소프트웨어 설계 명세서(캡스톤 디자인)

2023. 11. 03

프로젝트명: 실시간 영상 분석을 통한 인원 밀집도 공유 시스템

팀 원: 임수연(팀장), 성열암, 김윤희

<table>
<tr><th colspan="3">작품 우수성(40%)</th><th colspan="4">작품의 기술성(40%)</th><th colspan="3">개발문서 완성도(20%)</th></tr>
<tr><td>평가기준</td><td>배점</td><td>점수</td><td colspan="2">평가기준</td><td>배점</td><td>점수</td><td>평가기준</td><td>배점</td><td>점수</td></tr>
<tr><td>작품 창의성이 높고 상품화 가능한 우수한 작품이고, 취업에 도움을 줄 수 있는 수준이다.</td><td>(36~40)</td><td rowspan="6"></td><td>필요한 핵심 기술을 자체 개발하여 적용 수준</td><td>작품 개발의 핵심 기술인 Cloud, 임베디드 시스템, AI/DB 엔진, SE, 보안/통신, 영상처리 등을 직접 개발하여 적용하였다.</td><td>(36~40)</td><td rowspan="6"></td><td>시스템 정의서, 요구사항명세서(SRS), 설계명세서(SDD), 소스 코드 작성 등의 내용이 잘 완성되었고, 그 품질이 아주 우수하다.</td><td>(16~20)</td><td rowspan="6"></td></tr>
<tr><td>졸업작품으로 완성도가 높고 취업에 도움을 줄 수 있는 수준이다.</td><td>(31~35)</td><td>작품의 기능성과 사용성이 우수한 수준</td><td>기존의 상용/공개 플랫폼에서 지원하는 기술을 적용하고, 작품의 기능성과 사용성이 우수하다.</td><td>(26~35)</td><td rowspan="4">시스템 정의서, 요구사항정의서, 요구사항명세서(SRS), 설계명세서(SDD) 등의 내용이 완성되었으나, 다소 보완이 필요하다.</td><td>(11~14)</td></tr>
<tr><td>작품이 고객 중심으로 개발되었고, 주제와 결과물이 일치하고 실행에 문제가 없다.</td><td>(26~30)</td><td>상용/공개 플랫폼에서 제공하는 방법 적용한 응용 시스템 개발 수준</td><td>작품 개발에 필요한 핵심 기술을 기존의 상용/공개 플랫폼에서 지원하는 기술을 적용하였다.</td><td>(21~25)</td></tr>
<tr><td>구현물의 실행은 되나, 평범하다.</td><td>(21~25)</td><td>단순 구현에 중점을 두어 완성한 수준</td><td>전공 교과에서 학습한 수준의 기술을 사용하였다.</td><td>(11~20)</td><td>소스 코드만을 생성하였고, 산출물의 내용 이해에 인내와 노력이 필요하다.</td><td>(6~10)</td></tr>
<tr><td>코드가 실행되지만, 구현의 노력이 미흡하고 개발범위의 완성도가 부족하다.</td><td>(11~20)</td><td colspan="2">특별히 사용한 것으로 판단되는 기술을 발견하기 어렵다.</td><td>(0~10)</td><td>작품에 대한 산출물의 이해가 불가능하여 받아들이기 어렵다.</td><td>(0~5)</td></tr>
<tr><td>구현물의 완성도가 매우 낮다. 예로, 코드 작성이 미완성이고, 컴파일 에러가 발생한다.</td><td>(0~10)</td></tr>
<tr><td>총 점(100점)</td><td colspan="9"></td></tr>
<tr><td>Open Source 사용 내역</td><td colspan="3">• YOLOv4
• DeepSORT
• TensorFlow</td><td>자체 개발 내용</td><td colspan="5">• - YOLOv4 및 DeepSORT 알고리즘을 통한 인원 계수 알고리즘
• - 안드로이드 애플리케이션 시스템
• - 모델 서버, Back-End 및 DB 시스템</td></tr>
</table>

[실시간 영상 분석을 통한 인원 밀집도 공유 시스템]

SW 설계 명세서

2023년 10월 30일

문서번호 : 2023-SW_설계_명세서_001

소　　속 : 충북대학교 소프트웨어학과

팀　　명 : 시나브로

팀　　원 : 김윤희, 성열암, 임수연

교　　수 : Aziz Nasridinov

제/개정 이력

버전	날짜	작성자 성명	제/개정사항	비 고
ver1.0	03/17	임수연, 김윤희, 성열암	초안 작성	
ver1.1	04/15	임수연, 김윤희, 성열암	메소드 작성 및 문서 내용 변경	
ver1.2	5/25	임수연, 김윤희, 성열암	최종설계안 작성	
ver1.3	10/14	임수연, 김윤희, 성열암	수정 기능 안 작성 및 문서 오탈자 수정	
ver1.4	10/15	임수연, 김윤희, 성열암	목차 오탈자 수정	
ver1.5	10/30	임수연, 김윤희, 성열암	Use-case Diagram 수정 및 평기지표 표지 추가	

목 차

1. 서 론

1.1 개발 목표

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

1.2 개발 내용

- **인원 밀집도 분석 모델 개발**
  영상을 분석하여 인원을 식별 및 카운트하는 모델을 개발

- **장소 정보관리 기능 개발**
  장소를 등록, 삭제, 수정, 조회 및 장소 검색 등의 기능을 개발

- **리워드 시스템 개발**
  영상을 업로드하여 영상 분석 수행한 회원에게 보상하는 기능 개발

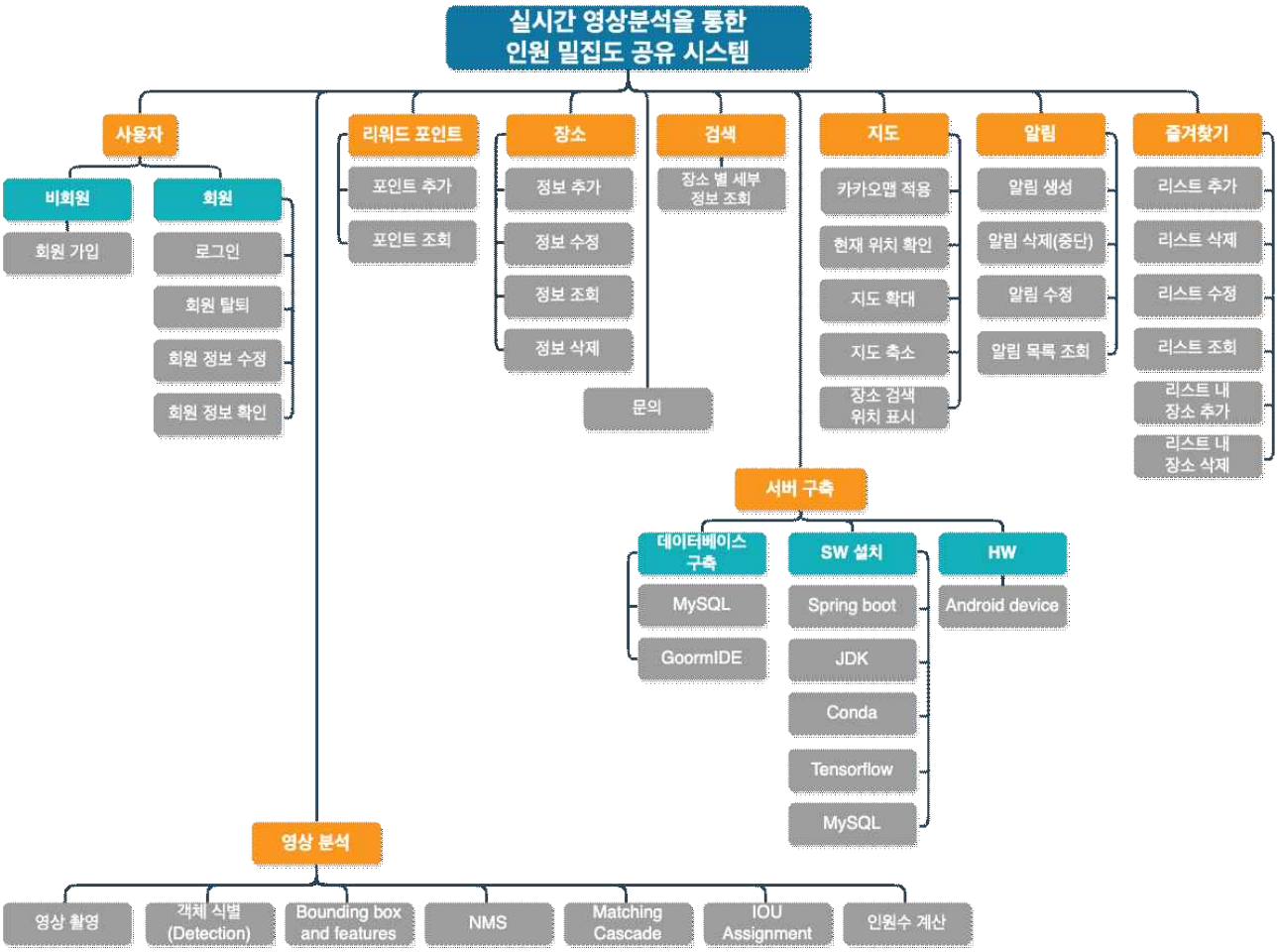
- **즐겨찾기 기능 개발**
  즐겨찾기를 등록, 삭제, 조회하는 기능과 즐겨찾기 목록을 등록, 삭제, 수정, 조회하는 기능 등을 개발

- **지도 UI 및 기능 개발**
  지도를 나타내고, 확대/축소/현위치 이동/장소 마커 출력 등의 기능 개발

- **알림 기능 개발**
  홍보 알림, 장소 등록 알림, 즐겨찾기 등록 알림, 인원수 공유 내용 알림과 관련된 기능 개발

- **사용자 관리 기능 개발**
  회원 등록, 조회, 수정, 삭제 등과 관련된 기능을 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 영상처리 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. 기능적 요구사항

No.	ID	Name	Description
1	-	회원관리	본 기능은 회원별 고유 아이디를 부여하고 사용자의 활동 내용과 범위를 관리하는 기능을 수행한다.
1.1	FR-001	회원 등록	비회원 사용자는 이름, 전화번호, 이메일, 아이디, 비밀번호를 통해 시스템에 등록한다.
1.2	FR-002	회원 탈퇴	회원은 본인 계정을 삭제할 수 있다.
1.3	FR-003	관리자 회원 정보 조회	관리자는 회원을 조회할 수 있다.
1.4	FR-004	회원 정보 조회	회원은 회원 정보를 조회할 수 있다.
1.5	FR-005	회원 개인정보 수정	회원은 본인의 개인정보를 수정할 수 있다.
1.6	FR-006	회원 권한 정보 수정	관리자는 회원의 접근 범위, 권한 등을 수정할 수 있다.
2	-	영상	사용자가 카메라로 주변 환경을 스캔하여 영상을 업로드 한 뒤, 인원 밀집 상태(인원수) 데이터를 수집, 분석하여 공유하는 기능이다.
2.1	FR-007	카메라	시스템은 카메라를 켤 수 있다.
2.2	FR-008	사람 객체 탐지	시스템은 카메라 스캔을 통해 사람 객체를 탐지해야 한다.
2.3	FR-009	사람 객체 인식(구별)	시스템은 카메라 스캔에서 탐지한 사람 객체를 각각 구별(인식)할 수 있어야 한다.
2.4	FR-010	사람 객체별 고유 번호 부여	시스템은 카메라 스캔에서 탐지한 사람 객체별로 고유 번호를 부여한다.
2.5	FR-011	사람 수 카운트	시스템은 탐지한 사람 객체의 수의 총합을 계산해야 한다.

