

[초보자 및 학생들을 위한 천체 관측 교육 애플리케이션]

빌드 개발 명세서

2025년 11월 11일

문서번호 : Byeoldori_Doc_07

소 속 : 충북대학교 천문우주학과

팀 명 : 천문소프트

팀 원 : 서범수, 윤태영, 김채영

교 수 : 조오현 교수님

목 차

1. 1차 빌드 개발 개요 -----	1
2. 2차 빌드 개발 개요 -----	15
3. 3차 빌드 개발 개요 -----	32
4. 4차 빌드 개발 개요 -----	46
5. 5차 빌드 개발 개요 -----	76
6. 6차 빌드 개발 개요 -----	82
7. 7차 빌드 개발 개요 -----	90
8. 기타 구현 참고 사항 -----	109

1.1. 1차 빌드 개발 개요

태스크 번호	태스크 명	담당자	시작일	종료일	비고
T-101	기상청 API 호출	윤태영	25.03.05	25.03.11	-
T-102	예보 별 관측 적합도 반환	윤태영	25.03.23	25.03.30	-
T-103	관측 장소 제공	윤태영	25.05.21	25.06.03	-
T-104	관측지 상세 조회	윤태영	25.06.14	25.06.20	-
T-105	Naver map API 호출	김채영	25.03.05	25.03.11	-
T-106	지도에 광공해 오버레이	김채영	25.03.20	25.04.01	-
T-107	지도 UI	김채영	25.04.07	25.04.17	-
T-108	관측지 정보 출력 UI	김채영	25.08.18	25.08.27	-
T-109	관측지 화면 I.A 작성	김채영	25.08.18	25.08.21	-
T-110	Observation API 명세서 작성	윤태영	25.09.21	25.09.23	-
T-111	Observation Controller 연동	김채영	25.09.25	25.10.11	-

1.2. 분석 명세

1.2.1. 기능 명세

(1) Use Case Diagram (전체 시스템)



(2) Use Case 설명서 (구현 시스템)

- 관측지 검색 Usecase

Use Case Name: 관측지 검색	UC-004	Important Level: Mid
Primary Actor: 사용자		Use case type: Detail, Essential
Stakeholders and Interests:		
사용자: 키워드로 관측지를 빠르게 찾고 싶어함 시스템: 사용자가 검색한 키워드에 포함되는 관측지 목록을 반환해줌		
Brief Description: - 사용자가 키워드(지명/관측지명)로 관측지를 조회한다.		
Trigger: 관측지 메뉴에 들어간 후 검색창에서 키워드 입력 Type: External		
Relationships: Include: 관측지 상세 정보 조회 Extend:		
Normal Flow of Events: 1. 사용자가 관측지 화면으로 간다. 2. 사용자는 검색어(예: "보현산")를 입력한다. 3. 시스템은 키워드로 관측지 목록을 조회한다. 4. 시스템은 각 관측지의 기본 정보(관측지 이름, 리뷰 수, 좋아요 수, 평점 수, 현재 관측 적합도, 도로명 주소, 해당 위치의 현재 날씨, 해당 관측지에서 진행한 관측후기 정보를 페이지 단위로 반환한다. 5. 사용자는 검색 결과 목록에서 관측 후기를 선택하여 관측 후기 상세 화면으로 이동한다.		
Subflows: S-1: 관측 적합도 계산 1. 시스템은 각 관측지의 위치 좌표를 기반으로 기상청 API를 호출한다. 2. 기상 데이터(구름량, 강수 확률, 달의 위상, 광공해 정도)를 종합해 관측 적합도 점수를 계산한다. 3. 관측 적합도는 퍼센트로 변환되어 사용자에게 표시된다.		
Alternative / Exceptional Flows: E-1: 해당 관측지에 대한 관측 후기가 없을 경우, "해당 관측지에서 관측한 관측 후기가 없습니다."라고 메시지를 출력한다.		

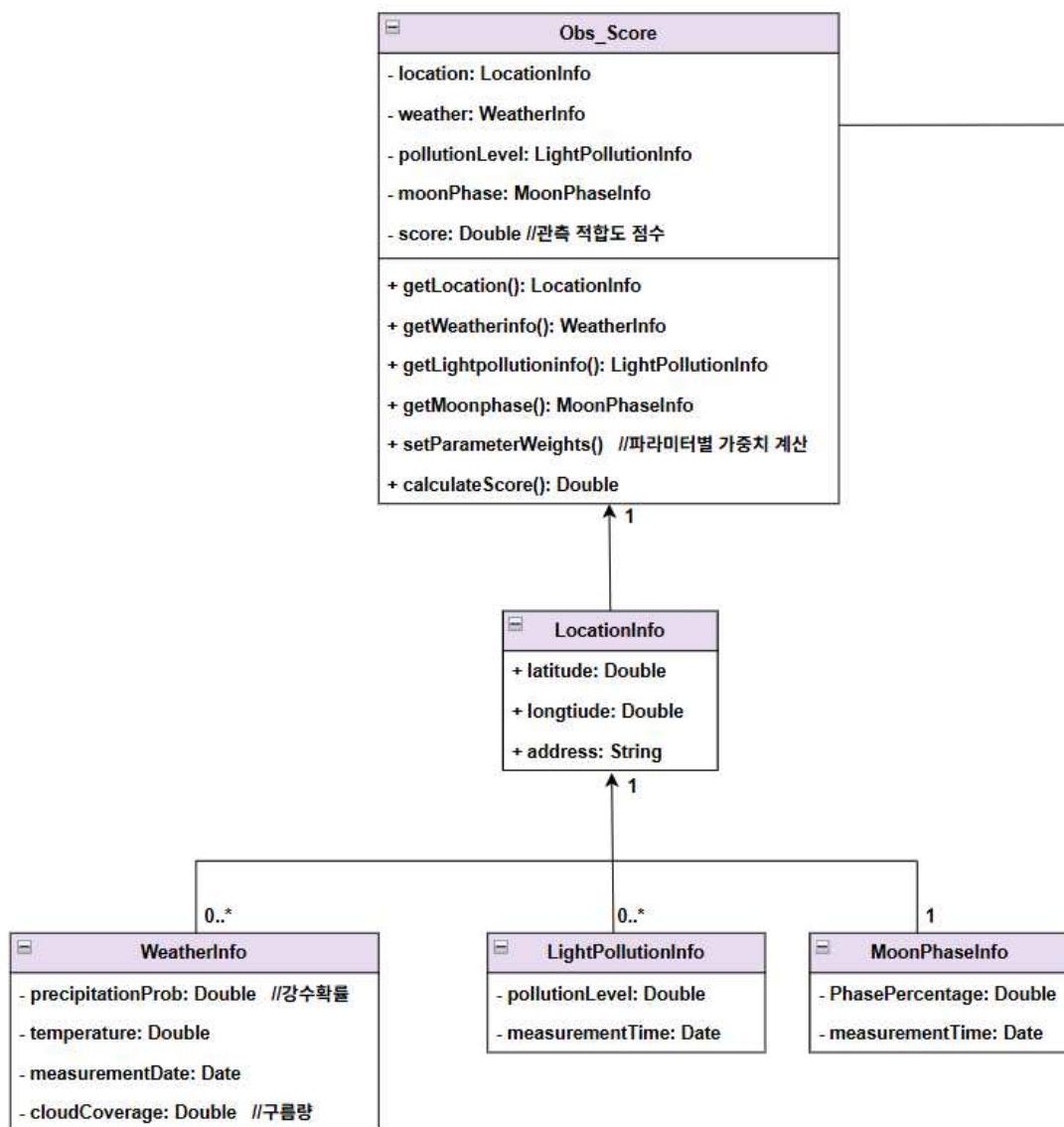
- 관측지 관측 적합도 확인 Usecase

Use Case Name: 관측 적합도 확인	UC-005	Important Level: High
Primary Actor: 사용자		Use case type: Detail, Essential
Stakeholders and Interests:		
<p>사용자: 관측지의 기본 정보와 함께 현재 날씨와 관측 적합도를 한 화면에서 확인하고 싶다.</p> <p>시스템: 관측지에 대한 자세한 정보를 사용자에게 제공한다.</p>		
Brief Description:		
<ul style="list-style-type: none"> 사용자가 특정 관측지의 이름, 위치와 함께 해당 좌표 기준의 날씨 정보와 관측 적합도 점수를 확인한다. 		
Trigger: 목록/지도에서 특정 관측지 선택 후 "상세 보기" 진입		
Type: External		
Relationships:		
<p>Include: 좌표 기반 날씨 조회</p> <p>Extend:</p>		
Normal Flow of Events:		
<ol style="list-style-type: none"> 사용자가 관측지 상세를 요청한다. 시스템은 관측지 식별자로 기본 정보(이름, 위·경도)를 조회 후 해당 좌표로 날씨 데이터를 받아온다. 시스템은 받아온 날씨 항목을 기준으로 관측 적합도를 계산한다. 시스템은 기본 정보, 날씨, 관측 적합도를 사용자에게 반환한다. 사용자는 상세 화면에서 결과를 확인한다. 		
Subflows:		
<p>S-1: 관측 적합도 산출</p> <ol style="list-style-type: none"> 가중치 기반 총점 및 세부 점수 계산한다.(날씨 / 달 / 광공해 고려) 		
Alternative / Exceptional Flows:		
<p>E-1: 서비스 영역 밖 좌표일 경우 "지원 지역이 아닙니다." 메시지 표시 및 지도 안내한다.</p> <p>E-2: 네트워크 또는 날씨 API 연결 실패 시, "현재 날씨 정보를 불러올 수 없습니다."라는 안내 메시지를 출력한다.</p>		

1.2.2. 구조 명세 (구현 시스템)

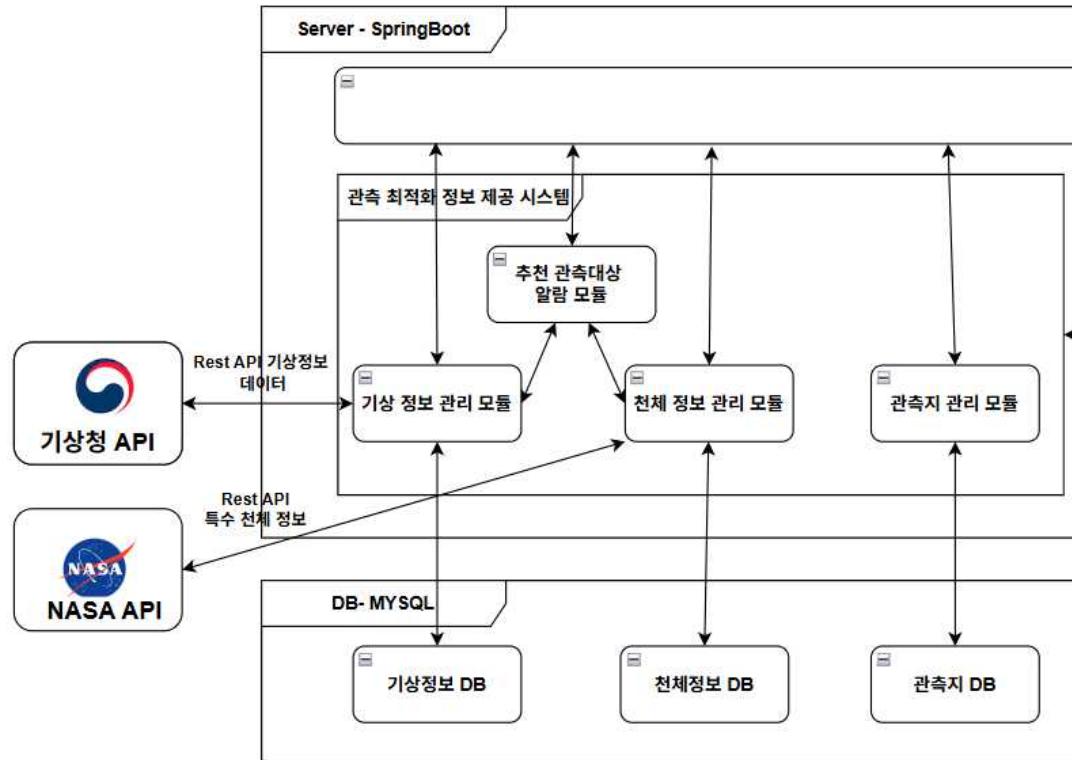
- 관측지 정보제공 Class Diagram

본 클래스 다이어그램은 사용자가 지도에서 선택한 관측지의 위치 정보를 입력으로 받아, 해당 지역의 기상 조건, 광공해 정도, 달의 위상 정보를 연계하여 관측 적합도(Obs_Score)를 계산하는 과정을 표현한 것이다. Obs_Score 클래스는 관측지(LocationInfo)를 중심으로 각종 환경 정보(WeatherInfo, LightPollutionInfo, MoonPhaseInfo)를 종합적으로 평가하고, 파라미터별 가중치를 고려하여 최종 점수를 산출한다.



- 관측지 정보제공 시스템 구성도

본 시스템 구성도는 기상청 API를 통해 기상 정보를 수집하고, 이를 기반으로 관측지의 날씨, 광공해, 짙위상 정보를 종합 분석하여 관측 적합도를 산출하는 구조로 구성되어 있다. 각 모듈은 MySQL DB와 연동되어 관측지별 환경 데이터를 저장 및 갱신한다. 이러한 구성은 사용자가 선택한 관측지의 환경 정보를 실시간으로 평가하여 사용자에게 관측지 정보를 제공하는 것을 목표로 한다.

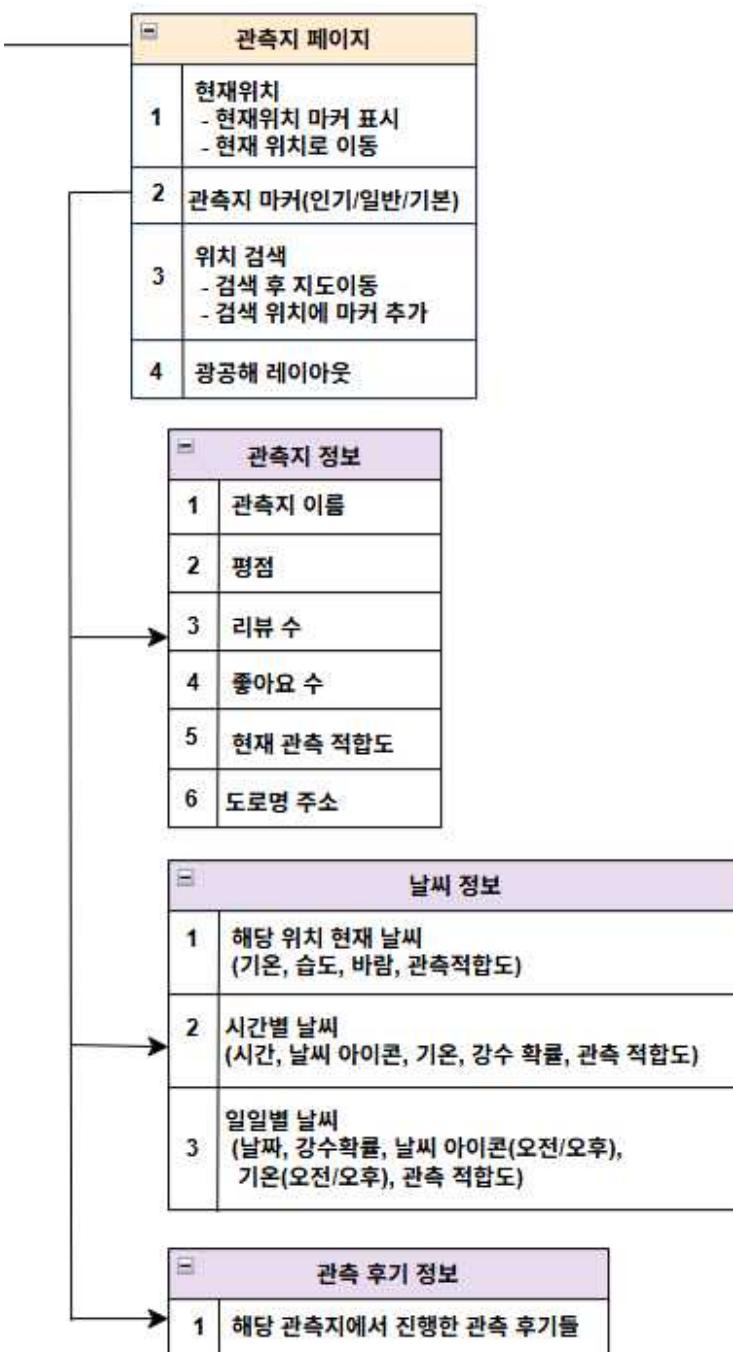


1.3. 설계 명세

1.3.1. 메뉴 전개도 (구현 시스템)

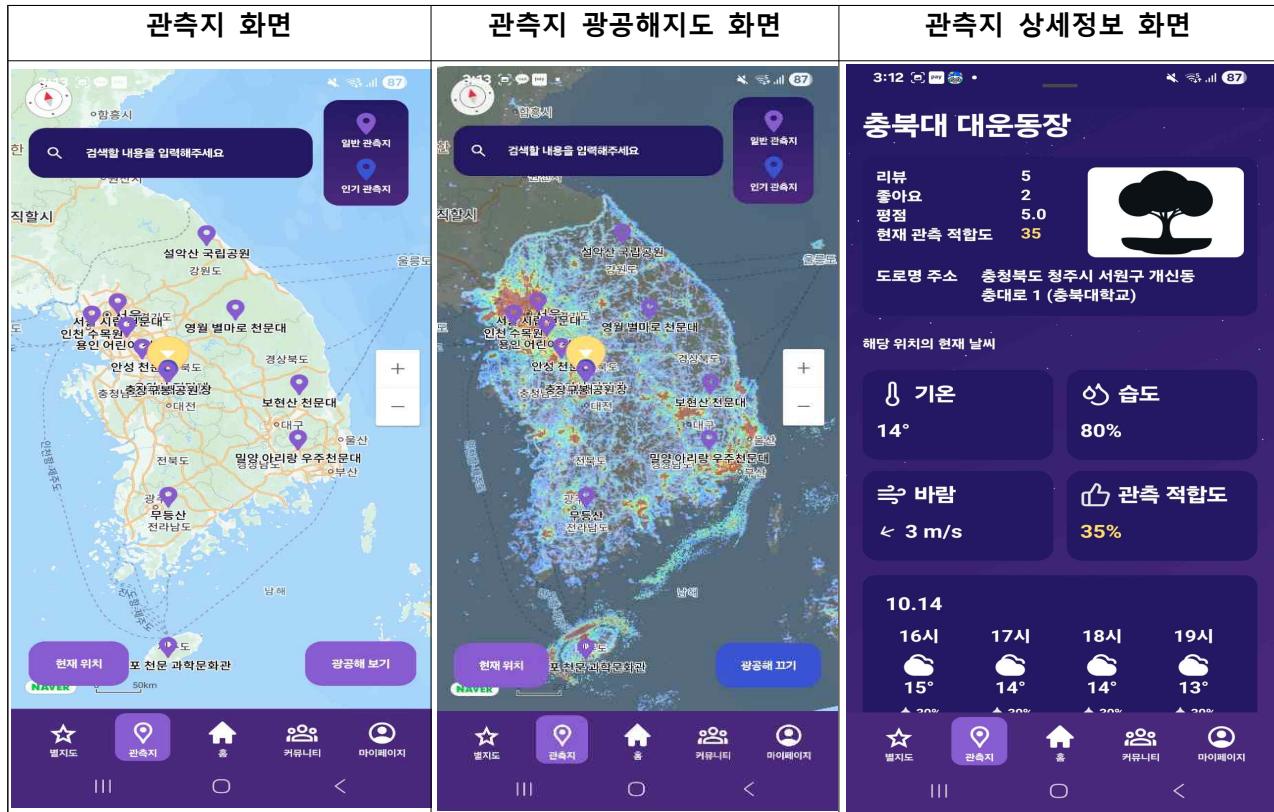
- 관측지 페이지 및 관측지 상세 조회 메뉴 전개도

본 메뉴 전개도는 관측지 검색 및 탐색부터 날씨 정보, 광공해, 관측 후기까지 사용자가 관측 환경을 종합적으로 파악할 수 있도록 구성된 관측지 페이지의 전체 기능 흐름을 표현한 것이다.



1.3.2. 사용자 인터페이스 설계 (구현 시스템)

- 관측지 UI

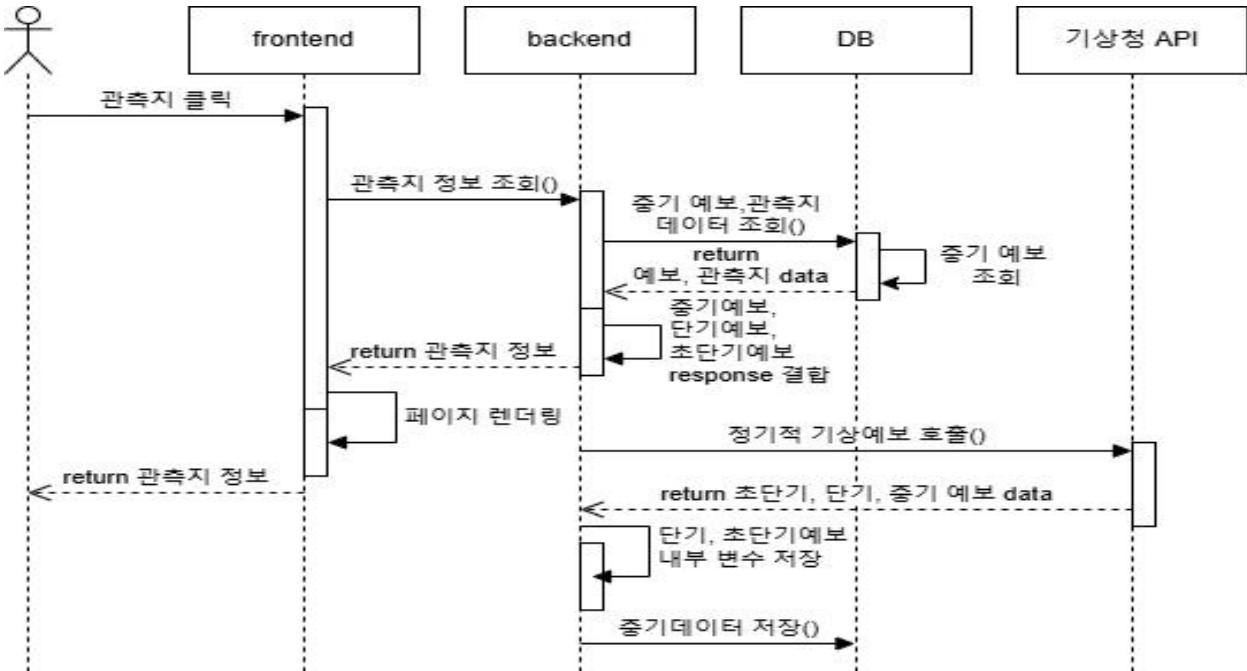


- 관측지 상세 조회 UI



1.3.3. 핵심 알고리즘 설계 (구현 시스템)

이번 빌드에서 구현된 핵심 알고리즘은 날씨 예보 호출, 관측 적합도 제공, NAVER MAP API 호출, 광공해 레이어 오버레이로 구성된다. 해당 알고리즘은 프론트엔드-백엔드-DB 계층 간의 명확한 역할 분리를 기반으로 동작하며, 사용자의 입력으로부터 데이터베이스 처리까지의 과정을 일관된 구조로 설계하였다. 아래 Sequence Diagram은 관측지 정보를 조회하는 절차를 시각화 한 것이다.



1.3.3.1. 초단기/단기 예보 호출 시퀀스

1. 시간 산정 (ForecastTimeUtil)

초단기: getStableUltraTmfc() - 현재시각에서 1시간을 뺀 후 00분, 30분중 가까운 시간에 스냅시킨다.

단기: getTMFCTimeForShort(), getTMEFTimesForShortForecast() - 한국 시간 기준으로

{02,05,08,11,14,17,20,23}의 시간에 가까운 발령시간(tmfc)/발효시간(tmef)을 계산한다.

2. 데이터 패치/병합

UltraGridForecastService / ShortGridForecastService에서 구름·습도·바람·기온 같은 변수를 동시에 호출해서 받아온 뒤 2차원 격자 형태에 맞게 하나로 합친다.

3. 캐시/메모리 업데이트

내부 TMEFGridMap에 ReentrantReadWriteLock으로 쓰기 락 후 원자적 교체한다.

조회(읽기)는 읽기 락으로 경합을 최소화 시킨다.

4. 재시도

호출이 실패하면 정해둔 간격(1분) 만큼 기다렸다가 다시 시도한다(최대 5회). 전부 실패하면 마지막으로 호출에 성공했던 예보를 반환해준다.

1.3.3.2. 중기 예보 호출 시퀀스

1. 예보 수집

육상 예보와 기온 예보를 각각 받아 '시 지역코드' 기준으로 하나의 예보로(결합 엔티티) 저장한다. 시간은 하루에 두번 6:10과 18:10에 스케줄러가 실행된다.

2. 이전 데이터 정리

이전 예보가 db에 쌓이는 걸 방지하기 위해서 예보 수집 후 20분 뒤, 24시간이 지난 중기 자료를 두 단계로 나눠 삭제한다. 기온 예보를 먼저 삭제한 후, 육상 예보를 삭제한다.

1.3.3.3 광공해 데이터 알고리즘 (요약)

전지구 광공해 위성 데이터(VIIRS Nighttime Lights Annual V2.2)를 이용하여 대한민국 영역의 광공해 분포를 시각화한다.

절차 요약	VIIRS Nighttime Lights Annual V2 (Version 2.2)데이터셋을 사용하여 GeoTIFF(지리정보 포함 이미지) 형식을 gzip으로 압축 해제한 뒤, 대한민국 위경도 영역(위도 33.0°~39.5°, 경도 124.5°~131.5°)으로 이미지를 자름 → 잘라낸 데이터를 Gamma Correction하여 컬러맵을 적용하여 시각화 → png파일로 저장한 광공해 이미지를 GroundOverlay()을 이용해 navermap에 오버레이
----------	--

1.3.3.4 관측지 리뷰 조회 알고리즘 (요약)

관측지 화면에서 마커를 클릭하면 관측지 정보를 BottomSheet로 띄우는데, 여기서 관측지 주소 정보, 날씨 정보, 해당 관측지에 해당하는 관측지 리뷰를 나타낸다.

절차 요약	마커를 클릭한 뒤, 관측지 카드(ObservatoryInfoCard)를 보여줌 → 관측지 카드 내부해서 해당 관측지의 siteId를 넘겨 ReviewViewModel.postState의 전체 후기 중에서 obsesrvationSiteId == siteId인 후기만 필터링
----------	--

1.3.3.5 관측지 주소 조회 알고리즘 (요약)

관측지화면에서 Navermap이 표시되고 마커를 클릭했을 때, ObservatoryCard에 도로명 주소가 표시된다.

절차 요약	관측지화면에서 navermap이 표시되고 마커를 클릭했을 때, NavermapRepository.reverseAddressRoad(lat,lon)을 호출하여 roadaddr, addr 결과를 조합해 도로명 주소를 생성 → 만약 도로명 주소를 실패했을 시, 빈 문자열을 반환하고 안드로이드 Geocoder를 호출하여 위도,경도 정보를 주소(문자열)로 변환
----------	--

1.3.3.6 Navermap 불러오기 알고리즘 (요약)

NavermapScreen에서 Navermap을 MapView형태로 화면에 지도를 불러온다.

절차 요약	Naver Cloud Platform에서 Maps 사용을 신청하여 API Key를 발급받음 → NavermapScreen 화면 접입 시 MapView를 생성하고 NaverMap을 준비 → RequestLocationPermission으로 위치 권한을 확인 후 사용자의 현재 위치로 카메라 이동
----------	--

1.3.4. 데이터 설계 (구현 시스템)

본 빌드의 데이터 설계는 관측지 정보 조회 및 기상청 API 호출 기능을 중심으로 이루어져 있다. 관측지 위치 조회, 날씨 정보 조회, 관측적합도를 지원하기 위해 3개의 핵심 테이블(Observation_site, Mid_forecast, Mid_temp_forecast)부가 테이블 및 관련 Repository 계층이 구성되어 있으며 초단기 예보와 단기 예보는 각각 10분, 3시간 마다 업데이트 되어 메모리에 저장이 된다. 각 테이블은 백엔드 서버(Spring Boot)와 MySQL DB를 기반으로 설계되었으며, Repository 계층을 통해 데이터 접근을 캡슐화하였다.

