

안녕하세요, 맡은 일은 깔끔하게, 말이 통하는 App 개발자 신세영 입니다.

하루 5cm씩이라도 멈추지 않고 꾸준한 성장을 지향합니다.

E-mail sinsy8989@gmail.com

Github <https://github.com/stevey-sy>

Blog https://velog.io/@shin_stealer/posts

Contents.

1 Projects.

- 한화 갤러리아 백화점 App 개선사업 수행 및 유지보수
- 오스템 임플란트 OneClick M 의사용 App 개발 사업
- 현대홈쇼핑 쇼핑 라이브 개선 사업
- LS 엠트론 조사료 플랫폼 카우잇츠 App 개발 사업
- IDAEYE 영유아 수면 자세 감지 App 개발 및 유지보수

2 Side Project.

- 독서 기록일지 App, 오독오독

한화 갤러리아 App 개선 사업 및 유지 보수 수행

2022.12 ~ 현재

Android Native App

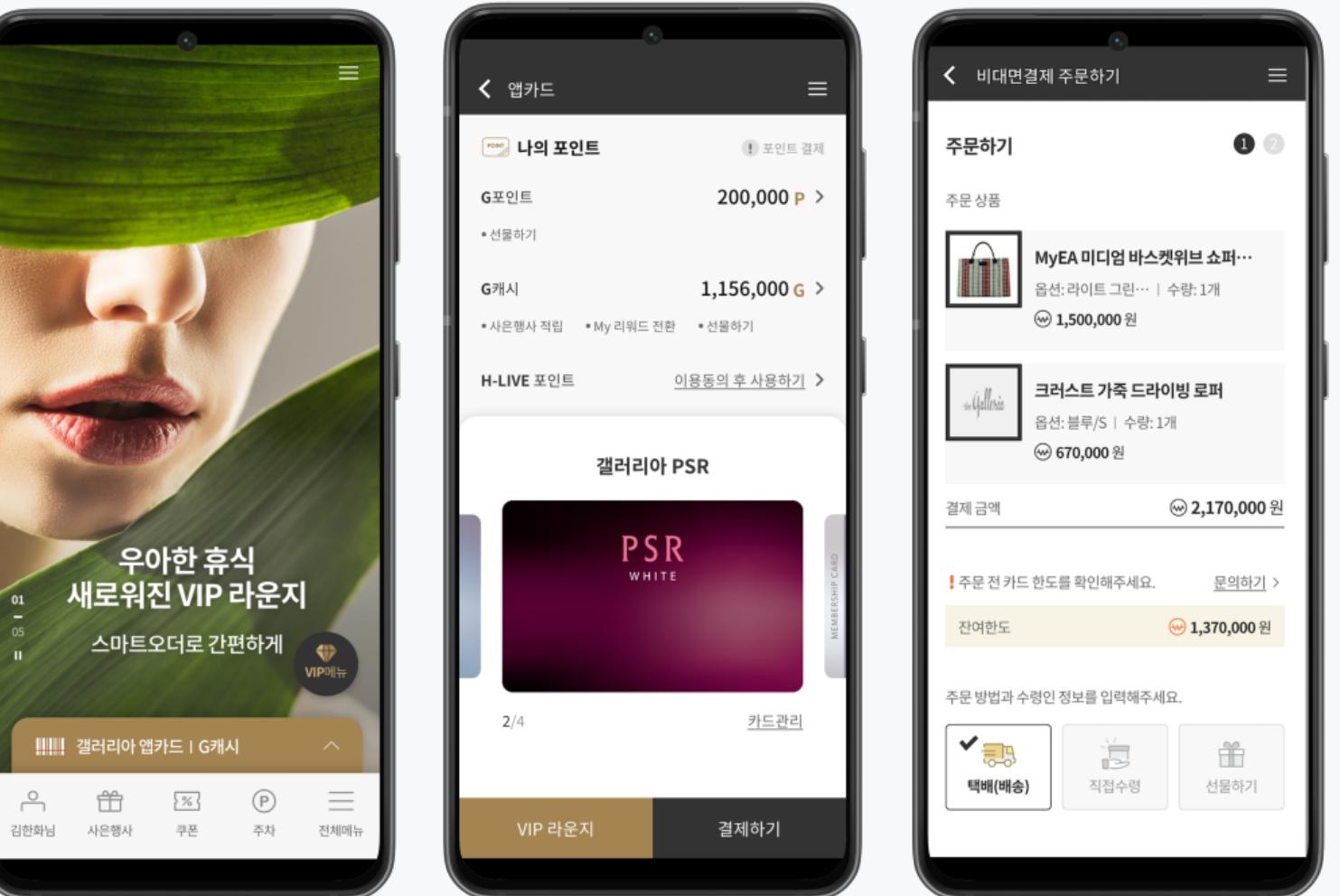
iOS Native App

개요

갤러리아 앱은 백화점 고객을 위한 스마트 쇼핑 도우미 앱입니다.

앱카드 결제/포인트, 쿠폰/비대면 결제를 한 번에 이용할 수 있으며, 주차, 매장 정보, VIP 이벤트 등 다양한 부가 기능도 제공합니다.

3차례의 앱 고도화 사업에 참여하였으며, 현재까지 유지 보수 업무를 담당해오고 있습니다.



기술스택

- Java / Kotlin / Objective-C
- Coroutines / Flow / MVVM Architecture (View - DataBinding - ViewModel - Model)
- Retrofit / OkHttp / Crashlytics / Firebase Analytics
- Gitlab CI CD

역할 / 기여

- Google Play Store 배포 관리, **총 다운로드 35만+ 달성**,
- 앱 비정상 종료율 0.11% 유지** (업계 평균 1% 이하 대비 안정적인 수준)
- 기존 레거시(Java 기반 코드) 소스를 **Kotlin으로 마이그레이션**하여 코드 품질 및 유지보수성 향상