2.6	FR-012	스캔 액션	시스템은 스캔 과정 중에 화살표 등으로 사용자의 액션 방향을 지시한다.
2.7	FR-013	스캔 완료	사용자는 카메라 스캔이 완료되었음을 시스템에 알릴 수 있어야 한다.
2.8	FR-014	인원 밀집 상태(인원수) 공유	사용자는 카메라 스캔을 통해 사용자가 현재 위치한 장소의 인원수를 공유할 수 있다.
2.9	FR-015	기타 공유	사용자는 혼잡도, 메모, 정확성 등 인원 밀집 상태(인원수) 정보 외에 부가적인 정보를 공유할 수 있다.
2.10	FR-016	공유 수정	사용자는 인원 밀집 상태(인원수) 정보를 수정할 수 있다.
3	-	알림	-
3.1	FR-017	이용 촉구(홍보) 알림	시스템은 사용자에게 장소의 인원 밀집 상태(인원수) 공유를 촉진하는 홍보 알림을 보낸다.
3.2	FR-018	새 장소 등록 알림	시스템은 새 장소를 등록한 사용자에게 등록 완료 알림을 보낸다.
3.3	FR-019	즐겨찾기 알림	시스템은 즐겨찾기를 추가한 사용자에게 등록 완료 알림을 보낸다.
3.4	FR-020	공유 내용 알림	시스템은 사용자가 즐겨찾기에 등록한 장소에 인원이 공유되면 알림을 보낸다.
4	FR-021	지도	시스템은 지도를 확대, 축소할 수 있어야 한다.
4.1	FR-022	지도표시	시스템은 지도를 띄울 수 있어야 한다.
4.2	FR-023	인원 공유 가능 장소	시스템은 인원 공유가 가능한 장소의 이름과 마커를 지도상에 표시할 수 있다.
4.3	FR-024	공유 내용 출력	시스템은 공유된 인원수를 지도상에 출력한다.
4.4	FR-025	시각화	시스템은 인원 밀집 상태(인원수) 정보를 기반으로 밀집 정도를 시각화한다.
5	-	위치	-

5.1	FR-026	현재 위치	시스템은 사용자의 현재 위치를 파악할 수 있다.
5.2	FR-027	공유 장소 확인	시스템은 사용자의 현재 위치와 공유 장소가 같은 곳인지 확인한다.
5.3	FR-028	현재 위치 이동	시스템은 사용자의 현재 위치로 지도의 시점을 변경할 수 있다.
6	FR-029	즐겨찾기	사용자는 원하는 장소를 즐겨찾기 등록, 삭제, 조회할 수 있다.
7	FR-030	장소 관리	사용자는 원하는 장소를 등록, 수정, 삭제, 조회할 수 있다.
8	FR-031	포인트 리워드	시스템은 공유를 진행한 회원에게 포인트를 줄 수 있다.
9	FR-032	사용법 안내 (튜토리얼)	시스템은 사용자에게 시스템 사용방법 가이드를 제공해야 한다.
10	FR-033	검색	사용자는 장소를 검색할 수 있다.
10.1	FR-034	검색 상세 내용	사용자는 검색한 장소 목록 중 장소에서 장소 상세 내용을 확인할 수 있다.
11	-	문의	사용자는 원하는 내용을 문의할 수 있다.
11.1	FR-035	이메일 전송	사용자가 문의한 내용을 1대1로 이메일로 전송한다.
12	-	회원	시스템에 가입한 회원 사용자이다.
12.1	FR-036	로그인	사용자는 회원가입 시 사용하였던 이메일과 비밀번호를 이용하여 시스템에 로그인할 수 있다.
12.2	FR-037	로그아웃	애플리케이션에 로그인된 상태에서 비회원 상태로 전환한다.
12.3	FR-038	정보 수정	회원은 본인의 회원 정보(프로필 이미지, 이름, 이메일, 비밀번호)를 수정할 수 있다.
12.4	FR-039	탈퇴	시스템으로부터 개인의 모든 정보들을 제거하여 비회원으로 전환한다.

12.5	FR-040	비밀번호 찾기	회원은 등록된 이메일을 통하여 새로운 비밀번호를 발급받을 수 있다.
12.6	FR-041	알림 설정	회원은 (공지사항, 추천/혜택, 야간 광고성, 즐겨찾기 장소 인원수 변동)에 대한 알림(notification)을 활성화 하여 수신할 수 있다.
12.7	FR-042	즐겨찾기	회원은 특정 장소에 대해 리스트 단위로 해당 정보를 추가/제거/조회할 수 있고, 그 리스트는 추가/제거/수정할 수 있다..
12.8	FR-043	장소 정보 제어	회원은 특정 위치에 대한 장소 정보를 추가/삭제/수정한다.
12.9	FR-044	장소 정보 확인	회원은 특정 위치에 대해 등록/미등록된 장소 정보를 확인한다.
12.10	FR-045	장소 검색	사용자는 검색란을 이용하여 장소를 조회할 수 있다.
12.11	FR-046	스캔	회원은 스캔 버튼을 이용하여 카메라로 촬영한 후 업로드할 수 있다.
13	-	비회원	시스템에 가입하지 않은 사용자이다.
13.1	FR-047	장소 정보 확인	사용자는 특정 위치에 대해 등록/미등록된 장소 정보를 확인한다.
13.2	FR-048	장소 검색	사용자는 검색란을 이용하여 장소를 조회할 수 있다.
14	-	관리자	시스템에 가입한 회원들의 정보를 관리하는 사용자이다.
14.1	FR-049	회원 정보 제어	시스템에 가입된 회원들의 정보를 조회 및 제거한다.

[표 2 – 기능적 요구사항]

3. 비기능적 요구사항

3.1 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 3.1 – 성능 요구사항]

3.2 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.
3	SR-003	보안 요구사항	시스템은 악의적인 사용자에 의하여 데이터베이스 정보의 추가/삭제/수정이 이뤄지지 않도록 제한한다.
4	SR-004	보안 요구사항	시스템은 의도치 않은 데이터의 손실 및 변경이 발생하지 않도록 서비스를 제공해야 한다.
5	SR-005	보안 요구사항	관리자 회원은 데이터베이스의 접근/관리 권한을 부여받아 시스템 데이터베이스를 제어할 수 있어야 한다.
6	SR-006	보안 요구사항	애플리케이션을 통한 회원가입 시 이메일 입력란에 추가되는 이메일은 반드시 이메일 인증을 통하여 확인 검증이 완료되어야 한다.
7	SR-007	보안 요구사항	애플리케이션 내에서 로그인 시 입력되는 비밀번호는 회원가입 시 사용되었던 SHA-256 알고리즘 및 Salt 문자열을 적용하여 암호화된 형태로 Request Body에 담겨야 한다.
8	SR-008	보안 요구사항	시스템은 애플리케이션 내에서 관리자 계정으로 로그인한 사용자의 경우 관리자 액티비티에 대한 접근 권한을 부여해야 한다.

[표 3.2 – 보안 요구사항]

3.3 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 3.3 – 신뢰성 요구사항]

3.4 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 회원가입된 개별 사용자의 정보를 조회할 수 있어야 한다.
5	DR-005	데이터베이스 요구사항	시스템에 회원가입된 전체 사용자의 정보를 조회할 수 있어야 한다.
6	DR-006	데이터베이스 요구사항	시스템에 회원가입된 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 3.4 – 데이터베이스 요구사항]

3.5 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 3.5 – SW 이식성 요구사항]

3.6 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 3.6 – 유지보수 요구사항]

3.7 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 3.7 – HW, SW 및 통신 요구사항]

3.8 동시성 요구사항

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

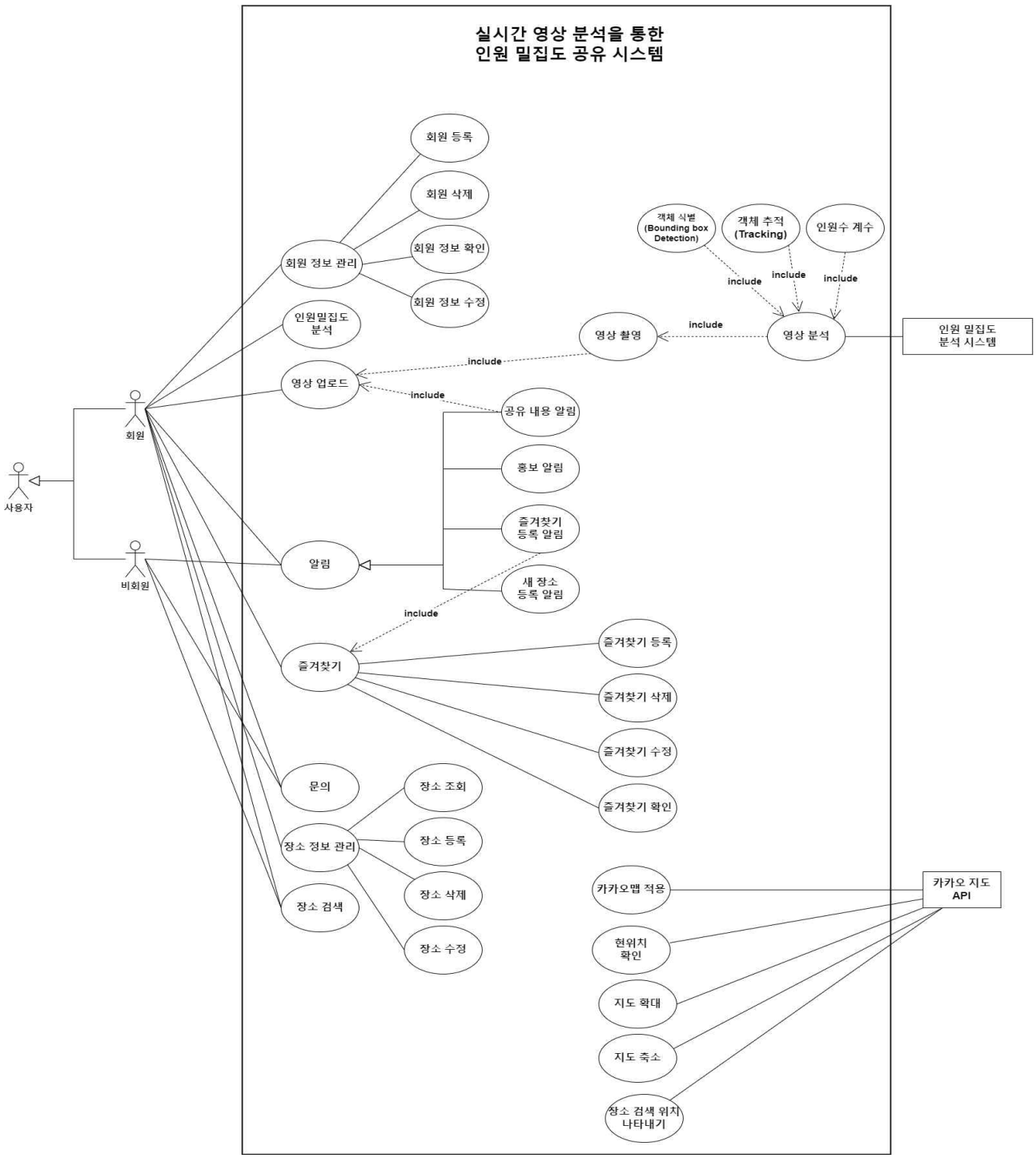
[표 3.8 – 동시성 요구사항]

3.9 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, 검색창)을 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 추가/삭제/수정되는 이벤트가 발생하였을 때 사용자에게 수행했던 작업에 대한 피드백을 실시간으로 전달할 수 있도록 확인 관련 안내 문구와 함께 화면의 하단에 Toast Message를 표시하여야 한다.

[표 3.9 – 사용자 인터페이스 요구사항]

4. 기능적 모델링 : Use–Case 다이어그램



[그림 4.1 – Use-Case 다이어그램]

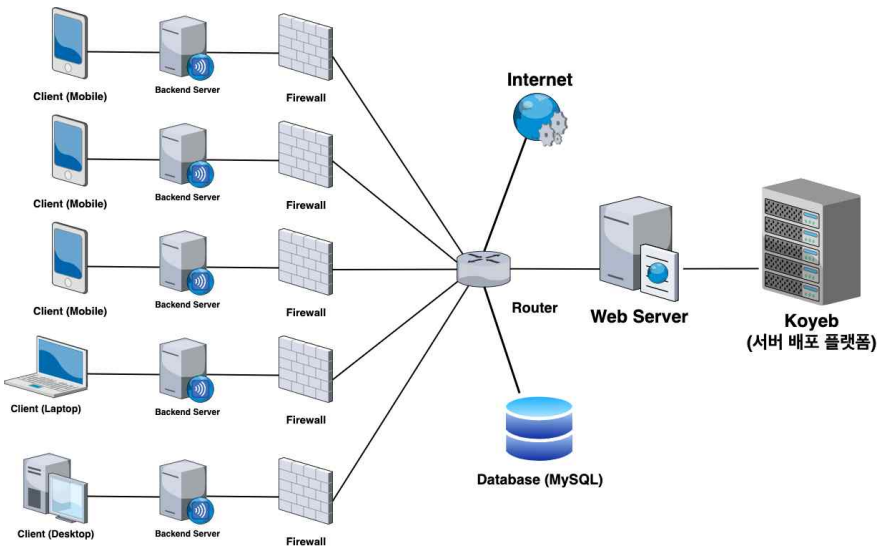
5. 아키텍처 설계

5.1 HW 아키텍처 설계

Specification	Standard Client	Standard Web Server	Standard Databse Server
Hardware	•2340x1080(FHD+) resolution • 10MP(화소) • 6 - 8GB memory • 128GB storage	• 32GB memory • 1TB disk drive • intel core i7	• 2048MB memory • 10GB disk drive

[표 5.1 – HW 아키텍처]

5.2 NW 아키텍처 설계



[그림 5.2 – NW 아키텍처]

Specification	Standard Client	Standard Web Server	Standard Databse Server
Network	wifi	100Mbps Ethernet	100Mbps Ethernet

