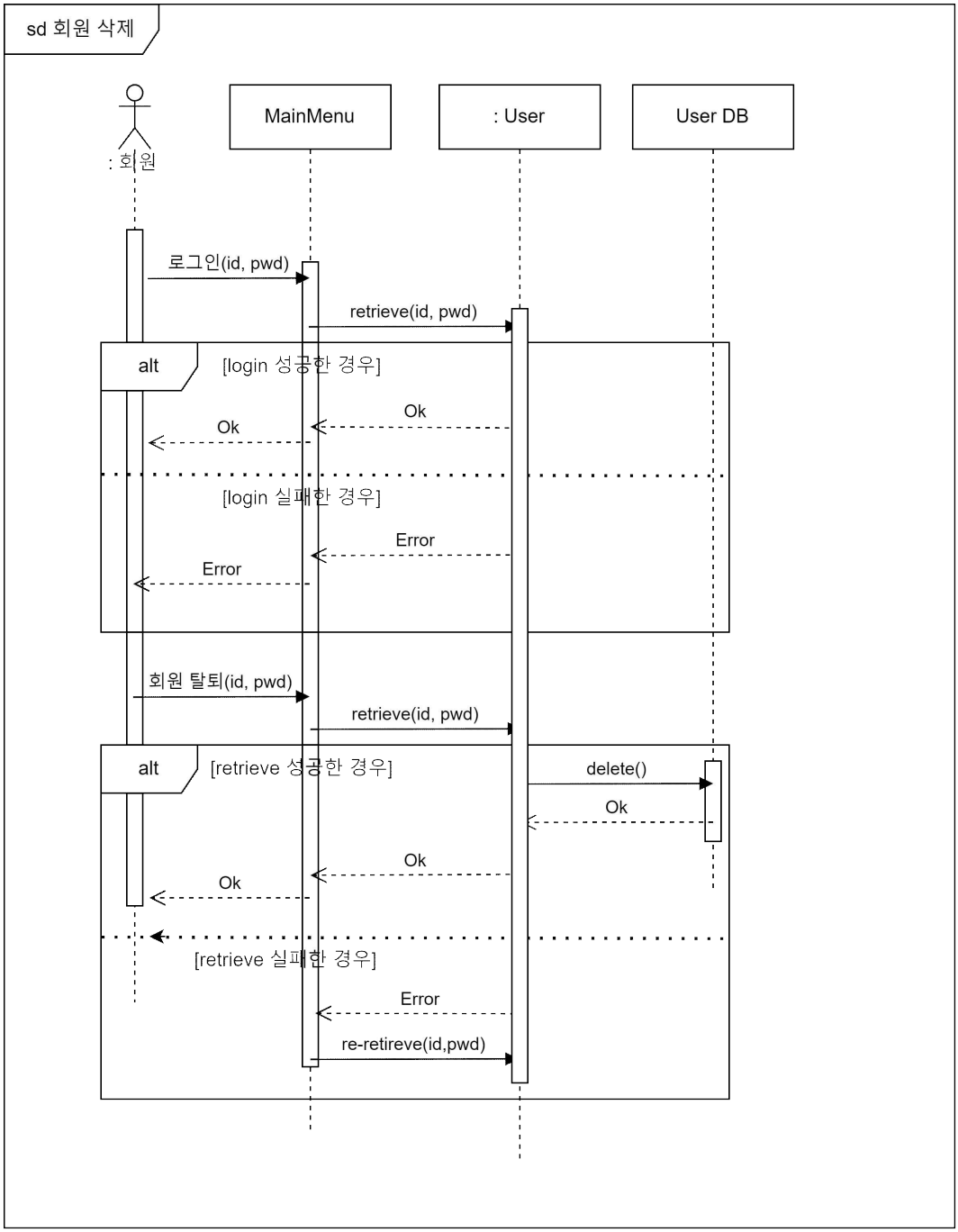
7. Use-Case에 대한 시퀀스 다이어그램 명세

7.1 Use-Case에 '회원 등록'에 대한 시퀀스 다이어그램 명세



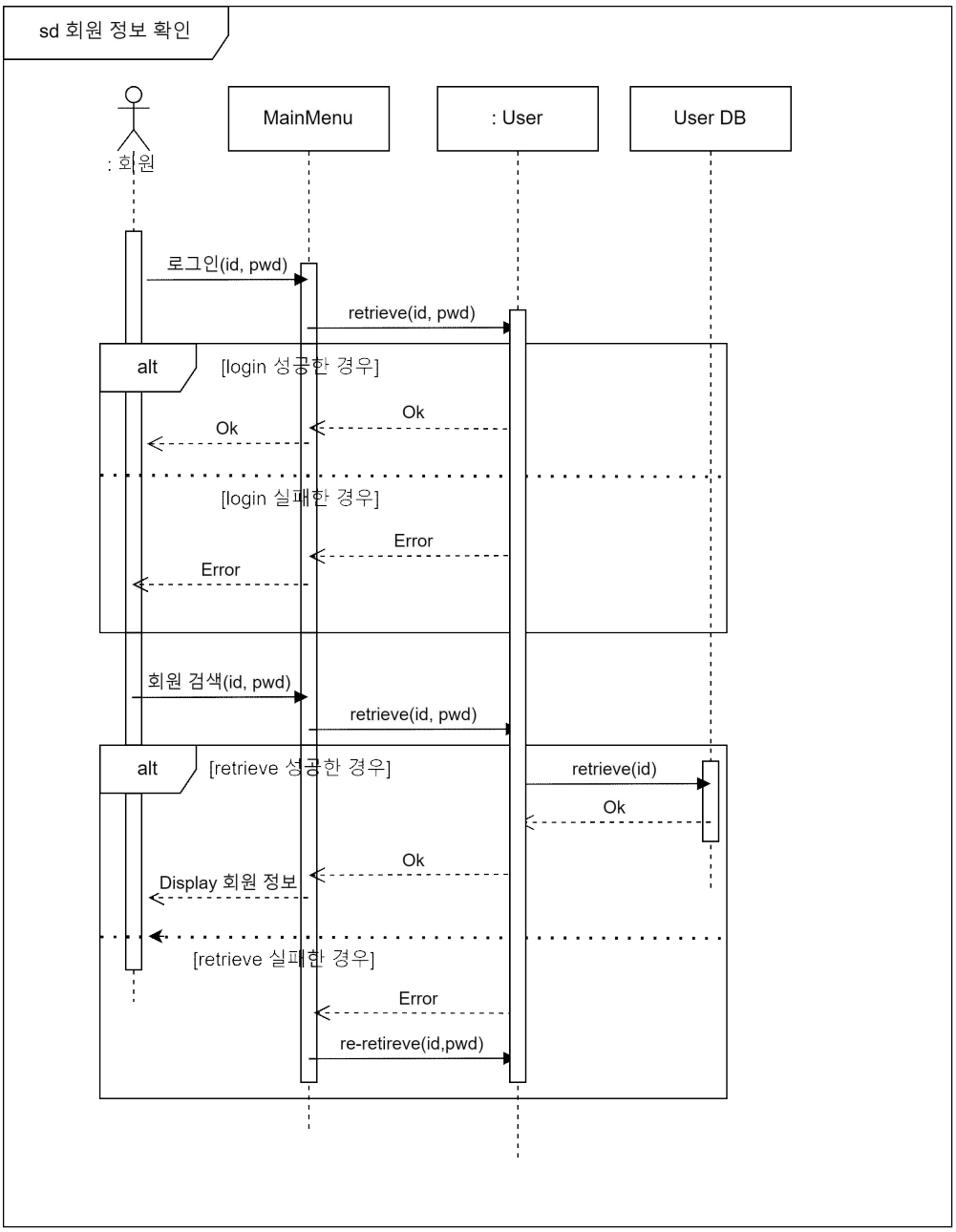
[그림 7.1 – "회원 등록" 시퀀스 다이어그램 명세]

7.2 Use-Case에 '회원 삭제'에 대한 시퀀스 다이어그램 명세



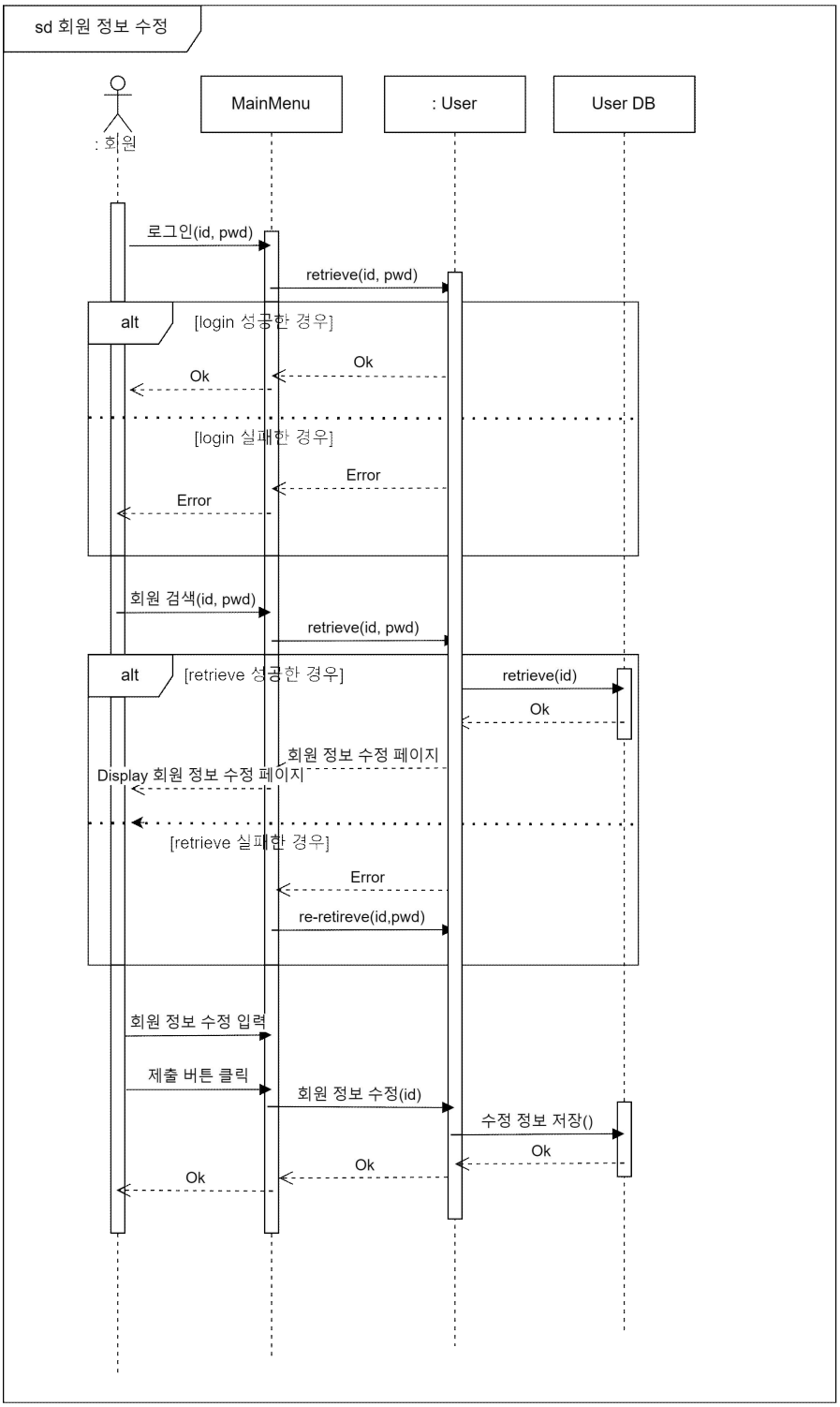
[그림 7.2 – "회원 삭제" 시퀀스 다이어그램 명세]