1.3.4.1. ERD(Entity Relationship Diagram)

- 관측지 테이블

관측지 테이블						
	관측지ID	obsite_id	Domain	bigint	1	고유 관측지 ID
관측지	ob_site	SiteName	varchar	Default value	Comment	
위도	latitude	Domain	double	Default value	Comment	
경도	longitude	Domain	double	Default value	Comment	

- 중기 예보 테이블

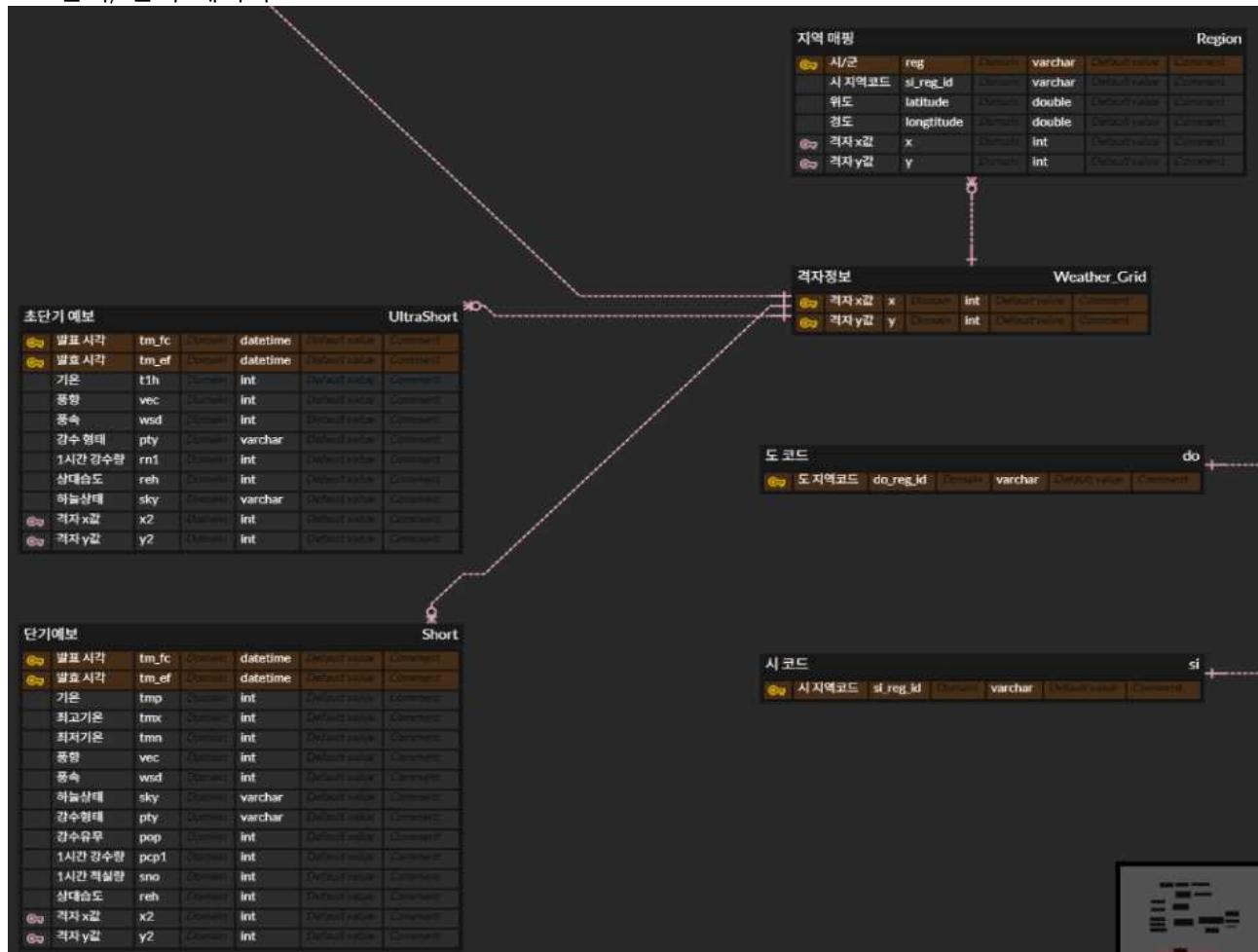
중기 옥상 예보

PK	land_id	datetime	bigint	1	Comment
발표 시각	tm_fc	Datetime	datetime	Default value	Comment
발표 시각	tm_ef	Datetime	datetime	Default value	Comment
하늘상태	sky	Datetime	varchar	Default value	Comment
강수형태	pre	Datetime	varchar	Default value	Comment
강수확률	tm_st	Datetime	int	Default value	Comment
생성일시	created_at	DatetimeNow	datetime	now()	Comment
도 지역코드	do_reg_id	Datetime	varchar	Default value	Comment

중기 기온 예보

PK	temp_id	datetime	bigint	1	Comment
발표 시각	tm_fc	Datetime	datetime	Default value	Comment
발표 시각	tm_ef	Datetime	datetime	Default value	Comment
최저기온	min	Datetime	int	Default value	Comment
최고기온	max	Datetime	int	Default value	Comment
생성일시	created_at	DatetimeNow	datetime	now()	Comment
시 지역코드	si_reg_id	Datetime	varchar	Default value	Comment

- 초단기, 단기 데이터



1.3.4.2. 테이블 설계 상세

테이블명	설명	주요 컬럼	제약조건 및 특징
Observation Site	관측지 저장	id, site, latitude, longitude	각 관측지의 경도, 위도 저장
Mid Forecast	중기 육상 예보 저장	id, tm_fc, tm_ef, sky, pre, rn,st, created_at, do.reg.id	중기 육상은 도 별로 예보를 하므로 각 시에 매핑을 시켜줘야함
Mid Temp Forecast	중기 기온 예보 저장	id, tm_fc, tm_ef, min, max, created_at, si.reg.id	시 지역코드를 격자 좌표로 변환

1.3.4.3. 데이터 설계 상세

데이터명	설명	주요 컬럼	제약조건 및 특징
Ultra Short Forecast	초단기 예보 호출	tm_fc, tm_ef, t1h, vec, wsdl, pty, rn1, reh, sky, x, y	10분마다 격자 별 호출
Short Forecast	단기 예보 호출	tm_fc, tm_ef, tmx, tmn, vec, wsdl, sky, pty, pop, pcp1, sno, reh,x ,y	3시간마다 격자 별 호출

1.4. 테스트 데이터 및 결과

MUT	Observation Site			
Tid	테스트 입력 / 테스트 시나리오	예상결과	실행결과	테스트 통과
FR-403 UIR-4102	GET /weather/ForecastData?lat=36.5&lon=125.5	200 OK, 초단기, 단기, 중기 예보, 관측적합도 반환	해당 경도/위도의 날씨 데이터를 보여줌	통과
FR-403	GET /weather/ForecastData?lat=1234.5&lon=1234.5 범위 밖 위도/경도 입력	위도 경도가 범위를 벗어나 418 I'm a Teapot, 에러 메시지 출력	한국 내 좌표만 지원합니다. (위도: 33.0~38.7, 경도: 124.0 ~ 132.0) 출력	통과
FR-405	GET /weather/ForecastData?lat=37.5665 &lon=126.9 780했는데 기상청 API에서 타임아웃이 났을 때	502 에러, 서비스 정의 에러, 재시도 가이드	500 에러 출력, 서버에서 에러가 났다고 알려줌	통과
FR-402 UIR-401	GET /observation-sites?keyword=산	200 OK, '산'이라는 키워드 포함하는 관측지 반환	200 OK, 보현산, 설악산을 반환해줌	통과
FR-402	관측지 상세 조회시	200 OK, 이름/위·경도 반환, 해당 위·경도의 날씨 데이터 반환	해당 관측지에 대한 상세한 정보를 볼 수 있음	통과
FR-402 UIR-4103	관측지 상세 조회시	해당 관측지의 관측 후 기 게시글을 리스트로 보여줌	해당 관측지의 게시글 을 확인할 수 있음	통과
UIR-401	임의의 장소 클릭 시	그 장소의 주소, 위도, 경도, 날씨 데이터를 반환	선택한 장소의 정보를 확인할 수 있음	통과
UIR-402	현재 위치 버튼 클릭 시	지도 상 현재 위치가 마커로 클릭됨	마커로 현재 위치 표시됨	통과
UIR-402	안드로이드 위치 권한 허용 동의를 하지 않고 현재 위치 버튼 클릭 시	현재 위치를 사용하려면 위치 권한이 필요하다는 메시지 출력	정상적으로 출력됨	통과
UIR-403	광공해 오버레이 버튼 클릭 시	지도에 광공해가 오버레이 됨	광공해 지수를 확인할 수 있음	통과

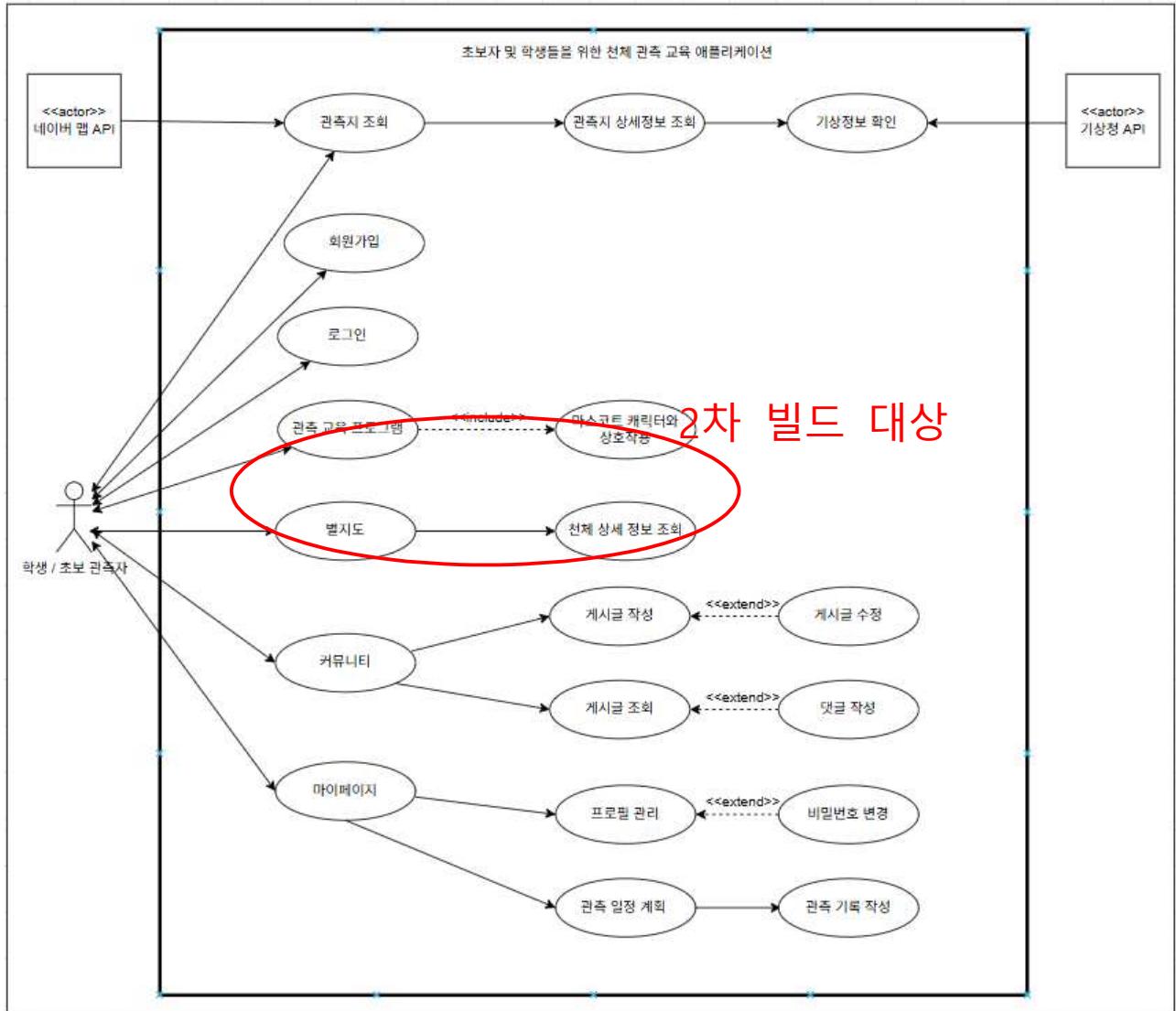
2.1. 2차 빌드 개발 개요

태스크 번호	태스크 명	담당자	시작일	종료일	비고
T-201	오픈소스 분석 및 기술 검토	서범수	25.04.01	25.10.03	Staroid 이식, 직접구현 실패로 인한 일정지연
T-202	별지도 오픈소스 선정 및 연결 (또는 자체 구현)	서범수	25.10.03	25.10.10	T-401
T-203	천체 렌더링 구현(별·행성·위성)	서범수	25.10.03	25.10.10	T-402
T-204	별자리 렌더링 구현(선·그림)	서범수	25.10.03	25.10.10	T-403
T-205	심천체 렌더링 구현(은하·성운·성단)	서범수	25.10.03	25.10.10	-
T-206	좌표계 구현 (적도, 지평좌표계)	서범수	25.10.03	25.10.10	-
T-207	대기효과 구현	서범수	25.10.03	25.10.10	T-403
T-208	천체 검색 기능 구현	서범수	25.10.03	25.10.10	T-403, T-404
T-209	관측 위치 조정 기능 구현	서범수	25.10.03	25.10.10	-
T-210	시간 조정 기능 구현	서범수	25.10.03	25.10.10	-
T-211	시선 트래킹 기능 구현	서범수	25.10.23	25.10.27	-
T-212	SkyMapController 모듈화	서범수	25.10.23	25.10.27	이전 전체
T-213	천체 상세정보 구현	서범수	25.10.29	25.10.29	T-408
T-214	천체명 번역 모듈	서범수	25.10.30	25.10.30	-
T-215	문서 작성	서범수	25.11.10	25.11.10	-

2.2 분석 명세

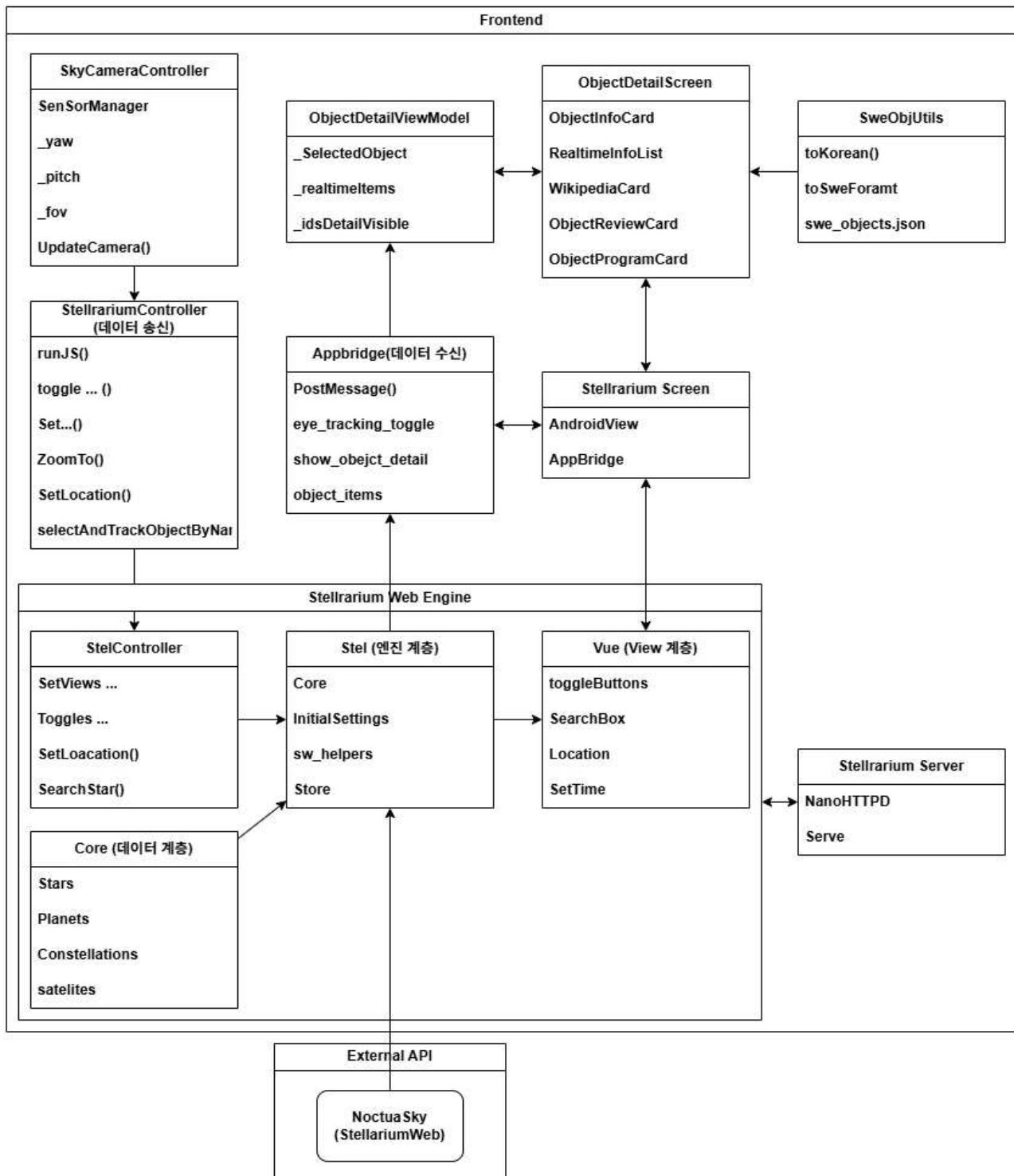
2.2.1 기능 명세

(1) Use Case Diagram (전체 시스템)



2.2.2 구조 명세 (구현 시스템)

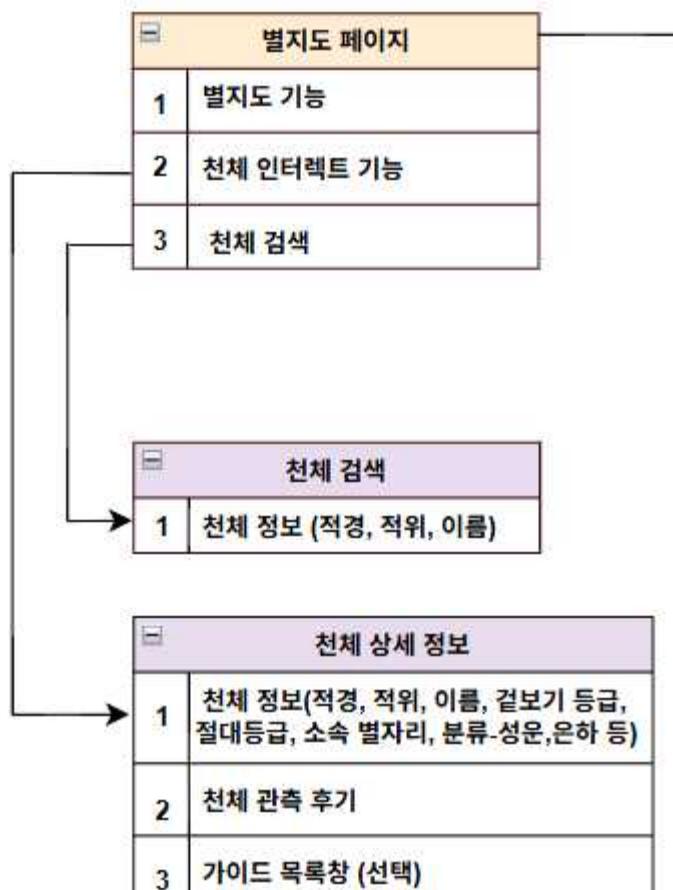
구현된 Skymap System에 대한 구조 명세이다. 오픈소스로 Stellarium-Web-Engine을 사용하였으며, StellariumController에서 데이터에대한 명령을 내리고, 엔진 내부의 stelController에서 처리한다. 또한 내부에서의 데이터 처리 및 제공해주는 데이터들은 AppBrige를 통해 앱으로 데이터가 보내지며, StellariumScreen에서 View를 처리한다.



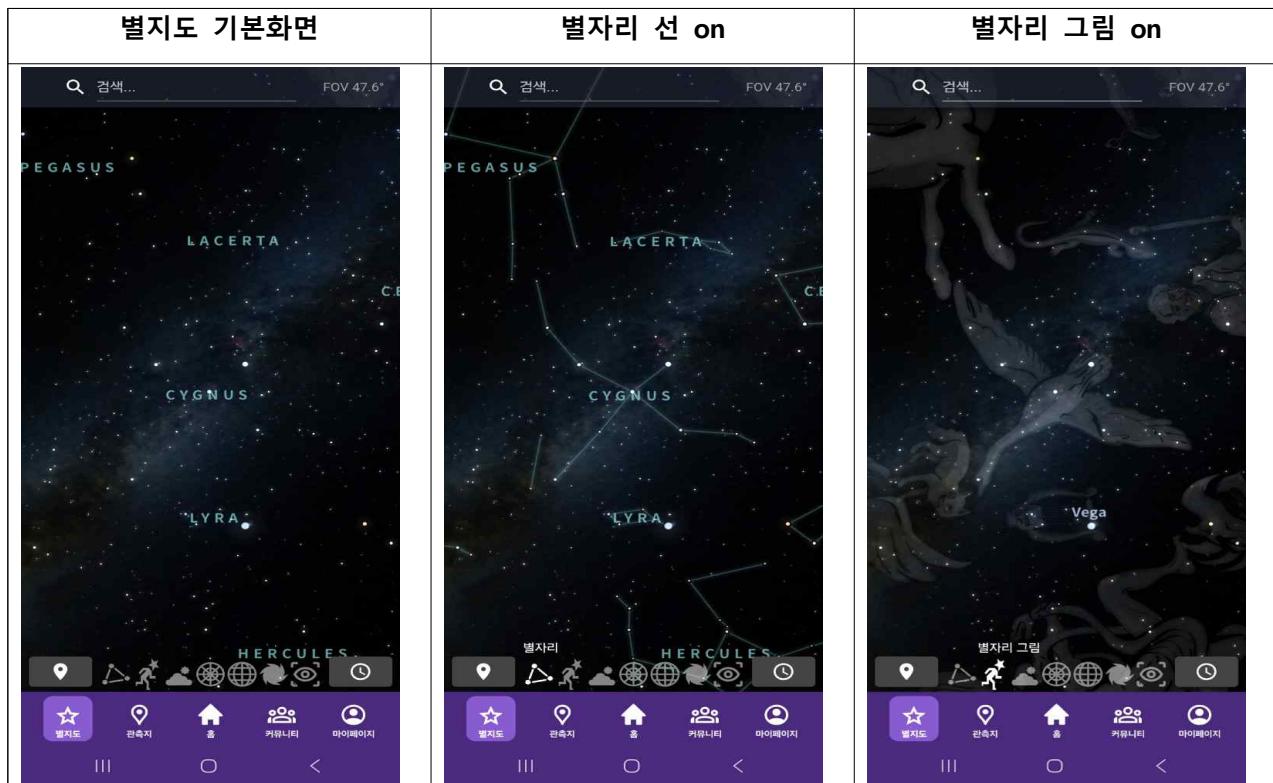
2.3. 설계 명세

2.3.1 메뉴 전개도 (구현 시스템)

- 별지도 및 천체 상세 페이지



2.3.2 사용자 인터페이스 설계 (구현 시스템)



별자리 선택	천체 선택	천체 상세정보

천체검색 UI	시간 설정 UI	관측 위치 UI

2.3.3 핵심 알고리즘 설계 (구현 시스템)

본 시스템의 별지도(SkyMap) 모듈은 모바일 환경에서 전체 데이터를 실시간으로 렌더링하고,

센서 정보 및 사용자 인터랙션을 반영하여 동적인 시야를 구성하도록 설계되었다.

별자리·행성·심천체 등의 시각화는 외부 렌더링 엔진(Stellarium Web Engine)을 기반으로 수행되며,

본 프로젝트의 구현 범위는 Android-JS 간 통신 구조, 센서 기반 시야 제어, 천체 정보 동기화 및 UI 반영 로직에 집중한다. 엔진 내부의 WebGL 렌더링 로직은 외부 모듈로 취급하며, 설계 범위에서 제외한다.

2.3.3.1 Android-JS 양방향 통신 알고리즘

구분	내용
목적	WebView 내부의 JavaScript 엔진과 Android Compose 환경 간의 양방향 데이터 교환 및 이벤트 처리를 담당한다.
핵심 구성요소	<ul style="list-style-type: none"> AppBridge : JS → Android 데이터 수신 인터페이스 StellariumController : Android → JS 명령 제어 클래스 ObjectDetailViewModel : 상태 데이터 관리 및 UI 반영
동작 과정	<ol style="list-style-type: none"> 엔진 초기화 완료 시, JavaScript에서 window.AndroidBridge.postMessage()를 통해 JSON 메시지 전송 Android의 AppBridge가 @JavascriptInterface로 메시지를 수신하고 JSON 파싱 수행 type 필드 값에 따라 이벤트 분기 처리 <ul style="list-style-type: none"> - stel_ready : 엔진 초기화 완료 신호 - show_object_detail : 천체 상세정보 표시 요청 - object_items : 실시간 천체 데이터 갱신 (밝기, 거리 등) - eye_tracking_toggle : 자이로 기반 시선 트래킹 제어 파싱된 데이터는 ObjectDetailViewModel의 StateFlow로 전달되어 Compose UI(ObjectDetailScreen)에서 자동 갱신
특징	<ul style="list-style-type: none"> - 메시지 기반의 비동기 처리 구조로 설계되어 데이터 흐름이 명확함 - AppBridge는 Android와 JS 간의 데이터 변환 및 전달을 담당하는 핵심 게이트웨이 역할 수행 - UI는 StateFlow를 통해 상태 변경에 자동 반응하도록 구성되어 실시간성과 일관성을 모두 확보
다이어그램	<pre> sequenceDiagram participant JS_Engine as JavaScript (Stellarium Engine) participant AB as AppBridge (Android) participant VM as ViewModel participant UI as Compose UI participant SC as StellariumController JS_Engine->>AB: postMessage(JSON) activate AB AB-->>VM: 파싱 후 상태 갱신 activate VM VM-->>UI: StateFlow → UI 자동 반영 activate UI UI-->>SC: 사용자 조작 이벤트 activate SC SC-->>JS_Engine: JS 함수 호출 (e.g. toggleConstellations) deactivate SC deactivate UI deactivate VM deactivate AB deactivate JS_Engine </pre>

2.3.3.2 센서 기반 시야 제어 알고리즘

구분	내용
목적	모바일 기기의 자이로스코프 센서 데이터를 이용해 사용자의 실제 방향에 맞게 천체 시야를 실시간으로 회전시킨다.
핵심 구성 요소	<ul style="list-style-type: none"> SkyCameraController : 센서 데이터 수집 및 각도 계산 담당 SensorManager : Rotation Vector 센서 관리 StellariumController : JS 엔진으로의 뷰포트 제어 명령 전달
동작 과정	<ol style="list-style-type: none"> ① Sensor.TYPE_ROTATION_VECTOR 이벤트 발생 ② getRotationMatrixFromVector()로 회전 행렬 계산 ③ remapCoordinateSystem()으로 좌표계 재매핑 후 getOrientation()으로 방위각·고도 계산 ④ 라디안을 도(degree)로 변환해 yaw, pitch 상태값으로 저장 ⑤ combine(yaw, pitch) Flow를 통해 StellariumController.setViewDirection(yaw, pitch) 호출 트리거 ⑥ JS의 \$stelController.setViewDirection() 호출로 엔진 내 카메라 시점 즉시 갱신
특징	<ul style="list-style-type: none"> - SENSOR_DELAY_GAME 주기로 시야 이동을 부드럽게 유지 - CoroutineScope + StateFlow 기반 비동기 구조로 UI 반응성과 성능을 모두 확보
다이어그램	<pre> sequenceDiagram participant SM1 as SensorManager (Rotation Vector) participant SKC as SkyCameraController participant VM as ViewModel (yaw/pitch Flow) participant SC as StellariumController participant JE as JS Engine (\$stelController) SM1->>SKC: TYPE_ROTATION_VECTOR 이벤트 activate SKC SKC-->>VM: 회전 행렬 계산 → 좌표계 재매핑 activate VM VM-->>SC: yaw, pitch 값 생성 activate SC SC-->>JE: setViewDirection(yaw, pitch) deactivate SC deactivate VM deactivate SKC deactivate JE </pre>

2.3.3.3 천체 정보 표시 및 UI 반영 알고리즘

구분	내용
목적	엔진에서 선택된 천체의 상세 정보를 Android UI에 실시간으로 반영한다.
핵심 구성요소	AppBridge : JS로부터 천체 선택 이벤트(show_object_detail) 수신 ObjectDetailViewModel : 천체 상세 데이터 관리 및 상태 저장 ObjectDetailScreen : Compose 기반 UI로 정보 표시
동작 과정	① 사용자가 Stellarium 화면에서 천체를 터치하거나 선택 ② JS 엔진이 "show_object_detail" 이벤트를 발생시키고, payload에 천체명·유형·위키 요약·별칭 목록 등을 포함 ③ AppBridge가 메시지를 수신해 JSON을 파싱하고, SkyObjectDetail 데이터 클래스로 변환 ④ 변환된 데이터가 ObjectDetailViewModel에 전달되어 상태(StateFlow)가 갱신 ⑤ Compose UI(ObjectDetailScreen)가 상태 변화를 감지해 자동으로 화면 갱신 ⑥ UI에 표시되는 정보: 천체 이름·유형, 실시간 좌표/밝기, 위키 요약, 관련 관측 후기 및 교육 콘텐츠 목록
특징	StateFlow-Compose 구조로 즉각적인 UI 반응성 확보 ViewModel 중심 데이터 파이프라인 으로 일관성 및 유지보수성 향상 비동기 처리로 JS–Android–UI 간 데이터 갱신 지연 최소화
다이어그램	<pre> sequenceDiagram participant JS_Engine as JS Engine (Stellarium) participant AppBridge as AppBridge (Android) participant ViewModel as ObjectDetailViewModel participant UI as Compose UI (ObjectDetailScreen) participant User JS_Engine->>AppBridge: show_object_detail(JSON payload) activate AppBridge AppBridge->>ViewModel: JSON 파싱 → SkyObjectDetail 변환 activate ViewModel ViewModel->>UI: ViewModel에 데이터 전달 activate UI UI->>User: StateFlow 갱신 → UI 자동 반영 activate User User-->>UI: 천체 이름·밝기·요약 등 표시 </pre> <p>The diagram illustrates the data flow for displaying celestial object details. It starts with the JS Engine (Stellarium) sending a 'show_object_detail' event with a JSON payload to the AppBridge (Android). The AppBridge performs JSON parsing and converts the data into a SkyObjectDetail object. This data is then passed to the ObjectDetailViewModel. The ViewModel triggers a StateFlow update, which results in the UI (Compose UI) automatically reflecting the changes. Finally, the UI displays the celestial object's name, brightness, and summary information to the User.</p>

2.3.3.4 천체명 번역 및 로컬라이징 알고리즘

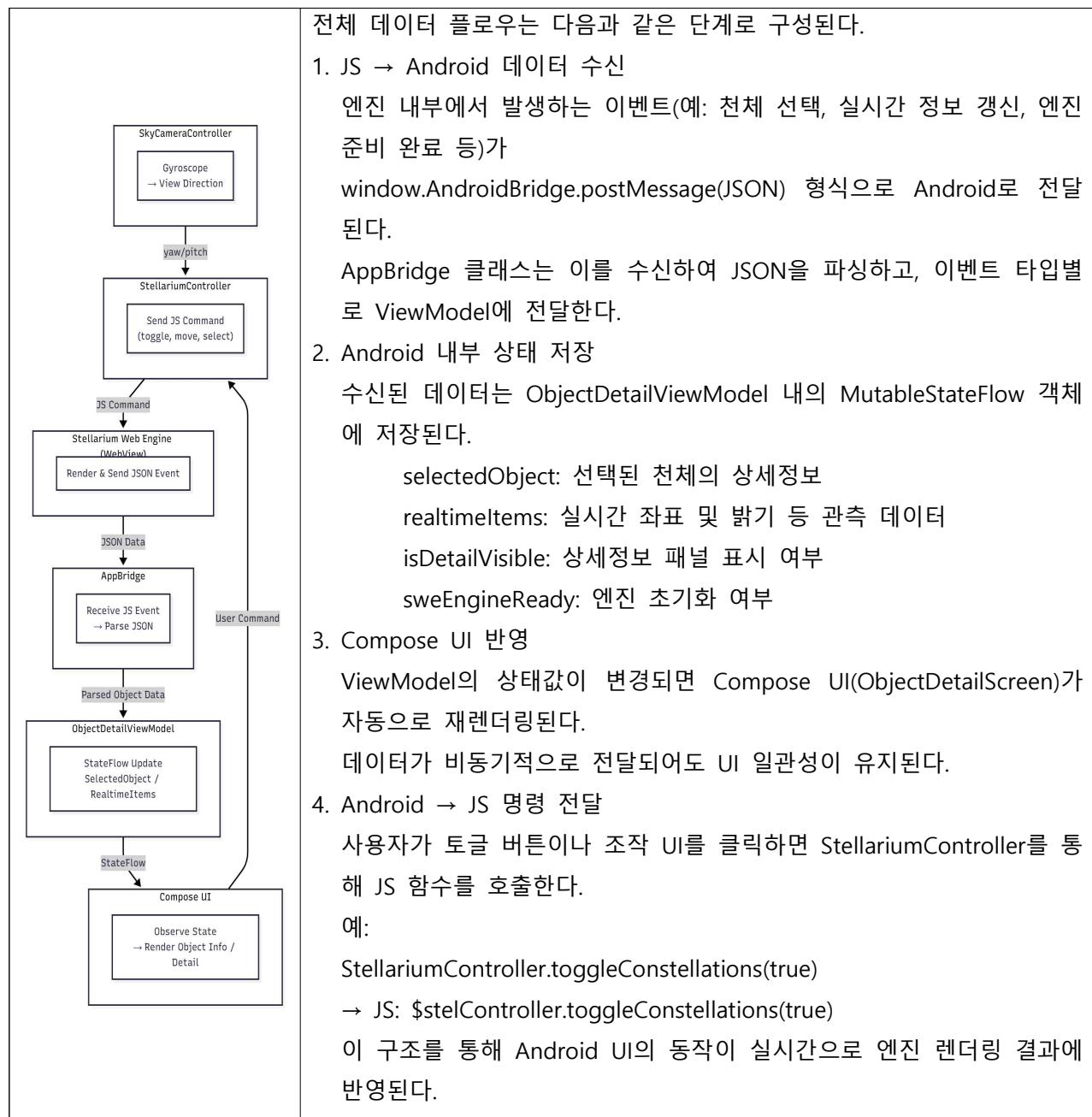
구분	내용
목적	엔진의 영어 천체명(NAME Vega)을 한국어(베가)로 자동 변환하여 사용자 가독성과 몰입도를 향상시킨다.
핵심 구성요소	SweObjUtils : 천체명 변환 및 맵핑 테이블 관리 유틸리티
동작 과정	<ul style="list-style-type: none"> ① 앱 초기 실행 시 assets/swe_objects.json 파일을 로드 ② "영문명-한글명" 맵핑 데이터를 메모리에 HashMap 형태로 캐싱 ③ 변환 요청 시 toKorean("NAME Deneb") → "데네브" 반환 ④ 역변환 요청 시 toSweFormat("베가") → "NAME Vega" 반환 ⑤ 변환 결과는 ObjectInfoCard, 검색 모듈 등 모든 UI에서 공통 사용되어 다국어 표현의 일관성을 유지
특징	<ul style="list-style-type: none"> 로컬 JSON 기반 오프라인 맵핑으로 빠른 응답 속도 확보 메모리 캐싱으로 중복 I/O 최소화 양방향 변환(영문↔한글)을 모두 지원해 글로벌 호환성 강화
ダイアグラム	<pre> sequenceDiagram participant AS1 as App Start participant SU as SweObjUtils participant OS as ObjectInfoCard / Search UI AS1->>SU: assets/swe_objects.json 로드 activate SU SU->>SU: 영문-한글 맵핑 테이블 캐싱 deactivate SU SU->>OS: toKorean("NAME Vega") 호출 OS-->>SU: "베가" 반환 SU->>OS: toSweFormat("데네브") 호출 OS-->>SU: "NAME Deneb" 반환 </pre> <p>The diagram illustrates the workflow for translating celestial names. It starts with the 'App Start' component loading the 'assets/swe_objects.json' file into the 'SweObjUtils' component. This triggers the caching of the English-Korean mapping table. Subsequently, when 'ObjectInfoCard / Search UI' requests the translation of 'NAME Vega' using the 'toKorean' method, 'SweObjUtils' returns the Korean name '베가'. Conversely, when 'ObjectInfoCard / Search UI' requests the reverse translation of '데네브' using the 'toSweFormat' method, 'SweObjUtils' returns the English name 'NAME Deneb'.</p>

2.3.4 데이터 설계 (구현 시스템)

본 시스템의 데이터 설계는 Android-JS 통신 구조를 중심으로 실시간 천체 데이터의 수집·전달·표시가 유기적으로 동작하도록 구성되었다. 데이터의 대부분은 비영속적(Non-persistent)이며, WebView 엔진에서 발생하는 이벤트 데이터를 실시간으로 반영하는 방식으로 처리된다.