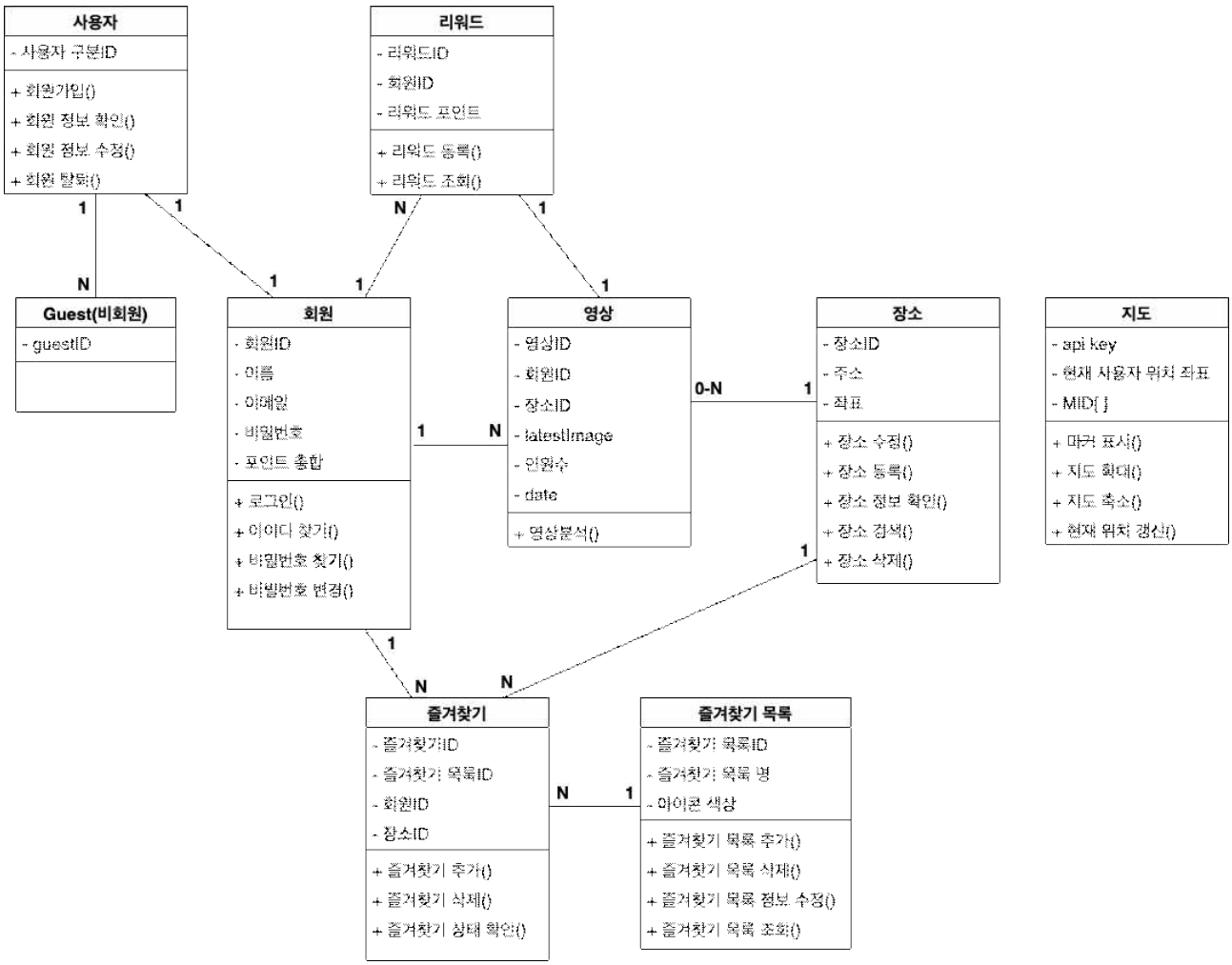
5.3 SW 아키텍처 설계

Specification	Standard Client	Standard Web Server	Standard Databse Server
Operating System	Android	Window	Window
Software	Android Studio	Node.js	MySQL

[표 5.3 – SW 아키텍처]

6. 구조적 모델링

6.1 CRC Card 및 클래스 명세



[그림 6.1 – 클래스 다이어그램

6.2 CRC Card 및 클래스 명세

6.2.1 사용자(Person)

속성	속성명	자료형	크기	제약조건	Collaborating Classe(s)
사용자 구분ID	id	boolean	1	default=0	비회원, 회원

Operations:	signUp(); // 회원가입
	read(); // 회원 정보 확인
	update(); // 회원 정보 수정
	delete(); // 회원 탈퇴				

명세	abstract class Person {
	private int id;
	signUp();
	read();
	update();
	delete();
	};				

[표 6.1 – "사용자" 클래스 명세]

6.2.2 회원(User)

속성	속성명	자료형	크기	제약조건	Collaborating Classe(s)
회원ID	uId	int	10	PK	사용자, 리워드, 즐겨찾기
이름	name	char	20	NN	
이메일	email	char	25	NN	
비밀번호	pwd	char	20	NN	
포인트 총합	reward	int	10	default=0	

Operations:	login(email, pwd); // 로그인
	findId(); // 아이디 찾기
	findPwd(email); // 비밀번호 찾기
	changePwd(); // 비밀번호 변경				

명세	class User extends Person {
	private int uId;
	private int reward;
	private char email[25];
	private char pwd[20];
	private char name[20];
	private char tel[15];
	login(email, pwd);
	findId();
	findPwd();
	changePwd();
	}				

[표 6.2 – "회원" 클래스 명세]

6.2.3 비회원(Guest)

속성	속성명	자료형	크기	제약조건	Collaborating Classe(s)
비회원ID	guestId	int	10	PK	사용자
Operations:					

명세	class NonUser extends Person {
	private int tId;
	}				

[표 6.3 – "비회원" 클래스 명세]

6.2.4 리워드(Reward)

속성	속성명	자료형	크기	제약조건	Collaborating Classe(s)
리워드ID	rewardId	int	10	PK	영상, 회원
회원ID	uId	int	10	FK	
리워드 포인트	point	int	10	-	

Operations:	create(); // 리워드 포인트 등록
	read(); // 리워드 조회				

명세	class Reward implements Temp {
	private int serialNo;
	private int imgId;
	private int point;
	read();
	create();
	}				

[표 6.4 – "리워드" 클래스 명세]

6.2.5 즐겨찾기(Bookmark)

속성	속성명	자료형	크기	제약조건	Collaborating Classe(s)
즐겨찾기ID	bId	int	10	PK	즐겨찾기 목록, 회원, 장소
즐겨찾기 목록 ID	bListId	int	10	FK	
회원ID	uId	int	10	FK	
장소ID	pId	int	10	FK	

Operations:	create(); // 즐겨찾기 추가
	read(); // 즐겨찾기 확인
	delete(); // 즐겨찾기 삭제				

명세	class Bookmark implements Temp {
	private int bId;
	private int bListId;
	private int pId;
	create();
	read();
	delete();
	}				

[표 6.5 – "즐겨찾기" 클래스 명세]

6.2.6 즐겨찾기 목록(BookmarkList)

속성	속성명	자료형	크기	제약조건	Collaborating Classe(s)
즐겨찾기 목록ID	bListId	int	10	PK	즐겨찾기
즐겨찾기 목록명	bListName	char	10	NN	
아이콘 색상	iconColor	char	10	NN	

Operations:	create(); // 즐겨찾기 목록 추가
	read(); // 즐겨찾기 목록 조회
	update(); // 즐겨찾기 목록 정보 수정
	delete(); // 즐겨찾기 목록 삭제				

명세	class BookmarkList implements Temp {
	private int bListId;
	private int uId;
	private char bListName[10];
	private char iconColor[10];
	create();
	read();
	delete();
	update();
	}				

[표 6.6 – "즐겨찾기 목록" 클래스 명세]

6.2.7 장소(Places)

속성	속성명	자료형	크기	제약조건	Collaborating Classe(s)
장소ID	pId	int	10	PK	즐겨찾기, 영상
주소	address	String	-	-	
좌표	location	String	-	-	

Operations:	create(); // 장소 등록
	read(); // 장소 정보 확인
	update(); // 장소 수정
	delete(); // 장소 삭제
	search(); // 장소 검색				

명세	class Places implements Temp {
	private int pId;
	private int bId;
	private String address;
	private String location;
	create();
	read();
	update();
	delete();
	search();
	}				

[표 6.7 – "장소" 클래스 명세]

6.2.8 지도(Map)

속성	속성명	자료형	크기	제약조건	Collaborating Classe(s)
api key	apiKey	char	150	PK	
현재 사용자 위치 좌표	currentUserLoc	String	-	NN	
MID[ ]	mId	String	-	-	

Operations:	makeMarker(); // 마커 표시
	zoomIn(); // 지도 확대
	zoomOut(); // 지도 축소
	currentLocation(); // 현재 위치 갱신				

명세	class Map {
	private char apiKey[150];
	private String currentUserLoc;
	private String mId;
	makeMarker();
	zoomIn();
	zoomOut();
	renewCurrentLocation();
	}				

[표 6.8 – "지도" 클래스 명세]

6.2.9 영상(Image)

속성	속성명	자료형	크기	제약조건	Collaborating Classe(s)
영상ID	imgId	int	10	PK	회원, 장소, 리워드
회원ID	uId	int	10	FK	
장소ID	pId	int	10	FK	
latestImage	latestImage	blob	20	NN	
인원수	peopleNum	int	12	-	
date	date	Date	-	-	
Operations:	analyzeImage(); // 영상 분석				

명세	class Image {
	private int imgId;
	private int uId;
	private int pId;
	private Object latestImage;
	analyzeImage()();
	}				

[표 6.9 – "영상" 클래스 명세]

6.3 데이터베이스 스키마 테이블 명세

6.3.1 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 총합	pointscore	integer	4	default=0
프로필 사진	file_name	blob	20	

[표 6.13 – "User" 스키마 테이블 명세]

6.3.2 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 6.14 – "Bookmark" 스키마 테이블 명세]

6.3.3 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 6.15 – "Label" 스키마 테이블 명세]

6.3.4 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 6.16 – "Places" 스키마 테이블 명세]

6.3.5 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 6.17 – "Headcount" 스키마 테이블 명세]

6.3.6 Reward

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

[표 6.18 – "Reward" 스키마 테이블 명세]

6.4 데이터구조 설계

(해당 사항 없음)

6.5 Interface 및 Abstract 클래스 설계

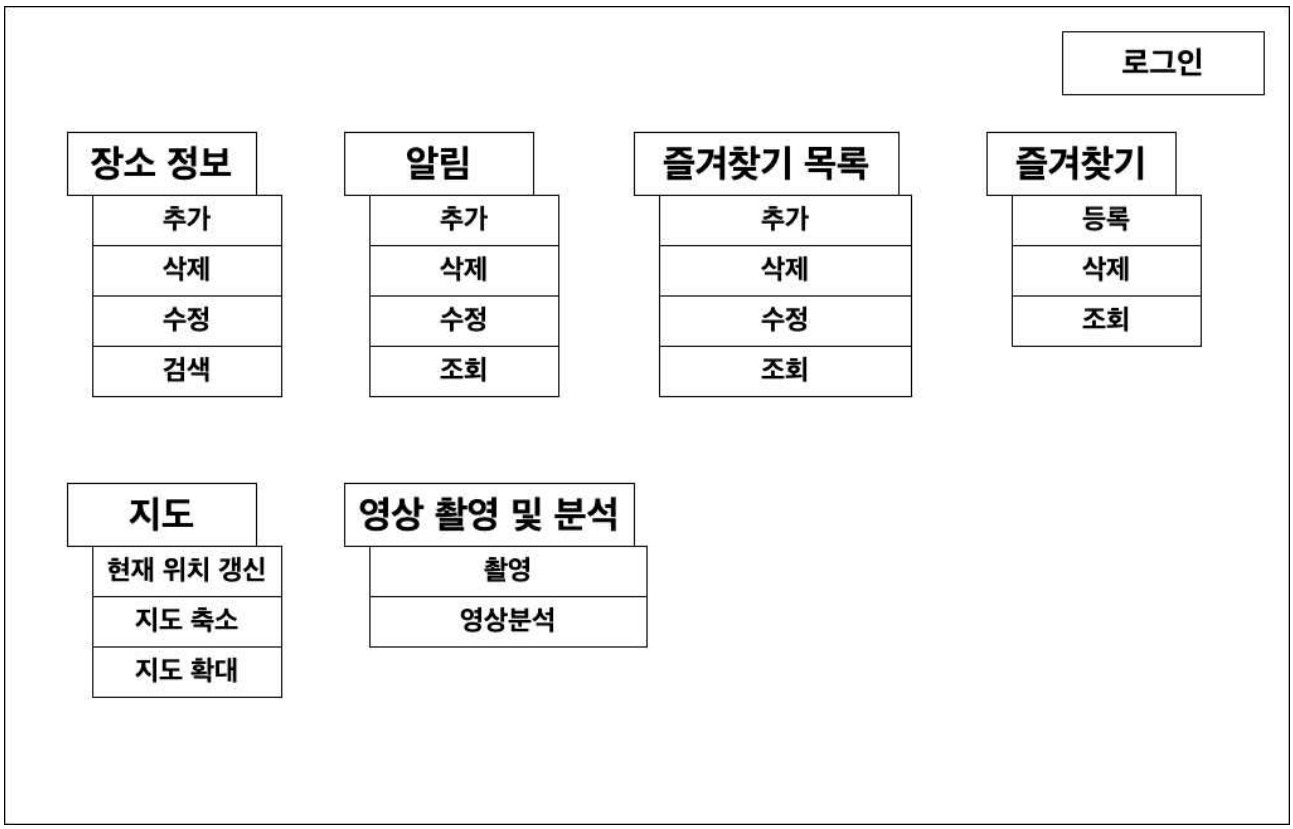
```
interface class Temp() {
  insert();
  delete();
  update();
  retrieve();
}

abstract class Person {
    int id;
    signUp();
    read();
    update();
    delete();
};

class Reward implements Temp {}
class Bookmark implements Temp {}
class BookmarkList implements Temp {}
class Place implements Temp {}
class HeadCount implements Temp {}

class User extends Person {}
clsss NonUser extends Person {}
```

7. 사용자 인터페이스(UI) 설계

7.1 관리자 사용자 인터페이스 명세

(해당 사항 없음)