7.3 Use-Case에 '회원 정보 확인'에 대한 시퀀스 다이어그램 명세



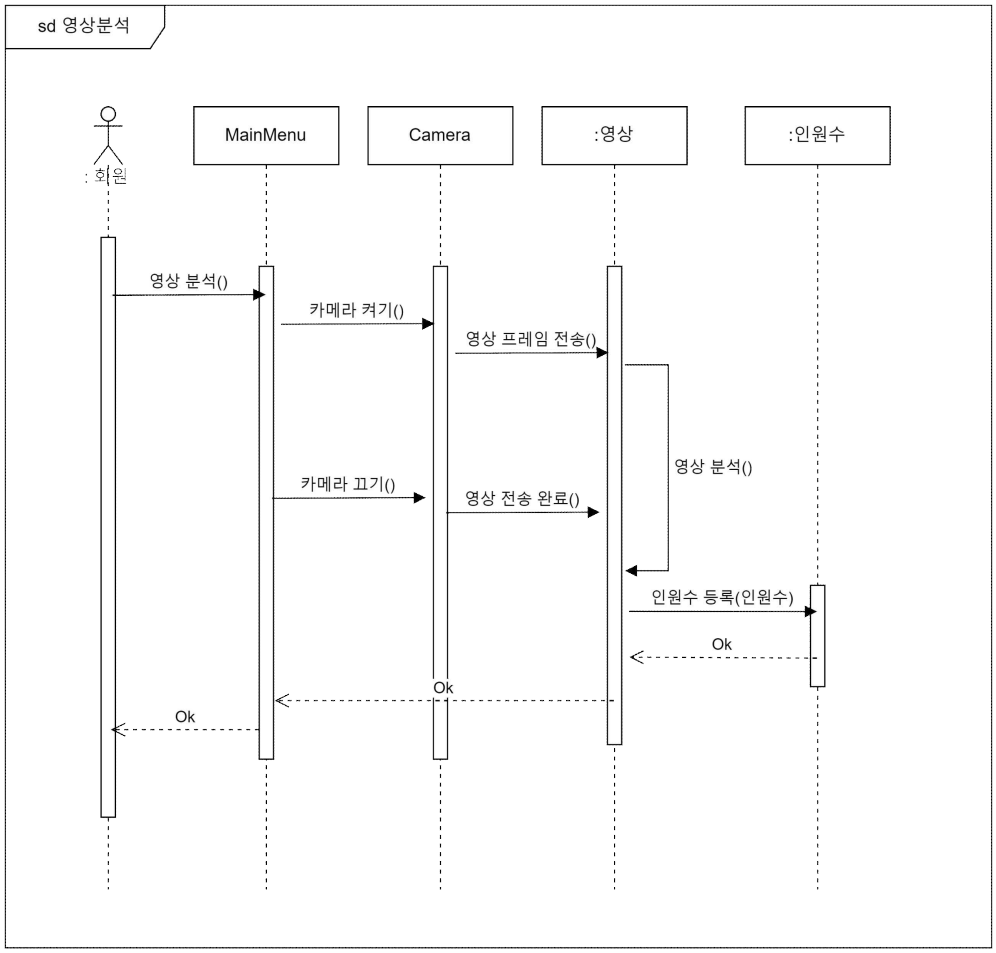
[그림 7.3 – "회원 정보 확인" 시퀀스 다이어그램 명세]

7.4 Use-Case에 '회원 정보 수정'에 대한 시퀀스 다이어그램 명세



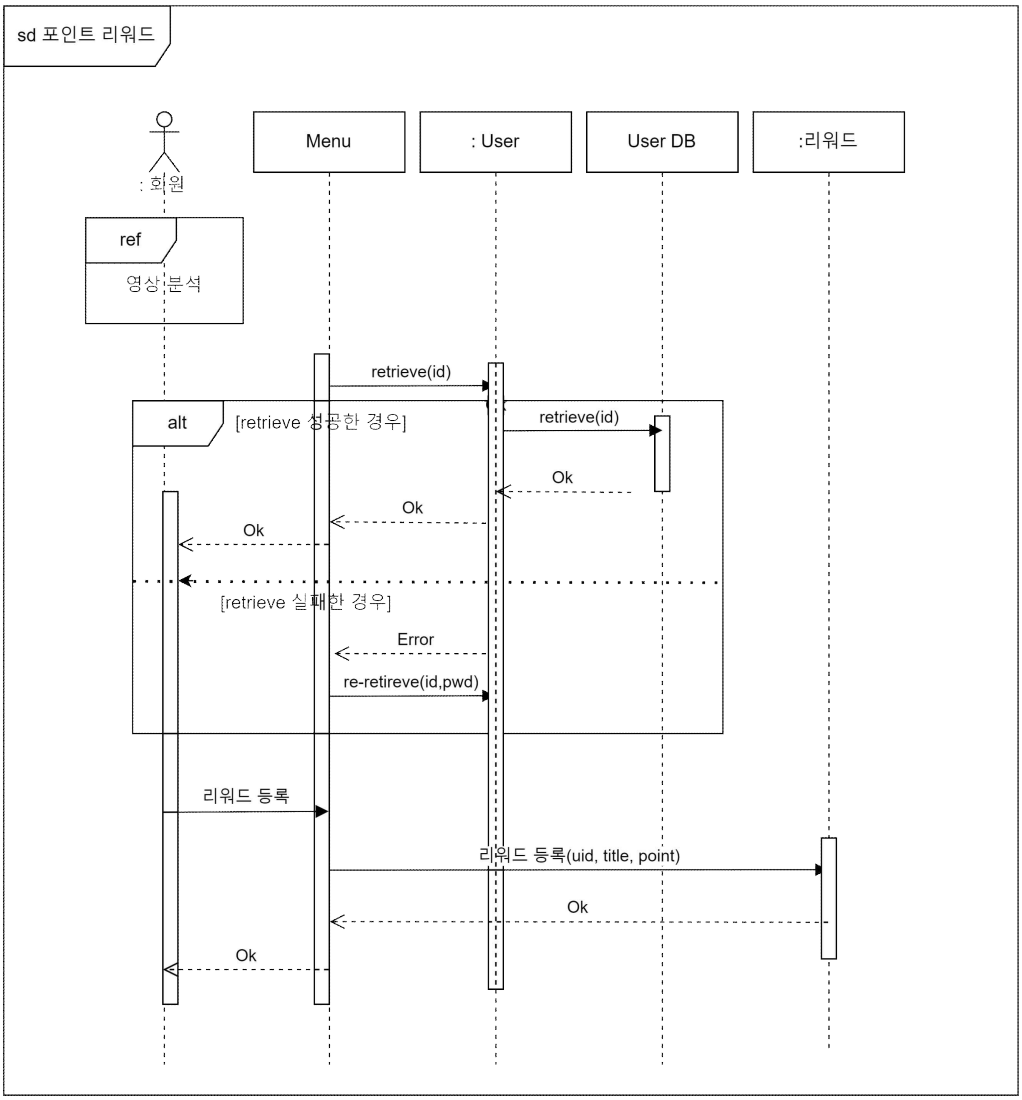
[그림 7.4 – "회원 정보 수정" 시퀀스 다이어그램 명세]

7.5 Use-Case에 '영상분석'에 대한 시퀀스 다이어그램 명세



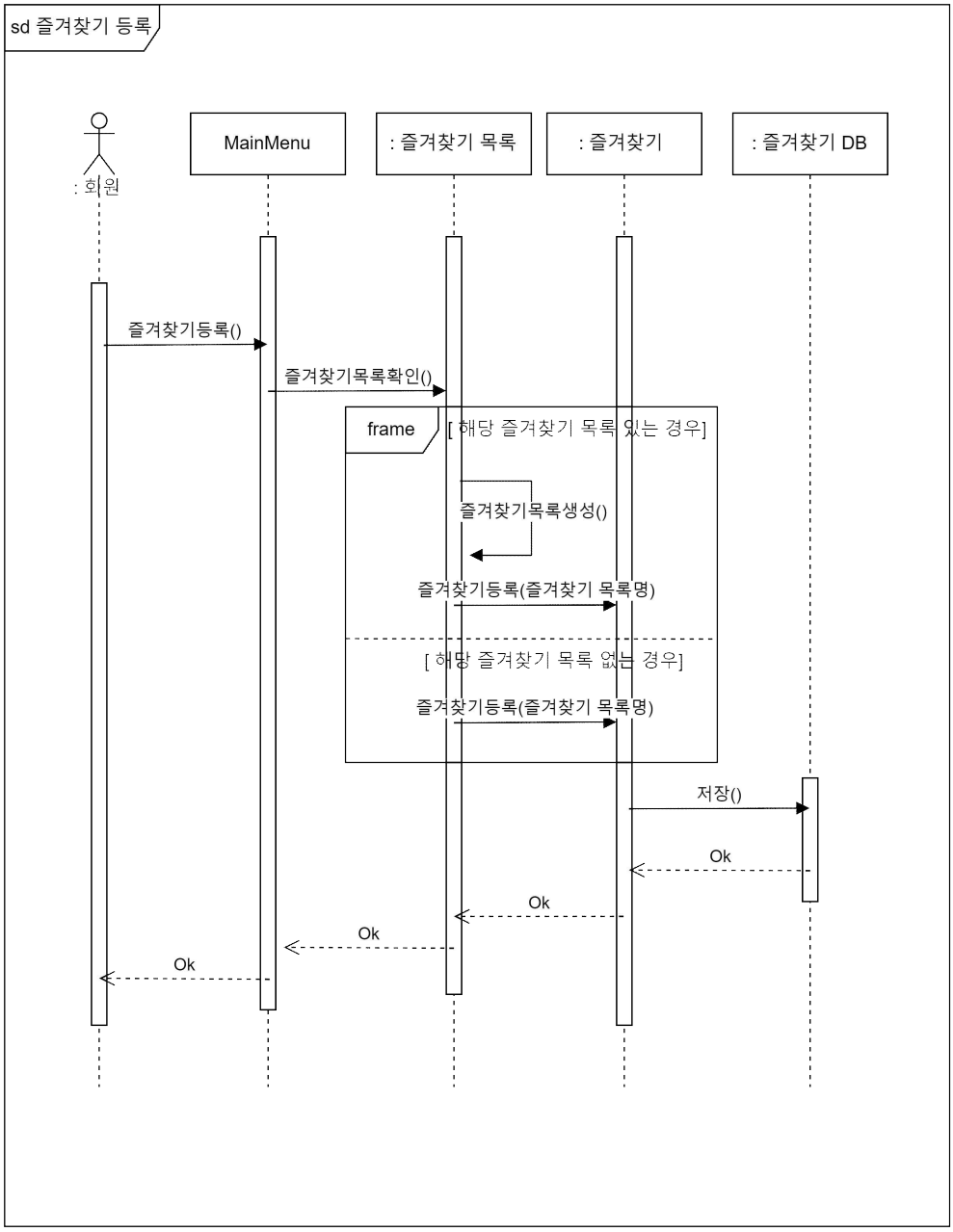
[그림 7.5 – "영상분석" 시퀀스 다이어그램 명세]