일부 천체 정보는 JSON 기반의 로컬 데이터(한글 변환 테이블, DSO 데이터 등)로 구성되어 있으며, 데이터베이스보다는 ViewModel-Flow-Compose 구조에 의한 상태 관리 중심으로 설계되었다.

2.3.4.1 데이터 흐름 구조 및 구조도



2.3.4.2 주요 데이터 구조 정의

구조체명	설명	주요 필드 / 구성 요소	예시
SkyObjectDetail	천체의 상세 정보를 저장하는 데이터 클래스	<ul style="list-style-type: none"> name: 천체 이름 (예: "NAME Vega") - type: 천체 유형 (Star, Galaxy 등) - wikipediaSummary: 위키 요약 정보 - otherNames: 별칭 목록 (List<String>) 	<pre>data class SkyObjectDetail(val name: String, val type: String, val wikipediaSummary: String val otherNames: List<String>)</pre>
ObjectItem	천체의 실시간 정보 항목을 표현하는 데이터 클래스	<ul style="list-style-type: none"> - key: 정보 항목명 (예: "Apparent Magnitude") - value: 해당 값 (예: "0.03 mag") 	<pre>data class ObjectItem(val key: String, val value: String)</pre>
센서 데이터 흐름	기기의 방향/시야 정보를 관리하고 결합하여 실시간 시야 제어에 활용	<ul style="list-style-type: none"> - yaw: 방위각 - pitch: 고도 - fov: 시야각 combine(yaw, pitch)으로 결합하여 시야 제어 수행 	StateFlow로 상태 관리 예: combine(yaw, pitch) { ... }
로컬 JSON 데이터 (SweObjU tils)	Stellarium 엔진 내 영문 천체명과 한글명을 매핑하는 JSON 파일	경로: assets/swe_objects.json 메모리 내 HashMap 캐싱으로 빠른 접근 지원	{ "NAME Deneb": "데네브", "NAME Vega": "베가", "NAME Altair": "알타이르" }

2.3.4.3 데이터 저장 및 관리 방식

구분	데이터 유형	저장 위치	지속성	주요 목적
천체 기본 정보	SkyObjectDetail	ViewModel (메모리)	일시적	선택된 천체 정보 표시
실시간 항목	ObjectItem List	ViewModel (메모리)	일시적	관측 중 실시간 업데이트
센서 정보	yaw, pitch, fov	SkyCameraController	일시적	시야 방향 계산
번역 매핑	swe_objects.json	앱 Assets	영속적	영문-한글 이름 변환
사용자 입력	위도/경도, 시간	Compose UI	일시적	관측 환경 설정

2.3.4.4 데이터 통신 포맷 (JSON 규격)

1. JS → Android 전송 데이터 예시

```
{  
    "type": "show_object_detail",  
    "payload": {  
        "name": "NAME Vega",  
        "type": "Star",  
        "wikipediaData": "Vega는 거문고자리의 알파별로...",  
        "otherNames": ["Alpha Lyrae", "HD 172167"],  
        "show": true  
    }  
}
```

2. Android → JS 명령 데이터 예시

```
$stelController.setViewDirection(45.2, -10.4);  
$stelController.setTime(new Date("2025-05-25T21:00:00Z"));  
$stelController.toggleConstellations(true);
```

2.3.4.5 설계적 특징 요약

비영속 데이터 중심 구조: 실시간 반응성과 성능을 우선으로 하여 메모리 기반 상태 관리 위주로 설계
표준화된 JSON 인터페이스: JS ↔ Android 간 데이터 호환성과 확장성 확보

단방향 데이터 흐름 유지: JS → ViewModel → Compose UI의 구조적 일관성

모듈화된 관리: 천체 정보, 센서 정보, 번역 정보가 독립적으로 관리되어 유지보수가 용이

확장 가능성: 향후 교육 콘텐츠, AR 기반 관측 기능 등과의 데이터 연계가 용이

2.4. 테스트 데이터 및 결과

MUT	Skymap			
Tid	테스트 입력 / 테스트 시나리오	예상결과	실행결과	테스트 통과
UIR-301	별자리와 주요 천체가 시각적으로 구분되어 표시되는지 확인	별자리 선과 항성이 명확히 구분되어 표시됨	실제 렌더링 화면에서 구분 확인	통과
UIR-302	교육 가이드 진행 시 시야 자동 이동 테스트	가이드 단계에 맞춰 시야가 자동 이동	StellariumController를 통한 시야 이동 정상 작동	통과
UIR-303	천체 터치 시 이름·밝기·거리 정보 표시 확인	선택 천체의 기본 정보 팝업 표시	팝업 정상 출력 및 데이터 정확	통과
UIR-304	다크테마 적용 상태에서 시각적 검증	어두운 배경과 명암비가 적절히 유지됨	저휘도 모드 정상 동작	통과
UIR-305	줌 인/아웃 시 텍스트 크기 유지 여부 확인	모든 텍스트가 고정 크기(14sp)로 유지	확대/축소 후에도 크기 동일	통과
UIR-306	적경·적위 격자 On/Off 토글 테스트	격자 토글 시 표시/숨김 전환됨	정상 작동	통과
UIR-307	방위각·고도 격자 표시 토글 테스트	격자 표시 및 숨김 정상 동작	정상 작동	통과
UIR-308	별지도에 N/E/S/W 방향 표시 확인	화면 상단에 방향 텍스트 표시	올바른 방향 표시 확인	통과
UIR-309	별자리 선 토글 기능 테스트	연결선 표시/숨김 전환됨	정상 작동	통과
UIR-310	별자리 그림 토글 기능 테스트	그림 이미지 표시/숨김 가능	정상 작동	통과
UIR-311	대기 효과 On/Off 테스트	Atmosphere 렌더링 효과 변경 확인	색상 변화 정상 반영	통과

UIR-312	심천체 표시 옵션 테스트	온하·성운·성단 표시 여부 제어 가능	설정에 따라 표시 정상	통과
UIR-313	관측 위치 직접 입력 및 지도 수정 테스트	위도·경도 입력 시 시야 재설정	위치 반영 정상	통과
UIR-314	시선 트래킹 기능 활성화 테스트	자이로센서 입력에 따라 시야 이동	실시간 추적 정상	통과
UIR-315	시간 변경 UI 작동 테스트	시간 변경 시 하늘 상태 갱신	해당 시점의 별자리 렌더링	통과
UIR-316	천체 상세보기 인터페이스 표시 확인	상세 정보 및 교육 콘텐츠 표시	정보 표시 정확	통과
FR-301	구면좌표 기반 별자리·격자 렌더링 확인	별자리와 격자가 구면상에 정확히 표시됨	시각 정합성 확보	통과
FR-302	센서 정합성 검증 (자이로/GPS 연동)	기기 방향과 시야 방향 일치	실시간 동기화 정상	통과
FR-303	천체 터치 인식 정확도 테스트	터치 좌표의 천체 식별 정확	$\pm 1^\circ$ 이내 정합	통과
FR-304	렌더링 위치 오차 검증	실제 별자리 위치와 비교 시 오차 $\pm 1^\circ$ 이내	정합성 확보	통과
FR-305	관측 위치 변경 처리	위도·경도 변경 시 천구 중심 재계산	재설정 정상 반영	통과
FR-306	천구 경사 보정 기능 검증	위도 변화에 따른 기울기 반영	정확한 보정 확인	통과
FR-307	천체 검색 기능 테스트	입력된 천체명으로 좌표 탐색 및 이동	좌표 계산 정확	통과
FR-308	핀치 줌 제스처 테스트	두 손가락으로 확대/축소	FPS 유지,	통과

		부드럽게 동작	자연스러움	
FR-309	시간 변경 처리 검증	시간 변경 시 천체 위치 재계산	즉시 반영	통과
FR-310	별 등급 필터링 기능 테스트	6등성 이하 별만 표시	필터 정상 작동	통과
FR-311	대기효과 렌더링 검증	Atmosphere On/Off 시 셰이더 변화	시각적 효과 확인	통과
FR-312	심천체 렌더링 검증	은하·성단·성운 렌더링 정상	DSO 데이터 반영 정상	통과
FR-313	시선 트래킹 처리 검증	자이로센서 값에 따라 View Matrix 갱신	실시간 반응	통과
FR-314	천체 고정(Pin) 기능 테스트	선택 천체가 화면 중앙 유지	정상 고정	통과
FR-315	천체 출몰 시각 계산 테스트	관측 위치·시간 기준으로 Rise/Set 계산	결과 정확	통과
FR-316	렌더링 주기 동기화 테스트	30Hz 주기로 센서/렌더링 동기화	부드러운 시야 유지	통과

2.5 구현 참고 사항

Stellarium Web Engine 라이선스 관련 (AGPL v3)

- 본 시스템의 별지도(Skymap) 기능은 Stellarium Web Engine (SWE)을 기반으로 구현되었다.
- SWE는 GNU Affero General Public License v3 (AGPL v3) 하에 배포되는 오픈소스 소프트웨어로, 해당 라이선스 조건에 따라 소스 코드 수정 및 배포 시 전체 공개 의무가 발생한다.
- 본 프로젝트에서는 Stellarium Labs(Fabien Chéreau)로부터 비영리 교육 목적의 사용 허가를 정식으로 받았으며,
- 소스 코드 수정 내역은 내부 저장소에 별도 관리하고 있다.
- 단, AGPL 조항에 따라 상용 목적 배포는 제한되며,
- 추후 공개 배포 시 소스코드와 라이선스 고지를 함께 포함해야 한다.

외부 API 및 기술적 안정성

- 현재 사용 중인 SWE 내부 API 중 일부는 공식 문서화가 완료되지 않은 비공식 인터페이스(unstable API)로 향후 엔진 버전 업데이트 시 호환성 문제가 발생할 가능성이 있다.
- 또한, 별이름 검색(searchByName) 등 특정 기능은 NoctuaSky API를 통해 구현되었으나, 해당 API는 안정화(stable)되지 않은 테스트용 앤드포인트로, 네트워크 불안정 시 연결이 자주 끊기거나 결과 응답이 지연될 수 있음을 확인하였다.

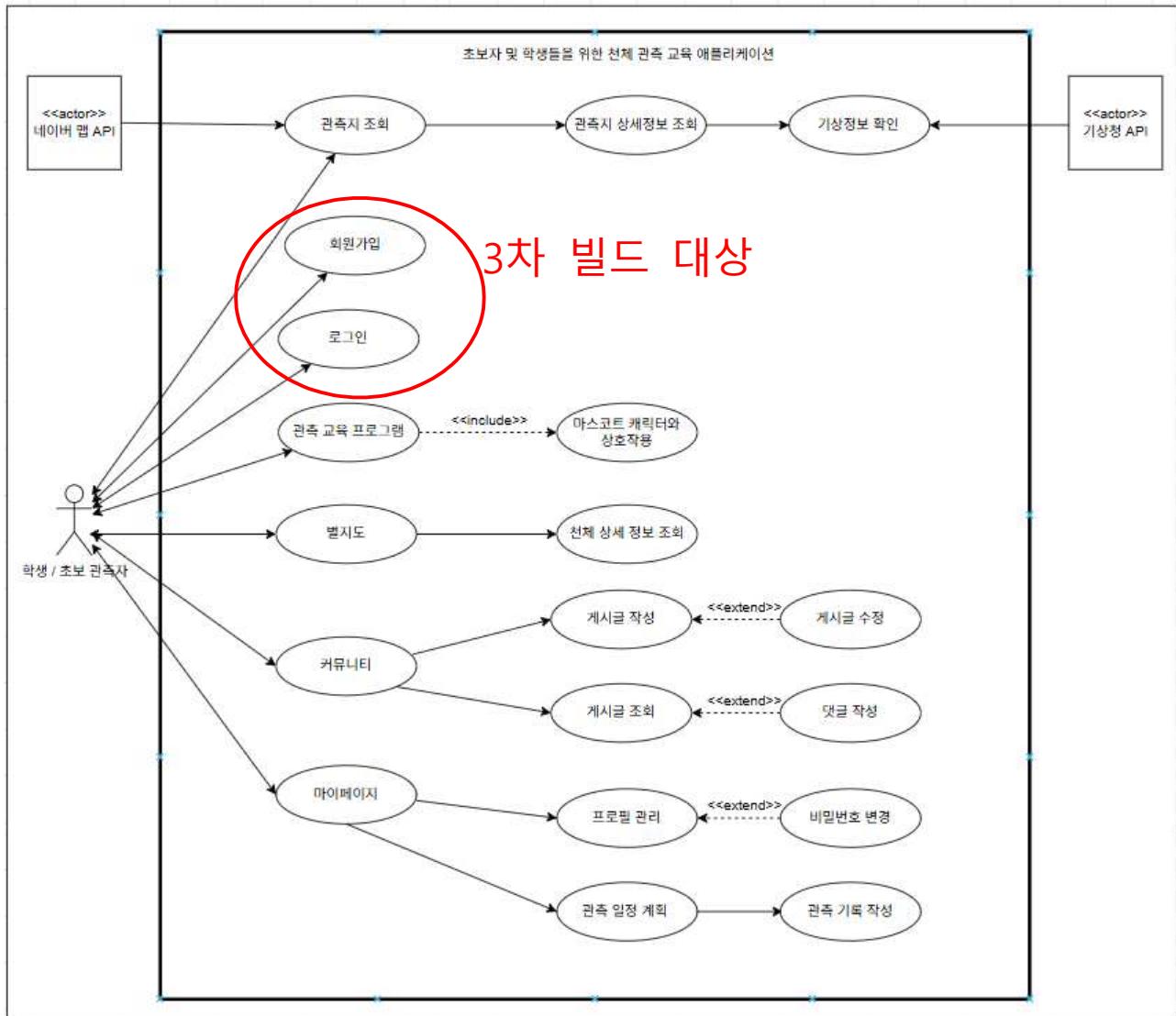
3.1. 3차 빌드 개발 개요

태스크 번호	태스크 명	담당자	시작일	종료일	비고
T-301	회원가입	윤태영	25.07.01	25.07.10	-
T-302	로그인	윤태영	25.07.05	25.07.14	-
T-303	이메일 인증	윤태영	25.07.10	25.07.18	-
T-304	아이디 찾기	윤태영	25.07.18	25.07.25	-
T-305	비밀번호 찾기	윤태영	25.07.25	25.08.01	-
T-306	로그인 UI	서범수	25.07.20	25.08.05	이전 Skymap 연동문제로 연기됨.
T-307	회원가입 UI	서범수	25.07.25	25.08.10	-
T-308	약관 동의 UI	서범수	25.08.01	25.08.15	-
T-309	이메일 인증 UI	서범수	25.08.10	25.08.20	-
T-310	이메일 찾기 UI	서범수	25.08.15	25.08.25	-
T-311	비밀번호 찾기 UI	서범수	25.08.20	25.08.30	-
T-312	TokenDataStore	서범수	25.08.25	25.09.05	-
T-313	Auth Controller 연동	서범수	25.09.01	25.09.15	-
T-314	Auth 문서 작성	서범수, 윤태영	25.09.10	25.09.21	-

3.2. 분석 명세

3.2.1 기능 명세

(1) Use Case Diagram (전체 시스템)



(2) Use Case 설명서 (구현 시스템)

- 회원가입 UseCase

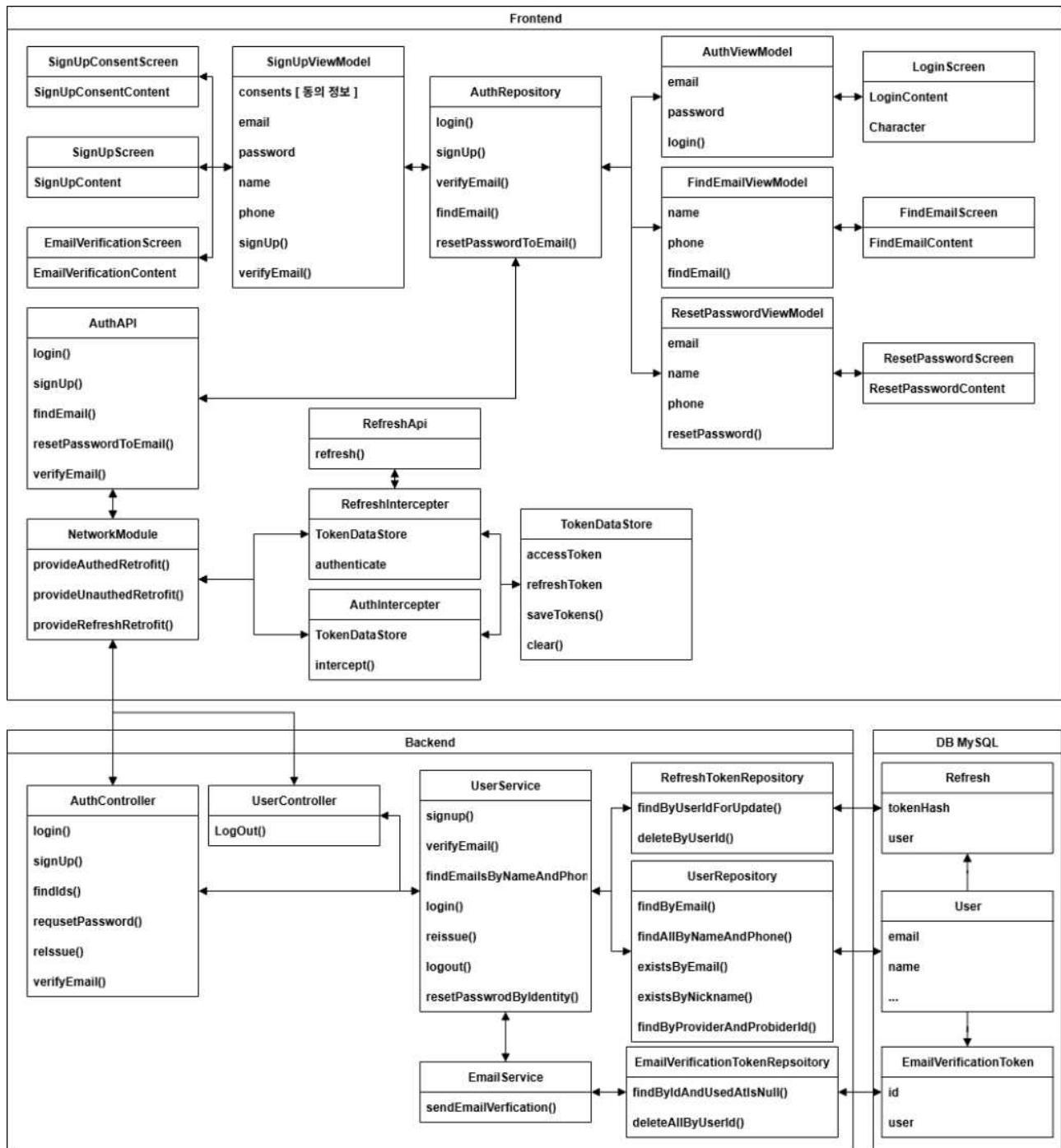
Use Case Name: 회원가입	ID: 1 UC-007	Important Level: LOW
Primary Actor: 사용자		Use case type: Detail, Essential
Stakeholders and Interests:		
<p>사용자: 애플리케이션 사용을 위한 개인 계정 생성 필요</p> <p>시스템: 사용자 정보를 안전하게 저장하고, 인증 과정을 통해 서비스 접근을 통제</p>		
Brief Description:		
<ul style="list-style-type: none"> - 사용자가 개인정보를 입력하여 회원가입을 완료하고 앱의 기능을 사용할 수 있다. 		
Trigger: 사용자가 회원가입 버튼을 클릭		
Type: External		
Relationships:		
<p>Associations: 학생 / 초보 관측자</p> <p>Include: 아이디 중복검사</p>		
Normal Flow of Events:		
<ol style="list-style-type: none"> 1. 사용자가 회원가입 버튼을 클릭한다. 2. 이름, 이메일, 비밀번호 등의 필수 항목을 입력한다. 3. 시스템은 입력된 이메일의 중복 여부를 확인한 후, 이메일 인증을 메일을 보낸다. 4. 사용자는 이메일 인증을 완료한 후 약관에 동의하고 '회원가입' 버튼을 클릭한다. 5. 회원가입이 성공적으로 처리되면 로그인 화면으로 이동한다. 		
Subflows:		
<p>S-1: 아이디 중복 검사 – 입력과 동시에 실시간 검사 후 결과를 표시한다.</p> <p>S-2: 약관 동의 – 전체 동의 또는 개별 동의 항목 확인 후 제출한다.</p>		
Alternative / Exceptional Flows:		
<p>E-1: 이메일이 중복되었을 경우 "이미 사용중인 아이디입니다." 메시지를 출력한다.</p> <p>E-2: 필수 입력 항목 누락 시 오류 메시지와 함께 다음 단계로 진행을 차단한다.</p>		

- 로그인 UseCase

Use Case Name: 로그인	ID: 1 UC-006	Important Level: LOW
Primary Actor: 사용자		Use case type: Detail, Essential
Stakeholders and Interests:		
<p>사용자: 본인 계정으로 안전하게 앱 기능을 사용하고 싶다.</p> <p>시스템: 안전한 인증/인가, 비정상 접근 차단, 이메일 미인증 사용자 제한</p>		
Brief Description:		
<ul style="list-style-type: none"> - 사용자가 이메일/비밀번호 또는 구글 연동을 통해 본인임을 인증하고, 앱 기능을 사용할 수 있다. 		
Trigger: 사용자가 로그인 버튼을 클릭		
Type: External		
Relationships:		
Associations: 학생 / 초보 관측자		
Extend: 비밀번호 재설정, 이메일 인증 재전송		
Normal Flow of Events:		
<ol style="list-style-type: none"> 1. 사용자가 로그인 버튼을 클릭한다. 2. 이메일과 비밀번호를 입력한다. 3. 시스템은 해당 이메일 계정 존재 여부와 이메일 인증 여부를 확인한다. 4. 비밀번호가 올바르면 시스템은 토큰을 발급하여 클라이언트에 전달한다. 5. 사용자는 로그인 성공 상태로 홈 화면으로 이동한다. 		
Subflows:		
S-1: 이메일 인증 확인		
<ol style="list-style-type: none"> 1. 로그인 시 이메일 미인증이면 시스템은 인증 필요 메시지와 인증 메일 재전송 옵션을 제공한다. 2. 사용자가 인증을 완료하면 다시 로그인을 진행할 수 있다. 		
S-2: 비밀번호 재설정 진입		
<ol style="list-style-type: none"> 1. 사용자가 "비밀번호 찾기"를 선택한다. 2. 시스템은 비밀번호 재설정 플로우로 이동시킨다. 		
Alternative / Exceptional Flows:		
E-1: 이메일 또는 비밀번호가 일치하지 않을 경우 "이메일 또는 비밀번호가 일치하지 않습니다." 에러 메시지 출력		
E-2: 계정은 있으나 이메일 인증을 하지 않은 경우 "이메일 인증이 필요합니다" 메시지 출력		

3.2.2 구조 명세 (구현 시스템)

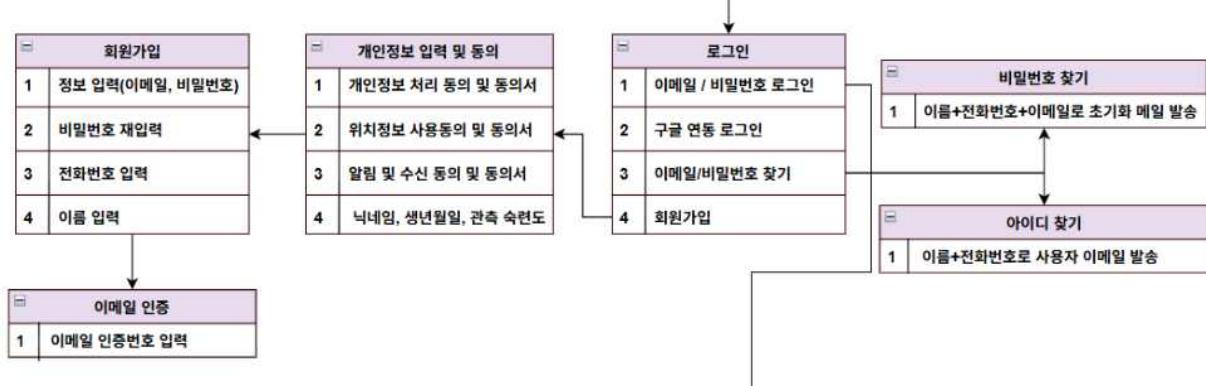
구현된 Auth 시스템에 대한 구조도. 해당 문서에서는 NetworkModule과 Auth Interceptor까지의 관계가 표현되어 있다. 이후 타 빌드에서 API를 사용할 경우, NetworkModule과 관련된 내용은 구조에서 제외하고 작성할 예정이다.



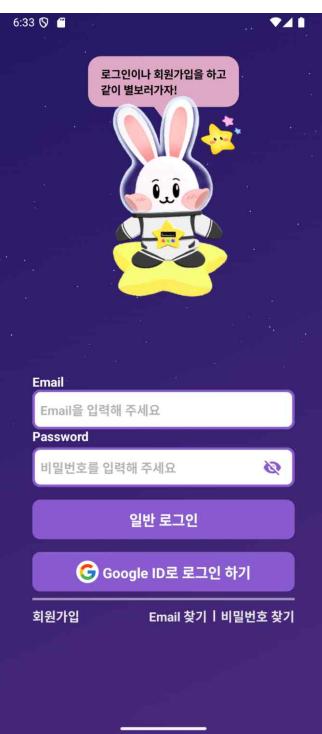
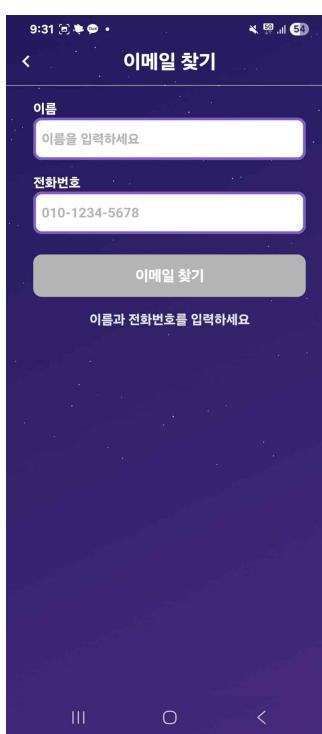
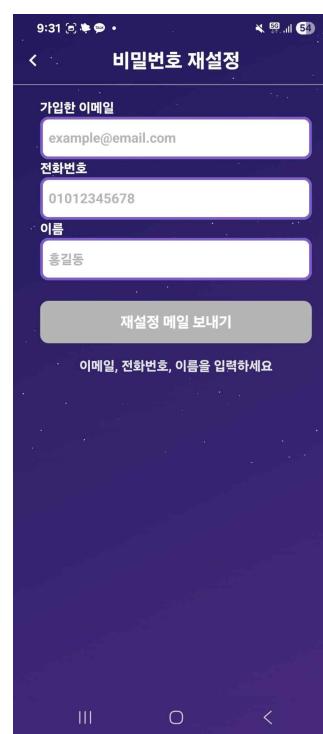
3.3. 설계 명세

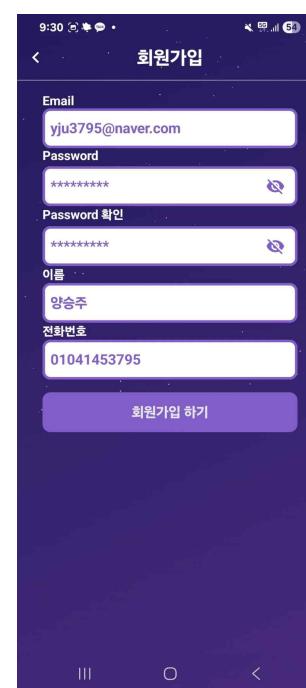
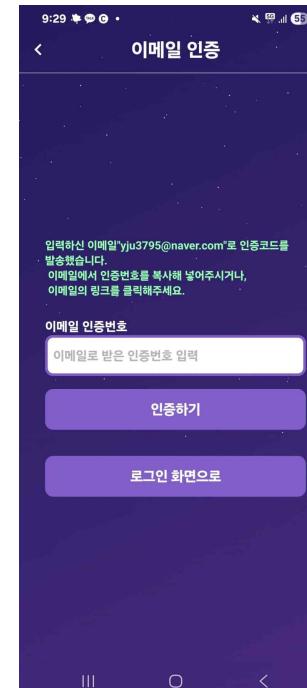
3.3.1 메뉴 전개도 (구현 시스템)

- 회원가입 및 로그인 메뉴 전개도



3.3.2 사용자 인터페이스 설계 (구현 시스템)

로그인 화면	이메일 찾기 화면	비밀번호 재설정 화면
 <p>로그인이나 회원가입을 하고 같이 봐보러가자!</p> <p>Email Email을 입력해 주세요</p> <p>Password 비밀번호를 입력해 주세요</p> <p>일반 로그인</p> <p>Google ID로 로그인 하기</p> <p>회원가입 Email 찾기 비밀번호 찾기</p>	 <p>이름 이름을 입력하세요</p> <p>전화번호 010-1234-5678</p> <p>이메일 찾기</p> <p>이름과 전화번호를 입력하세요</p>	 <p>가입한 이메일 example@email.com</p> <p>전화번호 01012345678</p> <p>이름 홍길동</p> <p>재설정 메일 보내기</p> <p>이메일, 전화번호, 이름을 입력하세요</p>

회원가입 중 약관동의	회원정보 입력	이메일 인증
 <p>교육 - 기술적 조치: SSL 암호화 전송, 개인정보 접근 통제, 웹입 차단 시스템 운영 - 물리적 조치: 서버실 접근 통제, 보안 설비 설치 및 운영</p> <p>8. 개인정보 보호책임자 및 담당자 - 책임자: 서범수 - 담당 부서: 고객지원팀 - 연락처: privacy@example.com / 02-123-4567</p> <p>9. 고지 및 변경 - 본 방침은 2025년 5월 25일부터 시행합니다. - 법령 및 내부 정책 변경 시 앱 공지사항 또는 이메일로 안내드립니다.</p> <p>개인정보 처리방침 전체 동의 (필수) <input checked="" type="checkbox"/></p> <p>프로필 정보(닉네임·생년월일 등) 제공 동의 (필수) <input checked="" type="checkbox"/></p> <p>위치정보 수집·이용 동의 (선택) <input checked="" type="checkbox"/></p> <p>알림(푸시·이메일) 수신 동의 (선택) <input checked="" type="checkbox"/></p> <p>다음으로</p>	 <p>회원가입</p> <p>Email yju3795@naver.com</p> <p>Password *****</p> <p>Password 확인 *****</p> <p>이름 양승주</p> <p>전화번호 01041453795</p> <p>회원가입 하기</p>	 <p>이메일 인증</p> <p>입력하신 이메일 'yju3795@naver.com'로 인증코드를 발송했습니다. 이메일에서 인증번호를 복사해 넣어주시거나, 이메일의 링크를 클릭해주세요.</p> <p>이메일 인증번호 이메일로 받은 인증번호 입력</p> <p>인증하기</p> <p>로그인 화면으로</p>

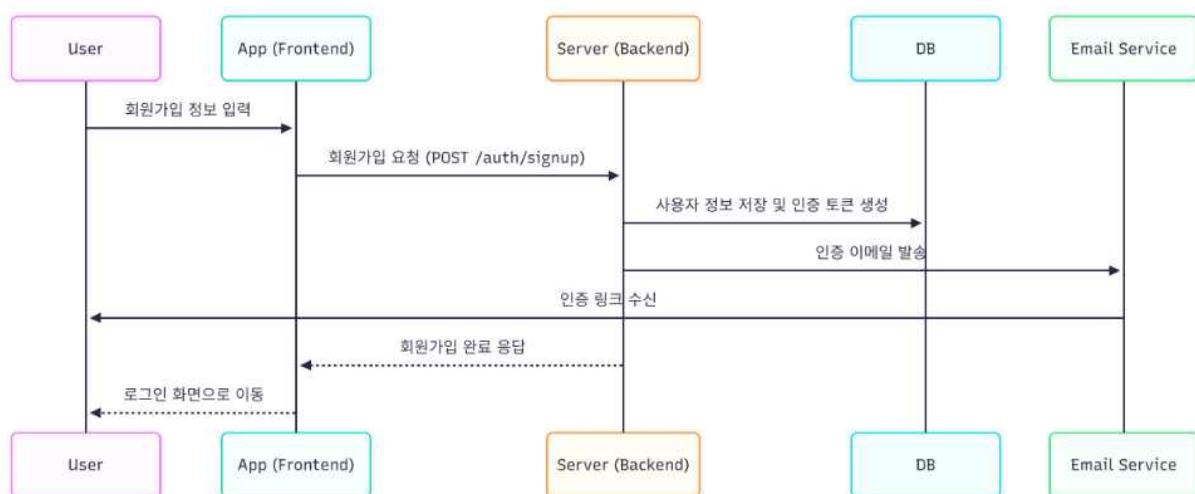
3.3.3 핵심 알고리즘 설계 (구현 시스템)

이번 빌드에서 구현된 핵심 알고리즘은 회원가입, 로그인, 이메일 인증, 토큰 재발급(Refresh Token), 비밀번호 재설정으로 구성된다. 해당 알고리즘은 프론트엔드-백엔드-DB 계층 간의 명확한 역할 분리를 기반으로 동작하며, 사용자의 입력으로부터 데이터베이스 처리까지의 과정을 일관된 구조로 설계하였다. 회원가입 알고리즘을 대표 알고리즘으로 설정하여, 상세하게 작성하고, 나머지 기능은 회원가입과 동일한 계층적 구조이므로, 간략히 작성함.