- 비대면 결제 기능 도입으로 전월 대비 **백화점 매출 5% 증가** 및 **앱 사용률 12% 증가**에 기여
- 앱 전체 UI/UX 관리, 백엔드 개발자와 협업하여 업데이트 된 API 반영, 앱 안정성 유지

한화 갤러리아 App 개선 사업 및 유지 보수 수행

2022.12 ~ 현재

Android Native App

문제 해결 및 조치

Case 1: Api 요청 데이터 암복호화 처리 실패 이슈

Why

- 결제 기능 추가로 인해 요청 데이터 크기가 증가하면서, 암호화 처리 시 RSA 블록 크기 한계(256바이트)를 초과하여 데이터 암복호화 실패 케이스 발생.

How

- 암호화할 요청 데이터를 UTF-8 인코딩 후, RSA 블록 크기를 245바이트 (256 - 11) 씩 분할하여, 암호화된 데이터를 결합하였지만 간헐적으로 실패 케이스 발생
- 내부적으로 추가되는 패딩 메타데이터, 헤더 이슈로 추정.
- 이를 해결하기 위해 블록 크기를 200바이트 씩 분할하는 것으로 조정
- 평문 데이터 → UTF-8 인코딩으로 바이트화 → 최대 200바이트 씩, 데이터 Chunking → 각 블록에 RSA 암호화 적용
→ 암호화된 블록 병합 → Hex 문자열로 변환

Outcome

- 패딩 및 인코딩 특성으로 인한 암호화 실패 문제를 해결, 데이터 암호화 처리의 안정성과 신뢰성을 확보.
- 요청 데이터 암호화 오류를 방지하여 앱의 보안성과 안정성을 강화하고, 문제 해결 역량을 배양.

암호화 로직 설계에서 데이터 크기, 인코딩 특성, 예외 처리를 포함한 안정성의 중요도를 깨달았습니다.

◀ 비대면결제 주문하기 ⌂

주문하기 1 2

주문 상품

 MyEA 미디엄 바스켓워크 쇼퍼…
옵션: 라이트 그린… | 수량: 1개
₩ 1,500,000 원

 크루스트 가죽 드라이빙 로퍼
옵션: 블루/S | 수량: 1개
₩ 670,000 원

결제 금액 ₩ 2,170,000 원

주의사항 주문 전 카드 한도를 확인해주세요. 문의하기 >

잔여한도 ₩ 1,370,000 원

주문 방법과 수령인 정보를 입력해주세요.