7.6 Use-Case에 '포인트 리워드'에 대한 시퀀스 다이어그램 명세



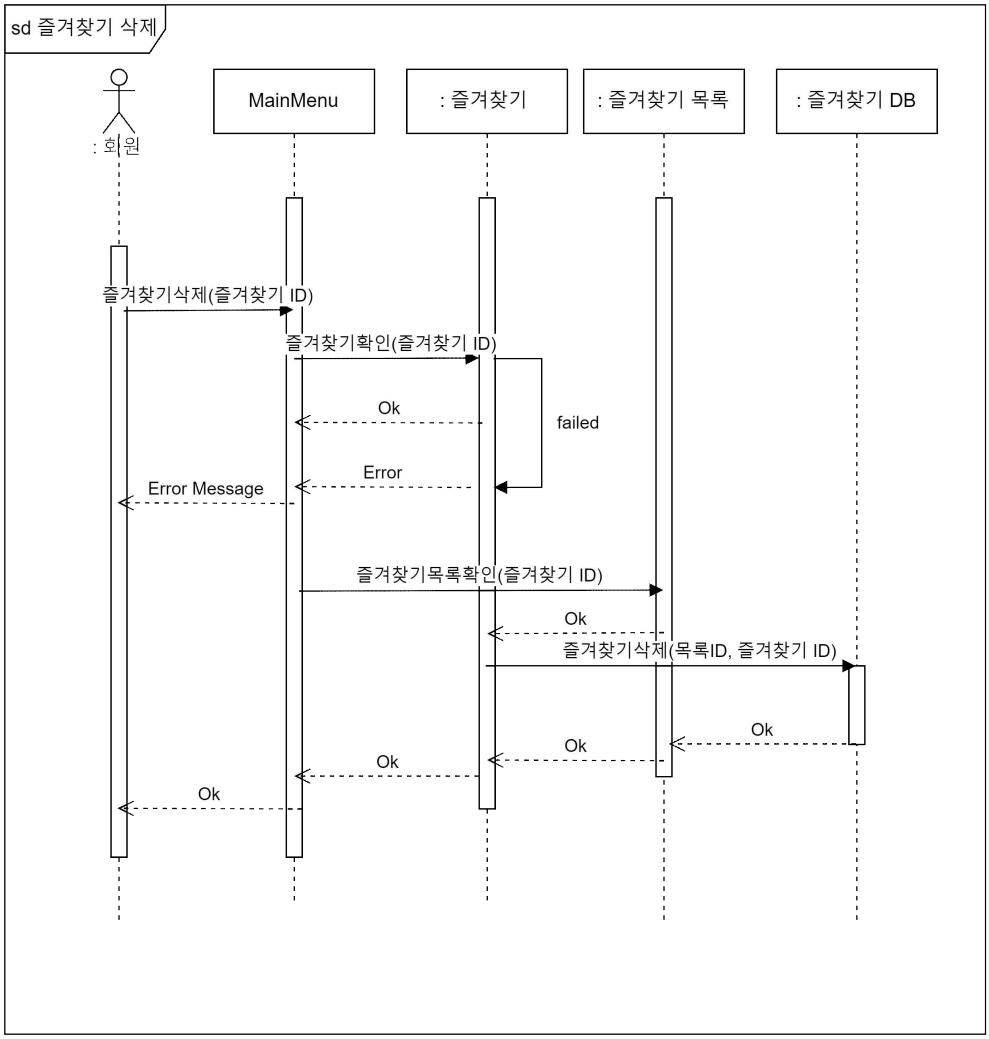
[그림 7.6 – "포인트 리워드" 시퀀스 다이어그램 명세]

7.7 Use-Case에 '즐겨찾기 등록'에 대한 시퀀스 다이어그램 명세



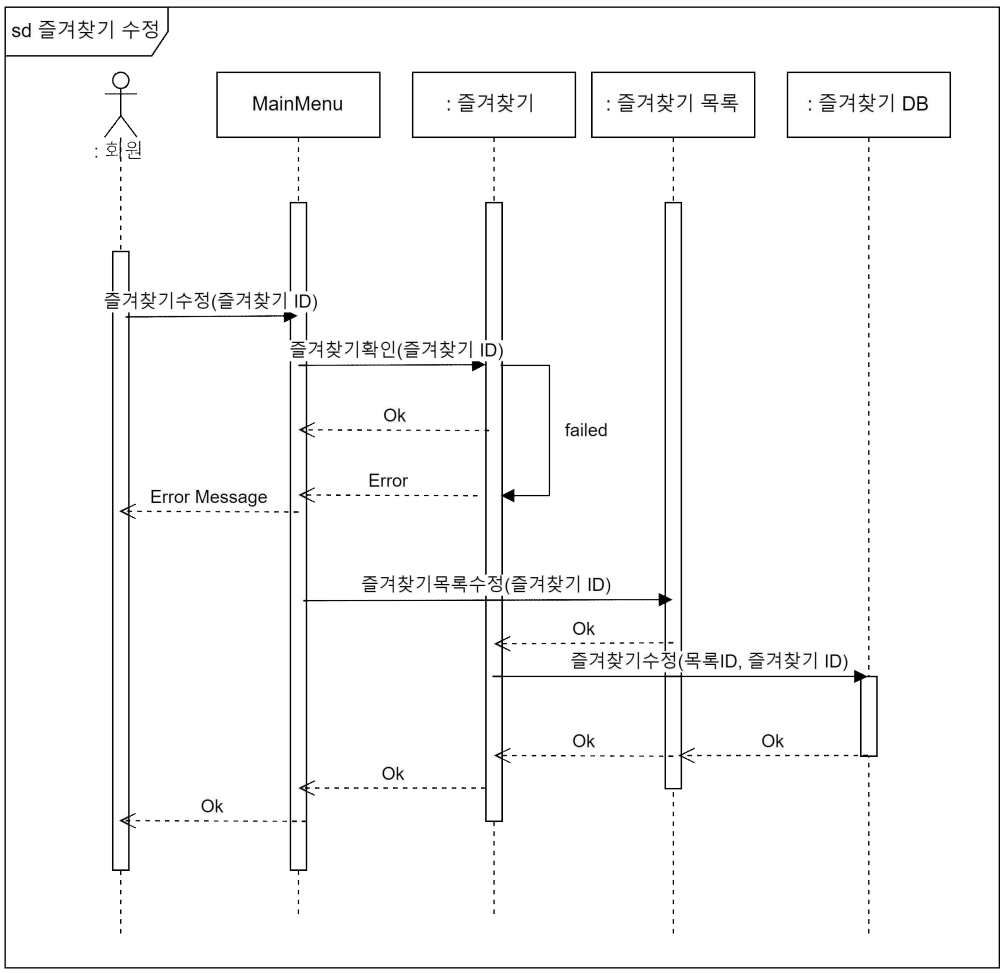
[그림 7.7 – "즐겨찾기 등록" 시퀀스 다이어그램 명세]

7.8 Use-Case에 '즐겨찾기 삭제'에 대한 시퀀스 다이어그램 명세



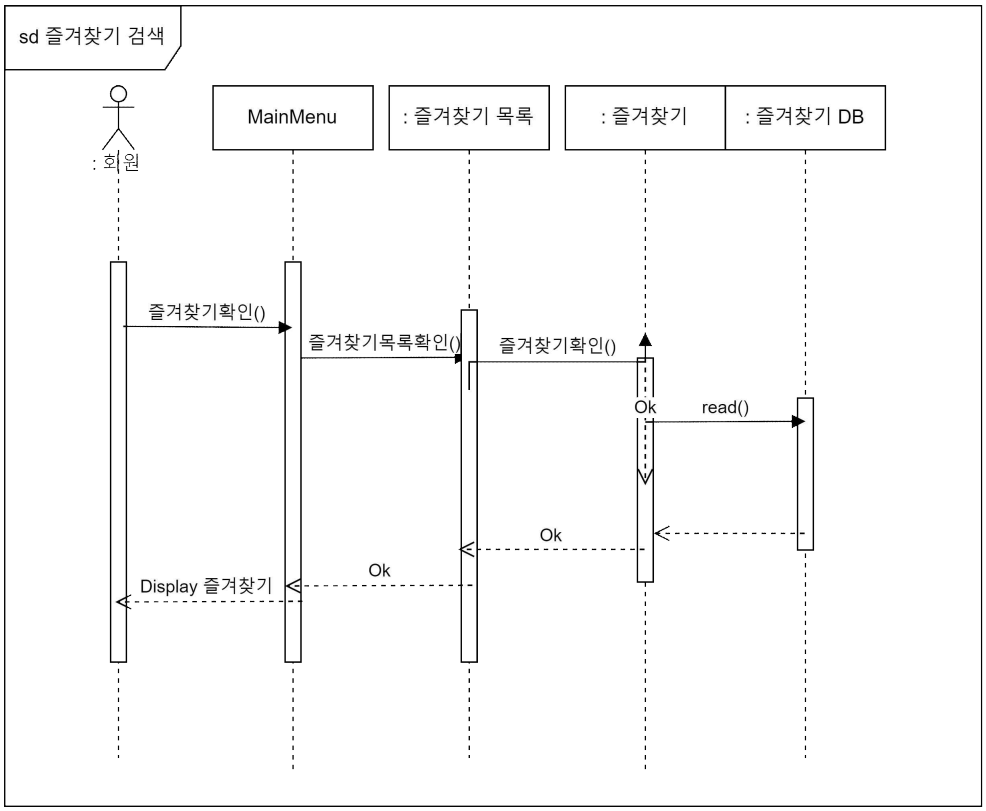
[그림 7.8 – "즐겨찾기 삭제" 시퀀스 다이어그램 명세]

7.9 Use-Case에 '즐겨찾기 수정'에 대한 시퀀스 다이어그램 명세



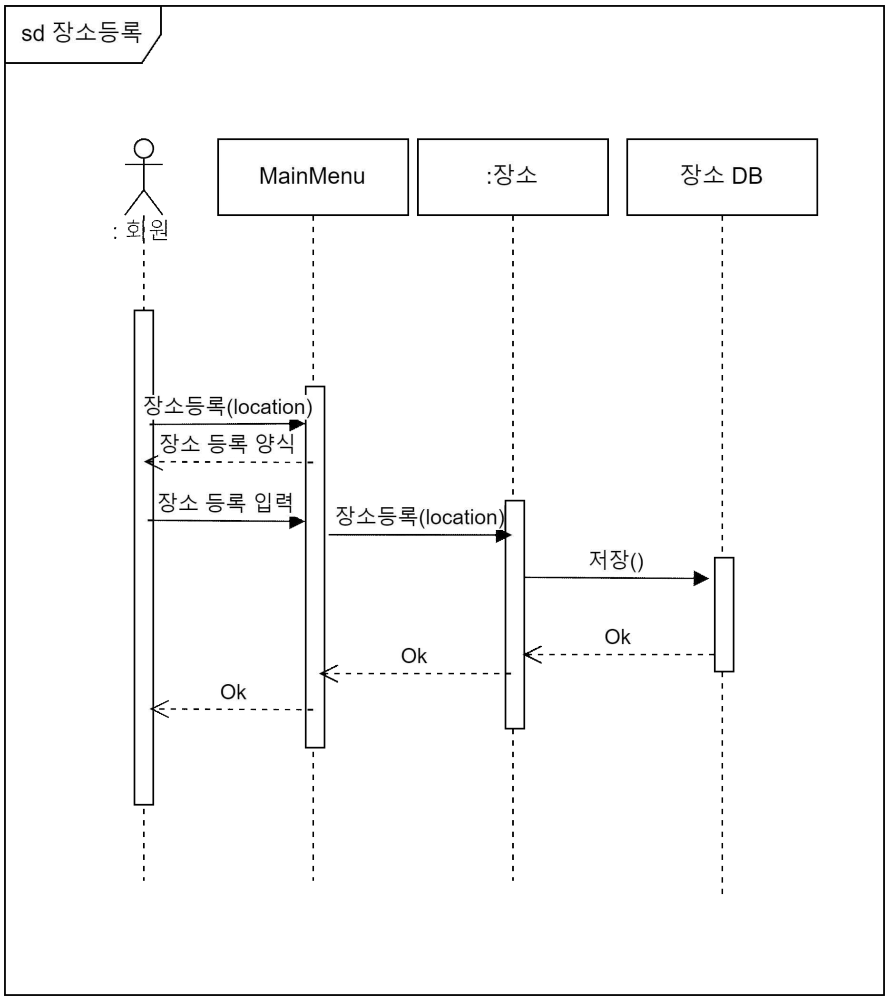
[그림 7.9 – "즐겨찾기 수정" 시퀀스 다이어그램 명세]