3.3.3.1 회원가입 알고리즘 (대표)

회원가입 알고리즘은 사용자의 기본정보 입력 및 약관동의 → 중복검증, → DB저장 및 인증 메일 발송 → 로그인 화면 이동의 순서로 진행된다. 프론트엔드에서 입력받은 사용자 정보를 백엔드로 전달하면, 백엔드에서는 중복 검사, 유효성 검사, 비밀번호 해싱 등의 과정을 거쳐 데이터를 저장하고, 인증 토큰을 생성해 이메일로 발송한다.

회원가입 알고리즘 절차 (Sequence Diagram)



1. 회원가입 정보 입력

사용자는 앱(프론트엔드) 화면에서 이메일, 비밀번호, 이름, 전화번호, 약관동의 등의 회원가입 정보를 입력한다. 필수 약관동의는 프론트단에서 검증을 진행한다.

2. 회원가입 요청 전송

프론트엔드는 사용자가 입력한 데이터를 AuthRepository를 통해 백엔드 서버의 /auth/signup 엔드포인트로 전송한다. 이때 요청에는 사용자 계정 생성에 필요한 모든 정보가 포함된다.

3. 사용자 정보 저장 및 인증 토큰 생성

백엔드 서버는 UserService를 통해 이메일 중복 여부와 비밀번호 유효성을 검사한 후, 비밀번호를 해시 처리하여 UserRepository를 통해 DB에 저장한다. 이후 EmailVerificationTokenRepository를 통해 인증 토큰을 생성한다.

4. 인증 이메일 발송

생성된 인증 토큰은 EmailService를 통해 사용자 이메일로 발송된다. 사용자는 해당 이메일을 통해 인증 절차를 진행하게 된다.

5. 인증 링크 수신 및 처리

사용자가 이메일에서 인증 링크를 클릭하면, 백엔드 서버에서 토큰 검증이 수행된다. 토큰이 유효할 경우 User엔티티의 emailVerified값이 true로 갱신된다.

6. 회원가입 완료 응답

백엔드는 회원가입 완료 응답을 프론트엔드로 반환한다. 프론트엔드는 이 응답을 기반으로 후속 처리를 수행한다.

7. 로그인 화면으로 이동

프론트엔드는 회원가입이 정상적으로 완료되면 로그인 화면으로 전환한다. 사용자는 로그인 절차를 통해 실제 서비스 이용을 시작할 수 있다.

예외 처리 설계

상황	처리 방식	결과
이메일 중복	UserRepository.existsByEmail() → 409 Conflict 반환	UI에 오류 메시지 출력
비밀번호 형식 오류	백엔드 유효성 검사 후 400 Bad Request 반환	사용자에게 재입력 요청
이메일 전송 실패	500 Internal Server Error 반환	사용자에게 재시도 안내
약관 미동의	프론트엔드에서 요청 차단 (ViewModel 단 처리)	UX 개선

3.3.3.2 로그인 알고리즘 (요약)

로그인 알고리즘은 사용자의 이메일과 비밀번호를 검증하고, 유효할 경우 Access Token과 Refresh Token을 발급하는 절차이다.

절차 요약	LoginScreen에서 입력받은 정보 → AuthViewModel → AuthRepository.login() AuthController에서 이메일/비밀번호 검증 후 JWT 토큰 발급 프론트엔드의 TokenDataStore에 토큰 저장 → 로그인 상태로 전환
예외처리	잘못된 비밀번호 → 401 Unauthorized 이메일 미인증 → Forbidden (이메일 인증 안내 메시지 출력)

3.3.3.3 이메일 인증 알고리즘 (요약)

회원가입 후 사용자의 이메일 주소를 검증하기 위해 인증 토큰이 포함된 링크를 발송하고, 사용자가 클릭하면 이메일 인증 상태를 업데이트한다.

절차 요약	회원가입 완료 후 EmailService가 인증 링크 발송 사용자가 인증 링크 클릭 시 → AuthController.verifyEmail() 호출 токен 검증 후 User.emailVerified = true 업데이트
예외처리	만료된 토큰 → 400 Bad Request 잘못된 토큰 → Invalid Token 반환)

3.3.3.4 토큰 재발급 알고리즘 (요약)

Access Token이 만료된 경우 Refresh Token을 이용하여 새로운 Access Token을 자동으로 발급받는 절차이다.

절차 요약	프론트엔드 AuthInterceptor에서 Access Token 만료 감지 AuthRepository.refresh() 호출 → AuthController.reissue() 요청 Refresh Token 검증 후 새로운 Access Token, Refresh Token 발급 및 저장
예외처리	Refresh Token 만료 또는 불일치 시 → 로그아웃 처리 및 재로그인 요구

3.3.3.5 비밀번호 재설정 알고리즘 (요약)

이메일과 이름, 전화번호를 통해 사용자 정보를 검증한 뒤, 임시 비밀번호를 발급하여 이메일로 전송하는 절차이다.

절차 요약	ResetPasswordScreen에서 입력 → ResetPasswordViewModel → AuthRepository.resetPasswordToEmail() 백엔드에서 사용자 정보 검증 후 임시 비밀번호 생성 EmailService를 통해 이메일로 임시 비밀번호 발송
예외처리	사용자 정보 불일치 → 404 Not Found 이메일 전송 실패 → 500 Internal Server Error

3.3.3.6 이메일 찾기 알고리즘 (요약)

이름과 전화번호를 통해 사용자 정보를 검증한 뒤, 해당 조건에 일치하는 사용자 이메일 목록을 반환하는 절차이다. 가입된 이메일을 잊은 사용자가 계정 정보를 복구할 수 있도록 돋는 기능이다.

절차 요약	FindEmailScreen에서 이름, 전화번호입력 → FindEmailViewModel → AuthRepository.findEmail() → 백엔드의 UserRepository에서 이름과 전화번호를 검증하고 일치하는 이메일목록 조회 → List<String>형태로 일치하는 이메일목록 반환 → 프론트엔드는 이메일목록 화면에 표시
예외처리	일치하는 계정이 없는 경우 → 빈 리스트 반환 입력값 형식 오류 → 400 Bad Request 서버 오류 발생 시 → 500 Internal Server Error

3.3.4 데이터 설계 (구현 시스템)

본 빌드의 데이터 설계는 사용자 인증 및 계정 관리 기능을 중심으로 이루어져 있다.

회원가입, 로그인, 이메일 인증, 토큰 발급 및 관리, 비밀번호 재설정기능을 지원하기 위해 **3개의 핵심 테이블(User, RefreshToken, EmailVerificationToken)부가 테이블 및 관련 Repository 계층이 구성되어 있다.** 각 테이블은 백엔드 서버(Spring Boot)와 MySQL DB를 기반으로 설계되었으며, Repository 계층을 통해 데이터 접근을 캡슐화하였다.

3.3.4.1 ERD(Entity Relationship Diagram)



3.3.4.2 테이블 설계 상세

테이블명	설명	주요 컬럼	제약조건 및 특징
User	사용자 계정 및 인증 정보 저장	id, email, passwordHash, name, phone, nickname, birthdate, emailVerified, provider, providerId	emailunique, 비밀번호는 해싱 처리 후 저장
Refresh Token	JWT Refresh Token 관리	id, tokenHash, expiresAt, revokedAt, rotatedAt, user_id	user_idFK, 토큰은 해시값으로만 저장(원본 미보관)
Email Verification Token	이메일 인증 토큰 관리	id, createdAt, expiresAt, usedAt, user_id	만료 시간 기반, 사용 후 usedAt 업데이트

3.3.4.3 데이터 설계상의 보안 고려사항

비밀번호는 원문을 저장하지 않으며 PasswordEncoder를 통해 해시 처리 후 저장.

Refresh Token은 원본을 저장하지 않고 SHA-256 해시값만 저장하여 탈취 위험을 최소화.

EmailVerificationToken은 만료 시간(expiresAt)과 사용 여부(usedAt)를 통해 단일 사용 보장.

DB 접근은 모두 Repository계층을 통해 이루어지며, 비즈니스 로직에서 직접 SQL을 다루지 않는다.

3.3.4.4 클라이언트(Local Storage) 연동 설계

프론트엔드(Android)에서는 TokenDataStore를 통해 Access Token, Refresh Token을 안전하게 로컬에 저장하고, API 호출 시 AuthInterceptor를 통해 자동으로 헤더에 토큰을 첨부한다.

저장소	저장 내용	특징
TokenDataStore	AccessToken, RefreshToken, 만료시간	Jetpack DataStore 기반, 암호화 가능
AuthInterceptor	토큰 자동 부착 및 만료 시 Refresh 요청 처리	백엔드와 무중단 인증 플로우 유지

3.4. 테스트 데이터 및 결과

Auth					
MUT	Tid	테스트 입력 / 테스트 시나리오	예상결과	실행결과	테스트 통과
FR-601, UIR-6304		POST /auth/signup{"email": " test@naver.com ", "password": "ab1234!!", "passwordConfirm": "ab1234!!", "name": "홍길동", "phone": "01012345678", "nickname": "관측러", "birthdate": "2010-01-01"}	200 Created, 인증 메일 발송	인증 메일 발송됨	통과
FR-601		GET /auth/verify-email?token=<token> 또는 인증 메일 링크 클릭	200 OK, emailVerified=true, 성공 페이지 출력	인증 성공 페이지 출력	통과
FR-601		이미 가입된 이메일로 회원가입 시도	409 Conflict, "이미 사용 중인 이메일입니다."	409 Conflict	통과
FR-602, UIR-6301		POST /auth/login{"email": " test@naver.com ", "password": "ab1234!!"}	200 OK, Access Token 발급	токен 정상 발급	통과
FR-602, UIR-6301		잘못된 비밀번호로 로그인 시도	401 Unauthorized, 오류 메시지 출력	"이메일 또는 비밀번호가 일치하지 않습니다."	통과
FR-605		만료된 Refresh Token으로 접속 시도	401 Unauthorized, "만료된 토큰입니다."	에러 메시지 출력	통과
FR-601		회원가입 시 이메일 중복체크 기능 확인	중복 시 409 Conflict, 오류 메시지 출력	오류 메시지 출력	통과
FR-601		비밀번호 규칙 미준수(ex. "abcd")	400 Bad Request, 형식 오류 메시지 출력	오류 메시지 출력	통과
FR-601,		필수 약관 미동의 상태에서 회원가입 시도	요청 차단(프론트), ViewModel 차단		통과

UIR-6305		서버 요청 없음	확인	
FR-602	이메일 인증되지 않은 계정으로 로그인 시도	403 Forbidden, 인증 요청 메시지 출력	인증 필요 메시지 출력	통과
FR-602	존재하지 않는 계정으로 로그인 시도	404 Not Found	에러 메시지 출력	통과
FR-603, UIR-6303	이름·전화번호로 이메일 찾기 요청	가입된 이메일(마스킹 처리) 반환	이메일 노출 확인	통과
FR-603	잘못된 정보로 이메일 찾기 요청	404 Not Found, 오류 메시지 출력	오류 메시지 출력	통과
FR-604, UIR-6302	이름·전화번호 일치 시 비밀번호 재설정 요청	200 OK, 임시 비밀번호 메일 발송	메일 발송 확인	통과
FR-604	정보 불일치 시 비밀번호 재설정 시도	404 Not Found, 오류 메시지 출력	오류 메시지 출력	통과
FR-605	유효한 Refresh Token으로 앱 재실행	자동 로그인 성공, токن 재발급	로그인 상태 유지	통과
FR-605	Refresh Token 만료 후 앱 재실행	자동 로그인 실패, 로그인 화면 이동	재로그인 유도	통과
FR-602	로그인 실패 시	에러 메시지 출력	오류 메시지 출력	통과
FR-604	비밀번호 찾기 성공 시	임시 비밀번호 발송 안내 메시지 출력	정상 출력	통과
FR-603	이메일 찾기 성공 시	마스킹된 이메일 목록 출력	정상 출력	통과
FR-601	회원가입 완료 후	로그인 또는 메인 화면 이동	정상 이동	통과
FR-601	약관 미동의 시	다음 단계 진행 불가	버튼 비활성화 확인	통과
FR-601	인증 코드 인증 시	회원가입 완료	정상 처리	통과
공통	뒤로가기 버튼 클릭 시	이전 단계로 이동	정상 이동	통과

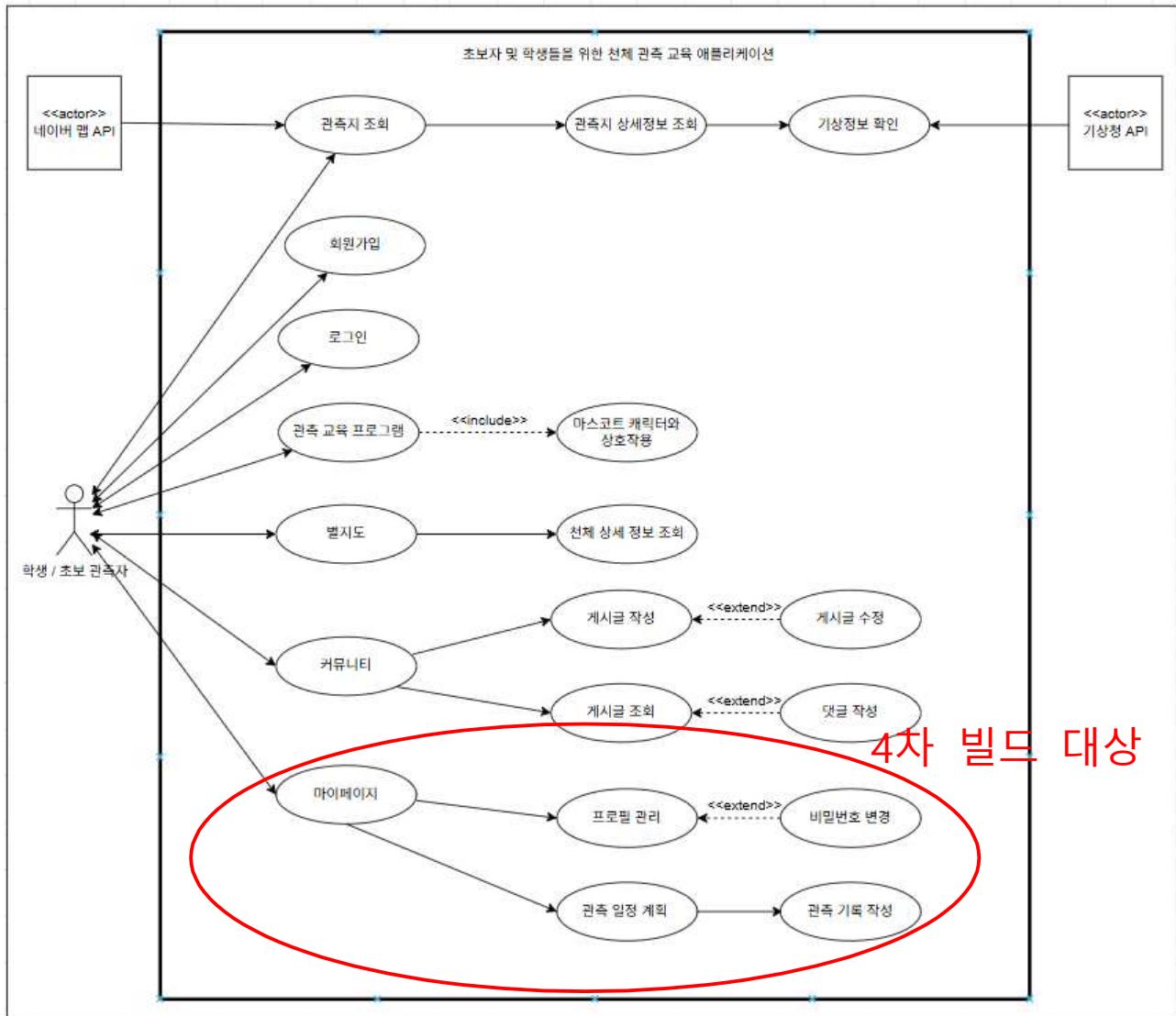
4.1. 4차 빌드 개발 개요

태스크 번호	태스크 명	담당자	시작일	종료일	비고
T-401	계정 관리	윤태영	25.09.01	25.09.05	-
T-402	비밀번호 변경	윤태영	25.09.10	25.09.13	-
T-403	로그아웃 및 회원탈퇴	윤태영	25.09.15	25.09.18	-
T-404	관측 일정 계획	윤태영	25.09.20	25.09.25	-
T-405	관측 기록	윤태영	25.09.26	25.09.30	-
T-406	캘린더 조회	윤태영	25.10.01	25.10.04	-
T-407	사용자 정보 표시 UI	김채영	25.10.20	25.10.22	-
T-408	회원 탈퇴 팝업	김채영	25.10.28	25.10.30	-
T-409	캘린더 UI	김채영, 윤태영	25.11.01	25.11.03	-
T-410	일정 상세조회 화면	김채영	25.11.04	25.11.05	-
T-411	일정 계획 작성 화면	김채영	25.11.3	25.11.06	-
T-412	마이페이지 I.A 작성	김채영	25.08.20	25.08.25	-
T-413	User API 명세서 작성	윤태영	25.09.03	25.09.08	-
T-414	User Controller 연동	김채영	25.09.20	25.09.26	-

4.2. 분석 명세

2.1 기능 명세

(1) Use Case Diagram (전체 시스템)



(2) Use Case 설명서 (구현 시스템)

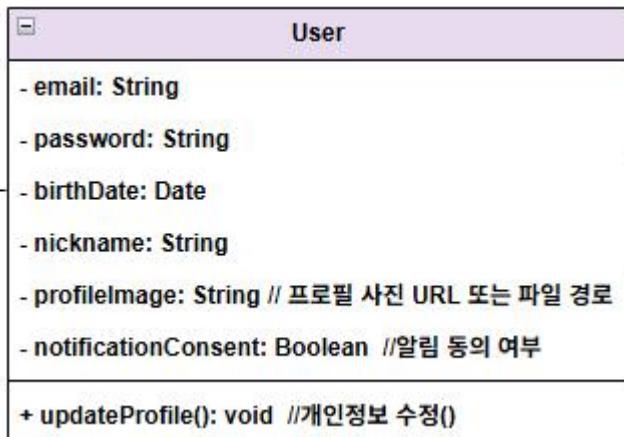
- 비밀번호 변경 Usecase

Use Case Name: 비밀번호 변경	ID: UC-008	Important Level: LOW
Primary Actor: 사용자		Use case type: Detail, Essential
Stakeholders and Interests:		
<p>사용자: 계정 보안을 위해 비밀번호를 변경하고 싶어함.</p> <p>시스템: 비밀번호 정책 준수, 인증 강도 확보, 세션 일관성 유지</p>		
Brief Description:		
<p>- 사용자가 현재 비밀번호 또는 비밀번호 재설정 과정에서 발급된 임시 비밀번호를 통해 새 비밀번호로 변경한다.</p>		
Trigger: 마이페이지 설정에서 “비밀번호 변경” 선택		
Type: External		
Relationships:		
<p>Include: 새 비밀번호 일치 확인</p> <p>Extend: 임시 비밀번호 발급(재설정)</p>		
Normal Flow of Events:		
<ol style="list-style-type: none"> 1. 사용자가 “비밀번호 변경”을 선택한다. 2. 시스템은 로그인 된 사용자인지 확인한다. 3. 사용자는 현재 비밀번호, 새 비밀번호, 새 비밀번호 확인을 입력한다. 4. 시스템은 현재 비밀번호 일치 여부를 검증한다. 5. 시스템은 새 비밀번호의 규칙을 검증하고, 확인 값과의 일치 여부를 확인한다. 6. 사용자에게 “변경 완료”를 알린다. 		
Subflows:		
<p>S-1: 임시 비밀번호</p> <ol style="list-style-type: none"> 1. 사용자가 임시 비밀번호, 새 비밀번호, 확인을 입력한다. 2. 시스템은 임시 비밀번호의 유효·미사용 상태와 만료 여부를 검증한다. 3. 통과 시 새 비밀번호로 갱신하고 임시 비밀번호를 무효화한다. 		
Alternative / Exceptional Flows:		
<p>E-1: 현재 비밀번호가 불일치 할 경우 “현재 비밀번호가 올바르지 않습니다.” 메시지 출력 후 재시도 한다.</p> <p>E-2: 비밀번호 규칙을 위반했을 경우 구체적인 가이드 메시지를 제공해준다.</p>		

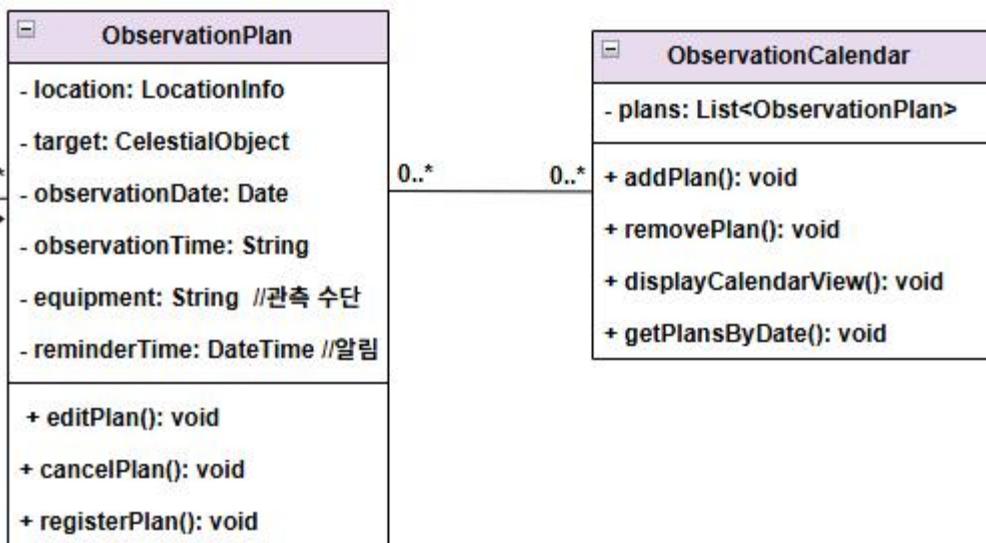
4.2.2 구조 명세 (구현 시스템)

마이페이지 기능은 사용자 개인 정보 관리 기능과 관측 캘린더에서 일정 계획, 기록 기능이 있으며, 구조는 아래 클래스 다이어그램으로 시각화 하였다. 비밀번호, 닉네임, 프로필 사진 등의 개인 정보를 수정 할 수 있으며, 관측 일정을 계획할 때에는 관측 장소, 대상, 날짜, 장비를 작성할 수 있다.

- User Class Diagram



- 관측 캘린더 Class Diagram



4.3. 설계 명세

4.3.1 메뉴 전개도 (구현 시스템)

- 마이페이지 메뉴전개도

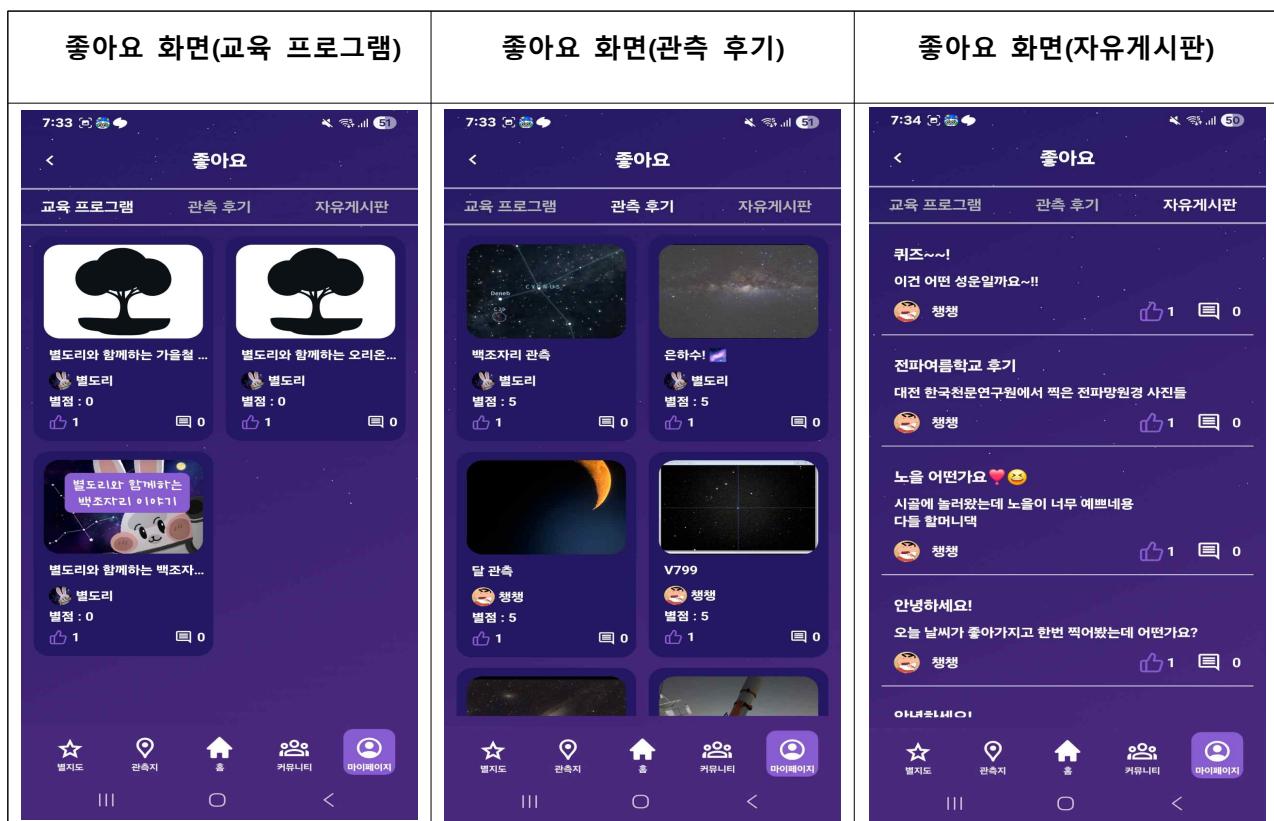
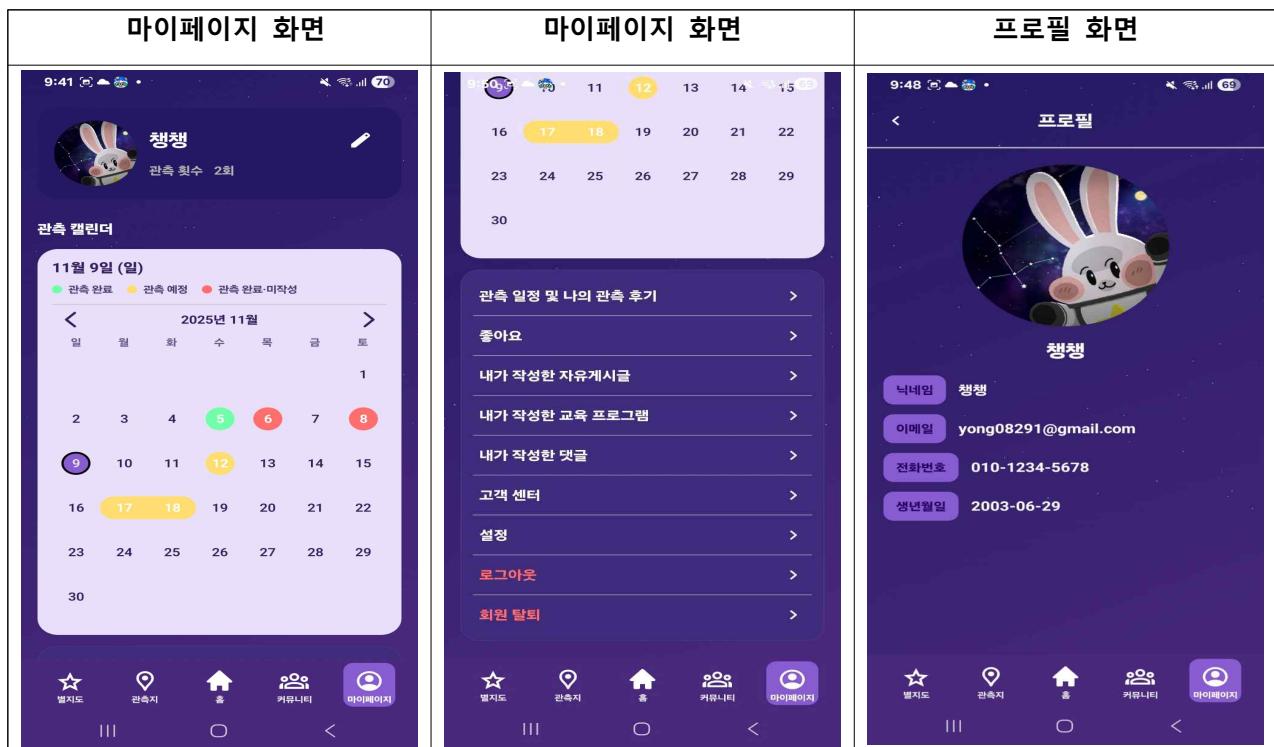
본 마이페이지 메뉴전개도는 사용자의 개인정보, 관측일정 캘린더, 사용자가 작성한 게시물들을 중점적으로 흐름을 나타낸 것이다. 여기서 관측일정 캘린더란, 사용자가 작성한 관측 일정에 해당하는 날짜들을 모아 캘린더 상에 표시한 것을 말한다.