7.2 일반 사용자 인터페이스 설계



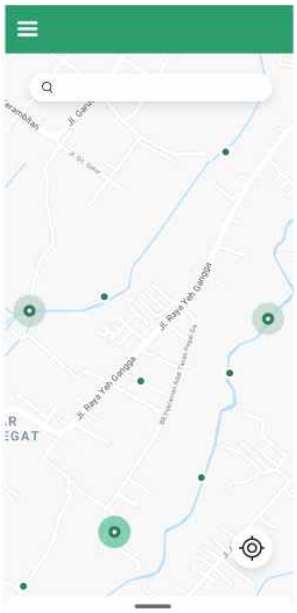
7.3 App용 사용자 인터페이스 설계

- App 시작

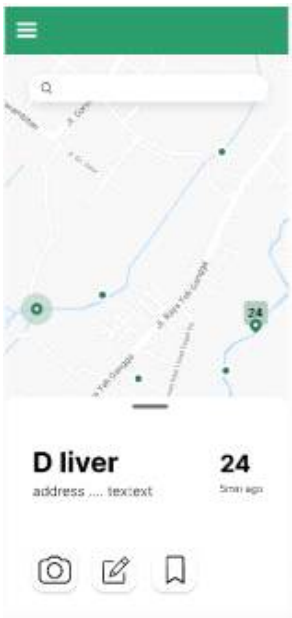
1. 시작화면

2. 튜토리얼(앱 사용법 안내)

3. 메인 시작 화면

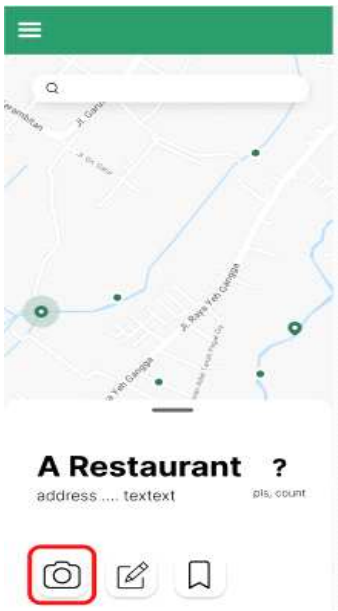


- 장소별 상세 정보 확인(하단 시트)

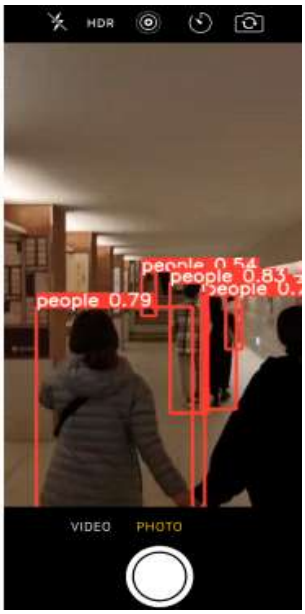


- 영상 등록 및 인원수 계수

1. 카메라 버튼 클릭

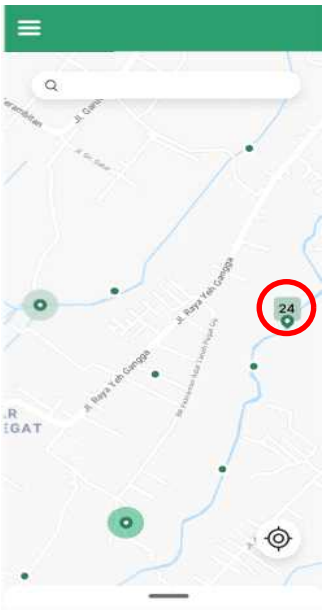


2. 카메라 스캔



3. 인원 계수 결과

4. 지도에 계수 결과 표시

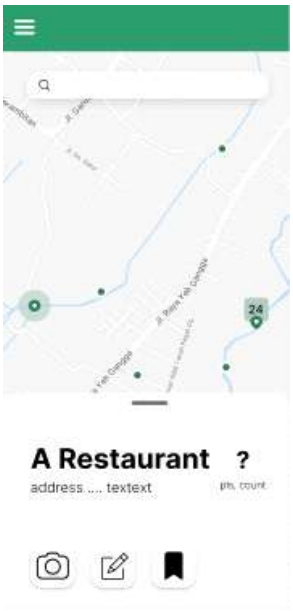


5.

6.