7.10 Use-Case에 '즐겨찾기 검색'에 대한 시퀀스 다이어그램 명세



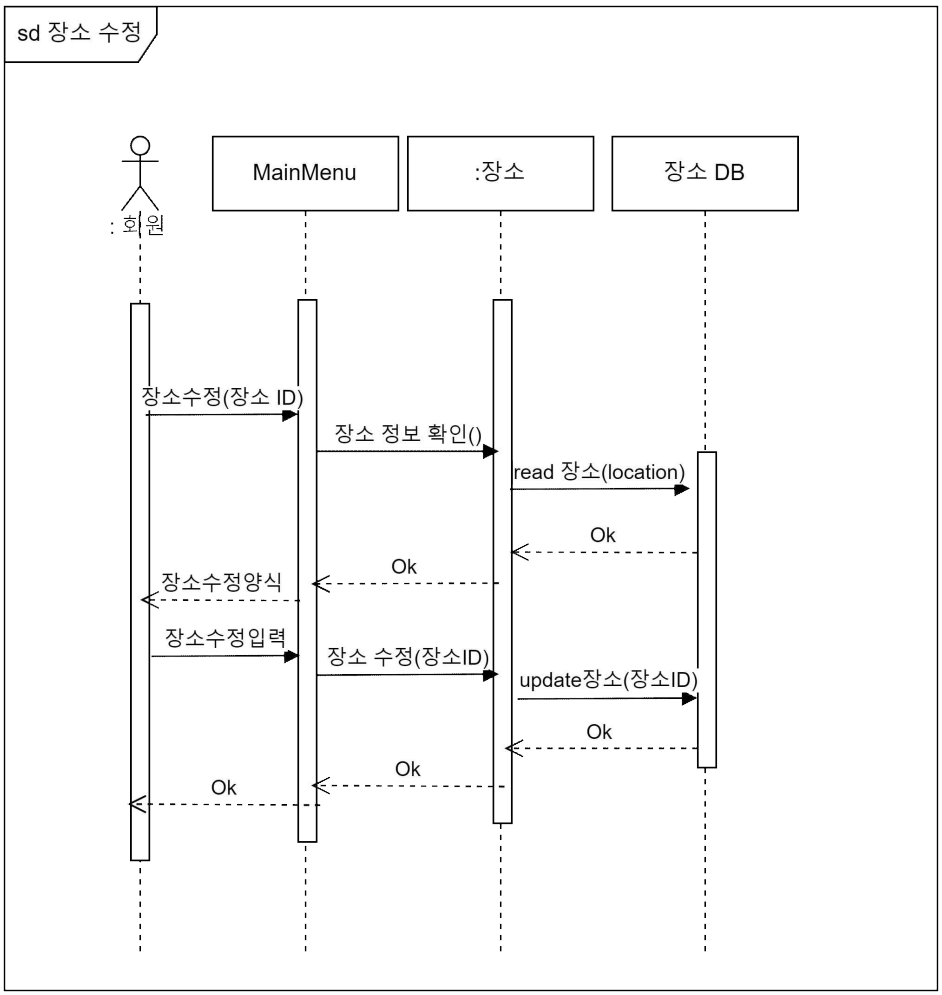
[그림 7.10 – "즐겨찾기 검색" 시퀀스 다이어그램 명세]

7.11 Use-Case에 '장소 등록'에 대한 시퀀스 다이어그램 명세



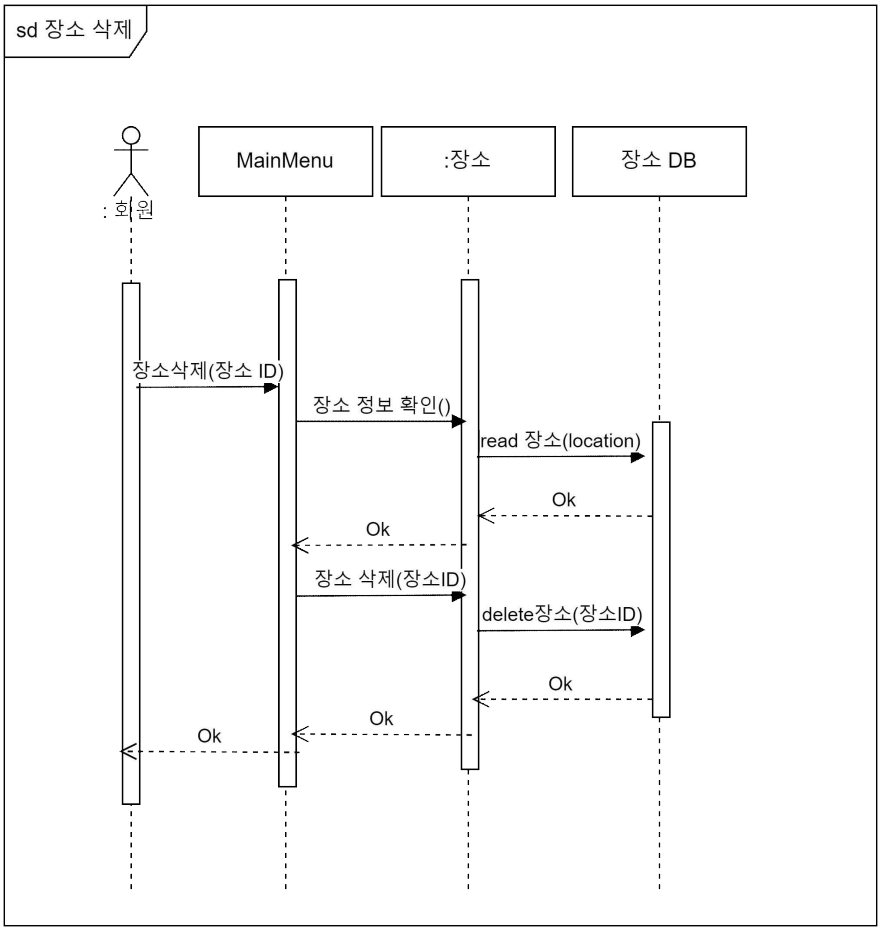
[그림 7.11 – "장소 등록" 시퀀스 다이어그램 명세]

7.12 Use-Case에 '장소 수정'에 대한 시퀀스 다이어그램 명세



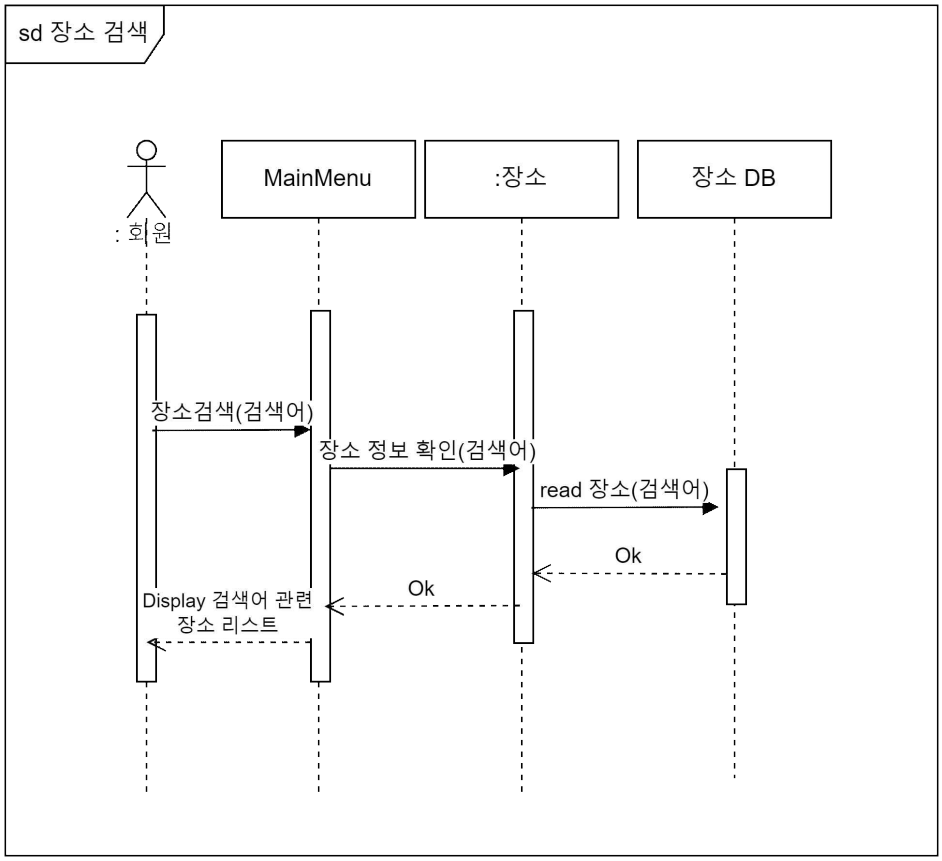
[그림 7.12 – "장소 수정" 시퀀스 다이어그램 명세]

7.13 Use-Case에 '장소 삭제'에 대한 시퀀스 다이어그램 명세



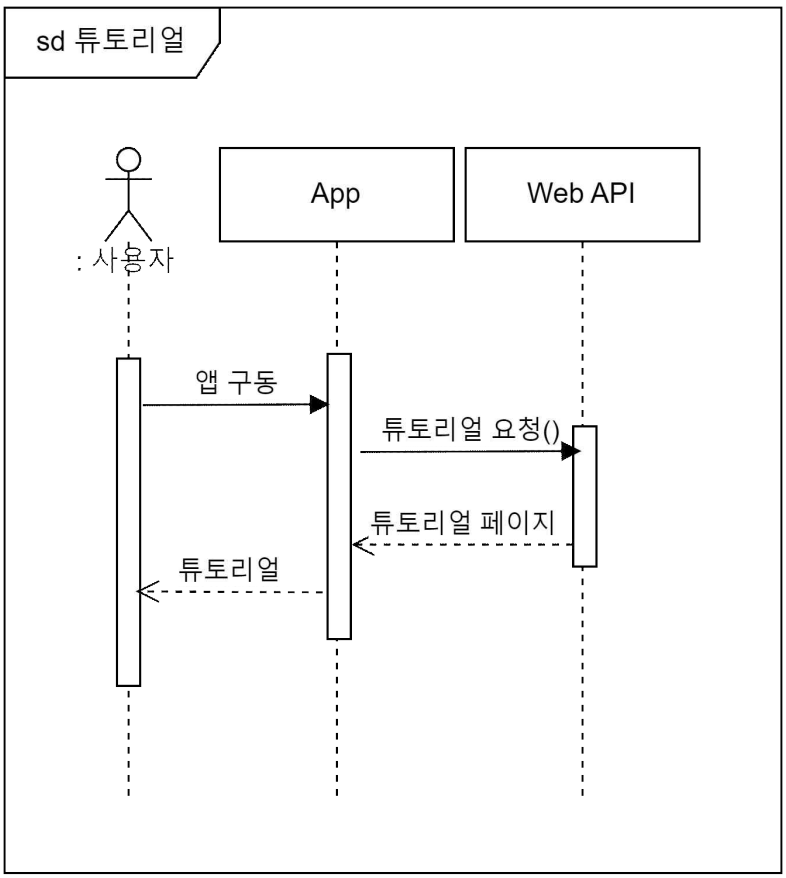
[그림 7.13 – "장소 삭제" 시퀀스 다이어그램 명세]