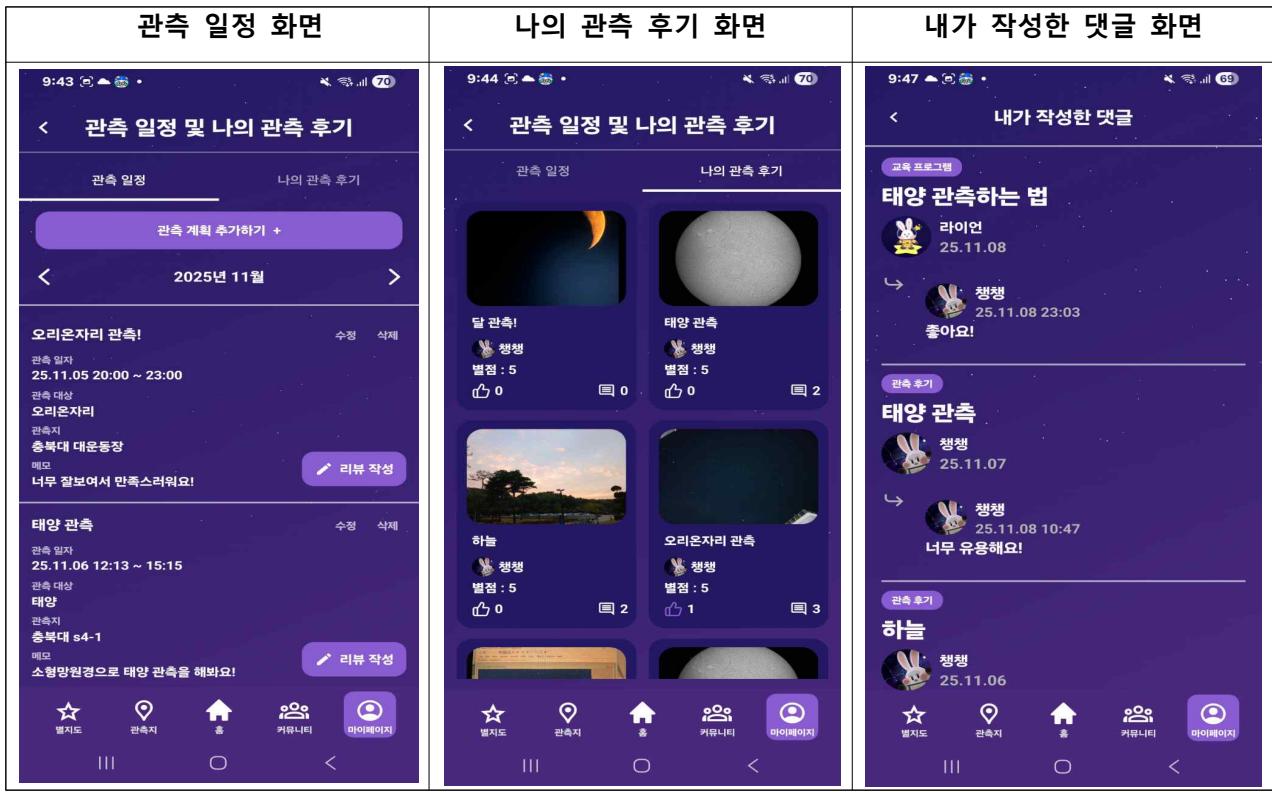
마이 페이지	
1	프로필 (개인정보) - 상단 둘니바퀴 -> 개인정보 수정
2	관측일정 캘린더
3	관측 일정 및 나의 관측 후기
4	좋아요
5	내가 작성한 자유게시글
6	내가 작성한 교육 프로그램
7	내가 작성한 댓글
8	고객 센터
9	설정
10	로그아웃
11	회원탈퇴

관측 일정 작성	
1	관측 대상
2	관측지
3	관측 날짜 및 시간 입력
4	메모(본문, 이미지)

개인정보 수정	
1	닉네임
2	생년월일
3	전화번호
4	프로필 이미지

4.3.2 사용자 인터페이스 설계 (구현 시스템)



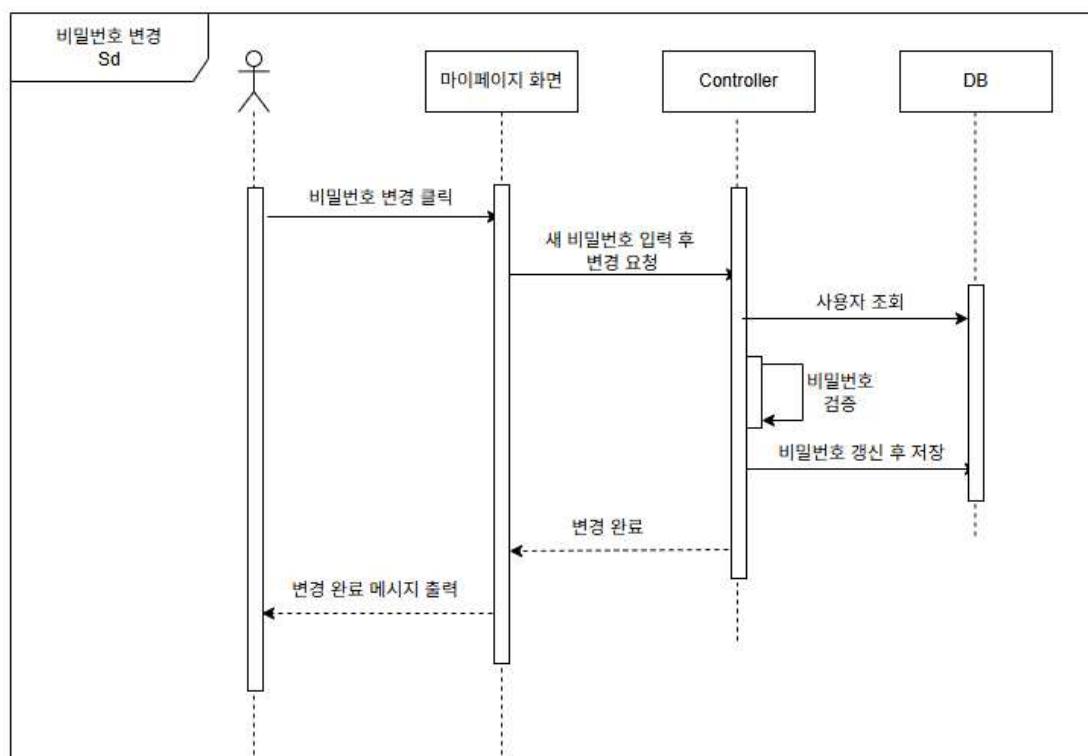


4.3.3 핵심 알고리즘 설계 (구현 시스템)

이번 빌드에서 구현된 핵심 알고리즘은 사용자의 계정 관리, 비밀번호 변경, 관측 일정 기록, 관측 캘린더 표시, 사용자가 작성한 게시글(관측 후기, 자유게시판, 교육 프로그램) 또는 댓글 열람, 로그아웃, 회원탈퇴로 구성된다.

- 비밀번호 변경 Sequence Diagram

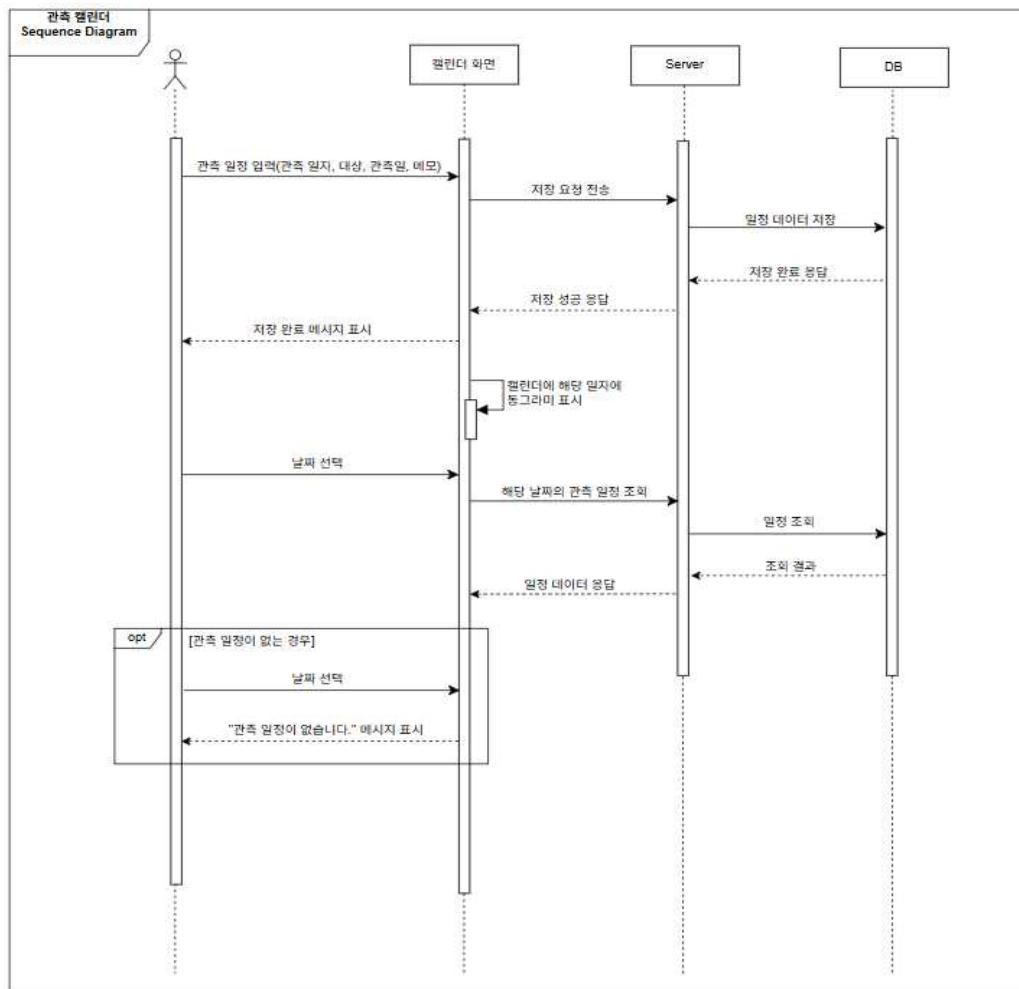
사용자가 비밀 번호 변경을 원하는 경우에, Controller를 통해 새 비밀번호 입력 후 변경을 요청하면 DB에서 비밀번호를 갱신 후 변경 완료 메시지를 출력하는 흐름을 담고 있다.



4.3.3.1 관측 캘린더 알고리즘 (대표)

관측 캘린더 알고리즘은 로그인한 사용자가 관측 일정을 추가해 관측 일자, 관측 대상, 관측지, 메모(텍스트/이미지)를 입력한 후 알림 시간을 설정하면, 해당 시간이 다가왔을 때 사용자에게 알림을 보내는 과정으로 진행된다.

관측 캘린더 알고리즘 절차 (Sequence Diagram)



1. 관측 일정 정보 입력

사용자는 앱(프론트엔드) 화면에서 관측 대상, 관측 자일, 관측 대상, 메모의 관측 일정 정보를 입력한다.

2. 관측 일정 요청 전송

프론트엔드는 사용자가 입력한 데이터를 PlanRepository를 통해 백엔드 서버의 /calendar/events 엔드포인트로 전송한다.

3. 관측 일정 저장

백엔드 서버는 CalendarService를 통해 관측지 정보와, 관측일자의 유효성 검증을 한 후, ObservationEventRepository를 통해 DB에 저장한다.

4. 관측 일정 저장 완료 응답

백엔드는 관측 일정 저장 응답을 프론트엔드로 반환한다.

5. 관측 캘린더에 일정 표시

캘린더 화면에서 관측 일정에 저장된 날짜를 기반으로 캘린더 상에 동그라미로 표시한다. 이때, 현재 날짜를 기준으로 이전의 관측 일정은 초록색, 관측 예정인 일정은 노란색, 이전의 관측일정에 대해서 관측 리뷰를 작성한 것은 빨간색, 현재 날짜는 보라색으로 구분하여 캘린더에 표시한다.

만약 해당 일자에 해당하는 관측 일정이 없는 경우, 캘린더 화면에서 "해당 일자에 관측 일정이 없습니다"라는 메시지를 출력한다.

4.3.3.2 사용자의 계정 관리 알고리즘 (요약)

사용자의 계정 관리 알고리즘은 사용자의 계정 정보를 프로필 화면에 표시하고, 프로필 사진, 닉네임, 전화번호, 생년월일을 변경할 수 있다.

절차 요약	<p>프로필 조회: MyPageScreen에서 로그인한 사용자가 ProfileCard를 클릭하면 UserViewModel → UserRepository.getMyProfile()를 통해 백엔드 서버의 GET("/users/me") 엔드포인트로 프로필 정보를 불러옴(프로필 이미지, 닉네임, 이메일, 전화번호, 생년월일)</p> <p>프로필 편집: ProfileEditCard에서 프로필을 수정한 후 저장 버튼을 클릭하면 UserViewModel → UserRepository.updateMe()를 통해 백엔드 서버의 PATCH("/users/me"), POST("/users/me/profile-image") 엔드포인트로 프로필 정보를 수정 → 수정 성공 시 getMyProfile()로 최신 상태를 받아 업데이트</p>
----------	---

4.3.3.3 사용자가 작성한 게시글 조회 알고리즘 (요약)

사용자가 작성한 게시글들(자유게시판, 관측후기, 교육 프로그램)을 조회하는 절차이다.

절차 요약	MyPageScreen에서 내가 작성한 게시글 메뉴 클릭하면 각 화면(MyBoardList, MyProgramList, MyReviewList)에서 UserViewModel → UserRepository.getMyProfile()를 호출하여 얻은 사용자 ID(me.id)와 각각의 게시글들의 ViewModel → Repository.getAllPosts()를 호출하여 전체 게시글을 불러온 뒤 authorId == me.id인 게시글만 필터링
----------	---

4.3.3.4 사용자가 작성한 댓글 조회 알고리즘 (요약)

사용자가 작성한 댓글들(자유게시판, 관측후기, 교육 프로그램)을 조회하는 절차이다.

절차 요약	MyPageScreen에서 내가 작성한 댓글 메뉴를 클릭하면 MyCommentList에서 UserViewModel → UserRepository.getMyProfile()를 호출하여 얻은 사용자 ID(me.id)와 CommentsViewModel.AllMyComments()를 통해 각 게시글의 댓글을 불러온 뒤, authorId == me.id인 댓글만 필터링
----------	--

4.3.3.5 사용자가 좋아요를 누른 게시글 조회 알고리즘 (요약)

사용자가 좋아요를 누른 게시글들(자유게시판, 관측후기, 교육 프로그램)을 조회하는 절차이다.

절차 요약	MyPageScreen에서 내가 좋아요 메뉴를 클릭하면 LikeSection에서 UserViewModel → UserRepository.getMyProfile()를 호출하여 얻은 사용자 ID(me.id)와 각 게시글들의 ViewModel(EducationViewModel, ReviewViewModel, CommunityViewModel)을 통해 Repository.getAllPost()로 전체 게시글 목록을 불러온 뒤, 각 게시글의 liked 속성이 true인 항목한 필터링
----------	---

4.3.3.6 로그아웃/회원탈퇴 알고리즘 (요약)

로그인한 사용자가 로그아웃 또는 회원탈퇴하는 절차이다.

절차 요약	로그아웃: MyPageScreen에서 로그아웃 메뉴를 클릭하면 UserViewModel → UserRepository.logout()을 통해 백엔드 서버의 POST("/users/logout") 엔드포인트로 로그아웃함 회원탈퇴: MyPageScreen에서 회원탈퇴 메뉴를 클릭하면 UserViewModel → UserRepository.resign()을 통해 백엔드 서버의 DELETE("/users/me") 엔드포인트로 회원탈퇴함
----------	--

4.3.3.7 관측 일정 알림 알고리즘 (요약)

관측 일정을 계획한 사용자가 정한 시간에 맞춰 알림을 선택할 수 있다.

절차 요약	PlanCheckScreen에서 사용자는 관측 계획을 작성(관측 일자, 관측지, 관측 대상, 메모) 후 알림 시간을 선택 → PlanViewModel.setAlarmMinutes()으로 캐시, PlanDataStore에 알림 시간을 저장 → 알림 권한 확인 후 PlanAlarm()으로 일정 시작 시각에서 알림 시간을 뺀 예약 시각 계산하고 AlarmManager에 등록 → 예약 시간이 되면 PlanAlarmReceiver가 실행되어 NotificationHelper를 통해 알림 채널을 생성하고 알림 표시
----------	---

4.3.4 데이터 설계 (구현 시스템)

본 빌드의 데이터 설계는 사용자 정보 관리, 캘린더 일정 생성 기능을 중심으로 이루어져 있다. 사용자 정보 수정 및 관리 기능을 수행하기 위해 **users** 테이블을 아래와 같이 설계 하였고, 관측 계획 관리를 수행하기 위해 **observation_plan** 테이블을 또한 아래와 같이 설계하였으며, 관련 **Repository** 계층이 구성되어 있다. 각 테이블은 백엔드 서버(Spring Boot)와 MySQL DB를 기반으로 설계되었으며, Repository 계층을 통해 데이터 접근을 캡슐화하였다.

4.3.4.1 ERD(Entity Relationship Diagram)

- 유저 테이블

유저 테이블						
	유저ID	user_id	UUID	bigint	1	UUID
이메일	email	Email@abc.com	varchar	Default value	로그인 ID	Comment
비밀번호	password	PasswordHash	varchar	Default value	Comment	Comment
닉네임	nickname	Domain	varchar	Default value	Comment	Comment
생년월일	brithdate	DateOnly	date	Default value	YYYY-MM-DD 형식	
마케팅 등의	marketing	Domain	boolean	false	마케팅 수신 등의 여부(알림)	
위치정보 등의	location	Domain	boolean	false	위치 이용 등의 여부	
이메일 인증 여부	email_verified	Domain	boolean	false	Comment	
생성일시	created_at	DatetimeNow	datetime	now()	계정 생성 시각	
수정일시	update_at	DatetimeNow	datetime	now()	Comment	

- 관측 계획 테이블

관측 계획 테이블						
	관측일정ID	plan_id	UUID	bigint	1	UUID
관측지ID	obsite_id	Domain	bigint	1	고유 관측지 ID	Comment
천체 ID	star_id	Domain	bigint	1	Comment	Comment
유저ID	user_id	UUID	bigint	1	UUID	
관측대상	target	Domain	varchar	Default value	Comment	Comment
관측지	name	SiteName	varchar	Default value	Comment	Comment
관측날짜	ob_date	DateOnly	date	Default value	Comment	Comment
작성일시	created_at	DatetimeNow	datetime	now()	Comment	Comment
수정일시	update_at	DatetimeNow	datetime	now()	Comment	Comment

4.3.4.2 테이블 설계 상세

테이블명	설명	주요 컬럼	제약조건 및 특징
users	사용자 계정 및 정보 관리	id, email, password, nickname, birthdate, marketing, location, email_verified, created_at, updated_at	-
observation plan	관측 일정 관리	id, observe, target, ob_date, created_at, updated_at	관측지 테이블과 연결

4.4. 테스트 데이터 및 결과

MUT	Test_Method() 또는 구현된 기능명			
Tid	테스트 입력 / 테스트 시나리오	예상결과	실행결과	테스트 통과
FR-607 UIR-601	GET /users/me{유효 토큰}	200 OK, 본인 프로필 반환	200 OK, 로그인 된 사용자의 프로필이 출력됨	통과
FR-607 UIR-6202	PATCH /users/me {nickname:"별돌이", birthday:2010-01-01, phone:"01012345678"}	200 OK, 변경 사항 DB 업데이트	200 OK, 내정보 조회 시 변경사항 업데이트 확인	통과
FR-607 UIR-6202	PATCH /users/me {nickname:"별돌이", birthday:2010-01-01, phone:"01012345678"} 에서 닉네임이 다른 사용자와 중복되는 경우	409 Conflict, 에러메시지 출력	409 Conflict, "이미 사용중인 닉네임입니다" 출력	통과
FR-607 UIR-6203	PATCH /users/me/password {current:"Aa123456!", new:"Bb234567!", confirm:"Bb234567!"}	200 OK, 비밀번호 변경 완료	200 OK, DB에 비밀번호 해시 갱신됨	통과
FR-606	비밀번호 변경할 때 임시 비밀번호 혹은 현재 비밀번호가 불일치 하는 경우	401, "현재 비밀번호가 일치하지 않습니다" 에러 출력	401, 에러 메시지 출력됨	통과
FR-607 UIR-605	DELETE /users/me로 회원 탈퇴할 경우	204, 탈퇴한 계정으로 로그인 불가	204, 정상적으로 회원 탈퇴 후 첫 화면으로 이동	통과
FR-608	프로필 사진 업로드 시	200 OK, 새로운 프로필 사진 생성	프로필 사진 정상적으로 업로드 됨	통과
FR-608	프로필 사진 업로드 할 때 파일 형식이 이미지가 아닐 경우	400, 지원하지 않는 파일 형식입니다. 에러 출력	에러 메시지 출력되면서 업로드 되지 않음	통과
FR-608	용량을 초과하는 이미지 업로드 시	413, 파일이 너무 큽니다 에러 출력	파일이 업로드 되지 않음	통과
FR-609 UIR-6103	캘린더 계획 생성	200 OK, 정상적으로 계획이 저장됨	새로운 관측 계획 생성	통과

FR-609	지난 날짜에 관측 계획을 생성할 경우	에러메시지 출력, 계획이 생성되지 않음	계획 생성이 되지 않음	통과
UIR-605	회원 탈퇴 시	정상적으로 회원 탈퇴되고 로그인 페이지로 이동	회원 탈퇴가 정상적으로 이루어짐	통과
UIR-6101	관측 계획 조회	캘린더의 날짜 클릭 시 해당 날짜의 계획을 조회할 수 있음	해당 날짜의 관측 계획 상세 조회 가능	통과
FR-607 UIR-6202	사용자 정보 수정 시 필수 정보를 입력 안한 경우	에러 메시지 출력	에러 메시지 출력되고, 필수 입력해야함	통과

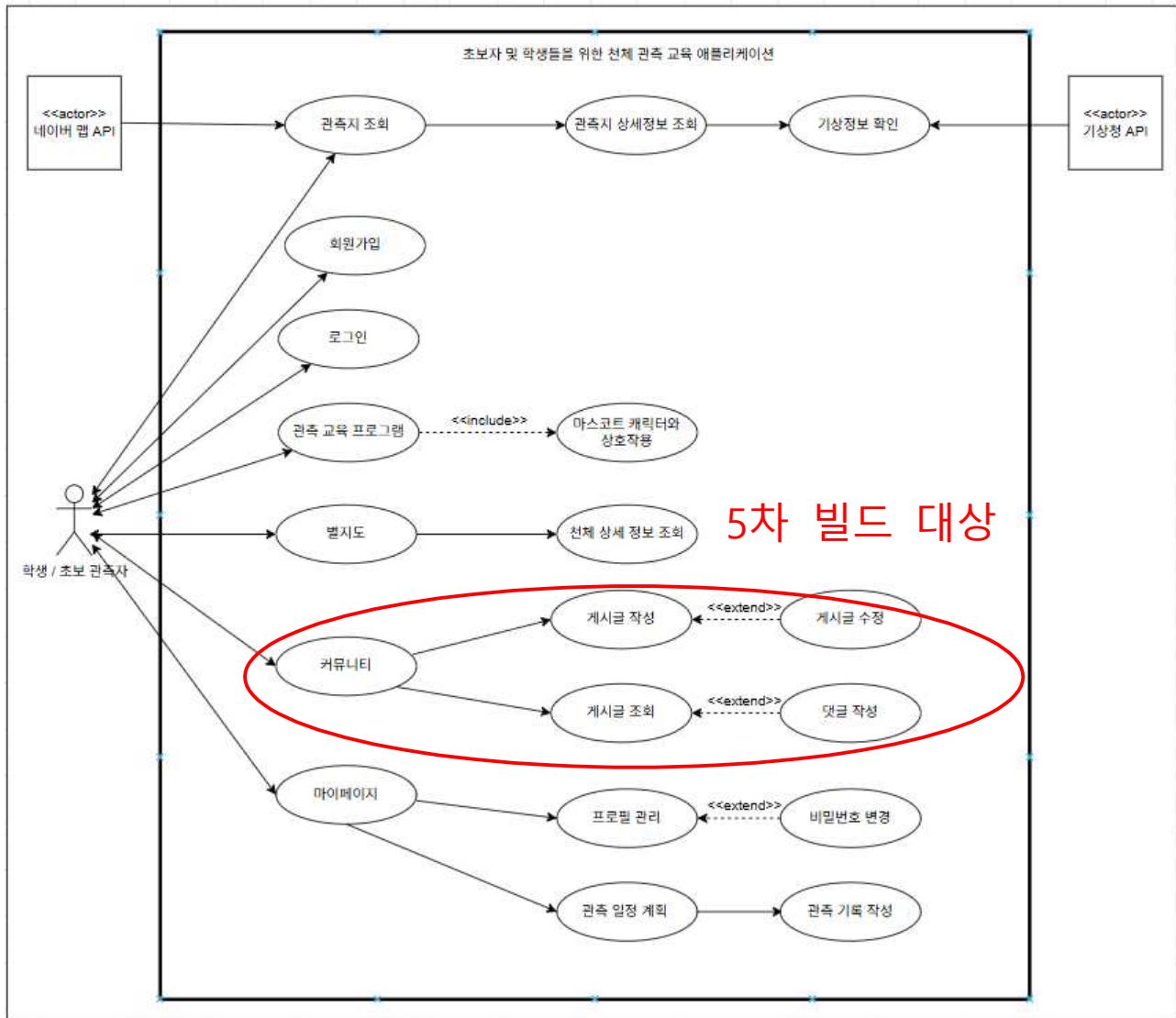
5.1. 5차 빌드 개발 개요

태스크 번호	태스크 명	담당자	시작일	종료일	비고
T-501	게시물 작성/수정/삭제	윤태영	25.10.01	25.10.06	-
T-502	게시글 조회 및 평점, 좋아요	윤태영	25.10.10	25.10.13	-
T-503	댓글 작성/수정/삭제	윤태영	25.10.15	25.10.20	-
T-504	이미지 업로드	윤태영	25.10.05	25.10.10	-
T-505	커뮤니티 홈 UI	김채영	25.09.16	25.09.25	-
T-506	게시글 목록 UI	김채영	25.09.26	25.10.02	-
T-507	게시글 작성 UI	김채영	25.10.03	25.10.08	-
T-508	게시글 상세 조회 UI	김채영	25.10.19	25.10.12	-
T-509	커뮤니티 페이지 화면 I.A 작성	김채영	25.07.15	25.07.20	-
T-510	Community API 명세서 작성	윤태영	25.09.20	25.09.25	-
T-511	Community Controller 연동	윤태영	25.09.26	25.10.02	-

5.2. 분석 명세

5.2.1 기능 명세

(1) Use Case Diagram (전체 시스템)



(2) Use Case 설명서 (구현 시스템)

- 관측 후기 작성 UseCase

Use Case Name: 관측 후기 작성	UC-010	Important Level: High
Primary Actor: 사용자		Use case type: Detail, Essential
Stakeholders and Interests:		
<p>사용자: 천체 관측을 한 후 관측에 대한 후기를 작성하고 싶어한다.</p> <p>시스템: 사용자에게 후기 작성을 위한 품을 제공하고, 작성한 후기를 저장한다.</p>		
Brief Description:		
<ul style="list-style-type: none"> - 사용자가 자신의 관측 후기를 게시글로 작성하여 게시한다. 		
Trigger: 사용자는 관측을 진행한 후 커뮤니티 페이지에서 관측 후기 작성 버튼을 클릭한다.		
Type: External		
Relationships:		
Include:		
Extend: 관측 기록 수정		
Normal Flow of Events:		
<ol style="list-style-type: none"> 1. 사용자는 관측 후기 작성 버튼을 클릭한다. 2. 시스템은 게시글 작성 페이지를 표시한다. (제목, 관측 대상, 관측지, 관측 장비, 관측 일자, 관측 평점, 후기, 이미지) 3. 사용자는 각각의 항목들을 입력한다. 4. 사용자는 작성물을 완료한 후 등록 버튼을 클릭한다. 5. 시스템은 입력 데이터에 대한 유효성을 검증 후 관측 후기를 저장한다. 6. 시스템은 작성이 완료되었다는 메시지를 화면에 표시한다. 		
Subflows:		
S-1. 날짜 선택		
<ol style="list-style-type: none"> 1. 사용자가 관측 일자 입력칸을 클릭하면, 시스템은 달력 팝업을 표시한다. 2. 사용자는 자신의 관측 일자에 맞는 날짜를 선택한다. 3. 시스템은 선택된 날짜를 형식(YY.MM.DD)에 맞게 자동 변환하여 저장한다. 		
S-2. 이미지 첨부		
<ol style="list-style-type: none"> 1. 사용자가 이미지 첨부 버튼을 클릭하면 시스템은 기기 내 갤러리 또는 파일 탐색기를 연다. 2. 사용자는 첨부할 관측 사진을 하나 이상 선택한다. 3. 시스템은 선택된 이미지의 파일 개수, 용량, 형식을 검사한 후, 통과한 이미지에 대해 업로드 목록 		

에 미리보기(썸네일)를 표시한다.

3. 사용자가 이미지를 잘못 선택했을 경우, 시스템은 해당 이미지를 삭제할 수 있도록 한다.

S-2. 평점 입력

1. 사용자가 별점을 클릭하면 시스템은 1~5점 중 선택된 평점을 시각적으로 표시한다.

Alternative / Exceptional Flows:

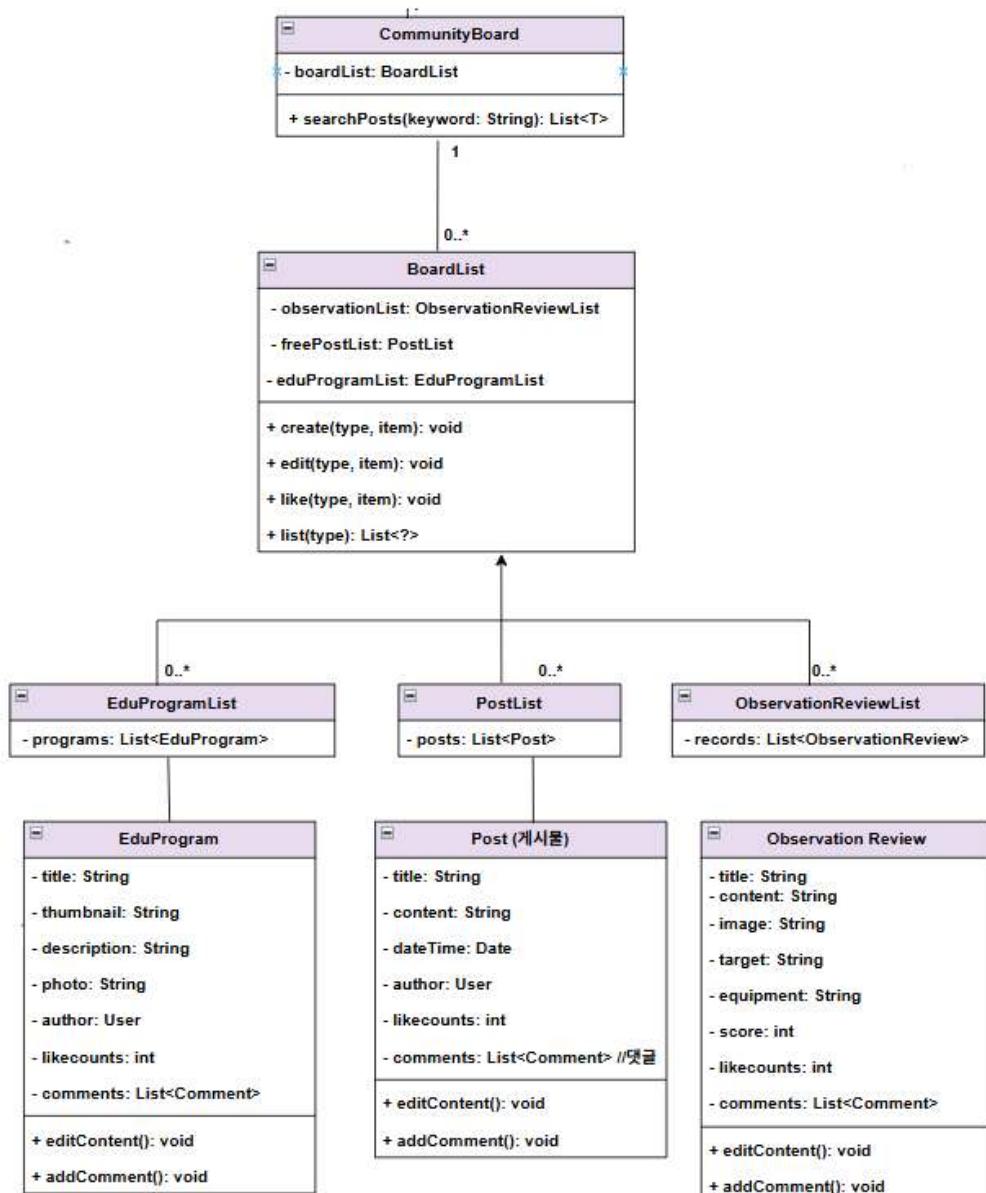
E-1: 게시글의 필수 항목을 누락했을 경우 시스템은 “필수 항목을 입력해주세요.”라는 메시지를 표시한다.

E-2: 용량이 초과하거나 지원하지 않는 형식의 이미지인 경우 에러 메시지를 표시한다.

5.2.2 구조 명세 (구현 시스템)

- 커뮤니티 Class Diagram

본 클래스 다이어그램은 커뮤니티 내에서 게시글을 관리하기 위한 전체 구조를 나타낸다. 커뮤니티 시스템은 CommunityBoard를 중심으로 자유게시판, 교육 프로그램 게시판, 관측 후기 게시판을 통합 관리하며, 게시글 작성, 수정, 댓글, 좋아요 기능을 공통화한 구조로 설계되어 있다.