택배(배송) 직접수령 선물하기

이름 *

받으실 분

연락처 *

‘-’ 제외 숫자만 입력

다음

한화 갤러리아 App 개선 사업 및 유지 보수 수행

2022.12 ~ 현재

Android Native App

문제 해결 및 조치

Case 2: RecyclerView의 무한 스크롤 이벤트 시 성능 저하 이슈

Why

- 초기 버전에서는 RecyclerView로 내역 데이터를 조회할 때, 스크롤이 하단에 도달하면 다음 데이터를 API로 불러와 리스트에 append 처리.
- 중복 호출 및 데이터 일관성 문제를 유발했고, 로딩 중 중복 요청이나 스크롤 성능 저하, 데이터 꼬임 현상 발생.

How

- 최신 Android 아키텍처 컴포넌트의 Paging 3 라이브러리를 도입하여, 효율적인 데이터 로드 및 페이지네이션 처리 구현.
- ListAdapter와 DiffUtil을 활용하여 데이터 변경을 효율적으로 감지하고 최소한의 View 갱신으로 성능 최적화.

Outcome

- 스크롤 성능 개선 및 중복 요청 방지, 안정적인 데이터 로드 구현.
- 코드 품질 및 유지보수성 향상, 최신 Android 아키텍처 대응력 강화.**
- 사용자 경험 개선 및 데이터 로드 오류 감소

Paging 3와 ListAdapter 도입 → 데이터 처리와 UI 성능 개선, 최신 Android 아키텍처로 문제 해결 능력 강화하였습니다.



항목	RecyclerView + append (기존 구현)	Paging 3
데이터 로딩 방식	스크롤 이벤트 감지 후 직접 다음 페이지 로딩 <small>PagingSource 가 자동으로 페이지 계산 및 요청</small>	
상태 관리	로딩/에러 상태 수동 처리 <small>로딩, 에러, 리프레시 상태 내장 (LoadState)</small>	
메모리/리소스	전체 리스트 메모리에 올라올 수 있음 <small>필요한 만큼만 메모리에 유지 (페이징 처리)</small>	
테스트/확장성	상태 처리 분산되어 어려움 <small>일관된 흐름으로 테스트 용이</small>	

오스템 임플란트 OneClick M 의사용 App 개발 사업

2021.04 ~ 2021.09

Android Native App

개요

치과 진료용 Tablet 전용 앱으로, 의사가 환자 접수부터 진료까지 효율적으로 관리할 수 있는 솔루션입니다.

오늘의 환자 리스트, 문진동의서, 전자 차트, 팬차트, 예약 관리 등 주요 기능을 탑재하여 진료 편의성을 극대화했습니다.