7.14 Use-Case에 '장소 검색'에 대한 시퀀스 다이어그램 명세



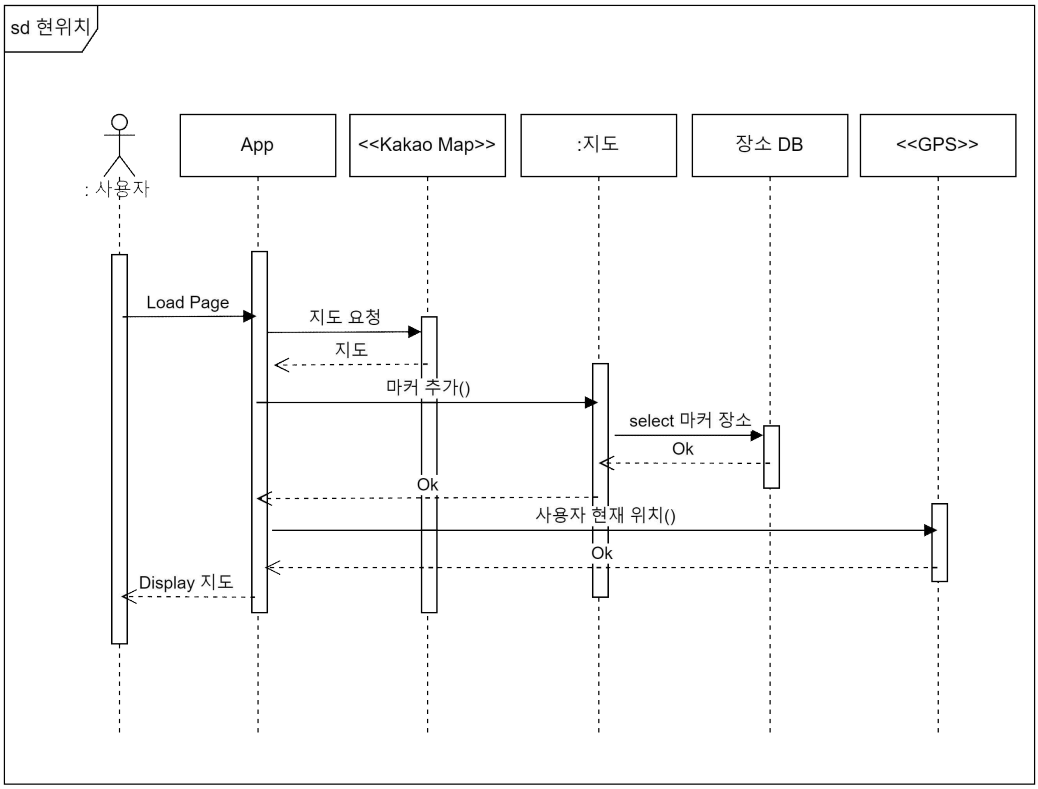
[그림 7.14 – "장소 검색" 시퀀스 다이어그램 명세]

7.15 Use-Case에 '튜토리얼'에 대한 시퀀스 다이어그램 명세



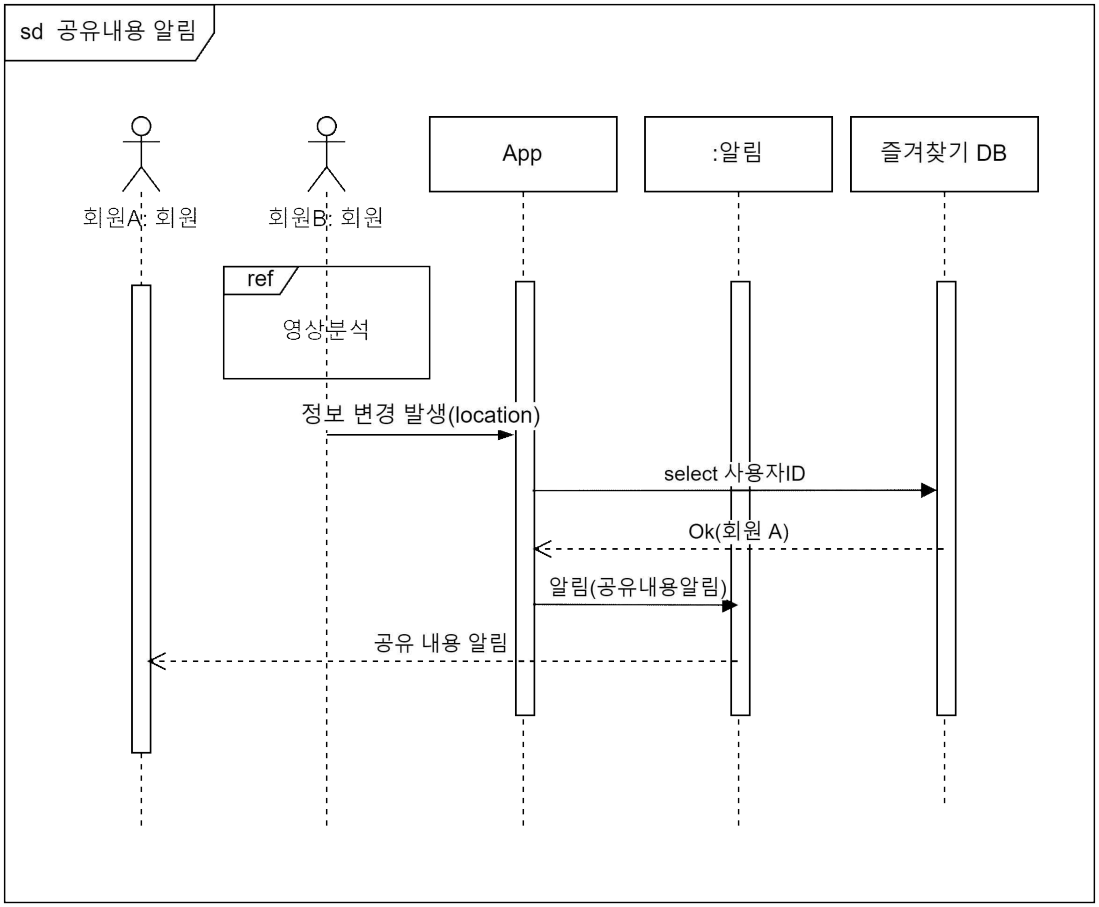
[그림 7.15 – "튜토리얼" 시퀀스 다이어그램 명세]

7.16 Use-Case에 '현 위치'에 대한 시퀀스 다이어그램 명세



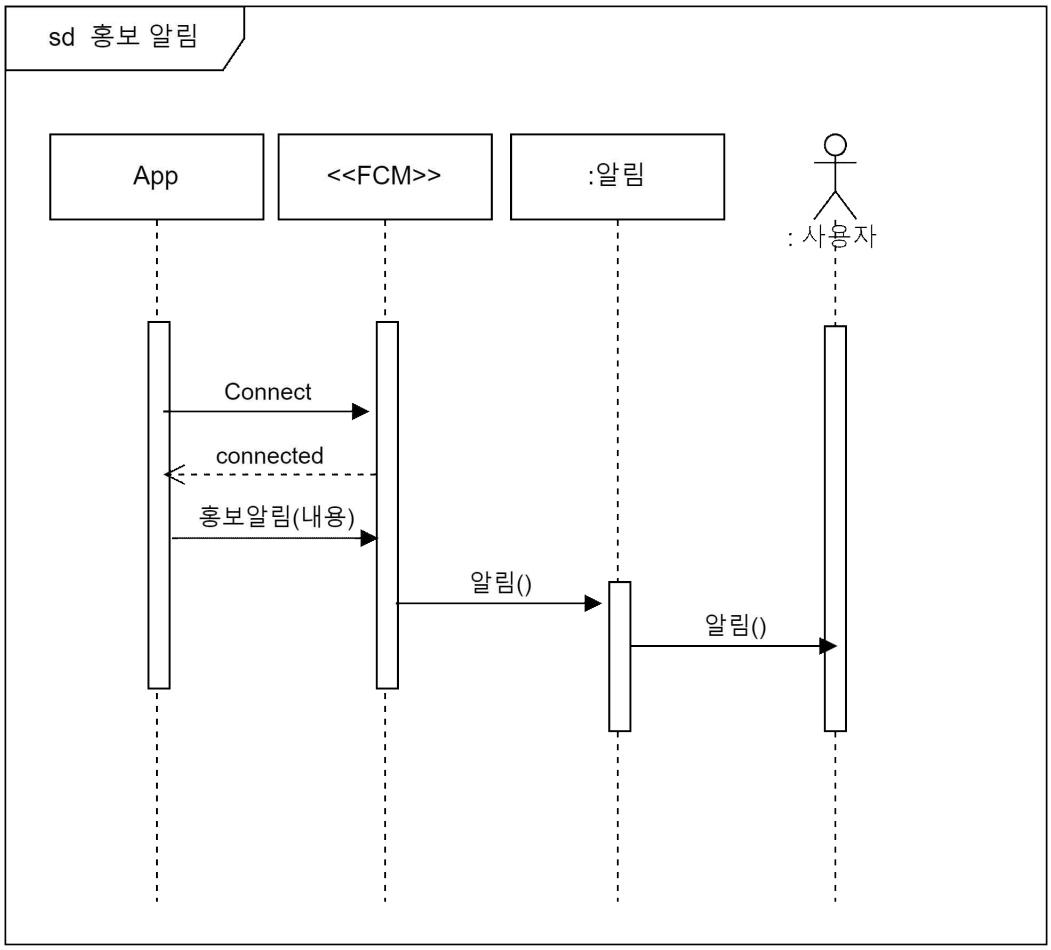
[그림 7.16 – "현 위치" 시퀀스 다이어그램 명세]

7.17 Use-Case에 '공유 내용 알림'에 대한 시퀀스 다이어그램 명세



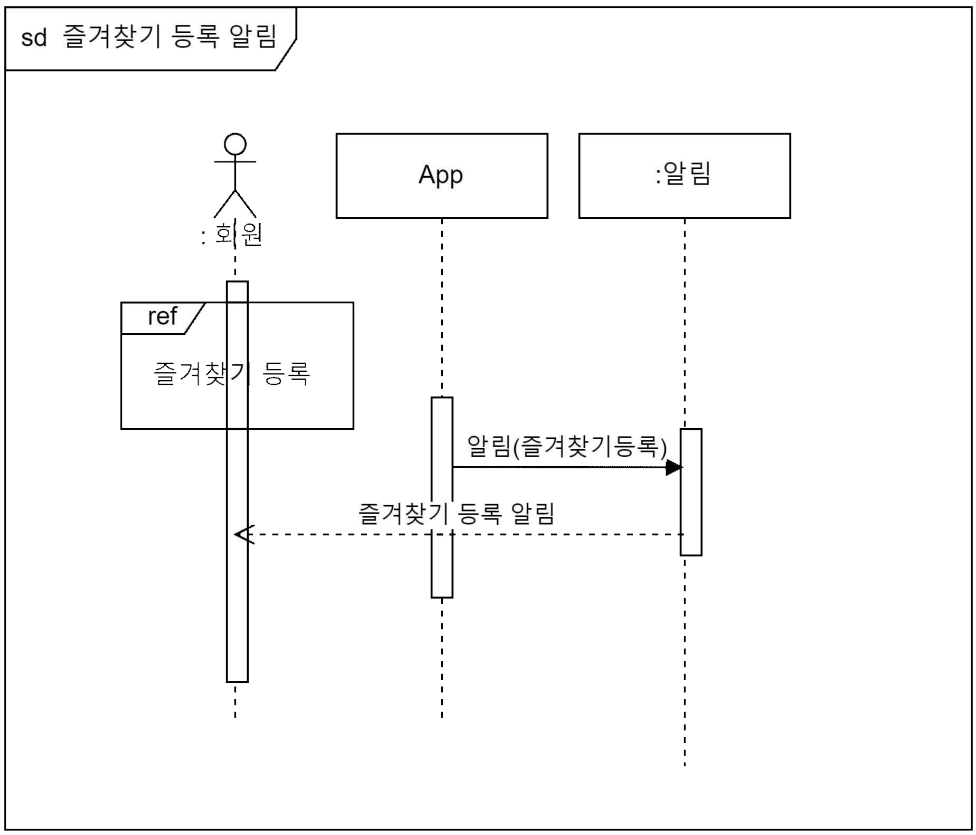
[그림 7.17 – "공유내용 알림" 시퀀스 다이어그램 명세]