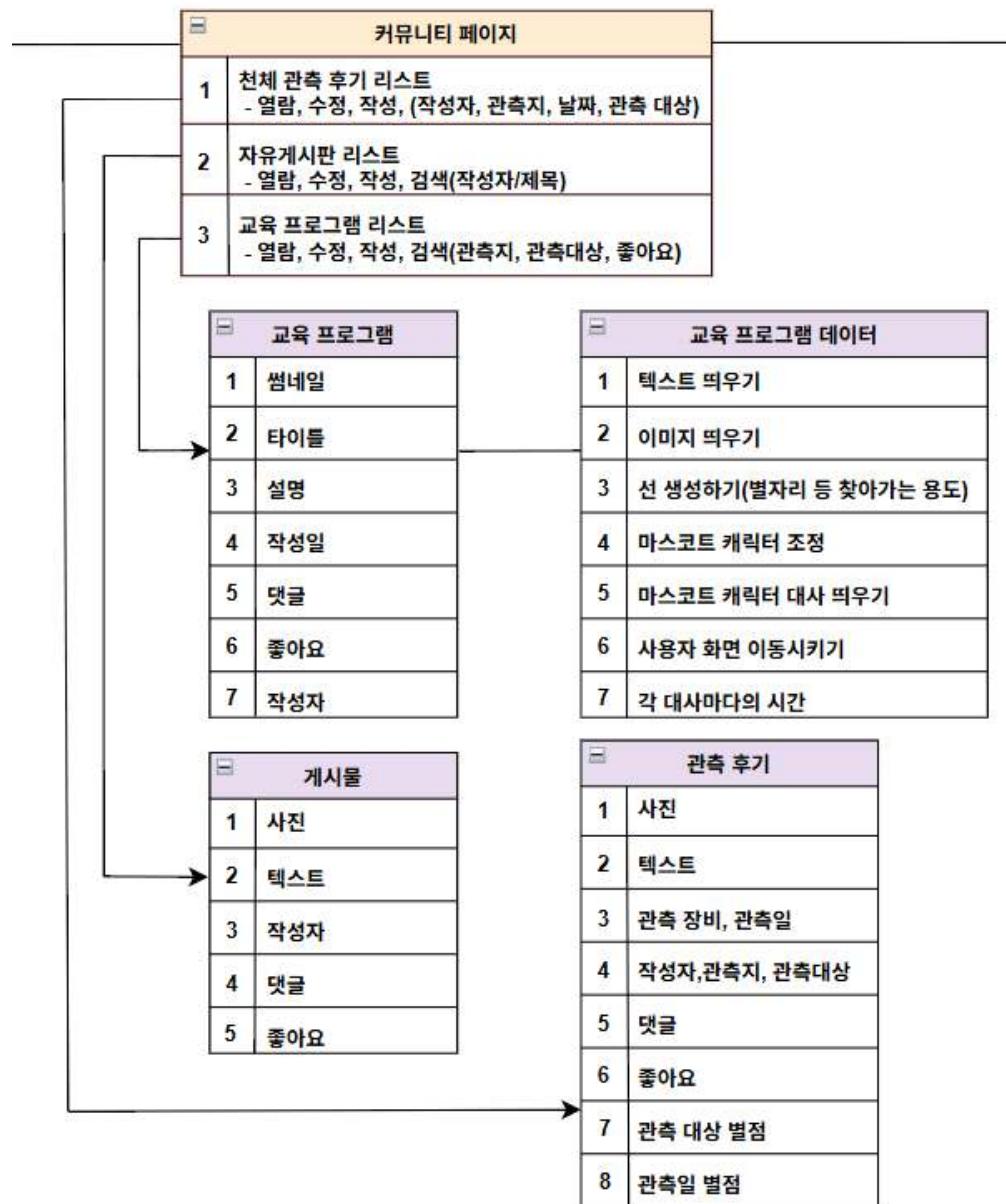


5.3. 설계 명세

5.3.1 메뉴 전개도 (구현 시스템)

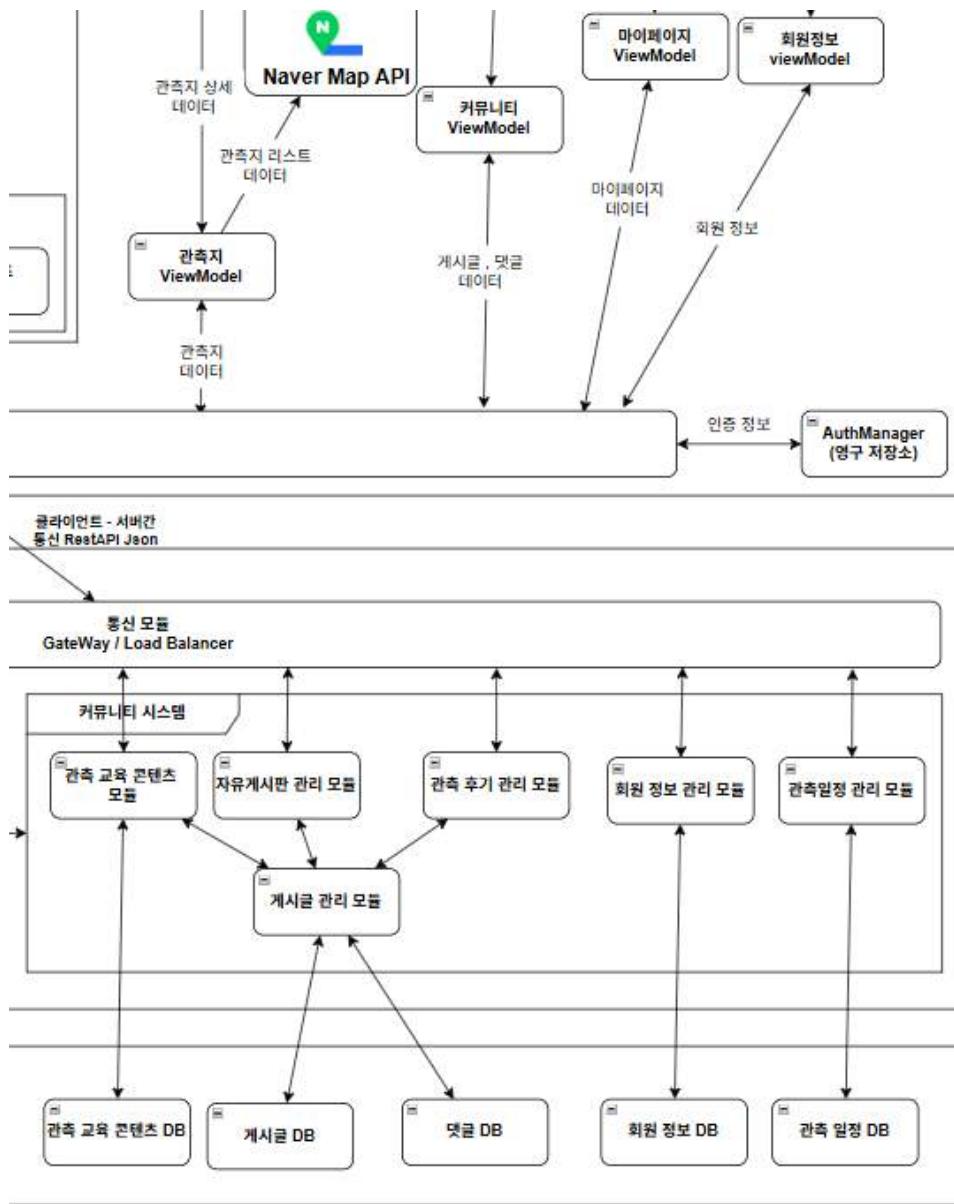
- 커뮤니티 게시판 메뉴 전개도

본 커뮤니티 메뉴 전개도는 관측 후기, 자유게시판, 교육 프로그램의 세 게시판 구조를 통합하여 사용자가 정보를 공유하고 소통할 수 있는 커뮤니티 기능의 전체 흐름을 나타낸 것이다. 각 게시판은 게시글 작성, 조회, 수정, 좋아요, 댓글 기능을 공통적으로 제공하며, 교육 프로그램 게시판은 7차 빌드에서 다룰 예정이다.



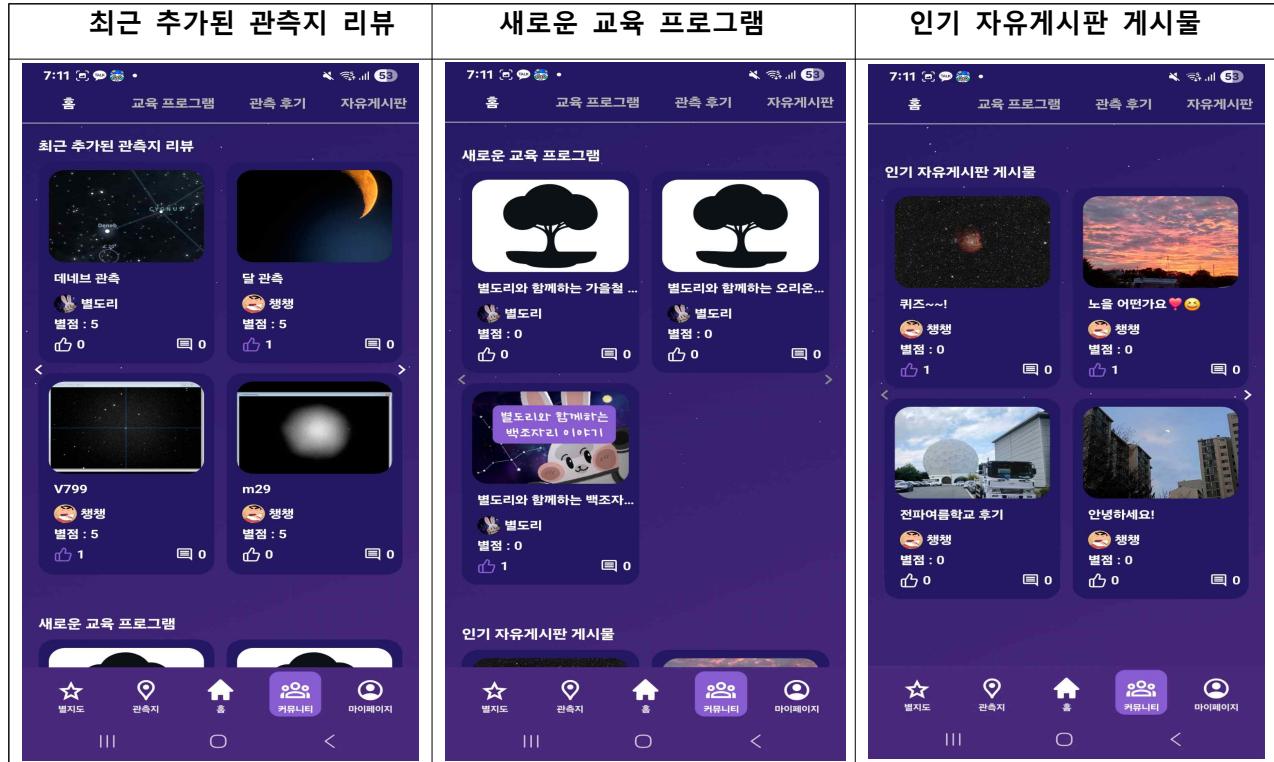
- 커뮤니티 시스템 구성도

본 커뮤니티 시스템 구성도는 관측 교육, 자유게시판, 관측 후기의 커뮤니티 기능 전반을 통합 관리하는 구조를 나타낸다. 클라이언트의 ViewModel과 서버의 관리 모듈이 REST API를 통해 상호 연동되며, 인증, 게시글, 댓글, 회원 정보가 모듈화되어 데이터 일관성과 서비스 확장성을 동시에 확보하고 있다.

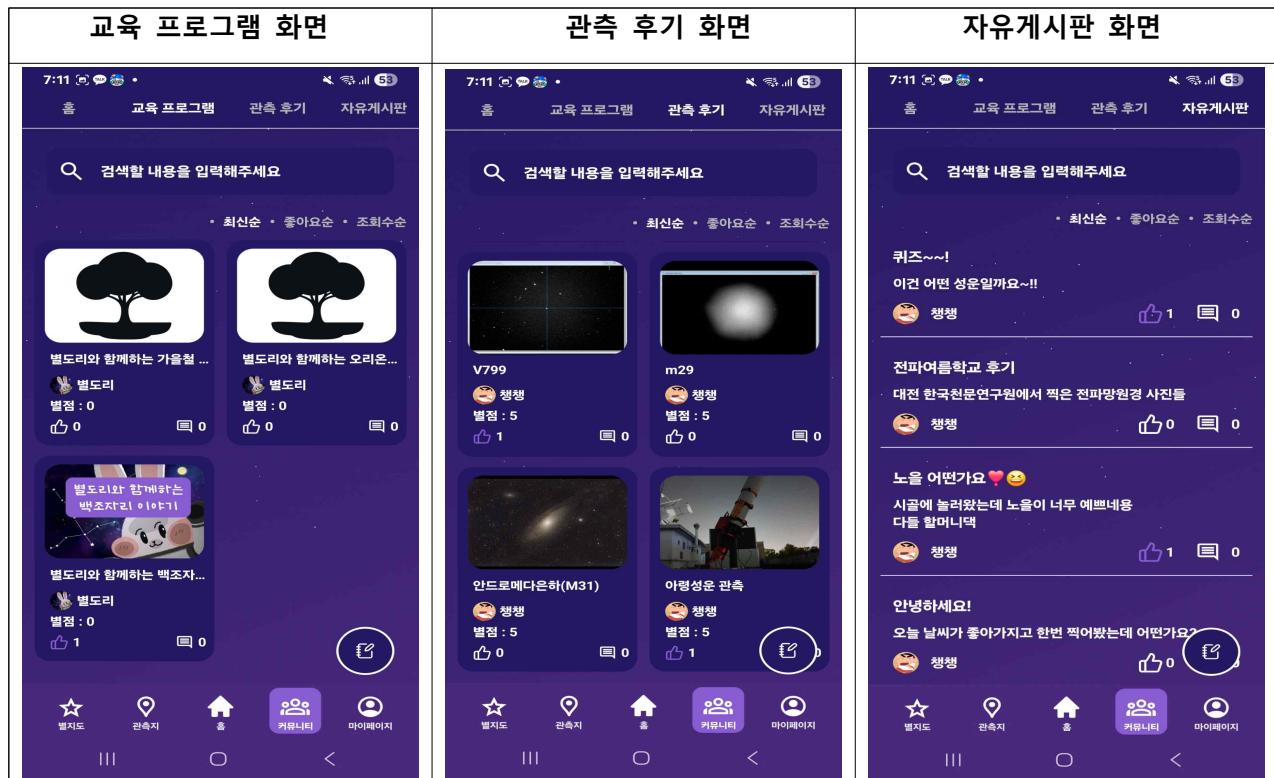


5.3.2 사용자 인터페이스 설계 (구현 시스템)

- 커뮤니티 홈



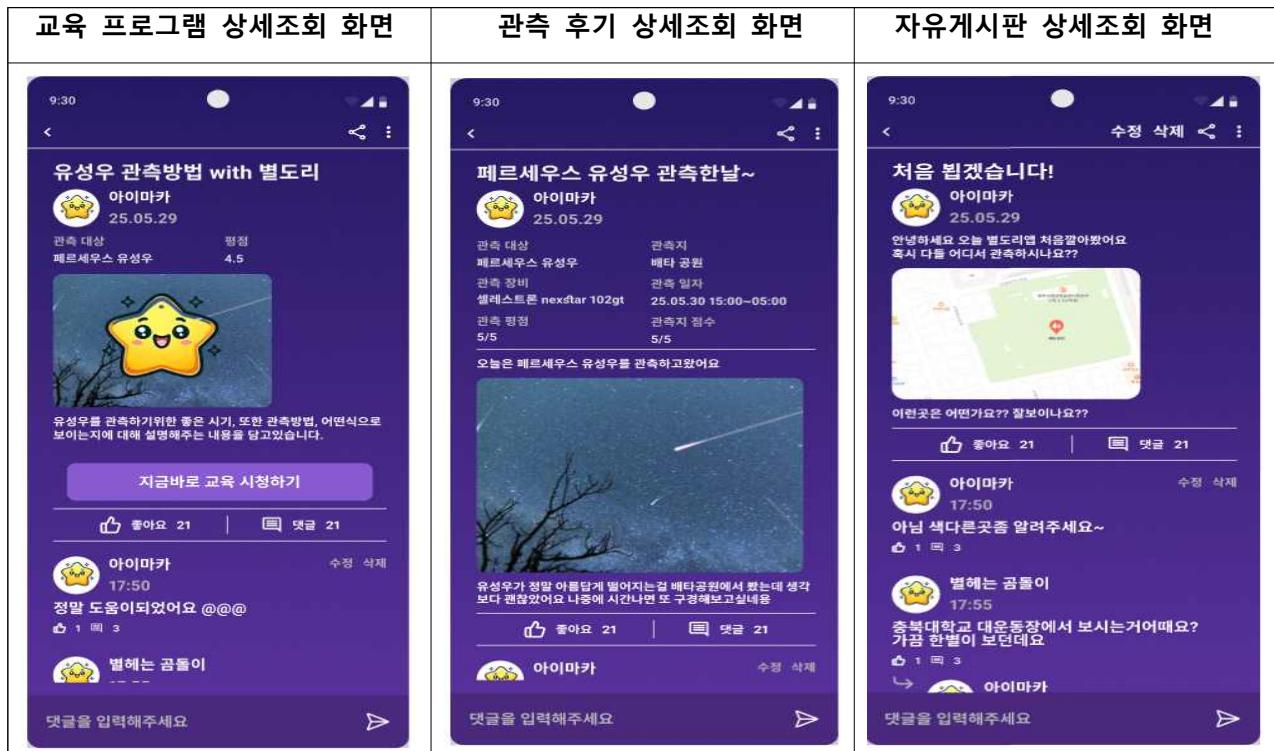
- 게시판 목록



- 게시글 작성



- 게시글 상세조회



5.3.3 핵심 알고리즘 설계 (구현 시스템)

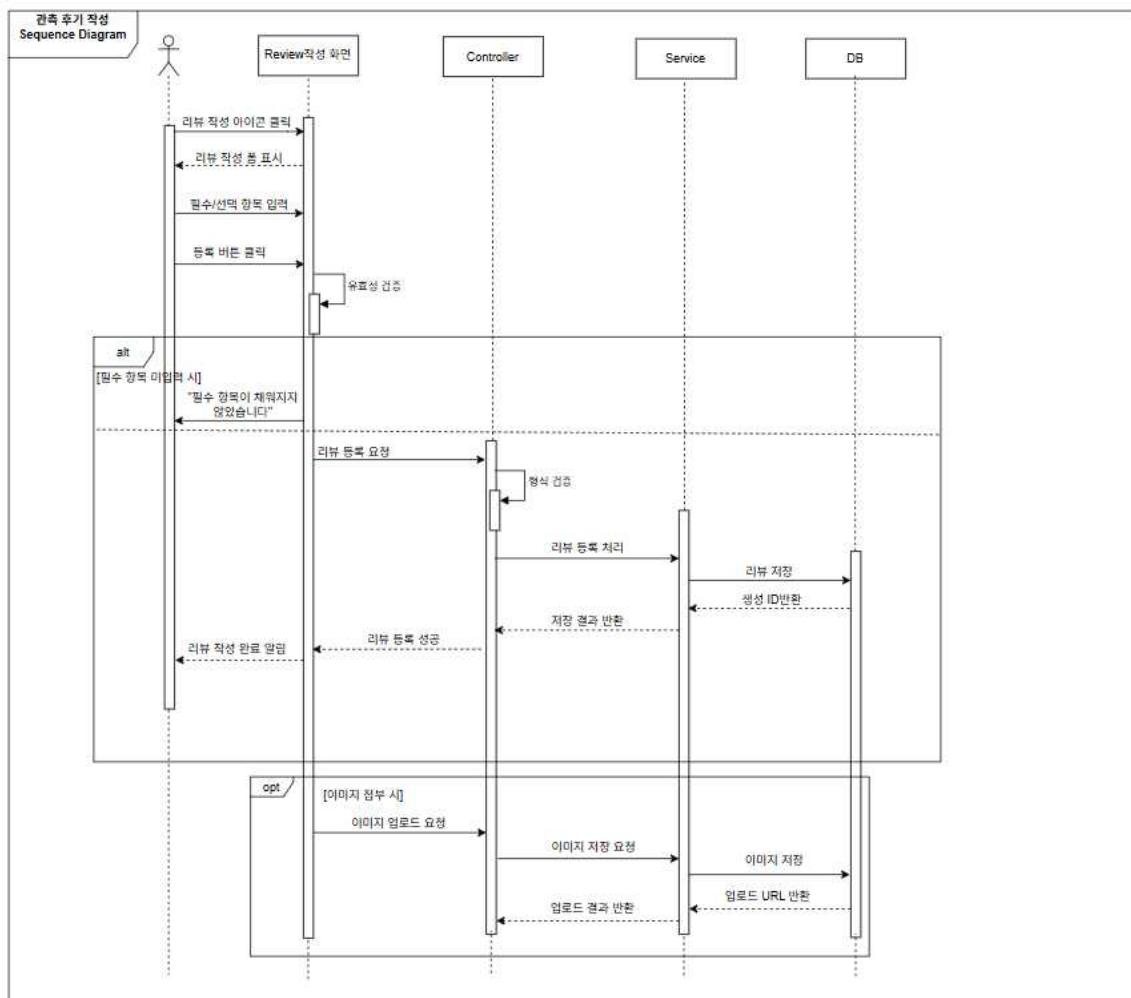
이번 빌드에서 구현된 핵심 알고리즘은 커뮤니티 기능 중심으로 설계되었으며, 사용자가 작성한 관측 후기, 자유게시글, 교육 프로그램 데이터를 표시함으로써 사용자 간의 소통과 참여를 촉진하도록 구성된다. 교육 프로그램에 대한 자세한 부분은 7차 빌드명세서에서 이루어진다.

5.3.3.1 게시글 작성 알고리즘 (대표)

게시글 작성 알고리즘은 로그인한 사용자가 관측을 한 다음 작성하는 관측 후기, 자유게시글에 대해 작성하는 과정으로 진행된다.

관측 후기 작성 알고리즘 절차 (Sequence Diagram)

본 시퀀스 다이어그램은 관측 후기 작성 시 발생하는 전체 데이터 흐름과 검증 과정을 표현한 것으로, 사용자의 입력 -> 서버 유효성 검증 -> DB 저장 -> 성공 메시지 반환의 순차적 로직을 통해 안정적으로 리뷰 데이터를 등록하는 핵심 알고리즘을 나타낸다.



1. 관측 후기 정보 입력

사용자는 ReviewWriteForm 화면에서 제목, 관측 대상, 관측지, 관측 장비, 관측 일자, 관측 평점, 내용을 입력한다.

2. 관측 후기 요청 전송

프론트엔드는 사용자가 입력한 데이터를 ReviewRepository를 통해 백엔드 서버의 /community/REVIEW/posts 엔드포인트로 전송한다.

3. 관측 후기 저장

백엔드 서버는 PostService를 통해 필수값 검증을 한 후, ReviewPostRepository를 통해 DB에 저장한다.

4. 관측 후기 저장 완료 응답

백엔드는 관측 후기 저장 응답을 프론트엔드로 반환한다.

5. 커뮤니티 페이지에 관측 후기 표시

커뮤니티 화면에서 새로 작성된 관측 후기를 표시한다. CommunityScreen → CommuReviewSection에서 ReviewViewModel을 통해 ReviewRepository.getAllReviews()를 호출해 관측 후기 목록을 불러오고, 받은 데이터를 ReviewCard의 형식으로 출력한다.

5.3.3.2 댓글 작성 알고리즘 (요약)

사용자는 게시글 상세 화면(FreeBoardDetail, ReviewDetail, EduProgramDetail) 하단의 댓글 입력창(CommentInput)에서 댓글 내용을 작성한다.

절차 요약	게시글 상세 화면에서 사용자가 댓글 내용을 입력, 전송하면 CommentsViewModel → CommentsRepository.createComment()를 통해 백엔드 서버의 POST("/community/posts/{postId}/comments") 엔드포인트로 요청이 보내지고 성공 응답을 받으면 CommentsViewModel이 내부 리스트와 댓글 수를 갱신하여 화면에 댓글이 표시된다.
----------	---

5.3.3.3 이미지 업로드 알고리즘 (요약)

사용자가 갤러리에서 선택한 이미지를 게시글 본문에 추가하고, 가장 첫번째 사진을 썸네일로 표시한다.

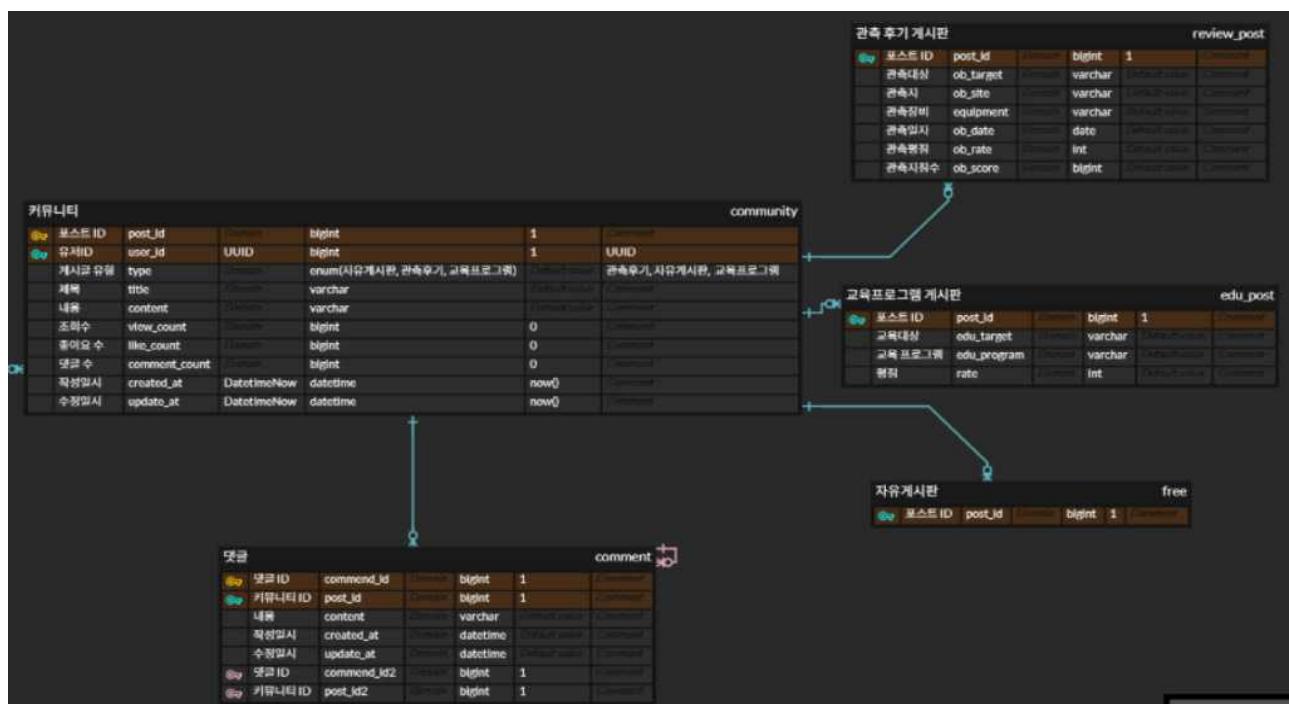
절차 요약	ActivityResultContracts.PickMultipleVisualMedia로 갤러리에서 이미지를 선택해 content Uri를 획득 → copyUriToCache(context, uri)로 Uri를 File 객체로 변환, UploadItem(status=UPLOADING, progress=0f)을 추가해 전송 중 바이트 수에 맞춰 progress를 갱신해 업로드 진행률을 표시 → FileUploadViewModel → FileRepository.uploadImage() 실행 → ProgressRequestBody(file, "image/jpeg")로 multipart/form-data 파트를 만들고 @Multipart @POST("files/image")로 서버에 업로드
----------	---

예외 처리	contentResolver.openAssetFileDescriptor()로 파일의 크기를 구해 10MB를 초과한 경우 업로드 취소
-------	---

5.3.4 데이터 설계 (구현 시스템)

본 빌드의 데이터 설계는 게시글/댓글 작성 및 조회 기능을 중심으로 이루어져 있다. 게시글/댓글 작성, 수정, 삭제, 조회를 지원하기 위해 중심 테이블인 **community** 테이블이 구성되어있고, 이 테이블을 재사용 하는 부가 테이블(**review_post**, **edu_post**, **free_post**, **comment**) 4개 및 관련 Repository 계층이 구성되어 있다. 각 테이블은 백엔드 서버(Spring Boot)와 MySQL DB를 기반으로 설계되었으며, Repository 계층을 통해 데이터 접근을 캡슐화하였다.

5.3.4.1 ERD(Entity Relationship Diagram)



5.3.4.2 테이블 설계 상세

테이블명	설명	주요 컬럼	제약조건 및 특징
Community	게시글 공통 입력 사항	id, user_id, type, content, view_count, like_count, comment_count, created_at, updated_at	3개의 type에서 공통되는 데이터 저장
Free	자유 게시글 데이터	id	커뮤니티 테이블 가져와 그대로 사용
Review Post	관측 후기 게시글 데이터	id, target, site, equipment, date, score	관측 후기 글 작성 시 필요한 데이터 저장
Edu	교육 프로그램 게시글	id, target, program, rate	교육 프로그램 글 작성

Post	데이터		시 필요한 데이터 저장
Comment	댓글 데이터	id, post_id, content, created_at, updated_at, comment_id2, post_id2	대댓글 지원을 위한 테이블 재참조

5.4. 테스트 데이터 및 결과

MUT	Test_Method() 또는 구현된 기능명			
Tid	테스트 입력 / 테스트 시나리오	예상결과	실행결과	테스트 통과
FR-501 UIR-5100	POST /community/FREE/posts 자유게시글 작성	200 OK, DB 저장, 게시글 id 반환	제목과 내용을 입력하고, 정상적으로 작성됨	통과
FR-502 UIR-503	GET /community/Review/posts?page=0 &size=10 관측 후기 게시글 목록 조회	200 OK, 제목, 내용, 조회수, 리뷰수 등 반환	페이지에 따른 게시글 수 변경 가능, 모든 항목 정상반환	통과
FR-504 UIR-5203	POST /community/posts/{postId}/likes 게시글 좋아요.	200 OK, likeCount = +1, liked=true	liked=false 상태일 때 좋아요, true일 때 취소 정상 작동	통과
FR-503 UIR-5203	POST /community/posts/{postId}/commen ts {content:"댓글123123", parentId:0} 게시글에 댓글 작성	201 Created, 댓글 저장, 게시글 commentCount +1	댓글 저장되고, 댓글 수 1 증가	통과
FR-503	작성자가 본인의 댓글을 삭제한 경우	204, 게시글 commentCount -1	204, 댓글이 삭제되었습니다. 에러 출력	통과
FR-501 UIR-5101	게시글을 작성할 때 제목, 내용 등의 필수 값이 누락되었을 경우	400 Bad Request, 제목은 필수 항목입니다. 메시지 출력	400 Bad Request와 에러 메시지 정상 출력	통과
FR-501	게시글 작성할 때 이미지 업로드 시 파일 형식이 이미지가 아닐 경우	413, 에러 메시지 출력, 이미지 업로드 되지 않음	이미지 업로드 되지 않음	통과
FR-502 UIR-503	관측 후기글에서 조회순으로 정렬 할 경우	조회수 높은 순으로 리스트로 반환함	조회수 높은순으로 게시글을 보여줌	통과
FR-504	자신의 글에 좋아요를 할 경우	좋아요 +1이 올라감	좋아요 반영이 됨	통과
FR-504	자신의 교육 프로그램 글에 피드백을 남길 경우	400, 에러 출력	자신이 작성한 글에 피드백을 남길 수 없다고 에러 출력	통과

FR-503	다른 사용자의 댓글을 삭제할 경우	에러 메시지 출력, 삭제되지 않음	삭제되지 않음	통과
FR-504 UIR-5203	좋아요 누른 게시글에 다시 좋아요를 누를 경우	토글 형식이라 좋아요 취소	좋아요 수 -1 되고 좋아요 취소됨	통과
FR-503	대댓글 작성 시	댓글 아래 대댓글이 달림	댓글 아래 대댓글 생성	통과
FR-502 UIR-5202	게시글 상세 조회 시	세부 내용을 확인할 수 있음	관측대상, 관측지, 내용 등 상세 정보 확인 가능	통과
FR-501	다른 사용자의 게시글을 삭제/수정 할 시	400, 에러 메시지 출력	자신이 쓴 글 외의 글은 삭제/수정되지 않음	통과