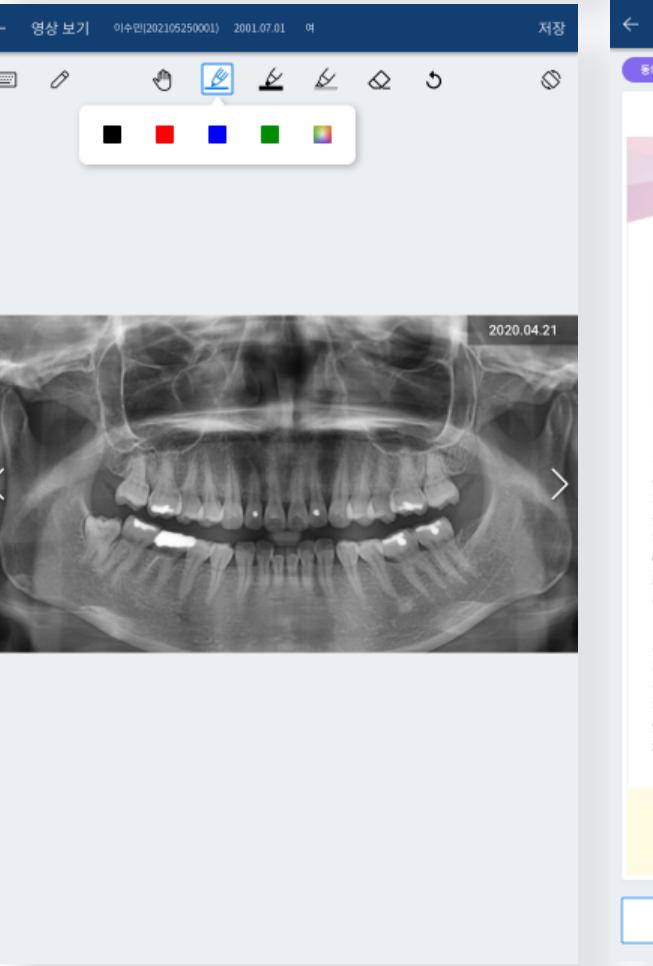
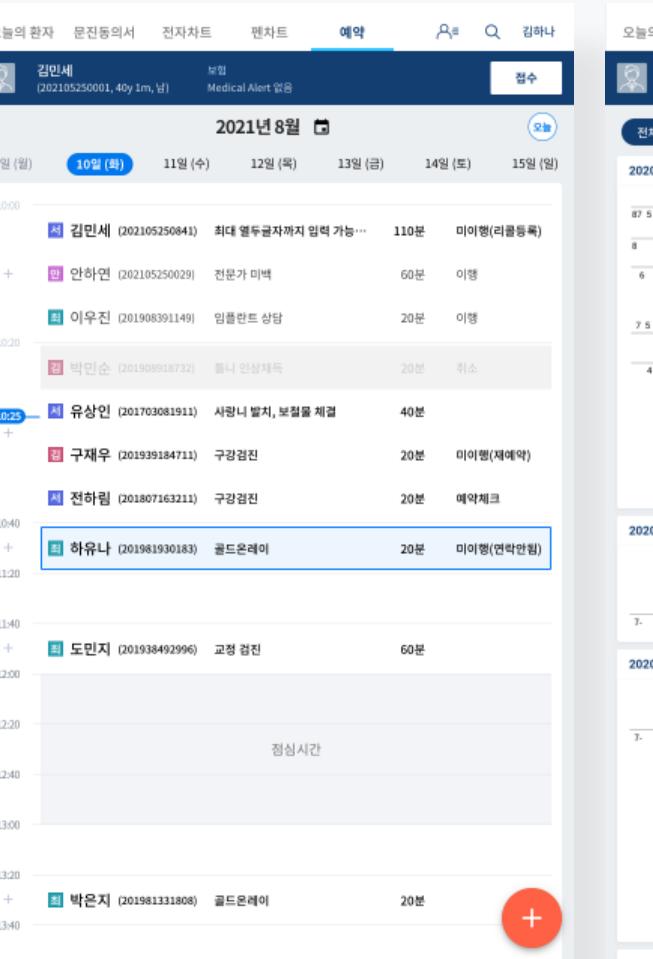
특히, 환자 차트 데이터 위에 팬으로 직접 드로잉 가능한 전자 차트 기능과, PC 솔루션과의 실시간 예약 동기화로 진료 흐름을 원활하게 지원합니다.

기술스택

- Kotlin
- Coroutines / Flow / MVVM Architecture (View - DataBinding - ViewModel - Model)
- Retrofit / OkHttp / ZeroMQ

역할 / 기여

- 앱의 기획, 설계, 개발, 배포 전 단계 참여 및 주도
- 드로잉 가능한 Canvas 기반 CustomView 모듈을 설계 및 구현하여 직관적 진료 기록 기능 제공
- UI/UX 전체 화면을 단독으로 설계 및 개발
- ViewPager 기반 예약 관리 기능 및 PC 솔루션과의 실시간 데이터 동기화 처리
- 앱 배포 전 단계 테스트, 오류 대응 및 사용자 피드백 반영 개선