7.18 Use-Case에 '홍보 알림'에 대한 시퀀스 다이어그램 명세



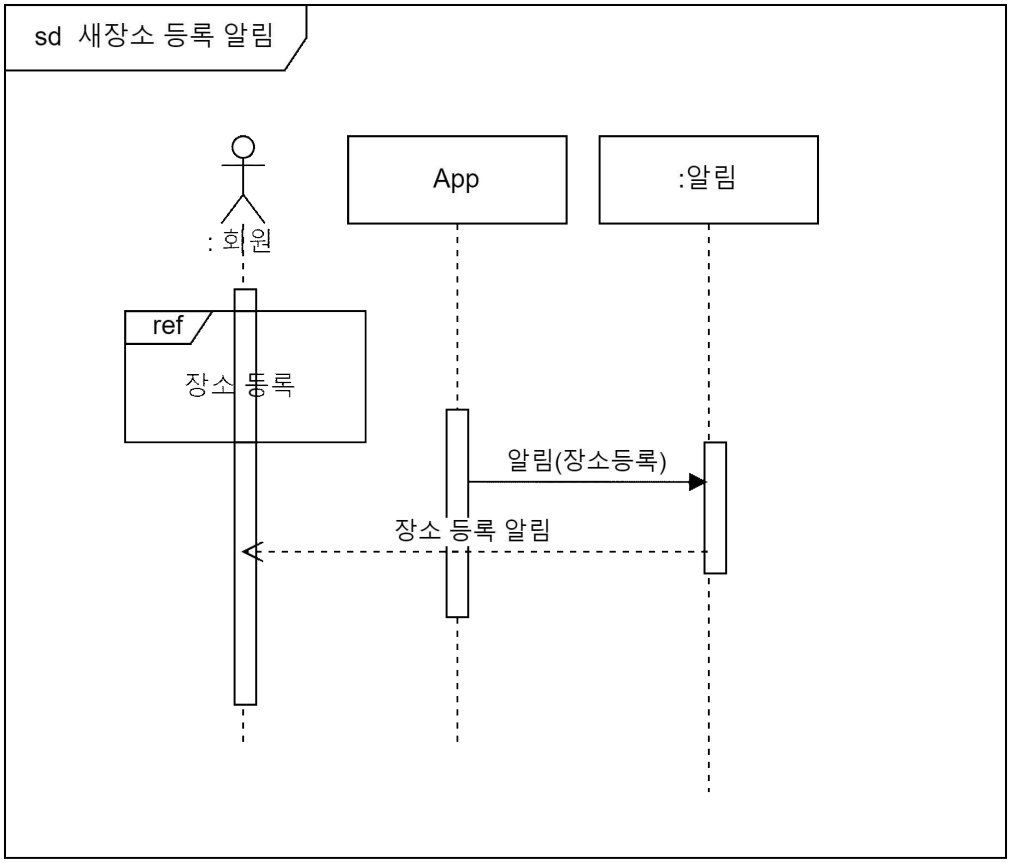
[그림 7.18 – "홍보 알림" 시퀀스 다이어그램 명세]

7.19 Use-Case에 '즐겨찾기 등록 알림'에 대한 시퀀스 다이어그램 명세



[그림 7.19 – "즐겨찾기 등록 알림" 시퀀스 다이어그램 명세]

7.20 Use-Case에 '새 장소 등록 알림'에 대한 시퀀스 다이어그램 명세



[그림 7.20 – "새 장소 등록 알림" 시퀀스 다이어그램 명세]

8. 메소드 명세

8.1 클래스 '알림'의 메소드 알고리즘 설계

8.1.1 alarm 메소드

메소드 이름	alarm
메소드 기능	파라미터로 받은 알림을 반환하는 메소드
알고리즘	

```
#NotificationManager 객체 생성
NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
# NotificationCompat.Builder 객체 생성
NotificationCompat.Builder builder = new NotificationCompat.Builder(context,
channelId).setSmallIcon(R.drawable.notification_icon)
            .setContentTitle("Notification Title")
            .setContentText("Notification Message")
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
            .setAutoCancel(true);
#NotificationChannel 생성
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    NotificationChannel channel = new NotificationChannel(channelId, channelName,
NotificationManager.IMPORTANCE_DEFAULT);
    notificationManager.createNotificationChannel(channel);
}
#NotificationManager를 사용하여 알림 표시
notificationManager.notify(notificationId, builder.build());
```

[표 8.1 – "alarm" 메소드 알고리즘 설계]

8.2 클래스 '사용자'의 메소드 알고리즘 설계

8.2.1 login 메소드

메소드 이름	login
메소드 기능	사용자로부터 아이디와 비밀번호를 입력받아 로그인 기능을 수행하는 메소드

알고리즘

```
# 사용자가 입력한 로그인 정보 가져오기
String username = editTextUsername.getText().toString().trim();
String password = editTextPassword.getText().toString().trim();
# 서버와 통신하여 로그인 요청 보내기
// 로그인 요청 URL
String url = "https://example.com/login";
// POST 요청으로 보낼 데이터 설정
Map<String, String> postData = new HashMap<>();
postData.put("username", username);
postData.put("password", password);

// Volley 라이브러리를 사용하여 POST 요청 보내기
StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                // 서버로부터 응답이 도착한 경우 호출됨
                // 로그인 성공 처리
            }
        }, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        // 서버로부터 응답이 도착하지 않은 경우 호출됨
        // 로그인 실패 처리
    }
}) {
```

```
    // POST 요청으로 보낼 데이터 설정
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        return postData;
    }
};

// Volley 라이브러리를 사용하여 요청 보내기
RequestQueue requestQueue = Volley.newRequestQueue(context);
requestQueue.add(stringRequest);
```

[표 8.2 – "login" 메소드 알고리즘 설계]

8.3 클래스 '회원'의 메소드 알고리즘 설계

8.3.1 find_id 메소드

메소드 이름	find_id
메소드 기능	회원이 회원가입을 통해 입력한 정보를 바탕으로 자신의 아이디를 찾는 메소드
알고리즘	

```
private void find_id() {
    // 아이디 찾기에 필요한 정보를 사용자가 입력할 수 있는 UI 요소 구현하기
    EditText emailEditText = findViewById(R.id.email_edittext);
    EditText phoneEditText = findViewById(R.id.phone_edittext);

    // 사용자가 입력한 이메일과 핸드폰 번호 가져오기
    String email = emailEditText.getText().toString().trim();
    String phone = phoneEditText.getText().toString().trim();

    // 서버와 통신하여 사용자 아이디 찾기 요청 보내기
    // 사용자 아이디 찾기 요청 URL
    String url = "https://example.com/find_my_id";
    // POST 요청으로 보낼 데이터 설정
    Map<String, String> postData = new HashMap<>();
    postData.put("email", email);
    postData.put("phone", phone);

    // Volley 라이브러리를 사용하여 POST 요청 보내기
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    // 서버로부터 응답이 도착한 경우 호출됨
                    // 사용자 아이디 표시
                    showMyId(response);
                }
```

```
  }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // 서버로부터 응답이 도착하지 않은 경우 호출됨
            // 오류 처리
            Toast.makeText(getApplicationContext(), "아이디 찾기에 실패했습니다. 다시 시도
해주세요.", Toast.LENGTH_SHORT).show();
        }
    }) {
        // POST 요청으로 보낼 데이터 설정
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            return postData;
        }
    };

    // Volley 라이브러리를 사용하여 요청 보내기
    RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
    requestQueue.add(stringRequest);
}
```

[표 8.3 – "find_id" 메소드 알고리즘 설계]

8.3.2 find_pw 메소드

메소드 이름	find_pw
메소드 기능	회원이 회원가입을 통해 입력한 정보를 바탕으로 자신의 비밀번호를 찾는 메소드

알고리즘

```
private void find_pw() {
    // 비밀번호 찾기에 필요한 정보를 사용자가 입력할 수 있는 UI 요소 구현하기
    EditText idEditText = findViewById(R.id.id_edittext);
    EditText emailEditText = findViewById(R.id.email_edittext);

    // 사용자가 입력한 아이디와 이메일 가져오기
    String id = idEditText.getText().toString().trim();
    String email = emailEditText.getText().toString().trim();

    // 서버와 통신하여 사용자 비밀번호 찾기 요청 보내기
    // 사용자 비밀번호 찾기 요청 URL
    String url = "https://example.com/find_my_password";
    // POST 요청으로 보낼 데이터 설정
    Map<String, String> postData = new HashMap<>();
    postData.put("id", id);
    postData.put("email", email);

    // Volley 라이브러리를 사용하여 POST 요청 보내기
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    // 서버로부터 응답이 도착한 경우 호출됨
                    // 사용자 비밀번호 표시
                    showMyPassword(response);
                }
            }, new Response.ErrorListener() {
```

```
        @Override
        public void onErrorResponse(VolleyError error) {
            // 서버로부터 응답이 도착하지 않은 경우 호출됨
            // 오류 처리
            Toast.makeText(getApplicationContext(), "비밀번호 찾기에 실패했습니다. 다시 시
도해주세요.", Toast.LENGTH_SHORT).show();
        }
    }) {
        // POST 요청으로 보낼 데이터 설정
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            return postData;
        }
    };

    // Volley 라이브러리를 사용하여 요청 보내기
    RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
    requestQueue.add(stringRequest);
}
```

[표 8.4 – "find_pw" 메소드 알고리즘 설계]

8.3.3 change_pw 메소드

메소드 이름	change_pw
메소드 기능	회원이 기존의 입력한 비밀번호에서 새로운 비밀번호를 변경하는 메소드

알고리즘

```
private void change_pw() {
    // 비밀번호 변경에 필요한 정보를 사용자가 입력할 수 있는 UI 요소 구현하기
    EditText currentPasswordEditText = findViewById(R.id.current_password_edittext);
    EditText newPasswordEditText = findViewById(R.id.new_password_edittext);
    EditText confirmPasswordEditText = findViewById(R.id.confirm_password_edittext);

    // 사용자가 입력한 현재 비밀번호, 새 비밀번호, 비밀번호 확인 가져오기
    String currentPassword = currentPasswordEditText.getText().toString().trim();
    String newPassword = newPasswordEditText.getText().toString().trim();
    String confirmPassword = confirmPasswordEditText.getText().toString().trim();

    // 새 비밀번호와 비밀번호 확인이 일치하는지 확인하기
    if (!newPassword.equals(confirmPassword)) {
        Toast.makeText(getApplicationContext(), "새 비밀번호와 비밀번호 확인이 일치하지 않
습니다.", Toast.LENGTH_SHORT).show();
        return;
    }

    // 서버와 통신하여 사용자 비밀번호 변경 요청 보내기
    // 사용자 비밀번호 변경 요청 URL
    String url = "https://example.com/change_my_password";
    // POST 요청으로 보낼 데이터 설정
    Map<String, String> postData = new HashMap<>();
    postData.put("current_password", currentPassword);
    postData.put("new_password", newPassword);
 // Volley 라이브러리를 사용하여 POST 요청 보내기
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
            new Response.Listener<String>() {
```

```
 @Override
                public void onResponse(String response) {
                    // 서버로부터 응답이 도착한 경우 호출됨
                    // 비밀번호 변경 성공 메시지 표시
                    Toast.makeText(getApplicationContext(), "비밀번호가 성공적으로 변경되
었습니다.", Toast.LENGTH_SHORT).show();
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // 서버로부터 응답이 도착하지 않은 경우 호출됨
            // 오류 처리
            Toast.makeText(getApplicationContext(), "비밀번호 변경에 실패했습니다. 다시 시
도해주세요.", Toast.LENGTH_SHORT).show();
        }
    }) {
        // POST 요청으로 보낼 데이터 설정
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            return postData;
        }
    };

    // Volley 라이브러리를 사용하여 요청 보내기
    RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
    requestQueue.add(stringRequest);
}
```

[표 8.5 – "change_pw" 메소드 알고리즘 설계]