- 즐겨찾기

1. 즐겨찾기 버튼 클릭

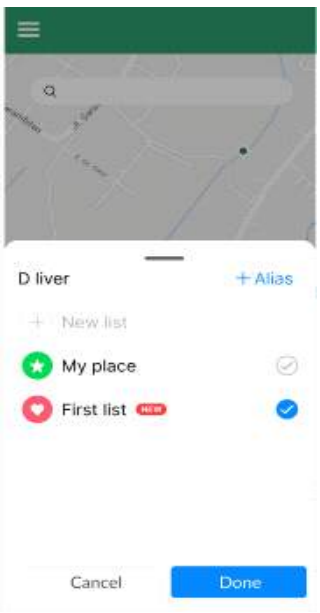


2. 즐겨찾기 리스트 목록 확인

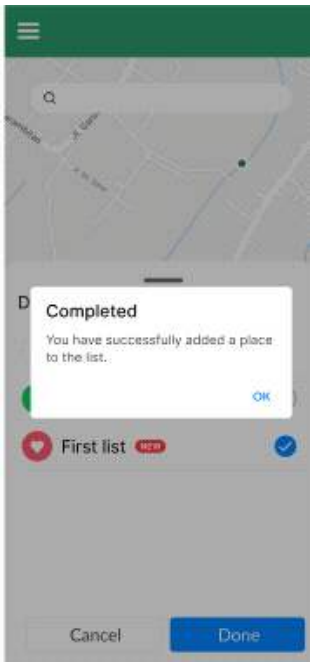
3.1. 즐겨찾기 목록 선택

3.2.1. 즐겨찾기 목록 생성

3.2.2. 즐겨찾기 목록 선택



4. 즐겨찾기 추가 완료 메시지

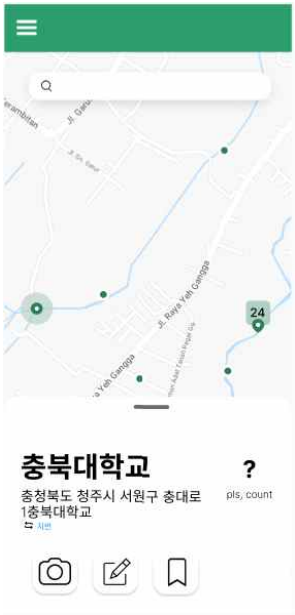


- 미등록장소 추가 등록

1. 등록 버튼 클릭

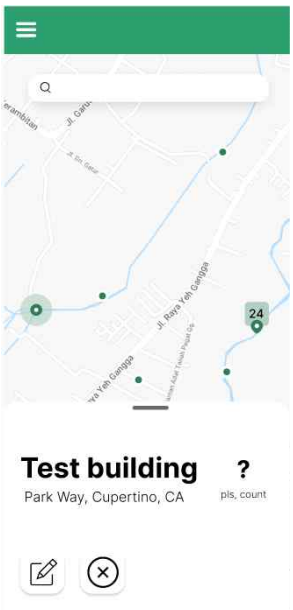
2. 주소 및 상세 정보 입력

3.등록 완료

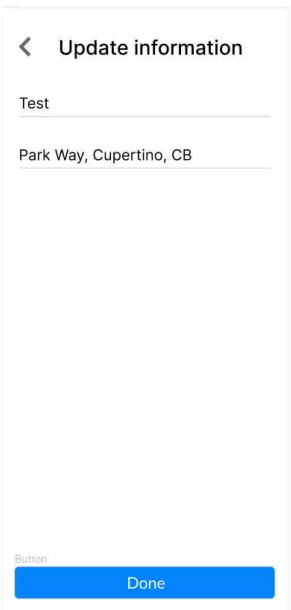


- 장소 정보 수정

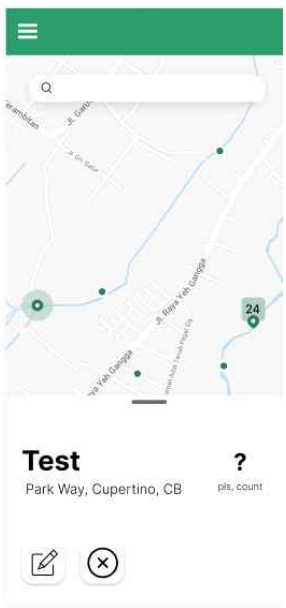
1. 수정 버튼 클릭



2. 내용 수정

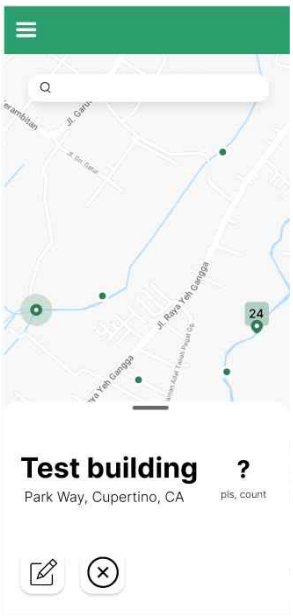


3. 수정 완료

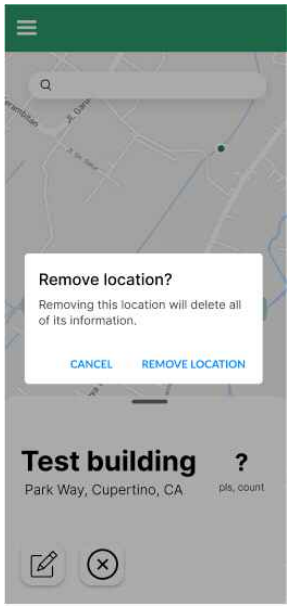


- 장소 삭제

1. 삭제 버튼 클릭



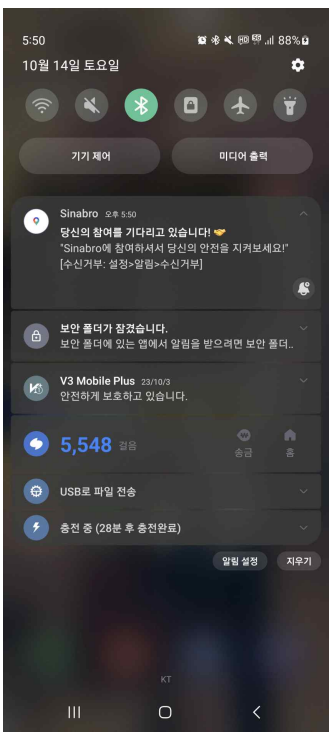
2. 삭제 재확인



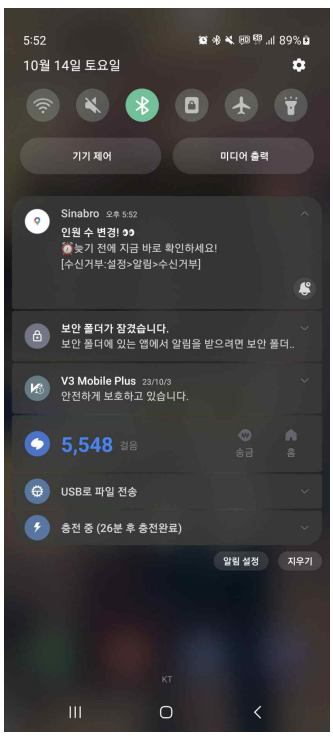
3.삭제 완료

- 푸쉬 알림

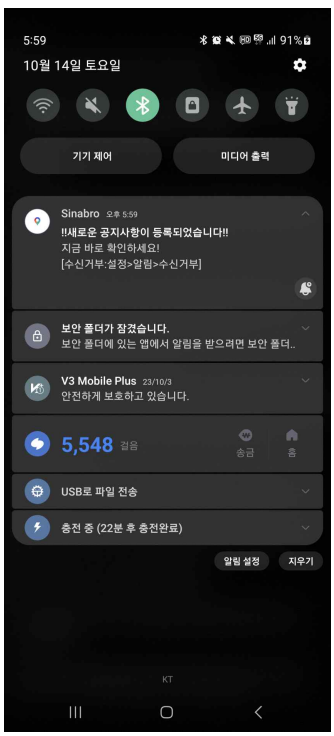
1. 광고성 Push



2. (인원수, 장소) Update Push



3. 공지사항 Push



8. 행위 모델링: 시퀀스 다이어그램에 대한 알고리즘 설계

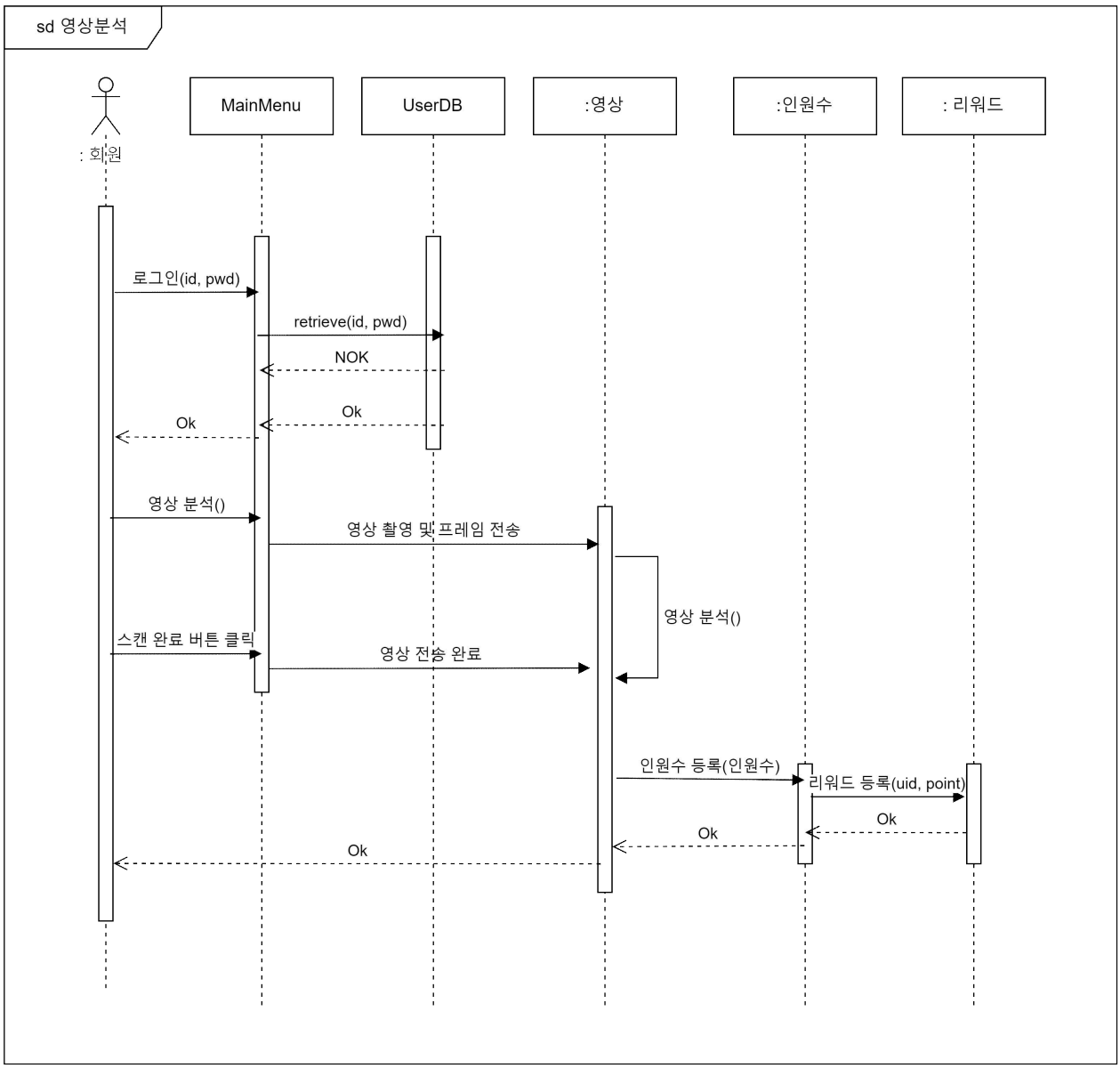
8.1 Use-Case '영상분석'에 대한 시퀀스 다이어그램의 알고리즘 설계

메소드 이름	imageAnalyze
메소드 기능	촬영된 영상을 통해 인원수 데이터를 분석 및 리워드를 제공하는 메소드

알고리즘

```
class ImageAnalysisService {
    private DBService dbService;
    String url = "jdbc:mysql://your_host:your_port/Label";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    public void imageAnalyze(Object image) {
        try {
            dbConnection = DriverManager.getConnection(url, user, password);
            if (dbConnection != null) {
                System.out.println("데이터베이스 연결 성공");
                // 분석을 위해 영상을 서버로 전송하는 코드 작성
                int numberOfPeople = extractNumberOfPeopleFromImageData(image);
                // 리워드 계산 및 부여
                dbService.addRewards(calculateRewards(numberOfPeople));
            }
        } catch (SQLException e) { e.printStackTrace(); }
    }
    private void extractNumberOfPeopleFromImageData(Object image) {
        // 이미지 데이터로부터 인원수를 추출하는 로직을 작성
    }
    private void calculateRewards(int numberOfPeople) {
        // 리워드 계산 로직을 작성
        return 0;
    }
}
```

8.1.1 메뉴 '영상분석'의 서브메뉴 '영상 촬영 및 분석'에 대한 시퀀스 다이어그램



[그림 8.1 – "영상분석" 시퀀스 다이어그램 명세]

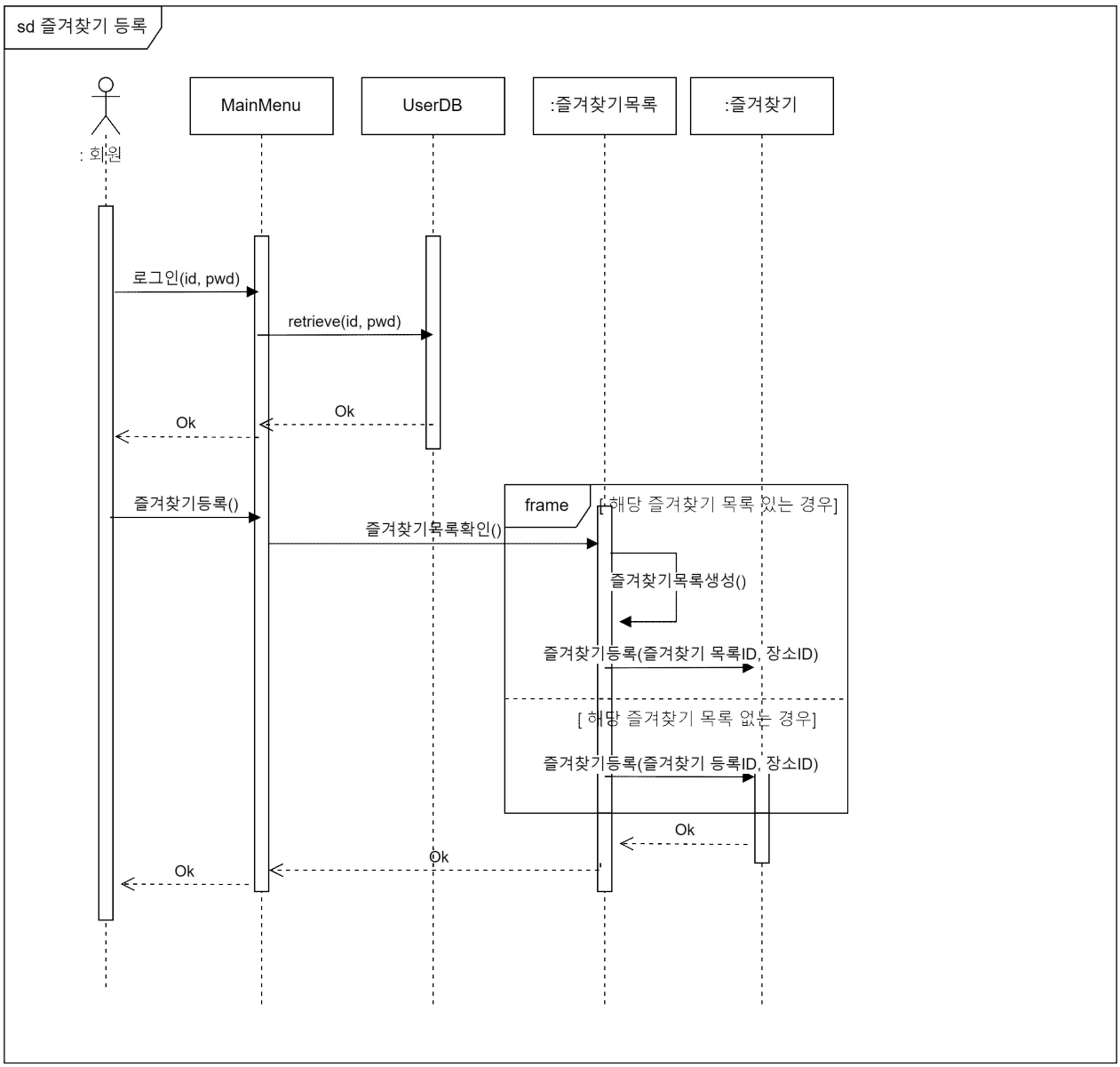
8.2 Use-Case '즐겨찾기 등록'에 대한 시퀀스 다이어그램의 알고리즘 설계

메소드 이름	addBookmark
메소드 기능	즐겨찾기 목록 내 특정 장소에 대한 즐겨찾기 정보를 추가하는 메소드

알고리즘

```
public void addBookmark(String bookmark) {
    // DB 연결
    String url = "jdbc:mysql://your_host:your_port/Label";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
            // "INSERT INTO bookmarks (bookmark) VALUES (?)"
            PreparedStatement ps = dbConnection.prepareStatement("INSERT INTO
bookmarks (bookmark) VALUES (?)");
            // PreparedStatement에 bookmark 값을 설정
            ps.setString(1, bookmark);
            // 쿼리 실행
            ps.executeUpdate();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        if (dbConnection != null) {
            try {
                dbConnection.close();
            } catch (SQLException e) { e.printStackTrace(); }
        }
    }
}
```

8.2.1 메뉴 '즐겨찾기'의 서브메뉴 '추가'에 대한 시퀀스 다이어그램



[그림 8.2 – "즐겨찾기 등록" 시퀀스 다이어그램 명세]

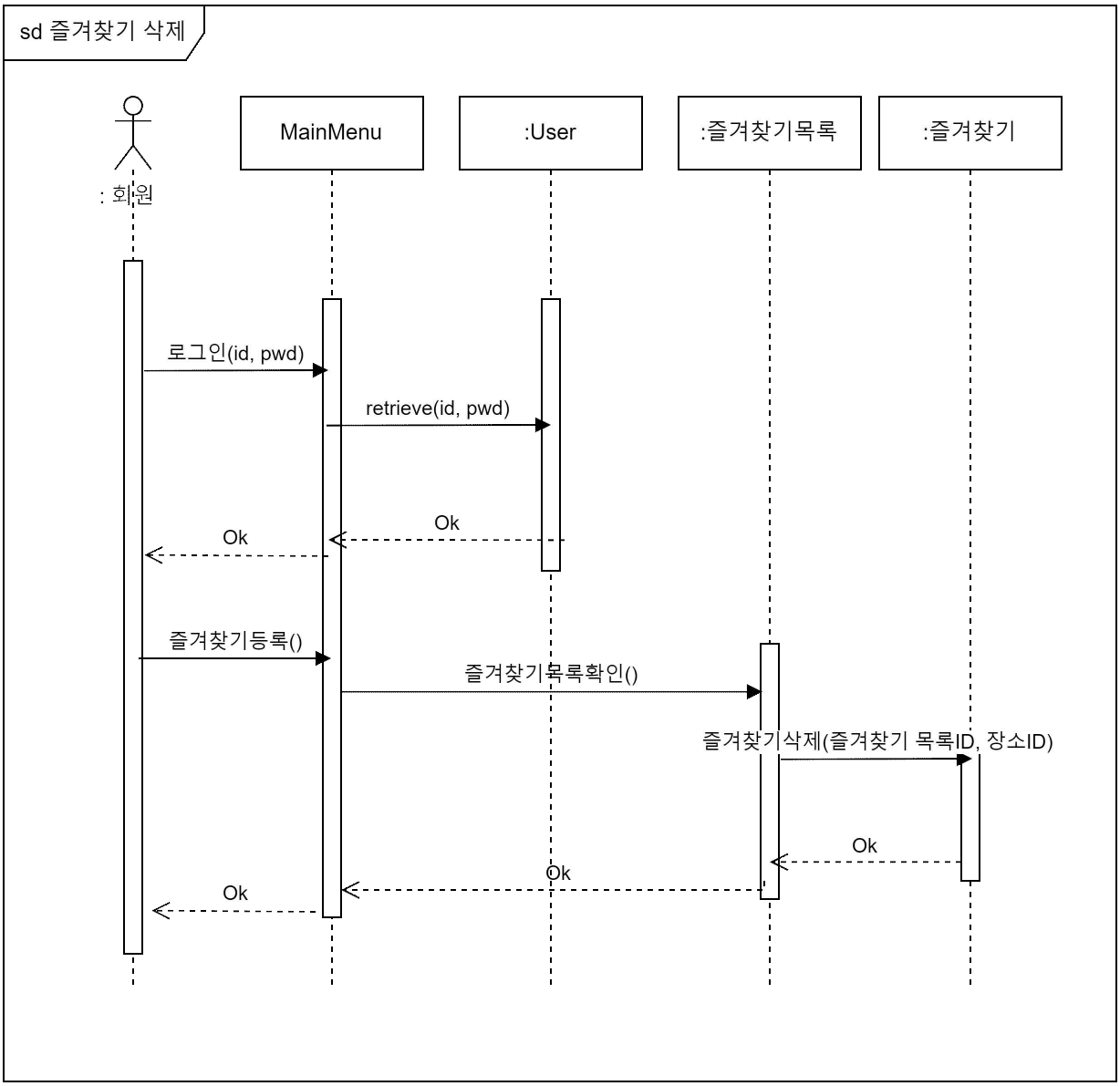
8.3 Use-Case '즐겨찾기 삭제'에 대한 시퀀스 다이어그램의 알고리즘 설계