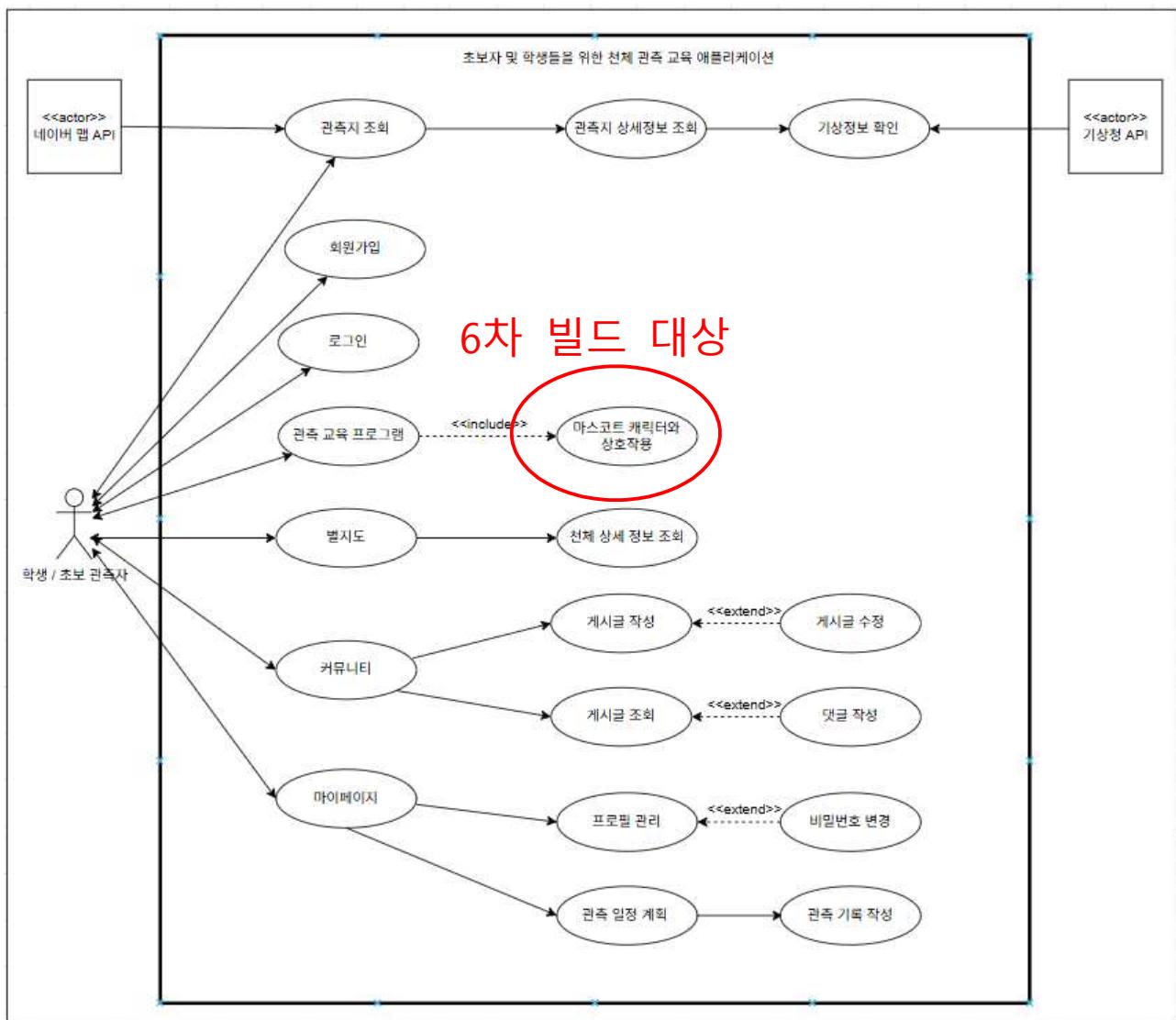
6.1 6차 빌드 개발 개요

태스크 번호	태스크 명	담당자	시작일	종료일	비고
T-601	마스코트 캐릭터 디자인	서범수	25.09.01	25.09.22	캐릭터 컨셉 수정으로 일정 일부 지연
T-602	Live2D App 이식	서범수	25.09.01	25.09.26	외부 SDK 초기세팅 이슈로 시간 소요
T-603	마스코트 캐릭터 디자인 Live2D화	서범수	25.09.22	25.09.23	-
T-604	Live2D 동작 구현	서범수	25.09.23	25.10.11	동작 수정으로 일정 지연
T-605	Live2D 동작 연동	서범수	25.09.26	25.09.30	테스트용 UI 추가
T-606	마스코트 캐릭터 말풍선	서범수	25.09.30	25.10.1	-
T-607	마스코트 캐릭터 UI	서범수	25.09.30	25.10.3	UI 상 캐릭터 외 영역 터치 이벤트 컨슘 문제 해결 “불가”
T-608	Live2D 모듈화	서범수	25.10.03	25.10.7	-
T-609	App Icon 변경	서범수	25.10.12	25.10.12	-
T-610	마스코트 캐릭터 문서 작성	서범수	25.10.12	25.10.12	-

6.2. 분석 명세

6.2.1 기능 명세

(1) Use Case Diagram (전체 시스템)



(2) Use Case 설명서 (구현 시스템)

- 마스코트 캐릭터를 통한 천체 관측 교육 UseCase

Use Case Name: 마스코트 캐릭터를 통한 천체 관측 교육	UC-001	Important Level: High
Primary Actor: 사용자		Use case type: Detail, Essential
Stakeholders and Interests:		
<p>사용자: 관심있는 천체나 지금 하늘에 보이는 별을 관측하고 싶어함.</p> <p>시스템: 사용자에게 마스코트 캐릭터가 진행하는 교육을 제공함.</p>		
Brief Description:		
<ul style="list-style-type: none"> - 사용자가 마스코트 캐릭터를 통한 천체 관측 교육을 받는다. 		
Trigger: 천체(오리온자리)를 선택하고 관측하려 가기 버튼을 사용자가 누른다.		
Type: External		
Relationships:		
Include: 관측 가이드 받기		
Extend:		
Normal Flow of Events:		
<ol style="list-style-type: none"> 1. 마스코트 캐릭터가 등장하고 사용자가 선택한 오리온자리의 관측 방법 및 위치, 관련한 이야기 등의 내용을 간단하게 설명한다. 2. 사용자는 캐릭터가 설명해 주는 것을 듣고 “다음” 버튼을 누른다. 3. 마스코트 캐릭터가 암적응에 대하여 설명한다. 4. 사용자는 암적응을 진행한 이후 설명을 듣는다. 5. 시스템은 핸드폰의 조도 시스템을 통해 화면의 밝기를 적절하게 맞춘다. 6. 캐릭터가 카시오페이아자리를 이용해서 북극성을 찾는 방법을 알려준다. 7. 시스템의 자기장 센서, 자이로스코프 기능, GPS 시스템을 통해 카시오페이아자리를 찾고 사용자 화면에 북극성의 방향을 지시한다. 8. 사용자는 배운 내용과 네비게이터를 활용해서 직접 하늘을 보며 북극성 찾는다. 9. 마스코트 캐릭터가 북극성을 통해서 오리온자리를 찾는 방법을 사용자에게 알려준다. 10. 시스템의 자기장 센서, 자이로스코프 기능, GPS 시스템을 통해 오리온자리의 위치를 방향으로 알려준다. 11. 사용자가 배운 내용과 네비게이터를 활용해서 직접 오리온자리를 찾는다. 		

Subflows:

S-1. 암적응 설명 스kip

1. 암적응 설명 듣기 요청 페이지에서 Skip 버튼을 누른다.

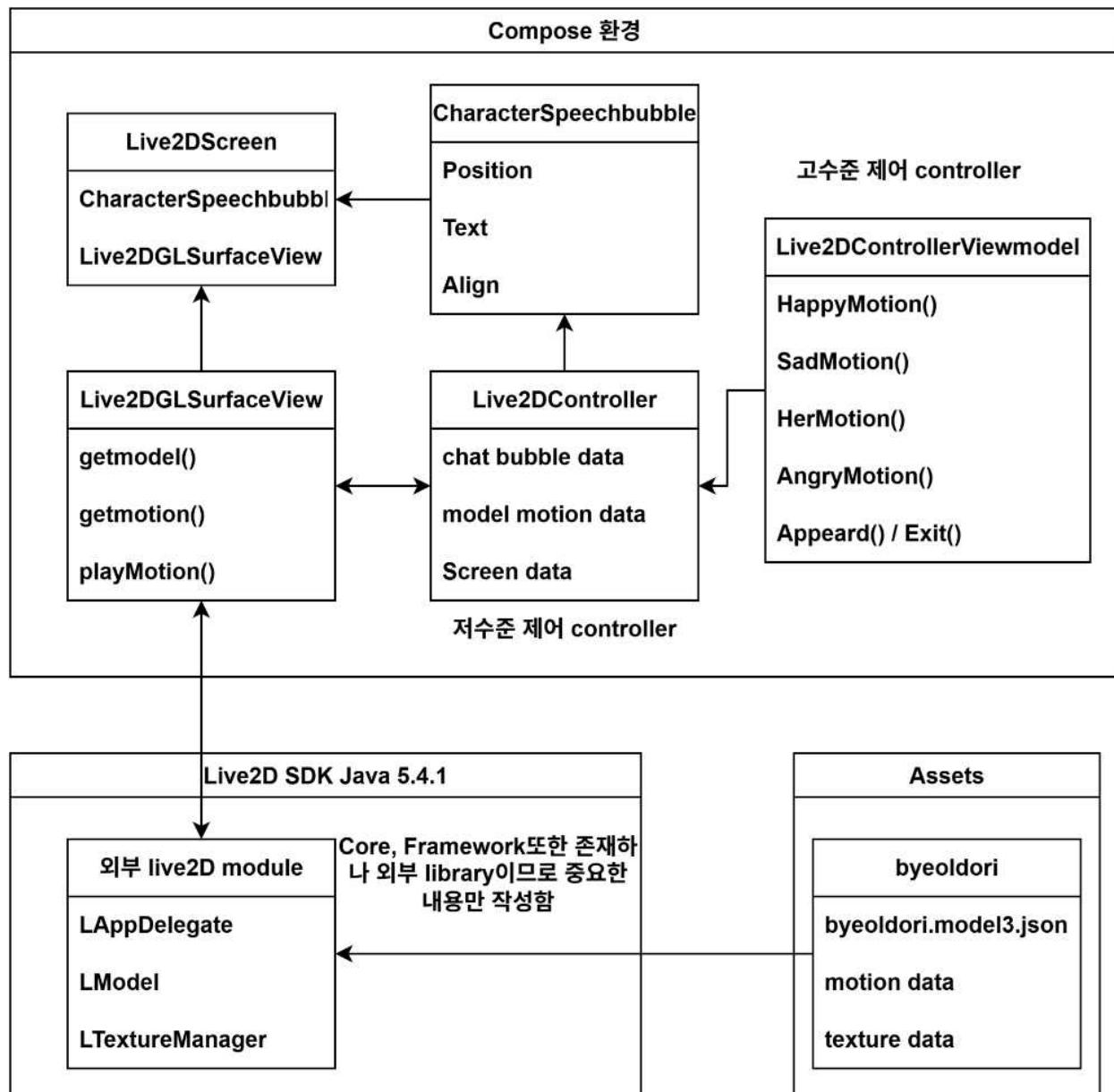
2. 별자리 찾는 방법 설명을 듣는다.

Alternative / Exceptional Flows:

E-1: 앱에서 GPS 시스템의 권한이 없을 경우 화면에 GPS 허용 권한 메시지를 띄운다.

6.2.2 구현 구조 명세

- Character package



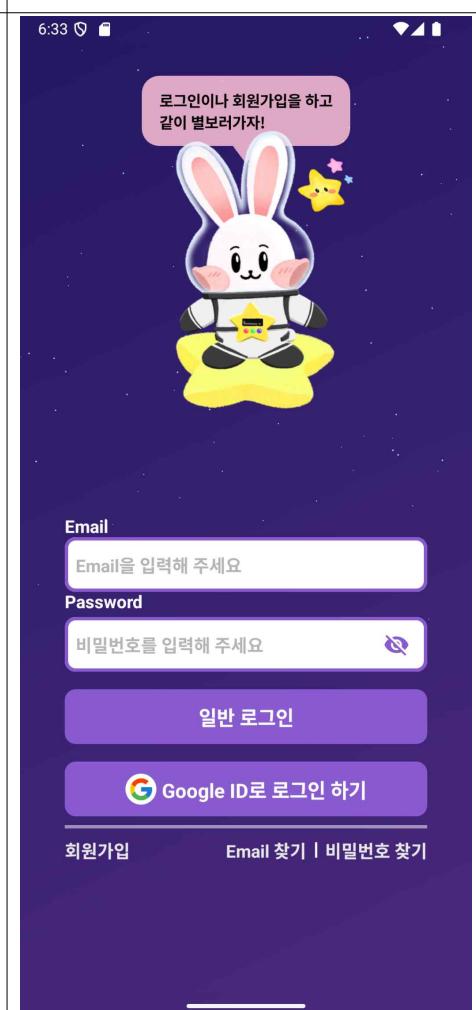
6.3. 설계 명세

6.3.1 메뉴 전개도 (구현 시스템)

마스코트 캐릭터는 별도의 메뉴 항목을 가지지 않으며, 메인 화면 및 교육 콘텐츠 화면에서 고정적으로 등장한다. 메뉴 기반 기능이 아니라 상태 기반 인터페이스 요소로 동작하므로, 메뉴전개도는 넘어간다. 추후 관련내용은 3.3, 3.4에서 서술한다.

6.3.2 사용자 인터페이스 설계 (구현 시스템)

마스코트 캐릭터 디자인

최종 디자인	실제 auth의 적용 모습
	

캐릭터 감정표현 디자인

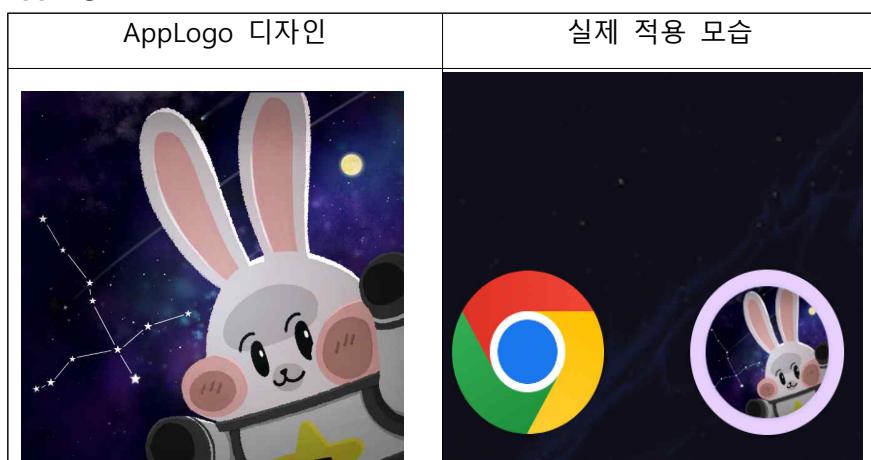
해당 감정표현은 motion이나, 본 문서에 gif를 추가하기 어려워 중간이미지로 대체함.



캐릭터 말풍선 디자인 / AppLogo 디자인



AppLogo 디자인



6.3.3 핵심 알고리즘 설계 (구현 시스템)

6.3.3.1 마스코트 캐릭터 제어 알고리즘 개요

본 알고리즘은 마스코트 캐릭터의 등장·동작·감정표현·말풍선 출력 등 UI와 상호작용하는 전체 동작 흐름을 관리하는 핵심 로직이다. **Jetpack Compose**와 **Live2D SDK**를 연동하여 GLSurfaceView 렌더링과 상태기반 UI 출력을 통합적으로 처리한다.

항목	내용
기능명	마스코트 캐릭터 제어
관련 태스크	T-1100 ~ T-1109
사용 기술	Kotlin, Jetpack Compose, Live2D SDK for Java 5.4.1, StateFlow, Coroutine
핵심 요소	Live2DScreen, Live2DController, Live2DGLSurfaceView, ViewModel

6.3.3.2 처리 흐름 요약



6.3.3.3 주요 알고리즘 절차

1) 캐릭터 표시/숨김

<pre>fun showCharacter() { live2DView?.visibility = View.VISIBLE _isVisible.value = true }</pre>	<pre>fun hideCharacter() { live2DView?.visibility = View.GONE _isVisible.value = false }</pre>
<p>역할 : 화면에 캐릭터를 표시하거나 숨긴다. 흐름 : 상태값 변경 → Compose UI 갱신 → GLSurfaceView 표시/숨김</p>	

2) 감정 표현 및 모션 실행

<pre>fun playMotion(group: String, index: Int) { live2DView?.playMotion(group, index) }</pre>	<pre>fun setExpression(exp: String) { live2DView?.setExpression(exp) }</pre>
---	--

역할 - Live2D SDK를 통해 감정 모션 및 표정 적용

흐름 - Controller 호출 → GLSurfaceView queueEvent → SDK → 렌더링 반영

3) 위치 및 크기 조정

<pre>fun moveBy(dx: Dp, dy: Dp) { _offsetX.value += dx _offsetY.value += dy refreshModifier() }</pre>	
---	--

역할 - UI 상 캐릭터의 좌표 및 크기를 실시간 조정

특징 - 화면 해상도와 비율에 따라 자동 보정

4) 말풍선 출력

<pre>fun showSpeech(text: String, tail: TailPosition, align: Alignment) { _speech.value = text _tailPosition.value = tail _alignment.value = align }</pre>	
--	--

역할 - 캐릭터가 텍스트를 출력할 때 말풍선 UI를 생성 및 정렬

흐름 - 상태 변경 → Compose UI 자동 반영

6.3.3.4 예외 및 제약사항

항목	내용	조치
터치 이벤트 충돌	캐릭터 외 영역 터치 시 UI 이벤트 소비 문제	해결 불가, 문서에 명시 (T-1106)
SDK 초기화 순서	초기화 전 모션 실행 시 NPE 발생	onStart() 강제 호출로 해결
화면 비율 문제	해상도 차이에 따른 위치 오차	centerHorizontally(), centerVertically() 제공

6.3.4 데이터 설계 (구현 시스템)

6.3.4.1 데이터 설계 개요

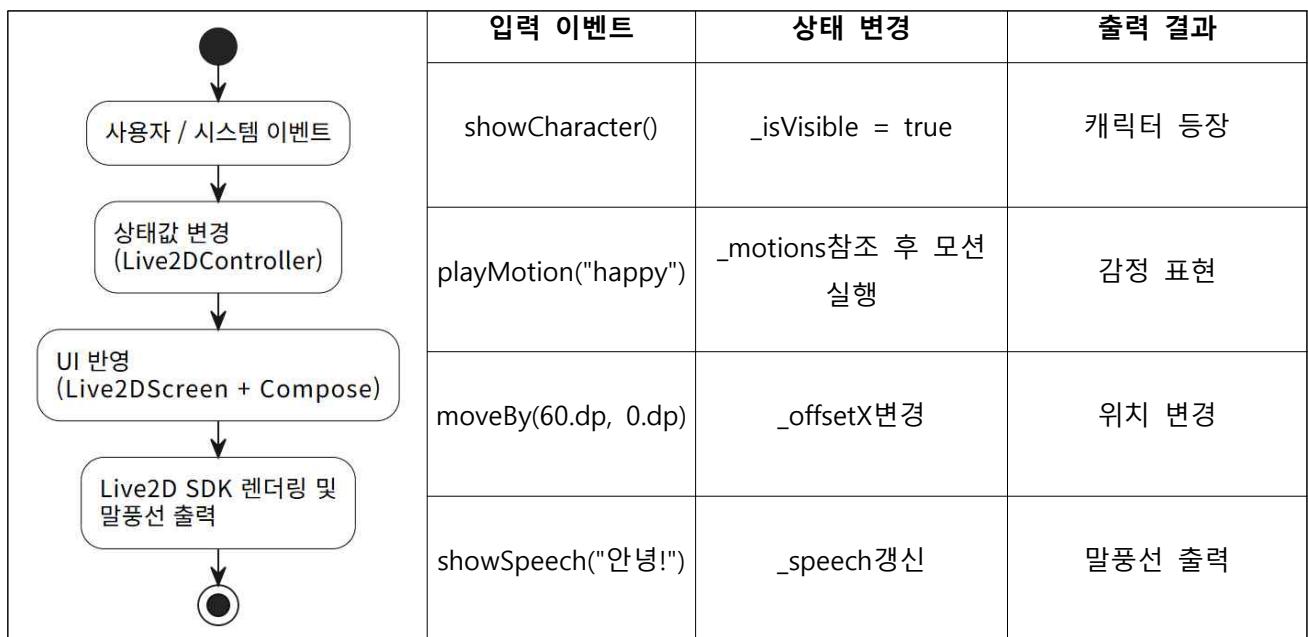
본 기능은 외부 데이터베이스 없이 상태 기반(StateFlow)으로 동작한다. 모든 상태값은 Live2DController내부에서 관리되며, Live2DScreen이 이를 실시간 구독하여 UI에 반영한다. 모델 및 모션 데이터는 assets폴더에 저장되어 Live2D SDK가 로드한다.

항목	내용
데이터 관리 방식	StateFlow 기반 실시간 상태 관리
외부 저장소	없음 (assets 디렉토리만 사용)
주요 상태	위치, 크기, 감정 표현, 말풍선 텍스트 등
데이터 사용 계층	Controller → Compose UI

6.3.4.2 상태 데이터 정의서

상태명	자료형	초기값	설명
_isVisible	MutableStateFlow<Boolean>	false	캐릭터 표시 여부
_speech	MutableStateFlow<String>	"오늘은 어떤 별을 관측해볼까?"	말풍선 텍스트
_tailPosition	MutableStateFlow<TailPosition>	Center	말풍선 꼬리 위치
_alignment	MutableStateFlow<Alignment>	TopCenter	말풍선 정렬 위치
_motions	MutableStateFlow<List<String>>	빈 리스트	Live2D 모션 그룹 목록
_offsetX, _offsetY	MutableStateFlow<Dp>	0.dp	캐릭터 위치 좌표
_width, _height	MutableStateFlow<Dp>	200.dp320.dp	캐릭터 크기
_viewModifier	MutableStateFlow<Modifier>	위치/크기 포함	UI 적용 상태

6.3.4.3 데이터 흐름



6.3.4.4 외부 리소스 데이터

항목	위치	설명
Live2D 모델 데이터	assets/byeoldori/	모델 설정(.model3.json), 모션, 텍스처
모션 데이터	assets/byeoldori/motion	모션 파일(.exp3.json)
SDK 모션 정보	런타임 호출	getAvailableMotionGroups()를 통해 로드

6.3.4.5 설계 의도

- 상태 기반 설계(StateFlow)를 통해 별도의 옵저버 없이도 UI와 상태 동기화 가능
- 데이터와 렌더링, UI를 명확히 분리하여 유지보수성을 높임
- 향후 대화 시나리오나 감정 표현을 외부 JSON/DB로 확장할 수 있는 구조를 보유
- 외부 DB가 없어도 내부 상태만으로 완결된 기능 동작 가능

6.4. 테스트 데이터 및 결과

MUT	Mascot			
Tid	테스트 입력 / 테스트 시나리오	예상결과	실행결과	테스트 통과
FR-201	showCharacter() 앱 실행 후 마스코트 캐릭터 표시 여부 확인	마스코트 캐릭터가 화면에 정상적으로 표시됨	캐릭터가 정상적으로 렌더링되며 말풍선 UI도 정상적으로 노출됨	통과
FR-201	hideCharacter() 마스코트 숨김 기능 실행	캐릭터가 화면에서 사라지고 말풍선도 사라짐	캐릭터와 말풍선이 동시에 제거됨	통과
FR-202	playMotion("Idle",0) Idle 모션 실행 시 캐릭터가 대기 동작 수행 여부 확인	캐릭터가 Idle 상태로 자연스럽게 움직임	Idle 모션이 자연스럽게 반복 재생되며 프레임 드랍이나 지연 없음	통과
FR-203	playMotion("Happy",0) Happy 모션 실행 시 감정 표현 확인	캐릭터가 웃는 표정과 함께 Happy 모션 출력	Happy 모션이 정상 재생됨	통과
FR-203	playMotion("Sad",0) Sad 모션 실행 시 감정 표현 확인	캐릭터가 우는 표정과 함께 슬픔 모션 출력	Sad 모션이 정상 재생됨	통과
FR-203	playMotion("Angryy",0) Angry 모션 실행 시 감정 표현 확인	캐릭터가 화난 표정과 함께 화남 모션 출력	Angry 모션이 정상 재생됨	통과
FR-203	playMotion("Her",0) Her 모션 실행 시 감정 표현 확인	캐릭터가 엉뚱한 표정과 함께 황당 모션 출력	Her 모션이 정상 재생됨.	통과
FR-204	CharacterSpeechBubble(text) 텍스트 입력 후 말풍선 출력 확인	입력된 텍스트가 말풍선에 정상적으로 표시됨	입력 텍스트가 실시간 반영되고 줄바꿈, 정렬 모두 정상 작동함	통과
UIR-201	controller.moveBy() 캐릭터를 X/Y축으로 이동	지정한 위치로 캐릭터가 이동함	화면 내에서 캐릭터가 자연스럽게 이동함	통과
UIR-202	controller.setSize() 캐릭터 크기 조정 기능 확인	비율 유지된 채 크기가 정상 조정됨	비율이 무너지지 않고 크기가 정상 변경되며 UI 레이아웃도 정상 유지됨	통과
UIR-204	controller.showSpeech() 긴 문장 입력 시 가독성 테스트	말풍선 내 텍스트가 자동 줄바꿈되어 읽기	긴 문장도 말풍선 영역에 자동	통과

		쉬움	줄바꿈되어 깨짐 없이 출력됨	
UIR-205	UI 터치 이벤트 캐릭터 외 UI 버튼 터치 테스트	캐릭터 영역 외 터치 시 다른 UI 정상 작동	캐릭터 주변으로 조금의 영역은 터치가 불가능함.	X (불가능)

6.5. 구현 참고 사항

5.1 터치 이벤트 미통과 항목에 대한 고찰

현 빌드에서는 Live2D 뷰(GLSurfaceView) 가 화면을 덮는 방식으로 붙어 있어, 캐릭터 외 영역의 터치 이벤트가 컨슘되는 문제를 해결하지 못했다. 원인은 GLSurfaceView의 터치 처리 특성(하드웨어 가속, 독립적인 Surface, onTouchEvent의 소비) + Compose 오버레이 구조의 결합에서 비롯된다.

GLSurfaceView는 별도 Surface로 그려지고, 이러한 GLsurfaceview는 특징상 모든 컴포넌트의 최상위 인덱스에서 구현된다 (이는 OpenGL 기반 렌더링이라 Compose와 렌더링 순서가 다름). 이를 해결하기 위해 Z-index 도입(실패), onTouchEvent의 view계층 반환(실패) 등을 실행해보았으나, 되지 않았음.

현실적인 해결방법으로 최대한 캐릭터의 크기와 view 크기가 일치하도록 하여, view가 잡아먹는 내용을 줄여야함. 그리하여 “UI 상 캐릭터 외 영역 터치 이벤트 컨슘 문제 해결 불가(T-1106)”로 명시하고 패스/폐일 관리하며, 사용자 플로우에서 캐릭터를 잠시 숨기거나(hideCharacter()), 특정 시점엔 말풍선만 노출하는방식으로 우회함.

5.2 Live2D Cubism SDK for Java 5.4.1 사용 및 라이선스 관련 사항

본 프로젝트는 Live2D Inc.의 Cubism SDK for Java 5.4.1을 사용하여 마스코트 캐릭터를 렌더링한다.

SDK 및 에셋 사용에는 제작사 라이선스/EULA의 조건 준수가 필수다.

본 빌드에서는 다음 항목을 준수·점검하며 사용한다.

1) 적용 범위

사용 SDK: Live2D Cubism SDK for Java 5.4.1

용도: 학술/졸업작품 목적의 비상업적 앱 데모(캐릭터 렌더링/모션/표정)

2) 준수 체크리스트(문서 보존용)

공식 EULA/이용약관 검토 및 버전 기록(날짜, 파일명)

상업/비상업 용도 구분 확인(졸업작품: 비상업 목적)

재배포 범위 확인: SDK 바이너리/라이브러리/샘플 코드의 포함 가능 범위

표기/크레딧 규정 준수 (필요 시 앱 내 ‘오픈소스/서드파티 고지’ 페이지에 표기)

금지 항목 확인(특정 용도 제한, 상표권·초상권·저작권 관련 주의사항 등)

저작권 표시문안 문서화

3) 권장 고지(크레딧) 문안(템플릿)

본 애플리케이션은 Live2D Cubism SDK for Java (ver. 5.4.1) 을 사용하여 구현되었습니다.

“Live2D” 및 관련 로고는 Live2D Inc. 의 상표입니다. 캐릭터 모델 및 모션 데이터의 저작권은 각 권리자에게 있습니다.

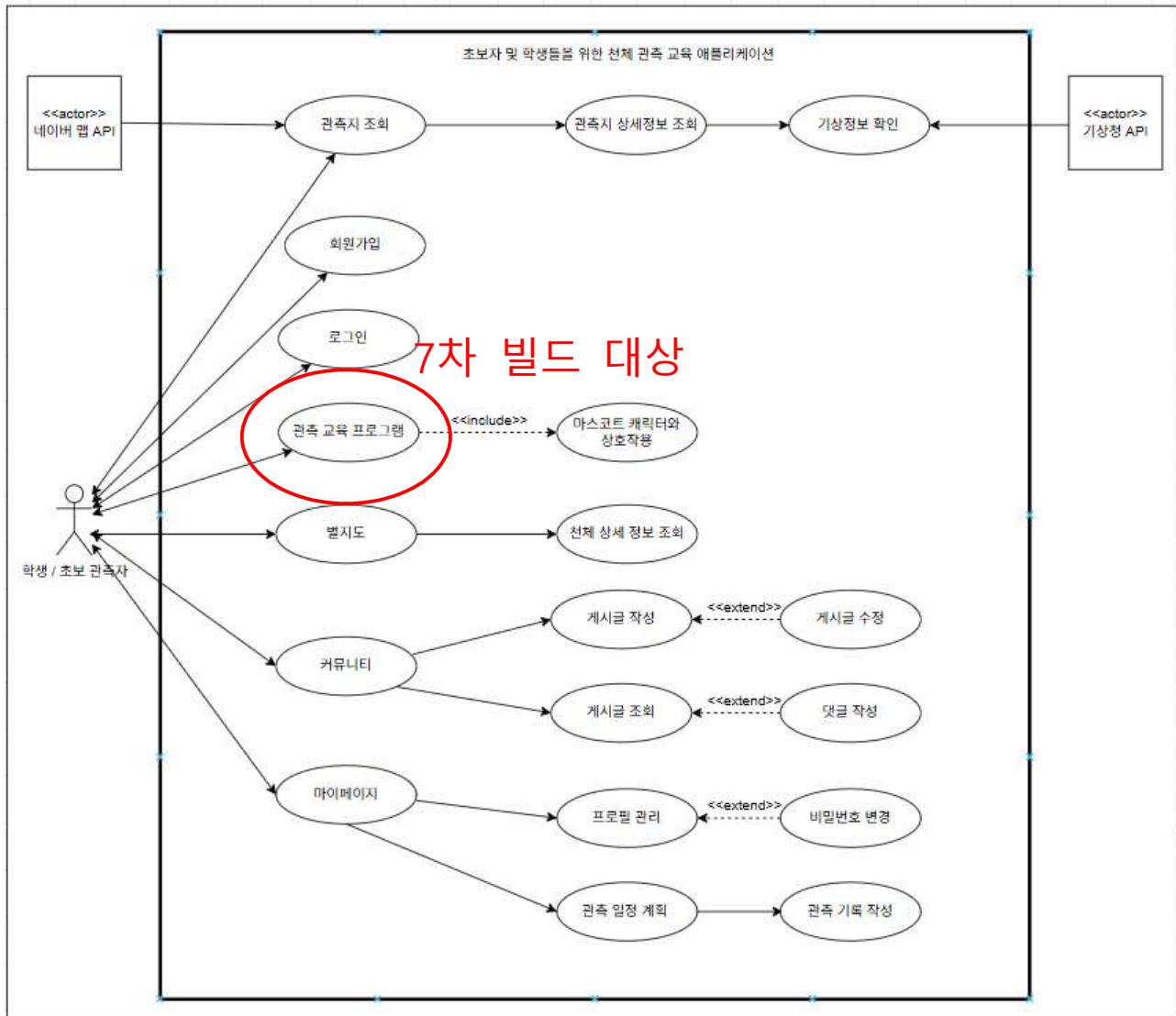
7.1. 7차 빌드 개발 개요

태스크 번호	태스크 명	담당자	시작일	종료일	비고
T-701	별지도 Controller 연동	서범수	25.10.22	25.11.01	-
T-702	Live2D Controller 연동	서범수	25.10.22	25.11.01	-
T-703	EduEngine 파서 구현	서범수	25.10.22	25.11.01	-
T-704	JSON 형식 구현	서범수	25.10.22	25.11.01	-
T-705	JSON 시나리오 로드 구현	서범수	25.11.08	25.11.08	-
T-706	교육프로그램 Overlay UI 구현	서범수	25.10.22	25.11.01	-
T-707	자동 진행 모드 구현	서범수	25.11.07	25.11.07	-
T-708	피드백 UI 구현 및 API 연동	서범수	25.11.08	25.11.08	-
T-709	교육프로그램 문서 작성	서범수	25.11.11	25.11.11	-

7.2. 분석 명세

7.2.1 기능 명세

(1) Use Case Diagram (전체 시스템)



(2) Use Case 설명서 (구현 시스템)

- 마스코트 캐릭터를 통한 천체 관측 교육 UseCase

Use Case Name: 마스코트 캐릭터를 통한 천체 관측 교육	UC-001	Important Level: High
Primary Actor: 사용자		Use case type: Detail, Essential
Stakeholders and Interests:		
<p>사용자: 관심있는 천체나 지금 하늘에 보이는 별을 관측하고 싶어함.</p> <p>시스템: 사용자에게 마스코트 캐릭터가 진행하는 교육을 제공함.</p>		
Brief Description:		
<ul style="list-style-type: none"> - 사용자가 마스코트 캐릭터를 통한 천체 관측 교육을 받는다. 		
Trigger: 천체(오리온자리)를 선택하고 관측하려 가기 버튼을 사용자가 누른다.		
Type: External		
Relationships:		
Include: 관측 가이드 받기		
Extend:		
Normal Flow of Events:		
<ol style="list-style-type: none"> 1. 마스코트 캐릭터가 등장하고 사용자가 선택한 오리온자리의 관측 방법 및 위치, 관련한 이야기 등의 내용을 간단하게 설명한다. 2. 사용자는 캐릭터가 설명해 주는 것을 듣고 “다음” 버튼을 누른다. 3. 마스코트 캐릭터가 암적응에 대하여 설명한다. 4. 사용자는 암적응을 진행한 이후 설명을 듣는다. 5. 시스템은 핸드폰의 조도 시스템을 통해 화면의 밝기를 적절하게 맞춘다. 6. 캐릭터가 카시오페이아자리를 이용해서 북극성을 찾는 방법을 알려준다. 7. 시스템의 자기장 센서, 자이로스코프 기능, GPS 시스템을 통해 카시오페이아자리를 찾고 사용자 화면에 북극성의 방향을 지시한다. 8. 사용자는 배운 내용과 네비게이터를 활용해서 직접 하늘을 보며 북극성 찾는다. 9. 마스코트 캐릭터가 북극성을 통해서 오리온자리를 찾는 방법을 사용자에게 알려준다. 10. 시스템의 자기장 센서, 자이로스코프 기능, GPS 시스템을 통해 오리온자리의 위치를 방향으로 알려준다. 11. 사용자가 배운 내용과 네비게이터를 활용해서 직접 오리온자리를 찾는다. 		