오스템 임플란트 OneClick M 의사용 App 개발 사업

2021.04 ~ 2021.09

Android Native App

문제 해결 및 조치

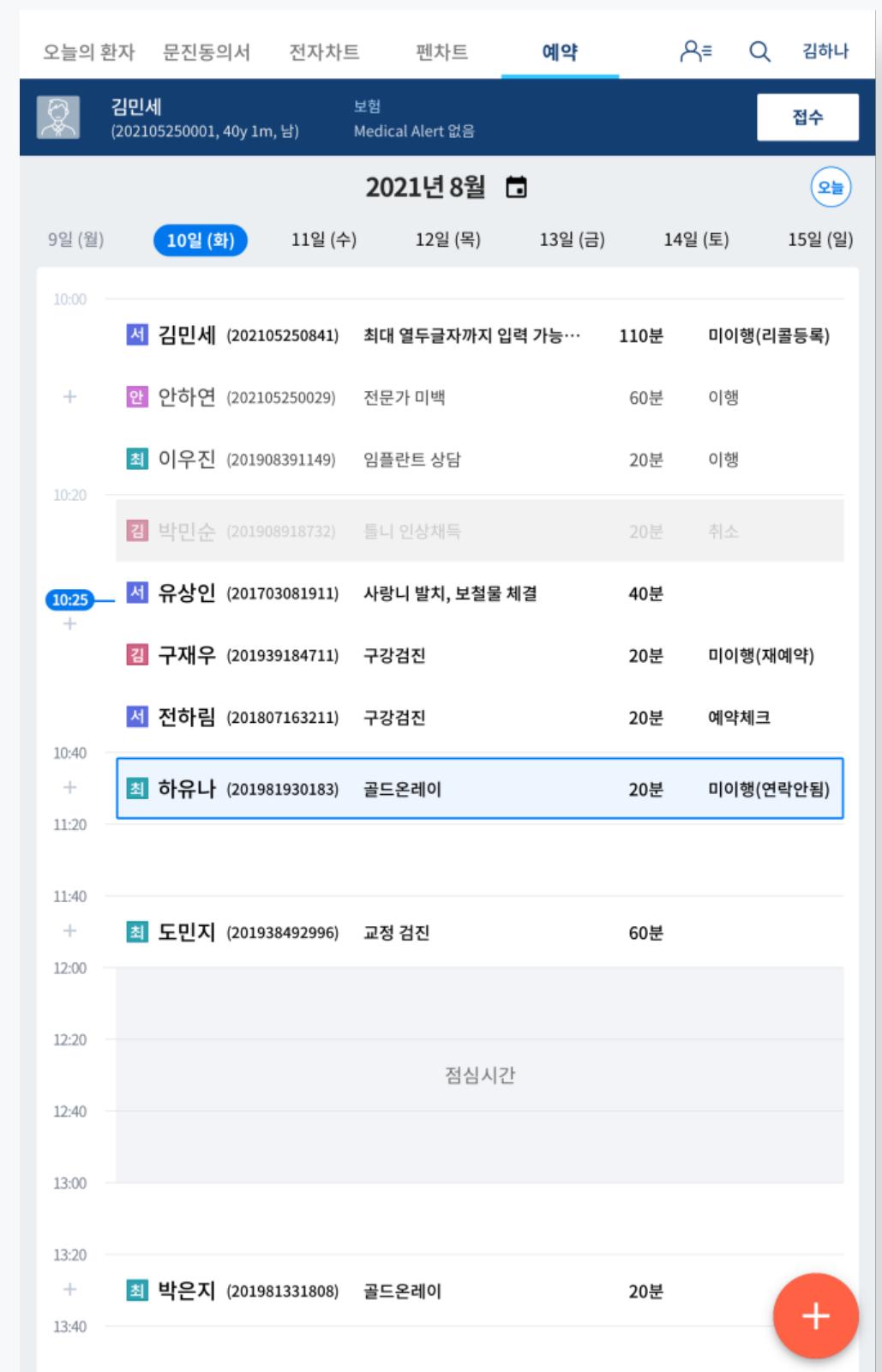
Case 1: ViewPager 내부 리사이클러뷰 실시간 동적 시간 표시

- Why**
- 리사이클러뷰 좌측의 시간 표시 뷰에 현재 시간을 반영하여, 실제 시간에 맞는 위치에 파란색 View(Indicator)를 표시해야 했음.
 - 그러나 시간대별 item height가 동적으로 달라져, 단순 계산으로는 View의 정확한 위치를 맞출 수 없었음.
 - 특히, 스크롤 시, 실시간 시간 표시 뷰가 올바른 위치를 찾지 못하고 위치 오류가 발생하는 문제가 있었음.