8.3.4 user_info 메소드

메소드 이름	user_info
메소드 기능	회원이 자신의 회원 정보를 확인할 수 있는 메소드
알고리즘	

```
// 회원 정보를 담는 클래스
public class UserInfo {
    private String username;
    private String email;
    private String phone;
    private int age;

    public user_info(String username, String email, String phone, int age) {
        this.username = username;
        this.email = email;
        this.phone = phone;
        this.age = age;
    }

    public String getUsername() {
        return username;
    }

    public String getEmail() {
        return email;
    }

    public String getPhone() {
        return phone;
    }

    public int getAge() {
        return age;
```

```
    }
}

// 회원 정보를 확인하는 액티비티
public class UserInfoActivity extends AppCompatActivity {

    private TextView tvUsername;
    private TextView tvEmail;
    private TextView tvPhone;
    private TextView tvAge;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_user_info);

        // UI 컴포넌트를 초기화합니다.
        tvUsername = findViewById(R.id.tv_username);
        tvEmail = findViewById(R.id.tv_email);
        tvPhone = findViewById(R.id.tv_phone);
        tvAge = findViewById(R.id.tv_age);

        // 현재 로그인한 사용자의 정보를 가져와서 화면에 표시합니다.
        UserInfo currentUser = getCurrentUser(); // 현재 로그인한 사용자 정보를 가져오는
메서드
        tvUsername.setText(currentUser.getUsername());
        tvEmail.setText(currentUser.getEmail());
        tvPhone.setText(currentUser.getPhone());
        tvAge.setText(String.valueOf(currentUser.getAge()));
    }

    // 현재 로그인한 사용자 정보를 가져오는 메서드
    private UserInfo getCurrentUser() {
        // 예시로 하드코딩된 회원 정보를 반환합니다.
        return new UserInfo("john.doe", "johndoe@example.com", "010-1234-5678",
```

```
30);
    }
}
```

[표 8.6 – "user_info" 메소드 알고리즘 설계]

8.3.5 modify_user_info 메소드

메소드 이름	moify_user_info
메소드 기능	회원이 자신의 회원정보를 수정하기 위해서 회원가입 데이터를 수정하는 메소드
알고리즘	

```
// 회원 정보를 담는 클래스
public class UserInfo {
    private String username;
    private String email;
    private String phone;
    private int age;

    public user_info(String username, String email, String phone, int age) {
        this.username = username;
        this.email = email;
        this.phone = phone;
        this.age = age;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
```

```
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}

// 회원 정보를 수정하는 액티비티
public class EditUserInfoActivity extends AppCompatActivity {

    private EditText etUsername;
    private EditText etEmail;
    private EditText etPhone;
    private EditText etAge;
    private Button btnSave;

    private UserInfo currentUser;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_user_info);
```

```
// UI 컴포넌트를 초기화합니다.
        etUsername = findViewById(R.id.et_username);
        etEmail = findViewById(R.id.et_email);
        etPhone = findViewById(R.id.et_phone);
        etAge = findViewById(R.id.et_age);
        btnSave = findViewById(R.id.btn_save);

        // 현재 로그인한 사용자의 정보를 가져와서 UI에 표시합니다.
        currentUser = getCurrentUser(); // 현재 로그인한 사용자 정보를 가져오는 메서드
        etUsername.setText(currentUser.getUsername());
        etEmail.setText(currentUser.getEmail());
        etPhone.setText(currentUser.getPhone());
        etAge.setText(String.valueOf(currentUser.getAge()));

        // 저장 버튼을 누르면 수정된 정보를 저장합니다.
        btnSave.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                saveUserInfo();
            }
        });
    }

    // 현재 로그인한 사용자 정보를 가져오는 메서드
    private UserInfo getCurrentUser() {
        // 예시로 하드코딩된 회원 정보를 반환합니다.
        return new UserInfo("john.doe", "johndoe@example.com", "010-1234-5678", 30);
    }

    // 수정된 회원 정보를 저장하는 메서드
    private void saveUserInfo() {
        // UI에서 입력된 값을 가져와서 현재 사용자 정보에 반영합니다.
        currentUser.setUsername(etUsername.getText().toString());
```

```
    currentUser.setEmail(etEmail.getText().toString());
        currentUser.setPhone(etPhone.getText().toString());
        currentUser.setAge(Integer.parseInt(etAge.getText().toString()));

        // 수정된 회원 정보를 서버에 전송하고 성공 여부를 확인합니다.
        boolean success = saveUserInfoToServer(currentUser); // 수정된 회원 정보를 서버에 저장하는 메서드
        if (success) {
            // 수정이 성공하면 화면을 종료합니다.
            finish();
        } else {
            // 수정이 실패하면 사용자에게
```

[표 8.7 – "modify_user_info" 메소드 알고리즘 설계]

8.3.6 withdrawal 메소드

메소드 이름	withdrawal
메소드 기능	회원이 자신이 가입한 정보를 삭제하고 회원 탈퇴를 하는 메소드

알고리즘

```
public void withdrawal(String UID, String PassWd) {
    // 사용자가 입력한 아이디와 비밀번호로 인증을 진행한다
    if (authenticateUser(userId, password)) {
        // 인증이 성공하면 회원탈퇴를 진행한다
        deleteUserData(userId);
        // 회원탈퇴 후에는 로그아웃 처리를 한다
        logout();
        // 회원탈퇴가 완료되었다는 메시지를 사용자에게 보여준다
        showWithdrawalCompleteMessage();
    } else {
        // 인증에 실패한 경우 오류 메시지를 사용자에게 보여준다
        showAuthenticationErrorMessage();
    }
}

private boolean authenticateUser(String userId, String password) {
    // 사용자 인증 로직을 구현한다
    // 예를 들면, 서버에 로그인 정보를 전송하여 인증 결과를 받아올 수 있다
    // 인증에 성공하면 true, 실패하면 false를 반환한다
    // 이 부분은 구체적인 구현 방식에 따라 달라질 수 있다
    return true; // 더미 데이터로 true를 반환한다
}

private void deleteUserData(String userId) {
    // 회원탈퇴를 위해 사용자 데이터를 삭제한다
    // 이 부분은 구체적인 구현 방식에 따라 달라질 수 있다
}
```

```
private void logout() {
    // 로그아웃 처리를 한다
    // 이 부분은 구체적인 구현 방식에 따라 달라질 수 있다
}

private void showWithdrawalCompleteMessage() {
    // 회원탈퇴가 완료되었다는 메시지를 사용자에게 보여준다
    // 이 부분은 구체적인 구현 방식에 따라 달라질 수 있다
}

private void showAuthenticationErrorMessage() {
    // 인증에 실패한 경우 오류 메시지를 사용자에게 보여준다
    // 이 부분은 구체적인 구현 방식에 따라 달라질 수 있다
}
```

[표 8.8 – "withdrawal" 메소드 알고리즘 설계]

8.4 클래스 '비회원'의 메소드 알고리즘 설계

8.4.1 join 메소드

메소드 이름	join
메소드 기능	비회원이 자신의 정보를 입력하여 회원가입을 하는 메소드

알고리즘

```
#사용자가 입력한 회원가입 정보 가져오기
String username = editTextUsername.getText().toString().trim();
String password = editTextPassword.getText().toString().trim();
String email = editTextEmail.getText().toString().trim();
#서버와 통신하여 회원가입 요청 보내기
// 회원가입 요청 URL
String url = "https://example.com/signup";
// POST 요청으로 보낼 데이터 설정
Map<String, String> postData = new HashMap<>();
postData.put("username", username);
postData.put("password", password);
postData.put("email", email);

// Volley 라이브러리를 사용하여 POST 요청 보내기
StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                // 서버로부터 응답이 도착한 경우 호출됨
                // 회원가입 성공 처리
            }
        }, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        // 서버로부터 응답이 도착하지 않은 경우 호출됨
        // 회원가입 실패 처리
```

```
    }
}) {
    // POST 요청으로 보낼 데이터 설정
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        return postData;
    }
};

// Volley 라이브러리를 사용하여 요청 보내기
RequestQueue requestQueue = Volley.newRequestQueue(context);
requestQueue.add(stringRequest);
```

[표 8.9 – "join" 메소드 알고리즘 설계]

8.5 클래스 '알림'의 메소드 알고리즘 설계

8.5.1 register_rewards 메소드

메소드 이름	register_rewards
메소드 기능	회원이 인원수를 공유하였을 때 임의의 포인트 보상을 제공하는 메소드
알고리즘	

```
#사용자에게 포인트를 제공
private void register_rewards(UID) {
    // 포인트 보상을 받을 사용자 정보 가져오기
    String UID = getUserId(); // 사용자 ID
    String userToken = getUserToken(); // 사용자 토큰

    // 서버와 통신하여 포인트 보상 요청 보내기
    // 포인트 보상 요청 URL
    String url = "https://example.com/reward_points";
    // POST 요청으로 보낼 데이터 설정
    Map<String, String> postData = new HashMap<>();
    postData.put("user_id", userId);
    postData.put("user_token", userToken);
    postData.put("points", String.valueOf(points));

    // Volley 라이브러리를 사용하여 POST 요청 보내기
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    // 서버로부터 응답이 도착한 경우 호출됨
                    // 포인트 보상 성공 처리
                    Toast.makeText(getApplicationContext(), "포인트 보상이 완료되었습니
다.", Toast.LENGTH_SHORT).show();
```

```
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // 서버로부터 응답이 도착하지 않은 경우 호출됨
            // 포인트 보상 실패 처리
            Toast.makeText(getApplicationContext(), "포인트 보상에 실패했습니다. 다시 시도
해주세요.", Toast.LENGTH_SHORT).show();
        }
    }) {
        // POST 요청으로 보낼 데이터 설정
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            return postData;
        }
    };

    // Volley 라이브러리를 사용하여 요청 보내기
    RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
    requestQueue.add(stringRequest);
}
2) #포인트 보상 기능 호출
int rewardPoints = 100; // 보상 포인트
giveRewardPoints(rewardPoints); // 포인트 보상 함수 호출
```

[표 8.10 – "register_rewards" 메소드 알고리즘 설계]

8.5.2 reward_lookup 메소드

메소드 이름	reward_lookup
메소드 기능	회원 자신이 보유하고 있는 포인트를 확인할 수 있는 메소드

알고리즘

1) #포인트 확인 기능을 호출

```
private void reward_lookup(UID) {
    // 포인트를 확인할 사용자 정보 가져오기
    String userId = getUserId(); // 사용자 ID
    String userToken = getUserToken(); // 사용자 토큰

    // 서버와 통신하여 사용자 포인트 정보 요청 보내기
    // 사용자 포인트 정보 요청 URL
    String url = "https://example.com/my_points";
    // POST 요청으로 보낼 데이터 설정
    Map<String, String> postData = new HashMap<>();
    postData.put("user_id", userId);
    postData.put("user_token", userToken);

    // Volley 라이브러리를 사용하여 POST 요청 보내기
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    // 서버로부터 응답이 도착한 경우 호출됨
                    // 사용자 포인트 정보 표시
                    showMyPoints(response);
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // 서버로부터 응답이 도착하지 않은 경우 호출됨
            // 오류 처리
```

```
 Toast.makeText(getApplicationContext(), "포인트 정보를 불러오는데 실패했습니다. 다시 시도
해주세요.", Toast.LENGTH_SHORT).show();
        }
    }) {
        // POST 요청으로 보낼 데이터 설정
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            return postData;
        }
    };

    // Volley 라이브러리를 사용하여 요청 보내기
    RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
    requestQueue.add(stringRequest);
}
3) 포인트 확인 기능 호출
checkMyPoints(); // 포인트 확인 함수 호출
```

[표 8.11 – "reward_lookup" 메소드 알고리즘 설계]

8.6 클래스 '영상분석'의 메소드 알고리즘 설계

8.6.1 video_analysis 메소드

메소드 이름	video_analysis
메소드 기능	영상을 분석하여 인원수를 계산하는 메소드
알고리즘	

```
public int video_analysis(Video video) { //video: 사용자가 입력한 영상
    // 영상 분석 모델 로드
    Model model = loadModel();
    // 영상 속 사람 객체 탐지
    Object object = detectObject(video, model);
    //  탐지 객체 트랙킹하여 중복 제거
    Object trackedObject = trackObject(video, object);
    // 사람 수 카운팅
    int numPeople = countPeople(video, trackedObject);
    // 인원수 리턴
    return numPeople;
}
```

[표 8.12 – "video_analysis" 메소드 알고리즘 설계]

8.7 클래스 '장소'의 메소드 알고리즘 설계

8.7.1 delete_place 메소드

메소드 이름	delete_place
메소드 기능	등록된 장소의 정보를 삭제하는 메소드
알고리즘	

```
public void delete_place(String PID) { //PID : 장소의 번호
    // DB에 장소가 저장되어 있는지 확인
    if (placeExists(PID)) {
        // 삭제
        deletePlaceFromDB(PID);
        // 장소 삭제 되었음을 알림.
        print_message("The place has been successfully deleted.");
    } else {
        //장소가 존재하지 않음.
        print_message("The place with ID " + PID + " does not exist.");
    }
}
```