메소드 이름	deleteBookmark
메소드 기능	즐겨찾기 목록 내 등록된 장소에 대한 즐겨찾기 정보를 삭제하는 메소드

알고리즘

```
public void deleteBookmark(String bookmark) {
    // DB 연결
    String url = "jdbc:mysql://your_host:your_port/Label";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
            // "DELETE FROM bookmarks WHERE bookmark = ?"
            PreparedStatement ps = dbConnection.prepareStatement("DELETE FROM
bookmarks WHERE bookmark = ?");
            // PreparedStatement에 bookmark 값을 설정
            ps.setString(1, bookmark);

            // 쿼리 실행
            ps.executeUpdate();
        }
    } catch (SQLException e) { e.printStackTrace(); } finally {
        if (dbConnection != null) {
            try {
                dbConnection.close();
            } catch (SQLException e) { e.printStackTrace(); }
        }
    }
}
```

8.3.1 메뉴 '즐겨찾기'의 서브메뉴 '삭제'에 대한 시퀀스 다이어그램



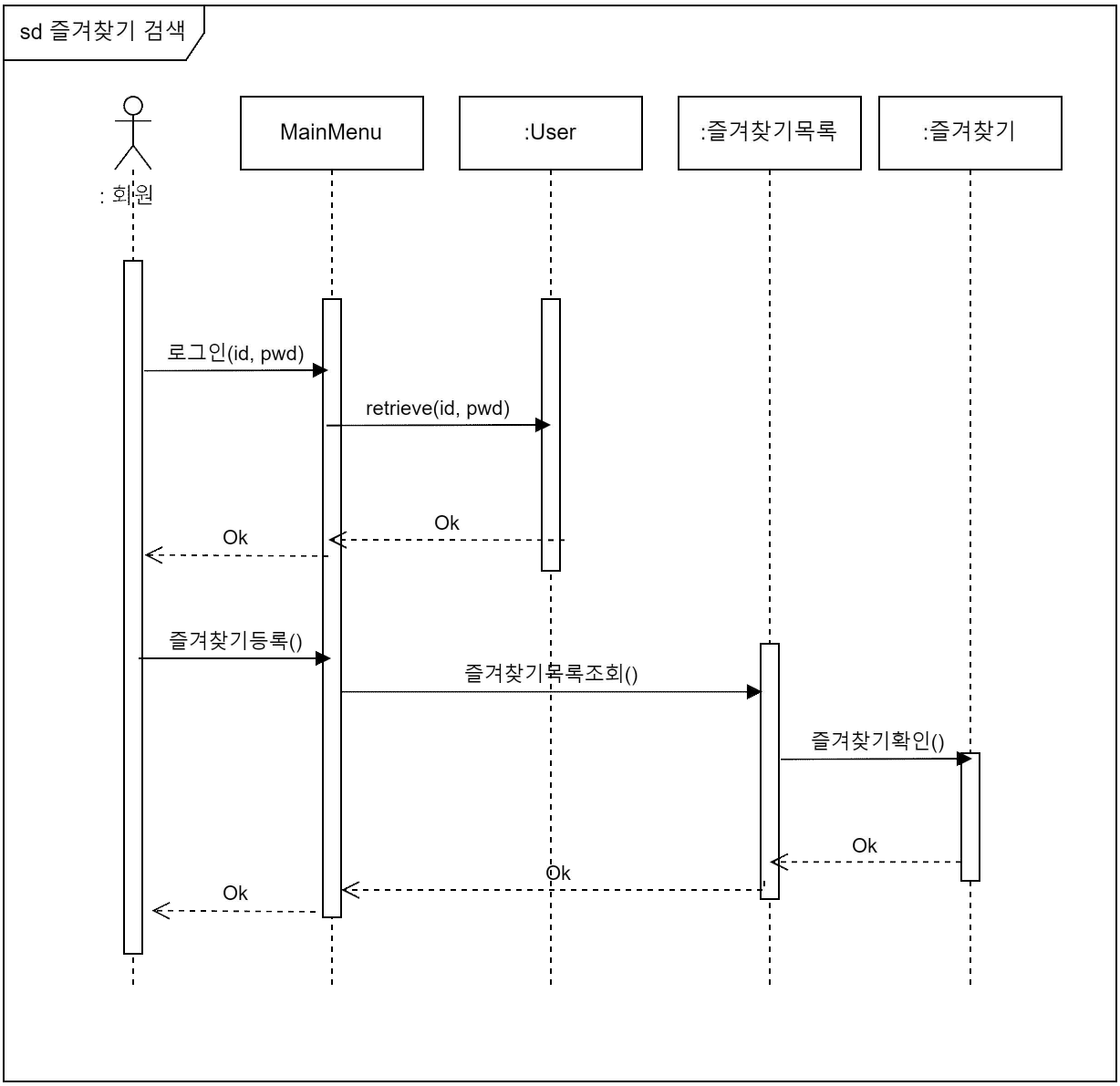
[그림 8.3 – "즐겨찾기 삭제" 시퀀스 다이어그램 명세]

8.4 Use-Case '즐겨찾기 검색'에 대한 시퀀스 다이어그램의 알고리즘 설계

메소드 이름	retrieve(search)Bookmark
메소드 기능	즐겨찾기 목록 내 등록된 장소에 대한 즐겨찾기 정보를 검색하는 메소드
알고리즘	

```
method retrieveBookmark(search) {
        // 데이터베이스 연결
        dbConnection = new DatabaseConnection();
        dbConnection.connect();

        // 로그인
        userService = new UserService();
        userService.login();

        // 키워드에 매칭되는 즐겨찾기 검색
        List<Bookmark> bookmarks = dbConnection.searchBookmarks(search);
        //검색 결과 리턴
        return bookmarks

    }
```

8.4.1 메뉴 '즐겨찾기'의 서브메뉴 '조회'에 대한 시퀀스 다이어그램



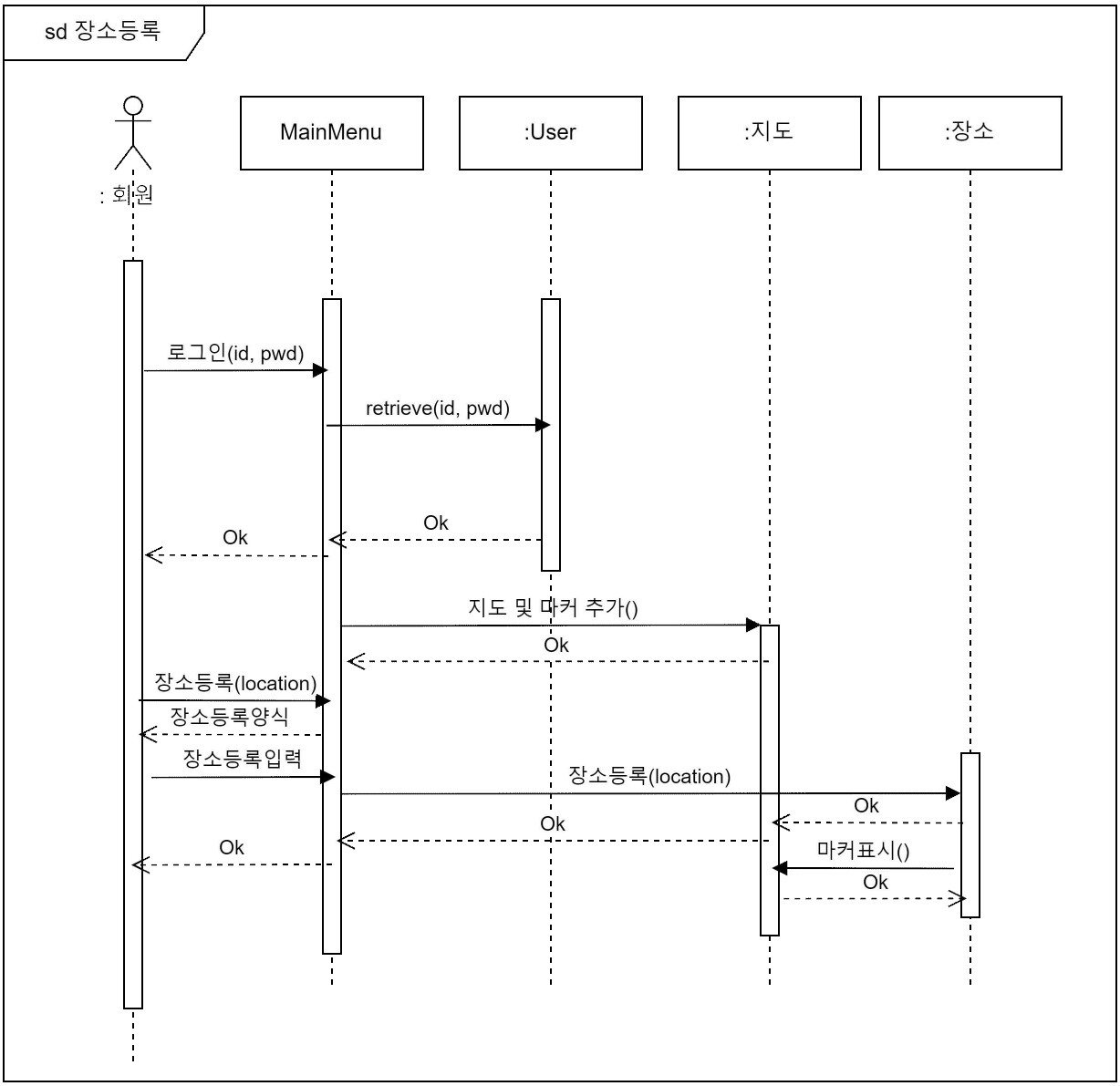
[그림 8.4 – "즐겨찾기 검색" 시퀀스 다이어그램 명세]

8.5 Use-Case '장소 등록'에 대한 시퀀스 다이어그램의 알고리즘 설계

메소드 이름	createPlace
메소드 기능	새로운 미등록되어있는 장소를 등록하는 메소드
알고리즘	

```
public void createPlace(String placeName, String placeAddress) {
        // 데이터베이스와 연결
        dbConnection = new DatabaseConnection();
        dbConnection.connect();

        // 로그인
        userService = new UserService();
        userService.login();

        // 등록된 마커들 지도 상에 표시
        map = new Map();
        map.addMarkers(placeName, placeAddress);

        // 장소 등록하기
        Place newPlace = new Place(placeName, placeAddress);
        newPlace.register();
    }
```

8.5.1 메뉴 '장소 정보'의 서브메뉴 '추가'에 대한 시퀀스 다이어그램



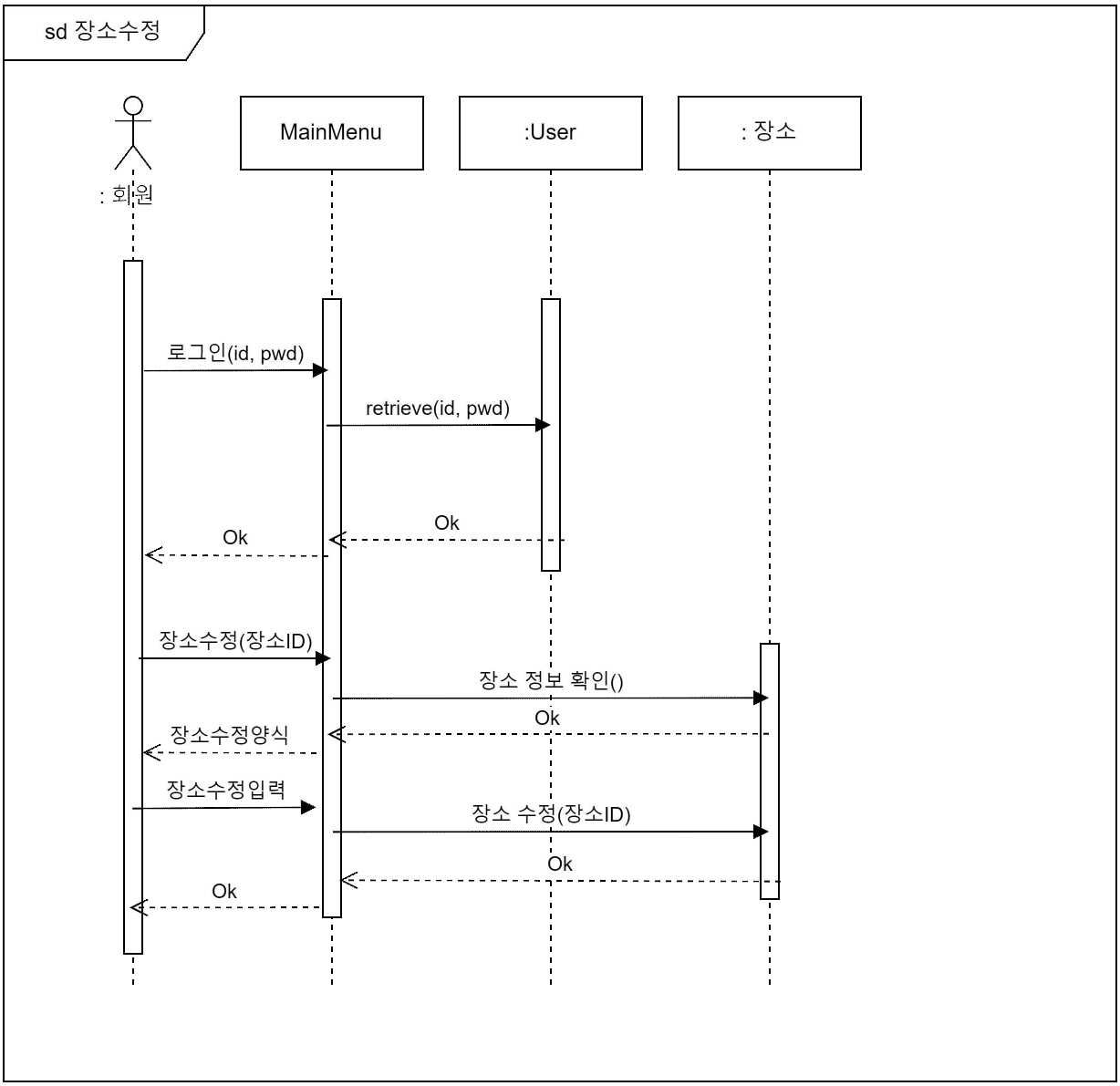
[그림 8.5 – "장소 등록" 시퀀스 다이어그램 명세]