- How**
- 시간 단위별 item height를 계산하여 누적 높이를 연산하고, 현재 시간에 해당하는 위치를 정확히 산출하도록 구현.
 - 스크롤 이벤트를 감지하여 리사이클러뷰의 visible 영역과 현재 시간 위치를 동기화하는 로직을 추가.
 - ViewHolder 바인딩과 LayoutManager의 아이템 위치 연산 로직을 최적화하여 스크롤 시에도 올바른 위치로 갱신되도록 수정

- Outcome**
- 리사이클러뷰의 시간대별 동적 높이와 실시간 시간 위치 연산을 안정적으로 처리하여 사용자 경험 향상.
 - 스크롤 이벤트 대응으로 시간 Indicator 위치 오류 문제를 해결하고, 앱의 완성도 및 사용성을 강화.

동적 UI를 정밀하게 계산하고 동기화하여, 복잡한 UI 제어와 문제 해결 능력을 강화했습니다.



오스템 임플란트 OneClick M 의사용 App 개발 사업

2021.04 ~ 2021.09

Android Native App

문제 해결 및 조치

Case 2: 드로잉 CustomView의 품질에 대한 고찰

Why

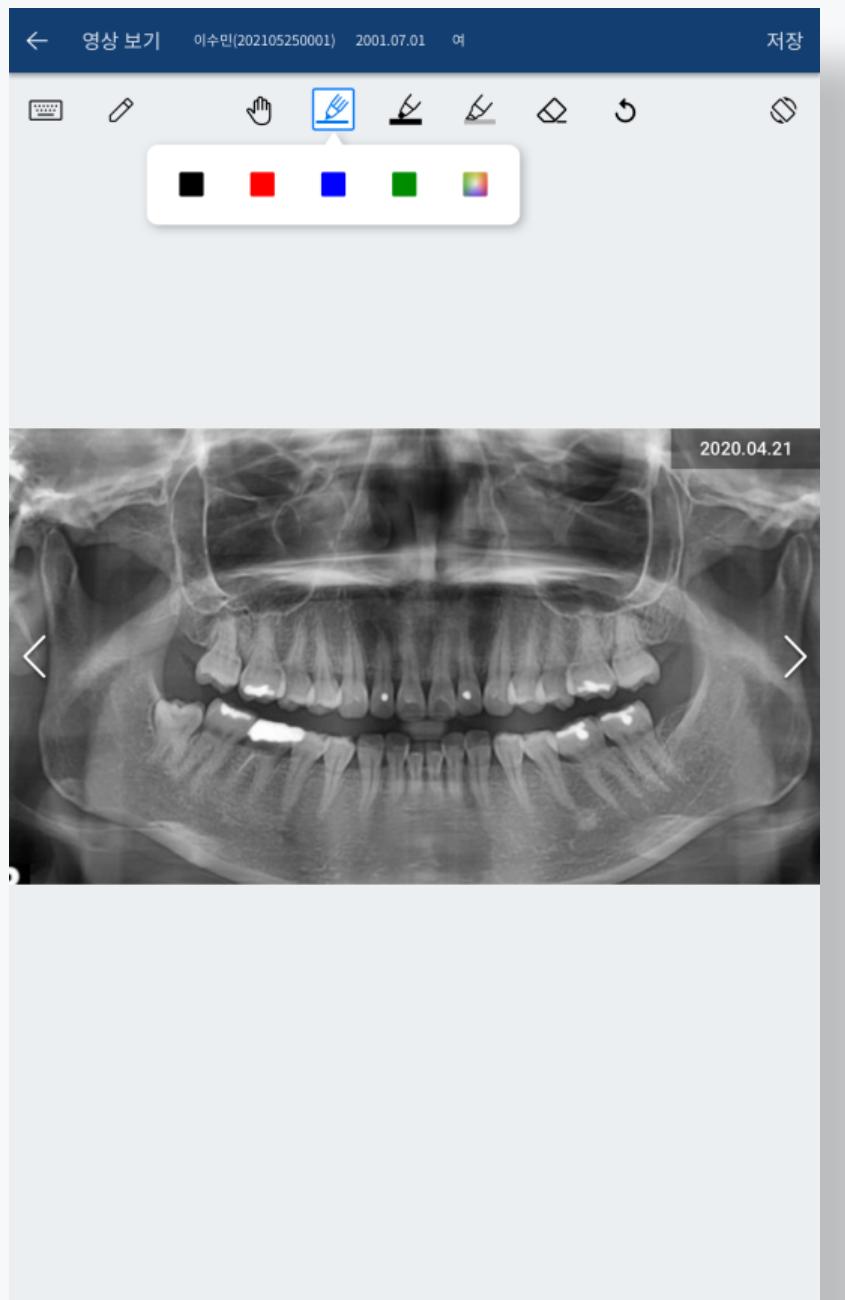
- 초기에는 Canvas 위에 drawPath를 이용해 선을 그렸으나, **선에 모서리가 생기고 곡선이 매끄럽지 않아** 사용자 경험이 저하됨.
- Path 대신 터치 지점에 point를 add하는 방식으로 곡선을 개선했지만, **터치 이벤트가 지나치게 많을 경우 메모리 부족 현상이 발생**.