Subflows:

S-1. 암적응 설명 스킵

1. 암적응 설명 듣기 요청 페이지에서 Skip 버튼을 누른다.

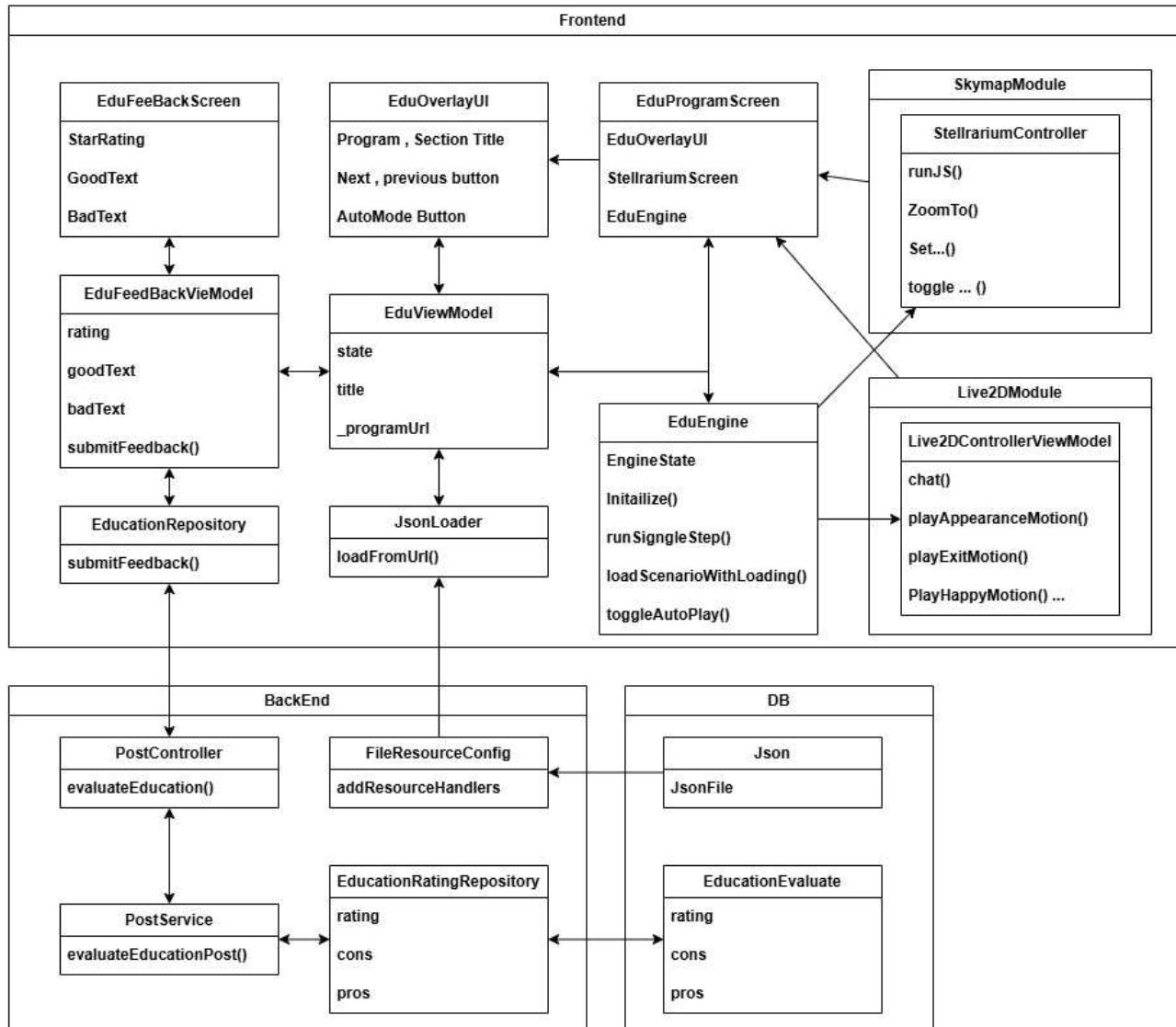
2. 별자리 찾는 방법 설명을 듣는다.

Alternative / Exceptional Flows:

E-1: 앱에서 GPS 시스템의 권한이 없을 경우 화면에 GPS 허용 권한 메시지를 띄운다.

7.2.2 구현 구조 명세

- Education Program package



7.3. 설계 명세

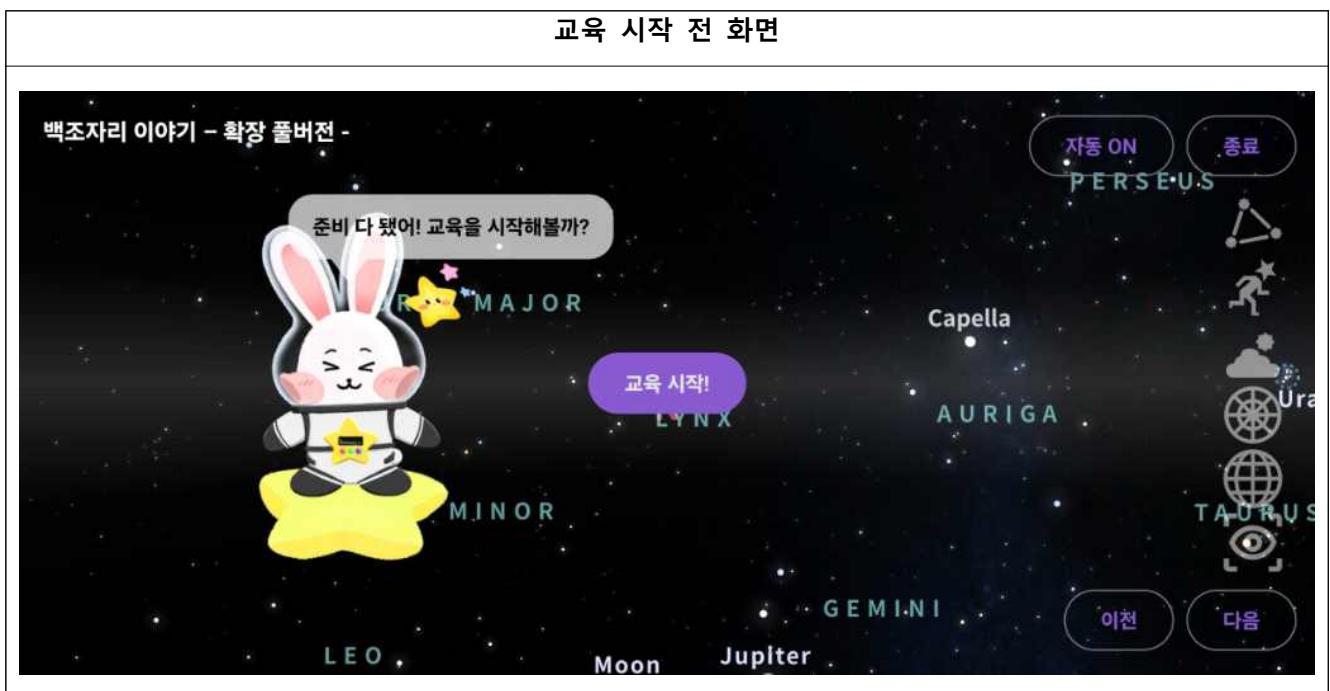
7.3.1 메뉴 전개도 (구현 시스템)

본 시스템의 교육 프로그램(EduProgram) 화면은 별도의 메뉴 계층 없이 하나의 통합 Compose 화면 내부에서 Live2D 캐릭터, 별지도(Sterrarium), 그리고 오버레이 UI를 동시에 제어하도록 구성되어 있다.

동작 흐름도	구성 요소	역할
<pre> graph TD A[프로그램 선택] --> B[시나리오 로딩 (EduEngine.loadScenario)] B --> C[교육 시작 (EduOverlayUI)] C --> D[Live2D 대사 + 별지도 시야 연동] D --> E[자동/수동 단계 진행] E --> F[교육 종료 → 피드백 입력] F --> G[피드백 전송 완료 → 종료 또는 재 시작] </pre>	StellariumScreen (SkyMap)	JS 기반 별자리 엔진으로 시야 이동, 천체 선택, 격자 표시 등 교육용 시각화 담당
	Live2DControllerView Model	마스코트 캐릭터의 대사, 감정 표현, 애니메이션 제어
	EduOverlayUI	학습 단계 표시, 자동 진행 토글, 다음/이전 버튼, 교육 종료 및 피드백 버튼 제공
	EduFeedbackScreen	학습 종료 후 별점 및 의견 입력 UI, Repository를 통해 서버로 전송
	EduViewModel / EduEngine	JSON 시나리오 로딩, 상태 관리(FSM), 시야·대사·타이머 제어

7.3.2 사용자 인터페이스 설계 (구현 시스템)

교육 프로그램 화면



교육 종료후 화면

백조자리 이야기 - 확장 풀버전 -

자동 ON

종료

모든 교육이 끝났어! 함께해줘서 고마워!



Polaris



교육 프로그램 평가하기

다시 보기

이전

다음

교육 프로그램 평가 화면

교육 프로그램 평가

교육 프로그램 별점



좋았던 점을 알려주세요

아쉬웠던 점을 남겨주세요

취소

제출하기

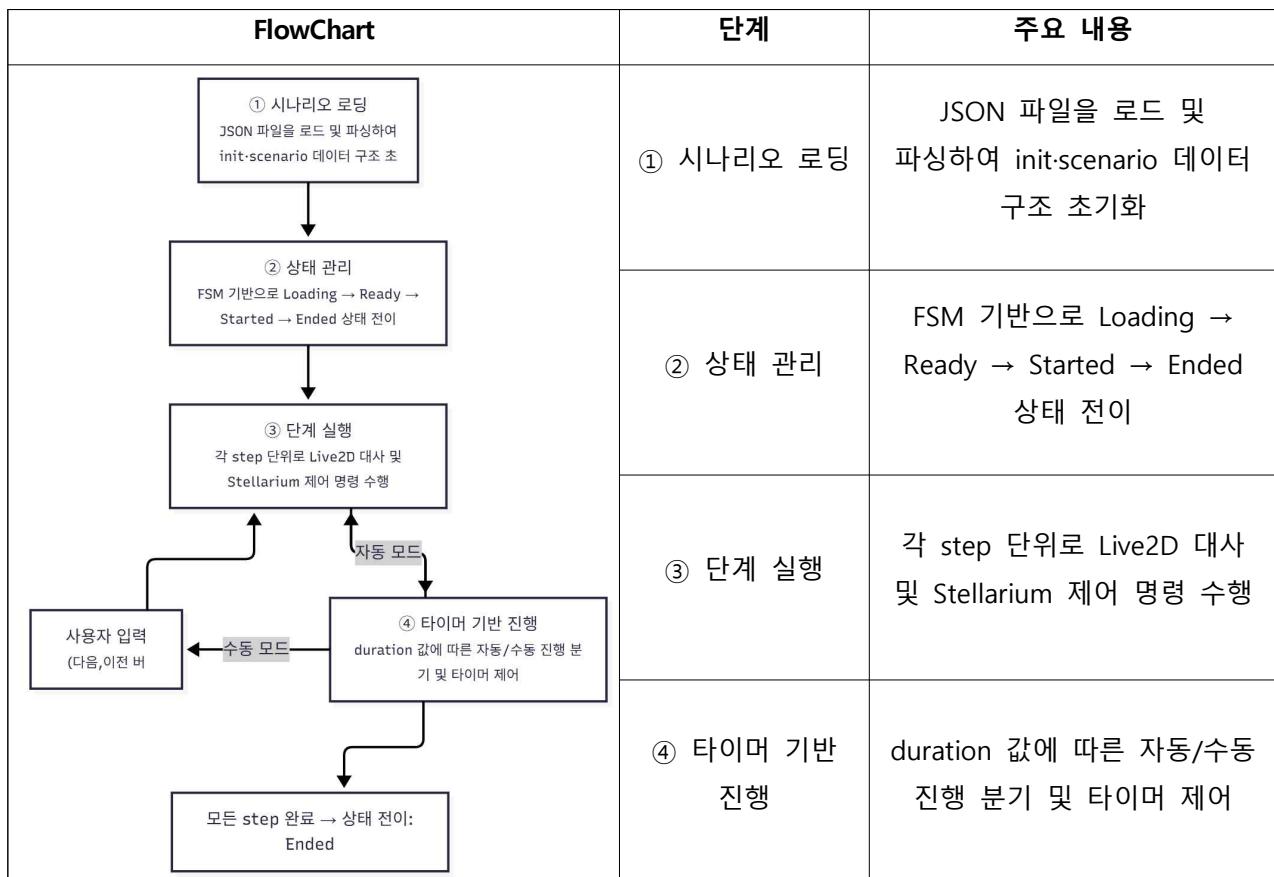
7.3.3 핵심 알고리즘 설계 (구현 시스템)

7.3.3.1概要

별도리 관측 교육 프로그램의 핵심 로직은 JSON 기반 교육 시나리오를 자동 실행하는 FSM(Finite State Machine) 구조의 EduEngine과 이를 제어·표시하는 EduViewModel 및 UI 계층으로 구성된다.

사용자는 Live2D 캐릭터의 설명과 함께 Stellarium 기반 별자리 화면을 보며 교육 과정을 단계적으로 진행할 수 있으며, 시스템은 시나리오에 정의된 시간(duration), 카메라 이동, 감정 표현, 토글 상태 등을 자동으로 처리한다.

이 알고리즘은 크게 4단계로 나뉜다:



7.3.3.2 FSM 기반 상태 전이 알고리즘 (EduEngine)

EduEngine은 교육 프로그램의 실행 단계를 FSM(Finite State Machine, 유한 상태 기계) 구조로 관리한다. 시스템은 JSON 시나리오 로드부터 종료까지의 전체 과정을 명확한 상태 전이로 구분하며, 각 상태 변화에 따라 Live2D 캐릭터의 대사와 Stellarium 제어가 동기적으로 수행된다.

상태정의	설명
sealed class EduState { object Loading : EduState() object Ready : EduState() object Started : EduState() object Ended : EduState() }	Loading : 시나리오 로드 및 초기 Stellarium 설정 단계
	Ready : 사용자 입력 대기 단계
	Started : 단계별 시나리오 실행 단계
	Ended : 모든 교육이 완료된 종료 상태

상태 변화는 내부 변수 `_state: MutableStateFlow<EduState>`로 관리되며, UI 계층(EduOverlayUI)에서 이를 실시간으로 구독하여 즉시 반영한다.

상태 전이 흐름 및 동작 요약

상태명	설명 (State Behavior)	Live2D 반응 / 출력 메시지	전이 조건 (Transition Trigger)
Loading	JSON 파일 로드 및 Stellarium 초기화 (FOV, Grid, ViewDirection 설정)	"교육 준비 중이에요!"	시나리오 로드 완료 → Ready
Ready	시나리오 로딩 완료 후 대기	"준비 다 됐어! 교육을 시작해볼까?"	start() 호출 → Started
Started	각 step 단위로 Live2D 대사, Stellarium 제어 실행	단계별 Live2D 및 Stellarium 동작 수행	모든 step 완료 → Ended
Ended	모든 교육 단계 종료	"모든 교육이 끝났어! 함께해줘서 고마워"	재시작 시 resetStateOnlyAndRestart() 호출 → Started
상태 전이 다이어그램	<pre> graph LR L[① Loading
JSON 로드 및 Stellarium 초기화] --> R[② Ready
교육 시작 대기] R --> S[③ Started
단계별 시나리오 실행] S --> E[④ Ended
모든 교육 종료] E -- "재시작" --> L </pre>		

7.3.3 시나리오 실행 알고리즘

별도리 교육 시스템의 핵심은 JSON 기반 시나리오(JSON Scenario)를 해석하고, 이를 바탕으로 Live2D 캐릭터와 Stellarium 별자리 엔진을 연동하는 것이다. 각 시나리오는 JSON 형식으로 구성되며, 교육 프로그램의 흐름을 완전히 정의한다. 시나리오는 다음과 같이 두 개의 주요 섹션으로 구분된다.

구분	설명
init	Stellarium 및 UI의 초기 상태를 정의 (시야각, 격자, 배경, 프로그램 제목 등)
scenario	교육 진행 단계를 정의하는 배열. 각 섹션에는 여러 step이 포함된다.
예시코드	<pre>{ "init": { "programTitle": "별도리와 함께하는 백조자리 이야기", "mode": "education", "fov": 60, "view": { "yaw": 0, "pitch": 45 }, "toggles": { "constellation": true, "landscape": true } }, "scenario": [{ "title": "오프닝", "steps": [{ "live2d": { "text": "안녕! 오늘은 백조자리를 배워보자!", "emotion": "Happy" }, "duration": 4000 }, { "sky": { "object": "NAME Deneb", "fov": 30 }, "duration": 6000 }] }] }</pre>

(1) 전체 구조

EduEngine은 위 JSON을 입력받아 다음 순서로 실행된다.

Flow Chart	단계 및 함수명	주요 기능
JSON 시나리오 로드 (loadScenarioWithLoading)	1. initialize(init)	초기 설정 로딩 및 Stellarium 환경 구성
init 블록 초기화 (initialize)		
시나리오 섹션 반복 실행 (runStep)	2. runStep(sectionIndex, stepIndex)	현재 단계 실행 및 자동 타이머 관리
step 내부 명령 실행 (runSingleStep)	3. runSingleStep(step)	Step 내부 명령(Live2D, Sky 등) 실행
duration 존재? Yes startTimer() 실행 No 사용자 입력 대기 다음 step으로 이동	4. startTimer (duration, onFinish)	duration 값 기반 자동 진행
모든 섹션 완료? Yes 종료 상태(Ended) 진입	5. moveToNextSectionOrEnd()	다음 섹션 전이 또는 종료 처리

(2) 초기화 알고리즘 — initialize(init: JSONObject?)

프로그램의 기본 환경과 Stellarium의 초기 상태를 설정한다.

교육 모드 전환, 시야각, 카메라 방향, 토글 요소 등을 모두 JSON에서 파싱한다.

항목	처리 내용	호출 함수
모드 설정	"education" 모드일 경우 우측 메뉴를 학습용 UI로 전환	setEducationRightBarMode()
시야각(FOV)	"fov" 값으로 초기 시야 설정	setFov()
시점(ViewDirection)	"yaw", "pitch"로 카메라 방향 지정	setViewDirection()
환경 토글	constellation, equatorialGrid, landscape, atmosphere 등을 제어	toggle*()
제목 표시	"programTitle"을 추출하여 UI 상단 표시	StateFlow 업데이트

(3) 단계 실행 알고리즘 — runStep(sectionIndex: Int, stepIndex: Int)

현재 섹션과 스텝 인덱스를 기준으로 하나의 step을 실행하고, 자동 진행 여부(_autoPlay)에 따라 다음 단계로 전환한다.

동작흐름	1. 현재 sectionIndex와 stepIndex를 업데이트 2. 코루틴 스코프 내에서 runSingleStep() 호출 3. duration 값 존재 시 startTimer()로 자동 진행 4. 스텝이 끝나면 moveToNextSectionOrEnd()로 다음 단계 이동
Flow Chart	<pre> graph LR A[① 인덱스 경신] --> B[② runSingleStep 실행] B --> C{③ duration 존재?} C -- Yes --> D[startTimer → 자동 진행] D --> E[④ moveToNextSectionOrEnd()] C -- No --> F[사용자 입력 대기] F --> E </pre>
의사코드	<pre> private fun runStep(sectionIndex: Int, stepIndex: Int) { val section = scenarioArray.getJSONObject(sectionIndex) val steps = section.getJSONArray("steps") val step = steps.getJSONObject(stepIndex) runSingleStep(step) val duration = step.optLong("duration", 0L) if (_autoPlay.value && duration > 0) startTimer(duration) { nextStep() } } </pre>

(4) 단일 스텝 처리 알고리즘 — runSingleStep(step: JSONObject)

step 내부의 각 요소(live2d, sky, duration 등)를 해석하고 Live2D 및 Stellarium에 명령을 전달한다.

항목	처리 내용	호출 함수
Live2D	"text"와 "emotion" 기반 대사 및 표정 출력	Live2DControllerViewModel.chat()
Sky.camera	"yaw", "pitch" 값으로 시야 이동	setViewDirection()
Sky.object	천체명으로 선택 및 추적	selectAndTrackObjectByName()
Sky.ra/dec	좌표 기반 이동 (포인팅/이동 선택)	moveToRaDec(), pointMoveToRaDec()
Sky.fov	시야각 조정	setFov()
Sky.toggles	격자, 대기, DSO 등 토글 제어	toggle*()

DataFlow Diagram	<pre> graph LR A[JSON Step 입력] --> B[요소 분석 (live2d / sky / duration)] B --> C{요소 타입 분기} C -- live2d --> D[Live2DControllerViewModel.chat() 대사-표정 출력] C -- sky.camera --> E[setViewDirection() 시야 이동] C -- sky.object --> F[selectAndTrackObjectByName() 천체 선택] C -- sky.ra/dec --> G[moveToRaDec() / pointMoveToRaDec() 좌표 이동] C -- sky.fov --> H[setFov() 시야각 변경] C -- sky.toggles --> I[toggleConstellation(), toggleGrid() 토글 제어] D --> J[실행 완료 → duration 확인] E --> J F --> J G --> J H --> J I --> J </pre>
예시 Step	<pre> { "live2d": { "text": "이제 데네브를 찾아볼까?", "emotion": "Happy" }, "sky": { "object": "NAME Deneb", "fov": 25, "toggles": { "constellation": true, "equatorialGrid": false } }, "duration": 7000 } </pre>

(5) 자동 진행 알고리즘 — startTimer(duration, onFinish) (요약)

각 단계의 duration 값을 기반으로 타이머를 실행하며, 시간이 만료되면 nextStep()을 자동 호출한다.

수동 모드일 경우 타이머는 단순히 시간만 갱신하고 종료 후 대기한다.

조건	동작
_autoPlay = true	타이머 종료 시 nextStep() 자동 호출
_autoPlay = false	남은 시간만 갱신 후 대기
Job 취소	타이머 즉시 중단

7.3.3.4 자동/수동 전환 알고리즘

사용자가 오버레이 UI의 “자동 ON/OFF” 버튼을 통해 교육 진행 방식을 전환할 수 있다. 현재 모드는 MutableStateFlow<Boolean>(_autoPlay)로 관리되어, 변경 시 Compose UI에 즉시 반영된다.

모드	동작 방식
자동 진행 (AutoPlay = true)	각 step의 duration 값에 따라 자동으로 다음 단계 실행
수동 진행 (AutoPlay = false)	사용자가 “다음/이전” 버튼을 눌러 직접 단계 이동

7.3.3.5 재시작 및 종료 알고리즘

학습 도중 또는 완료 후 프로그램을 다시 시작하거나 종료할 수 있다.

함수	동작
resetStateOnlyAndRestart()	모든 Job을 취소하고 상태를 초기화한 후 첫 Step부터 재시작
closeProgram()	EduEngine 실행을 중단하고 Live2D 종료 메시지를 출력 후 eduClose() 호출

7.3.3.6 EduViewModel 제어 알고리즘

EduEngine과 Compose UI를 연결하는 뷰모델로, 상태 변화와 사용자 입력을 관리한다.

함수	동작
preloadScenario()	JSON 로드 및 초기화
start(), next(), prev()	엔진 상태 전이 제어
state.collect	EduEngine 상태 변화 감시 → Ended 시 Live2D 종료 모션 실행

7.3.3.7 피드백 처리 알고리즘

교육 종료 후 사용자 피드백(별점, 의견)을 서버로 전송한다.

사용자 입력 수집 → FeedbackRequest(score, pros, cons) DTO 생성

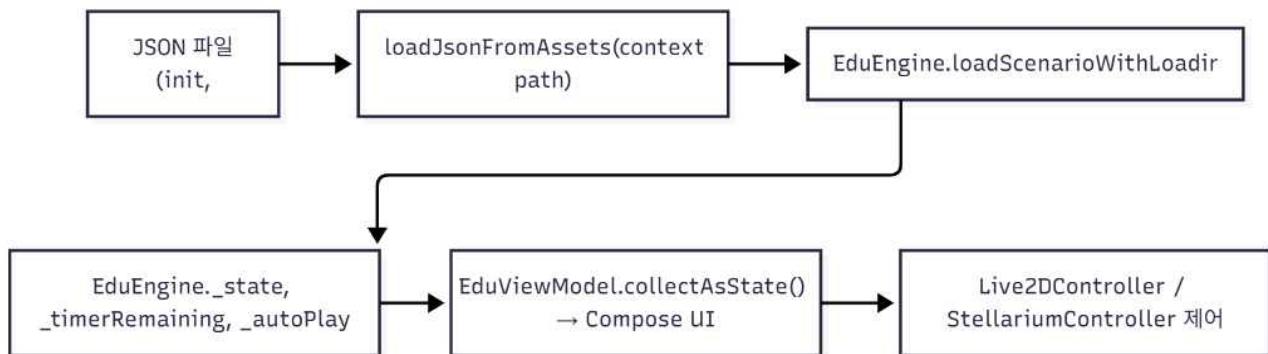
EducationRepository.submitFeedback() 호출

성공 시 “평가해주셔서 감사합니다!” 메시지 표시, 실패 시 예외 처리

7.3.4 데이터 설계 (구현 시스템)

7.3.4.1 데이터 구조 개요

본 시스템의 데이터 구조는 EduEngine을 중심으로 설계되어 있으며, JSON 기반 교육 시나리오를 로드하고, 실행 상태를 StateFlow로 관리하며, UI 계층(EduOverlayUI, Live2DControllerViewModel, StellariumController)에 실시간으로 반영한다. 주요 데이터 흐름은 다음과 같다.



7.3.4.2 주요 데이터 모델 정의

EduEngine 관련 내부 데이터

EduState	EduEngine
<pre> sealed class EduState { object Loading : EduState() object Ready : EduState() object Started : EduState() object Ended : EduState() } </pre>	<pre> class EduEngine @Inject constructor(private val stellarium: StellariumController, private val live2d: Live2DControllerViewModel) { private val _state = MutableStateFlow<EduState>(EduState.Loading) private val _autoPlay = MutableStateFlow(true) private val _timerRemaining = MutableStateFlow(0L) private var scenarioArray: JSONArray = JSONArray() private var initObject: JSONObject? = null private var sectionIndex = 0 private var stepIndex = 0 } </pre>

속성명	자료형	역할
_state	MutableStateFlow<EduState>	FSM 상태 관리 (Loading → Ready → Started → Ended)
_autoPlay	MutableStateFlow<Boolean>	자동/수동 진행 모드 플래그
_timerRemaining	MutableStateFlow<Long>	남은 시간 실시간 업데이트
scenarioArray	JSONArray	시나리오 전체 섹션 배열
initObject	JSONObject	초기 설정 정보 (FOV, 토플 등)
sectionIndex, stepIndex	Int	현재 진행 중인 섹션 및 단계 인덱스

7.3.4.3 JSON 시나리오 구조

EduEngine은 JSON 포맷을 기반으로 교육 프로그램 전체를 정의한다. JSON은 두 개의 최상위 키(init, scenario)로 구성된다.

Json	<pre>{ "init": { "programTitle": "별도리와 함께하는 백조자리 이야기", "mode": "education", "fov": 60, "view": { "yaw": 0, "pitch": 45 }, "toggles": { "constellation": true, "landscape": true } }, "scenario": [{ "title": "오프닝", "steps": [{ "live2d": { "text": "안녕! 오늘은 백조자리를 배워보자!", "emotion": "Happy" }, "duration": 4000 }, { "sky": { "object": "NAME Deneb", "fov": 30 }, "duration": 6000 }] }] }</pre>
-------------	---

7.3.4.4 Step 처리용 데이터 구조

각 Step은 runSingleStep() 내부에서 파싱되어 아래 구조에 매핑된다.

Live2DAction	SkyAction
<pre>data class Live2DAction(val text: String, val emotion: Emotion)</pre>	<pre>data class SkyAction(val objectName: String? = null, val yaw: Double? = null, val pitch: Double? = null, val fov: Double? = null, val toggles: JSONObject? = null)</pre>

구분	주요 필드	설명
Live2DAction	text, emotion	캐릭터 대사와 표정 정보
SkyAction	objectName, yaw, pitch, fov, toggles	Stellarium 카메라 이동 및 토글 설정
duration	Long	자동 진행 타이머 값(ms)

7.3.4.5 실행 중 상태 데이터 구조 (EduViewModel)

EduViewModel은 EduEngine의 상태를 UI 계층에 연결하는 역할을 수행한다.

코드	속성명	역할
<pre>@HiltViewModel class EduViewModel @Inject constructor(private val repository: EducationRepository, private val engine: EduEngine) : ViewModel() { val state = engine.state.asStateFlow() val autoPlay = engine.autoPlay.asStateFlow() val timerRemaining = engine.timerRemaining.asStateFlow() fun preloadScenario(context: Context, path: String) { val json = loadJsonFromAssets(context, path) engine.loadScenarioWithLoading(json) } fun start() = engine.start() fun next() = engine.nextStep() fun prev() = engine.prevStep() }</pre>	state	FSM 상태를 Compose UI에 전달
	autoPlay	자동/수동 모드 실시간 반영
	timerRemaining	남은 시간 표시용
	preloadScenario()	JSON 로드 및 파서 실행
	start() / next() / prev()	단계 전환 제어

7.4. 테스트 데이터 및 결과

MUT	EduProgram			
Tid	테스트 입력 / 테스트 시나리오	예상결과	실행결과	테스트 통과
UIR-100	천체관측 교육 프로그램 화면 로드	Live2D·별지도·오버레이 UI 통합 표시	정상 동작	통과
UIR-101	교육프로그램 전용 화면 구성	통합 Compose 화면에서 Live2D + Stellarium 동시 출력	정상 작동	통과
UIR-102	대화식 정보 제공	캐릭터가 단계별 시나리오에 맞춰 자연스럽게 설명	정상 동작	통과
UIR-103	단계 전환 기능	"이전/다음" 버튼으로 단계 이동 가능	정상 이동	통과
UIR-104	관측 교육 콘텐츠 반영	시나리오에 따라 Stellarium 화면이 실시간으로 변화	정상 반영	통과
UIR-105	피드백 입력 인터페이스	학습 종료 후 별점·의견 입력 및 서버 전송	정상 전송	통과
UIR-106	자동 진행 모드 토글	Auto ON → 자동 전환 / OFF → 수동 대기	정상 작동	통과
FR-100	관측 교육 프로그램 전체 실행	init → scenario → 종료까지 정상 동작	정상 실행	통과
FR-101	단계별 학습 기능	캐릭터 중심의 단계별 학습 진행	정상 진행	통과
FR-102	교육 프로그램 선택	사용자가 원하는 교육 선택 및 실행	정상 선택	통과
FR-103	Live2D 연동 제어	대사·표정이 정상 반영	정상 반영	통과
FR-104	별지도(Stellarium) 제어	천체 이동, 시야 변경, 토글이 정상 작동	정상 동작	통과
FR-105	가이드 건너뛰기	Skip 시 다음 단계로 즉시 이동	정상 동작	통과
FR-106	시야-대사 동기화	대사 출력과 시야 이동의 타이밍 일치	정상 일치	통과
FR-107	자동 진행 제어 기능	AutoPlay=true → 자동 전환, false → 수동 전환	정상 동작	통과
FR-108	확장성 확보	JSON 구조 유지 상태에서 신규 콘텐츠 추가 가능	정상 작동	통과

8. 기타 구현 사항 참고

해당없음.