8.6 Use-Case '장소 수정'에 대한 시퀀스 다이어그램의 알고리즘 설계

메소드 이름	updatePlace
메소드 기능	해당 장소의 정보를 수정하는 메소드
알고리즘	

```
public void updatePlace(String placeId, String newPlaceName, String newPlaceAddress) {
        //DB연결
        dbConnection = new DatabaseConnection();
        dbConnection.connect();

        // 로그인
        userService = new UserService();
        userService.login();

        // 수정하고자 하는 장소
        Place selectedPlace = dbConnection.selectPlace(placeId);

        // 정보 수정하기
        selectedPlace.updatePlace(modified contents);
    }
```

8.6.1 메뉴 '장소 정보'의 서브메뉴 '수정'에 대한 시퀀스 다이어그램



[그림 8.6 – "장소 수정" 시퀀스 다이어그램 명세]

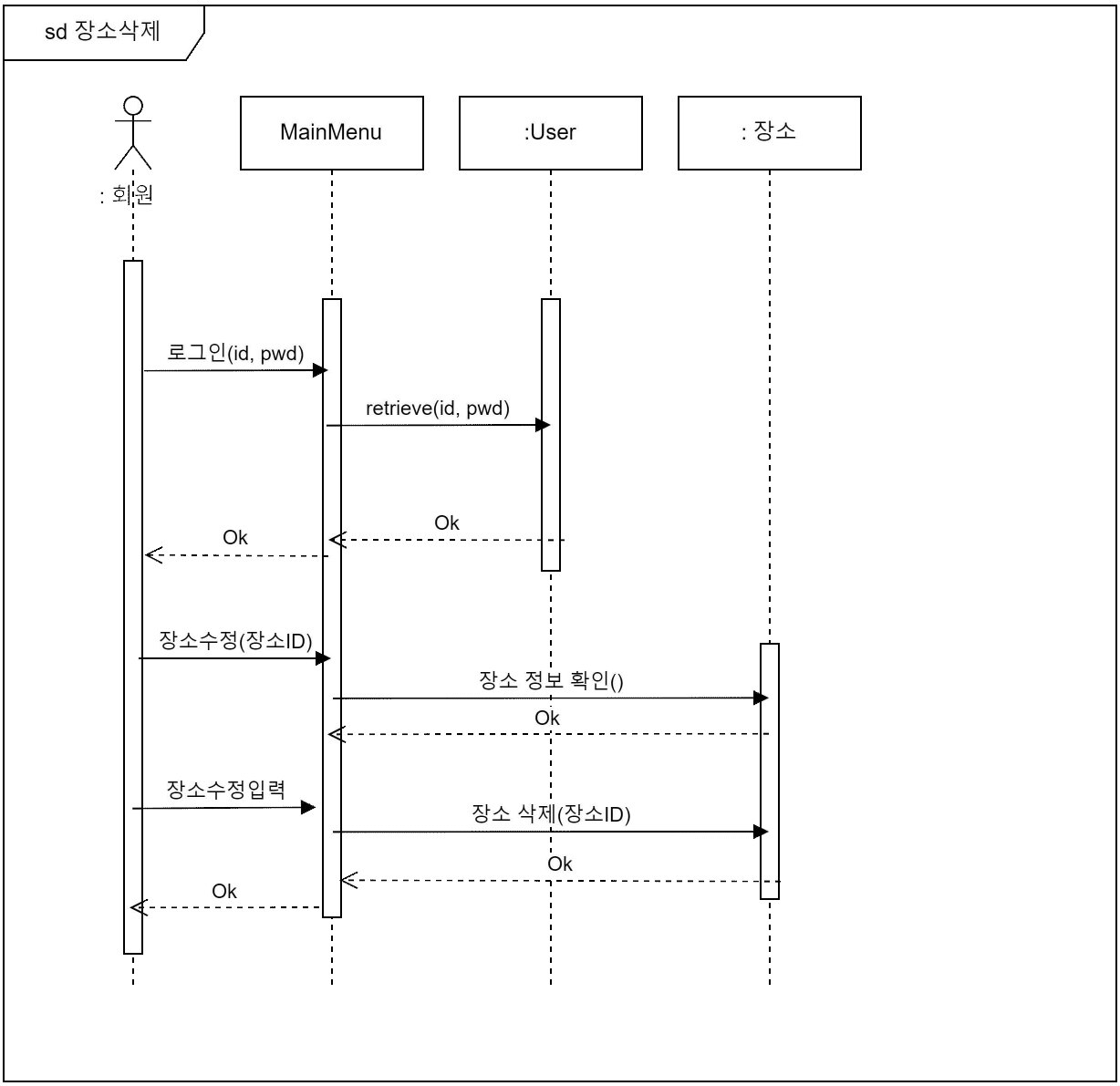
8.7 Use-Case '장소 삭제'에 대한 시퀀스 다이어그램의 알고리즘 설계

메소드 이름	delete_place
메소드 기능	지도에 등록된 장소 정보를 삭제하는 메소드

알고리즘

```
// 장소를 삭제하는 메소드
public void delete_place(String blistId) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Label";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     if (place.containsKey(placeName)) {
      //장소 삭제
          databse.deletebList(placeName);
     } else {
       //삭제 실패
     }
}
```

8.7.1 메뉴 '장소 정보'의 서브메뉴 '삭제'에 대한 시퀀스 다이어그램



[그림 8.7 – "장소 삭제" 시퀀스 다이어그램 명세]

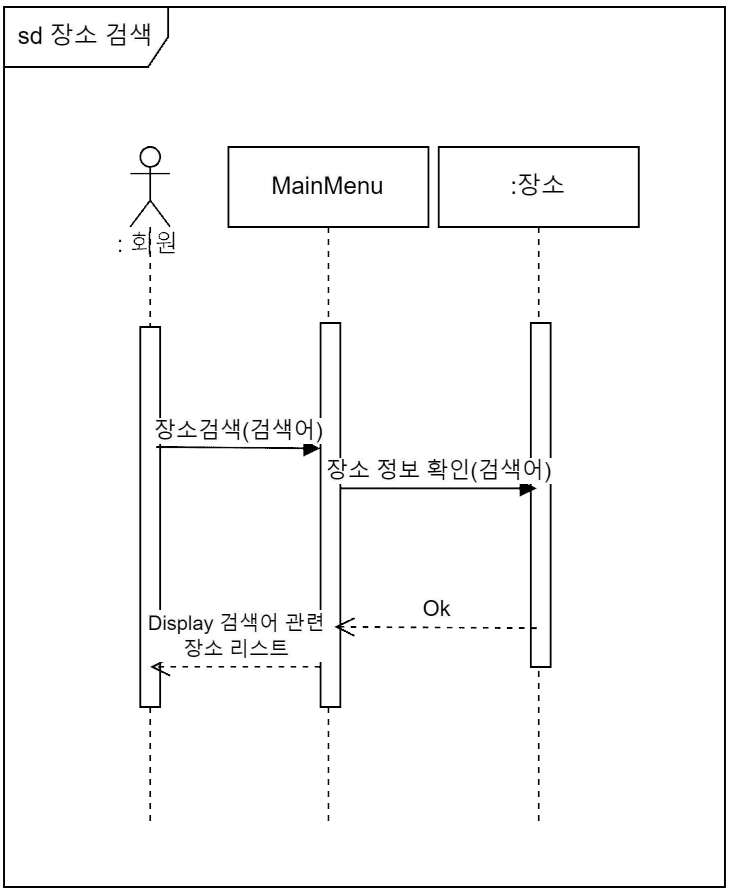
8.8 Use-Case '장소 검색'에 대한 시퀀스 다이어그램의 알고리즘 설계

메소드 이름	search
메소드 기능	지도에 등록된 장소 정보를 검색하는 메소드

알고리즘

```
public Place search(String PID) { //PID : 장소의 번호
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Places";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     /function search(placeName) {
  // Retrieve 모든 장소 정보
  List<Place> allPlaces = database.getAllPlaces();
  // 빈 리스트 생성
  List<Place> matchingPlaces = new ArrayList<>();
  for (Place place : allPlaces) {
    // 검색창에 입력한 장소이름과 유사한 장소들을 매칭장소 리스트에 추가
    if (place.getName().similarCase(placeName)) {
      matchingPlaces.add(place);
    }
  }
  // 매칭된 장소들 리턴
  return matchingPlaces;}}
```

8.8.1 메뉴 '장소 정보'의 서브메뉴 '검색'에 대한 시퀀스 다이어그램



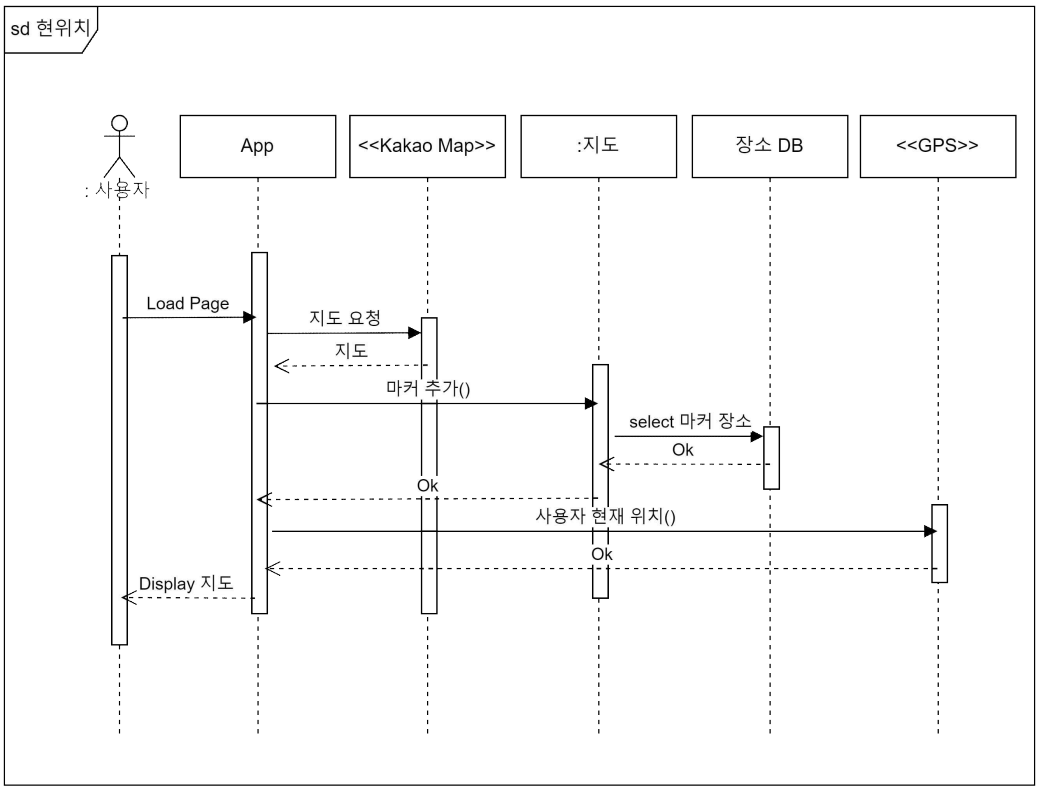
[그림 8.8 – "장소 검색" 시퀀스 다이어그램 명세]

8.9 Use-Case '현 위치 찾아가기'에 대한 시퀀스 다이어그램의 알고리즘 설계

메소드 이름	currentLocation
메소드 기능	현재 나의 위치를 지도에서 갱신한다.
알고리즘	

```
// 카카오맵의 시점을 현재 사용자의 GPS 상 위치한 곳으로 이동시키는 메소드
public currentLocation() {
    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;
}
```

8.9.1 메뉴 '지도'의 서브메뉴 '현재 위치 갱신'에 대한 시퀀스 다이어그램



[그림 8.9 – "현 위치 찾아가기" 시퀀스 다이어그램 명세]

9. 행위 모델링 : 메소드 알고리즘 설계

9.1 클래스 '사용자(Person)'의 메소드 명세

9.1.1 signUp 메소드

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

```
 function  signUp(email, password, name) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/User";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();    }
  // Retrieve 사용자 email 데이터(이메일 중복 확인)
  if (database.getUserByEmail(email) != null) {
    return false;}
  // 사용자 생성
  user = new User(email, password, name);
  // 데이터베이스 저장
  database.addUser(user);
  // 이메일 인증
  sendConfirmationEmail(user);
  // 회원 가입 성공
  return true;}
```

[표 9.1 – "signUp" 메소드 명세]