How

- 터치 포인트를 실시간으로 **버퍼(buffer)**에 저장하고, 일정량의 포인트만 유지하여 메모리 사용량을 최적화.
- 곡선의 부드러움을 위해 **interpolation(보간법)** 연산 로직 제작 후 적용, 포인트 간 곡선을 매끄럽게 연결.

Outcome

- 사용자 피드백 반영을 통한 전자 치과 드로잉 기능 완성도 향상.
- 최종적으로 사용자 경험을 고려한 **매끄러운 드로잉과 안정적인 메모리 관리를 모두 만족하는 CustomView**를 구현.



드로잉 기능에 필요한 복잡한 데이터 구조를 설계 / 메모리 효율 및 곡선 품질 최적화를 경험 / 안정성과 사용자 경험을 강화했습니다.

현대홈쇼핑 쇼핑 라이브 개선 사업

2021.12 ~ 2022.03

Android Native App

개요

현대홈쇼핑 App은 고객들이 모바일로 홈쇼핑 방송을 시청하고 상품을 구매할 수 있는 스마트 쇼핑 앱으로, 누적 **1000만+다운로드**를 기록한 대표 홈쇼핑 앱입니다.

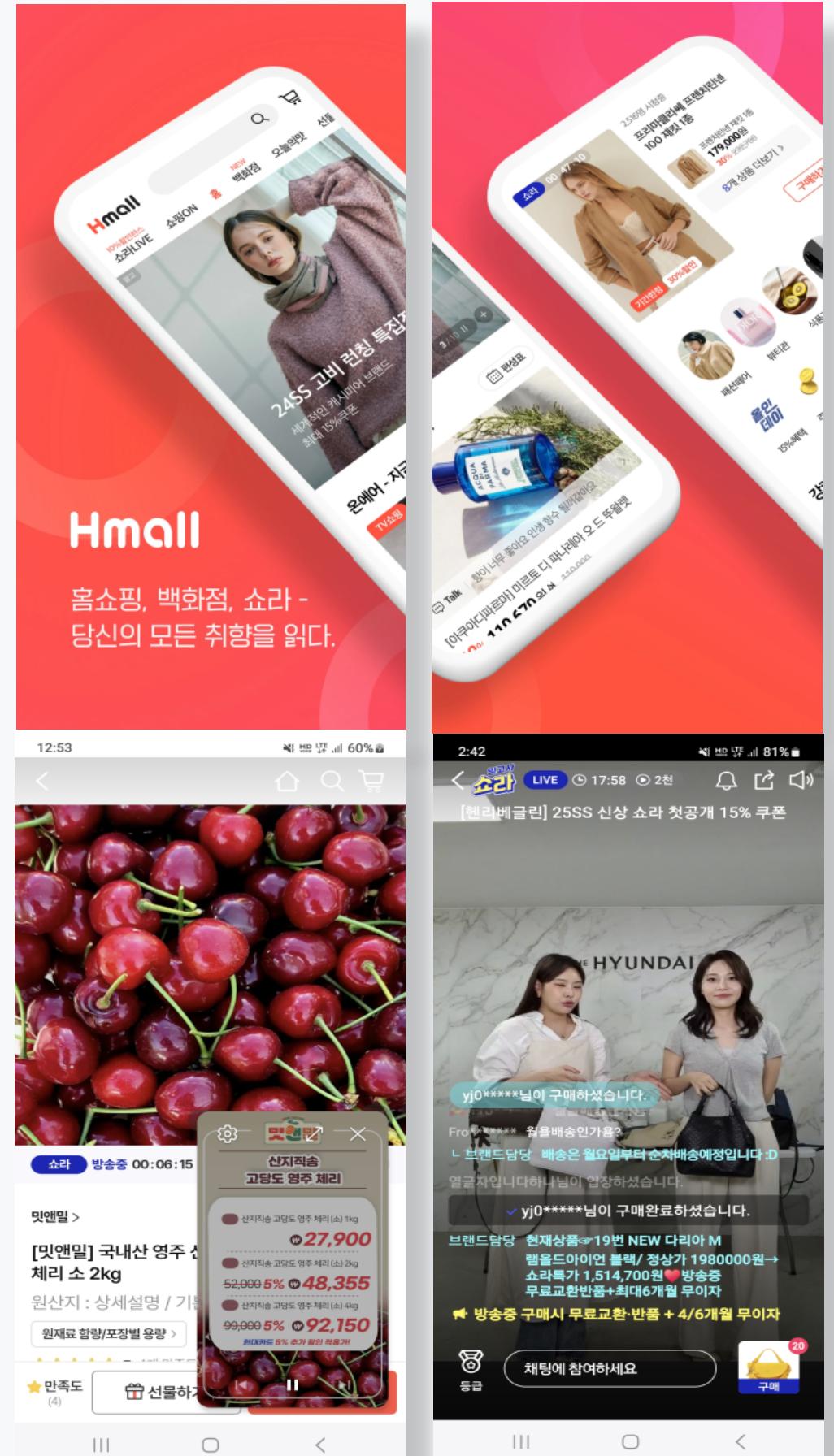
이번 프로젝트는 라이브 스트리밍 기반 쇼핑 컨텐츠(쇼핑 라이브) 개선 사업으로, **실시간 채팅 기능 강화와 방송 시청 이탈률 최소화 및 사용자 참여도 향상**을 목표로 진행되었습니다.

기술스택

- Java