[표 8.13 – "delete_place" 메소드 알고리즘 설계]

8.7.2 create_place 메소드

메소드 이름	create_place
메소드 기능	장소의 정보를 입력해 장소를 등록하는 메소드
알고리즘	

```
public void create_place(String PlaceName, String location) {//PlaceName: 장소 이름,
//location : 장소 위치, 주소
    // 같은 이름의 장소가 같은 위치에 존재하지 않는지 확인
    if (placeExists(PlaceName)) {
        // 이미 장소가 등록되어 있는 경우
        print_message(  PlaceName + " already exists.");
    } else {
        // 장소를 DB에 추가하여 등록
        addPlaceToDB(PlaceName, location);
        // 등록 완료 알림
        print_message("The place has been successfully added.");
    }
}
```

[표 8.14 – "create_place" 메소드 알고리즘 설계]

8.7.3 edit_place 메소드

메소드 이름	edit_place
메소드 기능	장소의 수정할 정보를 입력해 장소 정보를 업데이트하는 메소드
알고리즘	

```
public void edit_place(String PID, String newOne) {
    // DB에 등록된 장소 있는지 확인
    if (placeExists(PID)) {
        // 수정한 정보로 업데이트
        updatePlaceNameInDB(placeID, newOne);
        // 수정 완료 알림
        print_message("The place has been successfully updated.");
    } else {
        // DB에 등록된 장소 없음 알림.
        print_message("The place with ID " + placeID + " does not exist.");
    }
}
```

[표 8.15 – "edit_place" 메소드 알고리즘 설계]

8.7.4 get_place_info 메소드

메소드 이름	get_place_info
메소드 기능	특정 위치에 저장된 장소 정보를 가져오는 메소드
알고리즘	

```
public Place get_place_info(String location) {
    // 주어진 장소에 저장된 장소 있는지 확인
    Place place = getPlaceInfoFromDB(location);
    // 장소 정보 없으면 경우
    if (place == null) {
        print_message( "not found.");
    }
    // 장소 정보 리턴
    return place;
}
```

[표 8.16 – "get_place_info" 메소드 알고리즘 설계]

8.8 클래스 '지도'의 메소드 알고리즘 설계

8.8.1 show_marker 메소드

메소드 이름	show_marker
메소드 기능	카카오맵 상단 지정된 위치에 마커 아이콘을 표시하는 메소드
알고리즘	

```
// 카카오맵 상단에 마커 아이콘을 표시하는 메소드
public void show_marker(double latitude, double longitude) {
    // 카카오맵 뷰 생성
    mapView = new MapView();

    // 지도 중심 좌표 설정
    MapPoint mapPoint = MapPoint.mapPointWithGeoCoord(latitude, longitude);
    mapView.setMapCenterPoint(mapPoint, true);

    // 마커 생성
    marker = new Marker();
    marker.setItemName("마커");
    marker.setTag(0);
    marker.setMapPoint(mapPoint);

    // 마커 아이콘 설정
    MarkerImage markerImage =
        MarkerImage.defaultMarker(MarkerImage.ImageType.RED);
    marker.setImage(markerImage);

    // 마커 추가
    mapView.addMarker(marker);
}
```

[표 8.17 – "show_marker" 메소드 알고리즘 설계]

8.8.2 map_zoom_in 메소드

메소드 이름	map_zoom_in
메소드 기능	카카오맵의 시점을 확대시키는 메소드
알고리즘	

```
// 카카오맵의 시점을 확대시키는 메소드
public void map_zoom_in() {
    // 현재 줌 레벨 가져오기
    int currentZoomLevel = mapView.getMapZoomLevel();

    // 현재 줌 레벨에 1을 더하여 확대
    mapView.setMapZoomLevel(currentZoomLevel + 1, true);
}
```

[표 8.18 – "map_zoom_in" 메소드 알고리즘 설계]

8.8.3 map_zoom_out 메소드

메소드 이름	map_zoom_in
메소드 기능	카카오맵의 시점을 축소시키는 메소드
알고리즘	

```
// 카카오맵의 시점을 확대시키는 메소드
public void map_zoom_out() {
    // 현재 줌 레벨 가져오기
    int currentZoomLevel = mapView.getMapZoomLevel();

    // 현재 줌 레벨에 1을 더하여 확대
    mapView.setMapZoomLevel(currentZoomLevel - 1, true);
}
```

[표 8.19 – "map_zoom_in" 메소드 알고리즘 설계]

8.8.4 renew_current_location 메소드

메소드 이름	renew_current_location
메소드 기능	카카오맵의 시점을 현재 사용자의 GPS 상 위치한 곳으로 이동하는 메소드
알고리즘	

```
// 카카오맵의 시점을 현재 사용자의 GPS 상 위치한 곳으로 이동시키는 메소드
public void map_move_to_current_location() {
    if (currentLocation != null) {
        // 현재 위치의 위도와 경도를 이용하여 MapPoint 객체 생성
        MapPoint mapPoint = MapPoint.mapPointWithGeoCoord(currentLocation.latitude,
currentLocation.longitude);

        // MapPoint 객체를 사용하여 카카오맵 뷰의 중심 좌표를 설정하고, 줌 레벨을 설정하여
이동
        mapView.setMapCenterPointAndZoomLevel(mapPoint, 2, true);
    }
}

// 사용자의 현재 위치 정보를 갱신하는 메소드
public void updateCurrentLocation(LatLng location) {
    this.currentLocation = location;
}
```

[표 8.20 – "renew_current_location" 메소드 알고리즘 설계]

8.9 클래스 '즐겨찾기'의 메소드 알고리즘 설계

8.9.1 add_bookmark 메소드

메소드 이름	add_bookmark
메소드 기능	즐겨찾기 목록에 특정 장소에 대한 즐겨찾기를 추가하는 메소드
알고리즘	

```
// 장소 목록 리스트
private ArrayList<Place> placeList;

// 즐겨찾기로 추가하는 메소드
public void add_bookmark(Place placeId) {
    // 해당 장소가 이미 즐겨찾기에 있는지 확인
    if (isBookmarked(place)) {
        System.out.println("이미 즐겨찾기에 추가된 장소입니다.");
    } else {
        // 장소를 즐겨찾기에 추가
        place.setBookmark(true);
        addPlaceToBookmarkListDB(placeId);
        System.out.println(place.getName() + " 장소를 즐겨찾기에 추가하였습니다.");
    }
}
```

[표 8.21 – "add_bookmark" 메소드 알고리즘 설계]

8.9.2 delete_bookmark 메소드

메소드 이름	delete_bookmark
메소드 기능	즐겨찾기 목록에 특정 장소에 대한 즐겨찾기를 제거하는 메소드
알고리즘	

```
  // 장소 목록 리스트
private ArrayList<Place> placeList;

// 즐겨찾기를 제거하는 메소드
public void delete_bookmark(Place placeId) {
    // 해당 장소가 즐겨찾기에 있는지 확인
    if (isBookmarked(place)) {
        // 장소의 즐겨찾기 상태를 false로 변경
        place.setBookmark(false);
        deletePlaceToBookmarkListDB(placeId);
        System.out.println(place.getName() + " 장소의 즐겨찾기를 제거하였습니다.");
    } else {
        System.out.println("해당 장소는 즐겨찾기에 추가되어 있지 않습니다.");
    }
}
```

[표 8.22 – "delete_bookmark" 메소드 알고리즘 설계]

8.9.3 show_bookmark 메소드

메소드 이름	show_bookmark
메소드 기능	즐겨찾기 목록에 등록된 즐겨찾기 정보를 확인하는 메소드

알고리즘

```
// 장소 목록 리스트
private ArrayList<Place> placeList;

// 즐겨찾기 목록 리스트
private ArrayList<Place> bookmarkedList;

// 즐겨찾기 목록에 등록된 즐겨찾기 정보를 확인하는 메소드
public void show_bookmarked() {
    placeList = getBookmarkListInfoFromDB();
    // 즐겨찾기된 장소 정보 없으면 경우
    if (placeList == null) {
        print_message( "not found.");
        return;
    }
    // 즐겨찾기된 장소 장보 반환
    return placeList;
}
```

[표 8.23 – "show_bookmark" 메소드 알고리즘 설계]

8.10 클래스 '즐겨찾기 목록'의 메소드 알고리즘 설계

8.10.1 add_bookmark_list 메소드

메소드 이름	add_bookmark_list
메소드 기능	즐겨찾기 목록을 추가하는 메소드
알고리즘	

```
// 즐겨찾기 목록 정보를 저장하는 HashMap
private HashMap<String, String> bookmarkLists;

// 생성자
public BookmarkManager() {
    bookmarkLists = new HashMap<>();
}

// 즐겨찾기 목록을 추가하는 메소드
public void add_bookmark_list(String listName, String listId) {
    bookmarkLists.put(listName, listId);
    addBookmarkListToDB(listName ,listId);
    System.out.println("즐겨찾기 목록이 추가되었습니다.");
    System.out.println("목록명: " + listName);
    System.out.println("목록 아이디: " + listId);
}
```

[표 8.24 – "add_bookmark_list" 메소드 알고리즘 설계]

8.10.2 delete_bookmark_list 메소드

메소드 이름	delete_bookmark_list
메소드 기능	즐겨찾기 목록을 제거하는 메소드
알고리즘	

```
// 즐겨찾기 목록 정보를 저장하는 HashMap
private HashMap<String, String> bookmarkLists;

// 생성자
public BookmarkManager() {
    bookmarkLists = new HashMap<>();
}

// 즐겨찾기 목록을 삭제하는 메소드
public void delete_bookmark_list(String listId) {
    if (bookmarkLists.containsKey(listName)) {
        bookmarkLists.remove(listName);
        deleteBookmarkListFromDB(listId)
        System.out.println("즐겨찾기 목록이 삭제되었습니다.");
        System.out.println("목록명: " + listName);
    } else {
        System.out.println("해당 목록명의 즐겨찾기 목록이 존재하지 않습니다.");
    }
}
```

[표 8.25 – "delete_bookmark_list" 메소드 알고리즘 설계]

8.10.3 modify_bookmark_info_list 메소드

메소드 이름	modify_bookmark_info_list
메소드 기능	즐겨찾기 목록 정보를 변경하는 메소드
알고리즘	

```
// 즐겨찾기 목록 정보를 저장하는 HashMap
private HashMap<String, String> bookmarkLists;

// 생성자
public BookmarkManager() {
    bookmarkLists = new HashMap<>();
}

// 즐겨찾기 목록을 삭제하는 메소드
public void delete_bookmark_list(String listId) {
    if (bookmarkLists.containsKey(listName)) {
        bookmarkLists.remove(listName);
        deleteBookmarkListFromDB(listId)
        System.out.println("즐겨찾기 목록이 삭제되었습니다.");
        System.out.println("목록명: " + listName);
    } else {
        System.out.println("해당 목록명의 즐겨찾기 목록이 존재하지 않습니다.");
    }
}
```

[표 8.26 – "modify_bookmark_list" 메소드 알고리즘 설계]

8.10.4 show_bookmark_list 메소드

메소드 이름	show_bookmark_list
메소드 기능	즐겨찾기 목록 정보를 조회하는 메소드
알고리즘	

```
// 즐겨찾기 목록 정보를 저장하는 HashMap
private HashMap<String, String> bookmarkLists;

// 생성자
public BookmarkManager() {
    bookmarkLists = new HashMap<>();
}

// 즐겨찾기 목록을 확인하는 메소드
public void show_bookmark_lists(String listId) {
    System.out.println("즐겨찾기 목록:");
    if (bookmarkLists.isEmpty()) {
        System.out.println("등록된 즐겨찾기 목록이 없습니다.");
    } else {
        return showBookmarkListInfoFromDB(listId);
    }
}
```

[표 8.27 – "show_bookmark_list" 메소드 알고리즘 설계]