9.1.2 read 메소드

메소드 이름	read //회원 정보 확인
메소드 기능	회원이 자신의 회원 정보를 확인할 수 있는 메소드
알고리즘	

```
function read() {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/User";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  // retireve 사용사 ID
  user = database.getUserById(userId);
  //사용자 없는 경우 실패
  if (user == null) {
    return null;
  }
  // 사용자 데이터 조회 성공
  return {
    id: user.getId(),
    name: user.getName(),
    email: user.getEmail(),
    phoneNumber: user.getTel()}
}
```

[표 9.2 – "read" 메소드 명세]

9.1.3 update 메소드

메소드 이름	update //회원 정보 수정
메소드 기능	회원이 자신의 회원정보를 수정하기 위해서 회원가입 데이터를 수정하는 메소드
알고리즘	

```
 private void update(userId, newData) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/User";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");}
    } catch (SQLException e) {
        e.printStackTrace();}
  // retireve 사용사 ID
  user = database.getUserById(userId);
  // 사용자 검증 실패
  if (user == null) {
    return false;
}
  // 회원 데이터 수정
  if (newData.name != null) {
    user.setName(newData.name);
}
  if (newData.email != null) {
    user.setEmail(newData.email);
}
  if (newData.Tel != null) {
    user.setPhoneNumber(newData.phoneNumber);
}
```

```
// 사용자 데이터 베이스 저장(업데이트)
database.updateUser(user);
// 성공
return true;}
```

[표 9.3 – "update" 메소드 명세]

9.1.4 delete 메소드

메소드 이름	delete // 회원 탈퇴
메소드 기능	회원이 자신이 가입한 정보를 삭제하고 회원 탈퇴를 하는 메소드
알고리즘	

```
function delete() {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/User";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  // retireve 사용사 ID
  user = database.getUserById(userId);
  // 사용자 검증 실패
  if (user == null) {
    return false;
  }
  // 데이터베이스에서 회원 정보 삭제
  database.deleteUser(user);
  // 성공
  return true;
}
```

[표 9.4 – "delete" 메소드 명세]

9.2 클래스 '회원(User)'의 메소드 명세

9.2.1 login 메소드

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

```
function login(email, pwd) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/User";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  // Retrieve 사용자 email 데이터
  user = database.getUserByEmail(email);

  // 사용자 없거나 비밀번호 틀리면 false
  if (user == null || user.getPassword() != pwd) {
    return false;
  }

  // 이메일과 비밀번호 맞으면 true
  return true;
}
```

[표 9.5 – "login" 메소드 명세]

9.2.2 findId 메소드

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

```
function findId() {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/User";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  // Retrieve 사용자 email 데이터
  user = database.getUserByEmail(email);

  // 사용자 검증 실패한 경우
  if (user == null || user.getTel() != tel) {
    return null;
  }

  // 사용자 검증 성공하면 아이디 리턴
  return user.getId();
}
```

[표 9.6 – "findId" 메소드 명세]

9.2.3 findPwd 메소드

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

알고리즘

```
function findPwd(email) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/User";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  // Retrieve 사용자 email 데이터
  user = database.getUserByEmail(email);

  // 사용자 검증
  if (user == null || user.getTel() != tel) {
    return null;
  }

  // 검증 성공 시 비밀번호 찾기 성공
  return user.getPassword();
}
```

[표 9.7 – "findPwd" 메소드 명세]

9.2.4 changePwd 메소드

메소드 이름	changePwd
메소드 기능	회원이 기존의 입력한 비밀번호에서 새로운 비밀번호를 변경하는 메소드
알고리즘	

```
function changePwd(userId, currentPwd, newPwd) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/User";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  // Retrieve usrID 데이터
  user = database.getUserById(userId);
  // 비밀번호 검증 실패한 경우
  if (user == null || user.getPassword() != currentPwd) {
    return false;
  }
  //비밀번호 검증 성공한 경우 비밀번호 수정 및 db저장
  user.setPassword(newPwd);
  database.updateUser(user);
  return true;
}
```

[표 9.8 – "changePwd" 메소드 명세]

9.3 클래스 '리워드(Reward)'의 메소드 명세

9.3.1 read 메소드

메소드 이름	read //리워드 조회
메소드 기능	회원 자신이 보유하고 있는 포인트를 확인할 수 있는 메소드
알고리즘	

```
function read() {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Reward";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  // Retrieve reward ID 데이터
  reward = database.getRewardById(rewardId);

  // 리워드 데이터 없는 경우, 포인트 0
  if (reward == null) {
    return '0';
  }

  // 리워드 데이터 조회 성공
  return reward;
}
```

[표 9.9 – "read" 메소드 명세]

9.3.2 create 메소드

메소드 이름	create //리워드 포인트 보상 등록
메소드 기능	회원이 인원수를 공유하였을 때 임의의 포인트 보상을 제공하는 메소드
알고리즘	

```
function create() {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Reward";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  //리워드 포인트만큼 리워드 객체 생성
  reward = new Reward(rewardPoint);

  // 리워드 저장
  database.addReward(reward);

//리워드 총점 리턴
  return reward.getId();
}
```

[표 9.10 – "create" 메소드 명세]

9.4 클래스 '영상(Image)'의 메소드 명세

9.4.1 analyzeImage메소드

메소드 이름	analyzeImage
메소드 기능	영상을 분석하여 인원수를 계산하는 메소드
알고리즘	

```
public int analyzeImage() {
    // 영상 분석 모델 로드
    Model model = loadModel();
    // 영상 속 사람 객체 탐지
    Object object = detectObject(latestImage, model);
    //  탐지 객체 트랙킹하여 중복 제거
    Object trackedObject = trackObject(latestImage, detectedObject);
    // 사람 수 카운팅
    int headCount = countPeople(latestImage, trackedObject);
    // 인원수 리턴
    return headCount;
}
```

[표 9.11 – "analyzeImage" 메소드 명세]

9.5 클래스 '장소(Places)'의 메소드 명세

9.5.1 create 메소드

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

```
public void create(String PlaceName, String location) {
//PlaceName: 장소 이름, //location : 장소 위치, 주소
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Places";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     // 같은 이름의 장소가 같은 위치에 존재하지 않는지 확인
     if (database.placeExists(PlaceName)) {
         // 이미 장소가 등록되어 있는 경우
     } else {
         // 장소를 DB에 추가하여 등록
         database.createPlace(PlaceName, location);
         // 등록 완료 알림
         alarm("The place has been successfully added.");
     }
}
```

[표 9.12 – "create" 메소드 명세]

9.5.2 read 메소드

메소드 이름	read
메소드 기능	특정 위치에 저장된 장소 정보를 가져오는 메소드

알고리즘

```
public Place read(String location) {
/DB연결
    String url = "jdbc:mysql://your_host:your_port/Places";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     // 주어진 장소에 저장된 장소 있는지 확인
     Place place = databse.getPlaceInfo(location);
     if (place == null) {
// 장소 정보 없으면 경우 처리
     }
     // 장소 정보 리턴
     return place;
}
```

[표 9.13 – "read" 메소드 명세]

9.5.3 update 메소드

메소드 이름	update
메소드 기능	장소의 수정할 정보를 입력해 장소 정보를 업데이트하는 메소드

알고리즘

```
public void update(String PID, String newOne) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Places";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     // DB에 등록된 장소 있는지 확인
     if (database.getPlace(PID)!=null) {
         // 수정한 정보로 업데이트
         database.updatePlace(placeID, newOne);
     } else {
         // DB에 등록된 장소 없음 처리
     }
}
```

[표 9.14 – "update" 메소드 명세]

9.5.4 delete 메소드

메소드 이름	delete
메소드 기능	등록된 장소의 정보를 삭제하는 메소드
알고리즘	

```
public void delete_place(String PID) { //PID : 장소의 번호
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Places";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     // DB에 장소가 저장되어 있는지 확인
     if (database.getPlace(PID)!=null)) {
          // 삭제
          database.deletePlace(PID);
     } else {
          //장소가 존재하지 않음.
     }
}
```

[표 9.15 – "delete" 메소드 명세]

9.5.5 search 메소드

메소드 이름	search
메소드 기능	등록된 장소의 정보를 검색하는 메소드

알고리즘

```
public Place search(String PID) { //PID : 장소의 번호
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Places";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     /function search(placeName) {
  // Retrieve 모든 장소 정보
  List<Place> allPlaces = database.getAllPlaces();
  // 빈 리스트 생성
  List<Place> matchingPlaces = new ArrayList<>();
  for (Place place : allPlaces) {
    // 검색창에 입력한 장소이름과 유사한 장소들을 매칭장소 리스트에 추가
    if (place.getName().similarCase(placeName)) {
      matchingPlaces.add(place);
    }
  }
  // 매칭된 장소들 리턴
  return matchingPlaces;}}
```

[표 9.16 – "search" 메소드 명세]

9.6 클래스 '지도(Map)'의 메소드 명세

9.6.1 makeMarker 메소드

메소드 이름	makeMarker
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();
    marker.setMapPoint(mapPoint);

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

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

[표 9.17 – "makeMarker" 메소드 명세]

9.6.2 zoomIn 메소드

메소드 이름	zoomIn
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 9.18 – "zoomIn" 메소드 명세]

9.6.3 zoomOut 메소드

메소드 이름	zoomOut
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 9.19 – "zoomOut" 메소드 명세]

9.6.4 currentLocation 메소드

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

```
// 카카오맵의 시점을 현재 사용자의 GPS 상 위치한 곳으로 이동시키는 메소드
public currentLocation() {
    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 9.20 – "currentLocation" 메소드 명세]

9.7 클래스 '즐겨찾기(Bookmark)'의 메소드 명세

9.7.1 create 메소드

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

```
function create() {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/bookMark";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();}
  //retrieve 장소ID
  Place place = database.getPlaceById(placeId);

  // Retrieve 즐겨찾기 목록
  Blist selectedList = database.getLabelListById(bListId);
  // Create 북마크
  Bookmark newBookmark = new Bookmark(place, selectedList);
  // Add the new Bookmark
  selectedList.addBookmark(newBookmark);
  // 저장
  database.save();
}
```

[표 9.21 – "create" 메소드 명세]

9.7.2 read 메소드

메소드 이름	read
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 read() {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/bookMark";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     placeList = databse.getBookmarkList();
     // 즐겨찾기된 장소 정보 없으면 경우
     if (placeList == null) {
         return;
     }
     // 즐겨찾기된 장소 장보 반환
     return placeList;
}
```

[표 9.22 – "read" 메소드 명세]

9.7.3 delete 메소드

메소드 이름	delete
메소드 기능	즐겨찾기 목록에 특정 장소에 대한 즐겨찾기를 제거하는 메소드

알고리즘

```
// 즐겨찾기를 제거하는 메소드
public void delete(Place placeId) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/bookMark";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     // 해당 장소가 즐겨찾기 데이터베이스에 있는지 확인
     if (databse.getPlace(placeId)) {
          // 장소의 즐겨찾기 상태를 false로 변경
          database.deleteBookmark(placeId);
     } else {
        //제거 실패
     }
}
```

[표 9.23 – "delete" 메소드 명세]

9.8 클래스 '즐겨찾기 목록(BookmakrList)'의 메소드 명세

9.8.1 create 메소드

메소드 이름	create
메소드 기능	즐겨찾기 목록을 추가하는 메소드
알고리즘	

```
// 즐겨찾기 목록을 추가하는 메소드
public void create() {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Label";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     bookmarkLists.put(listName, listId);
//DB에 추가, 등록
     databse.addBookMarkList(listName ,listId);
}
```

[표 9.24 – "create" 메소드 명세]

9.8.2 read 메소드

메소드 이름	read
메소드 기능	즐겨찾기 목록 정보를 조회하는 메소드
알고리즘	

```
// 즐겨찾기 목록을 확인하는 메소드
public void read(String blistId) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Label";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     if (bookmarkLists.isEmpty()) {
         System.out.println("등록된 즐겨찾기 목록이 없습니다.");
     } else {
         return databse.readBList(blistId);
     }
}
```

[표 9.25 – "read" 메소드 명세]

9.8.3 delete 메소드

메소드 이름	delete
메소드 기능	즐겨찾기 목록을 제거하는 메소드
알고리즘	

```
// 즐겨찾기 목록을 삭제하는 메소드
public void delete_bookmark_list(String blistId) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Label";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     if (bookmarkLists.containsKey(blistName)) {
      //즐겨찾기 목록 삭제
          databse.deletebList(blistName);
     } else {
       //삭제 실패
     }
}
```

[표 9.26 – "delete" 메소드 명세]

9.8.4 update 메소드

메소드 이름	update
메소드 기능	즐겨찾기 목록 정보를 변경하는 메소드
알고리즘	

```
// 즐겨찾기 목록을 삭제하는 메소드
public void delete_bookmark_list(String blistId) {
//DB연결
    String url = "jdbc:mysql://your_host:your_port/Label";
    String user = "your_username";
    String password = "your_password";
    Connection dbConnection = null;
    try {
        dbConnection = DriverManager.getConnection(url, user, password);
        if (dbConnection != null) {
            System.out.println("데이터베이스 연결 성공");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
     if (bookmarkLists.containsKey(blistName)) {
         database.updatebList(blistId)
     } else {
         System.out.println("해당 목록명의 즐겨찾기 목록이 존재하지 않습니다.");
     }
}
```

[표 9.27 – "update" 메소드 명세]