- LiveData / MVVM Architecture (View - DataBinding - ViewModel - Model)
- Retrofit /OkHttp / WebSocket
- ExoPlayer / PIP (Picture-in-Picture)

역할/기여

- 동시간대 여러 스트리밍 채널 간 Swipe 이동을 위한 ViewPager UI 구현
- 방송 시청 중 구매 페이지로 이동 시 **PIP(Picture-in-Picture) 기능 연동**하여 구매 중에도 지속 시청 가능하도록 구현
- 전월 대비 **시청 이탈률 약 30% 감소**에 기여
- 방송 중 실시간 채팅 참여를 유도하기 위해 **실시간 좋아요 기능 추가** (애니메이션 인터렉션 효과)
- 업데이트 이전 대비 **채팅 참여도 약 20% 증가** 및 사용자 참여 경험 향상



현대홈쇼핑 쇼핑 라이브 개선 사업

2021.12 ~ 2022.03

Android Native App

문제 해결 및 조치

Case 1: ViewPager로 방송 채널 이동 시 데이터 꼬임 문제

Why

- ViewPager로 채널 이동 시 UI 초기화 이벤트와 Socket 연결 이벤트(데이터 수신)가 동시에 발생하여,
- UI와 데이터 상태 불일치로 인한 데이터 꼬임(중복 메시지, 위치 오류) 현상이 발생.

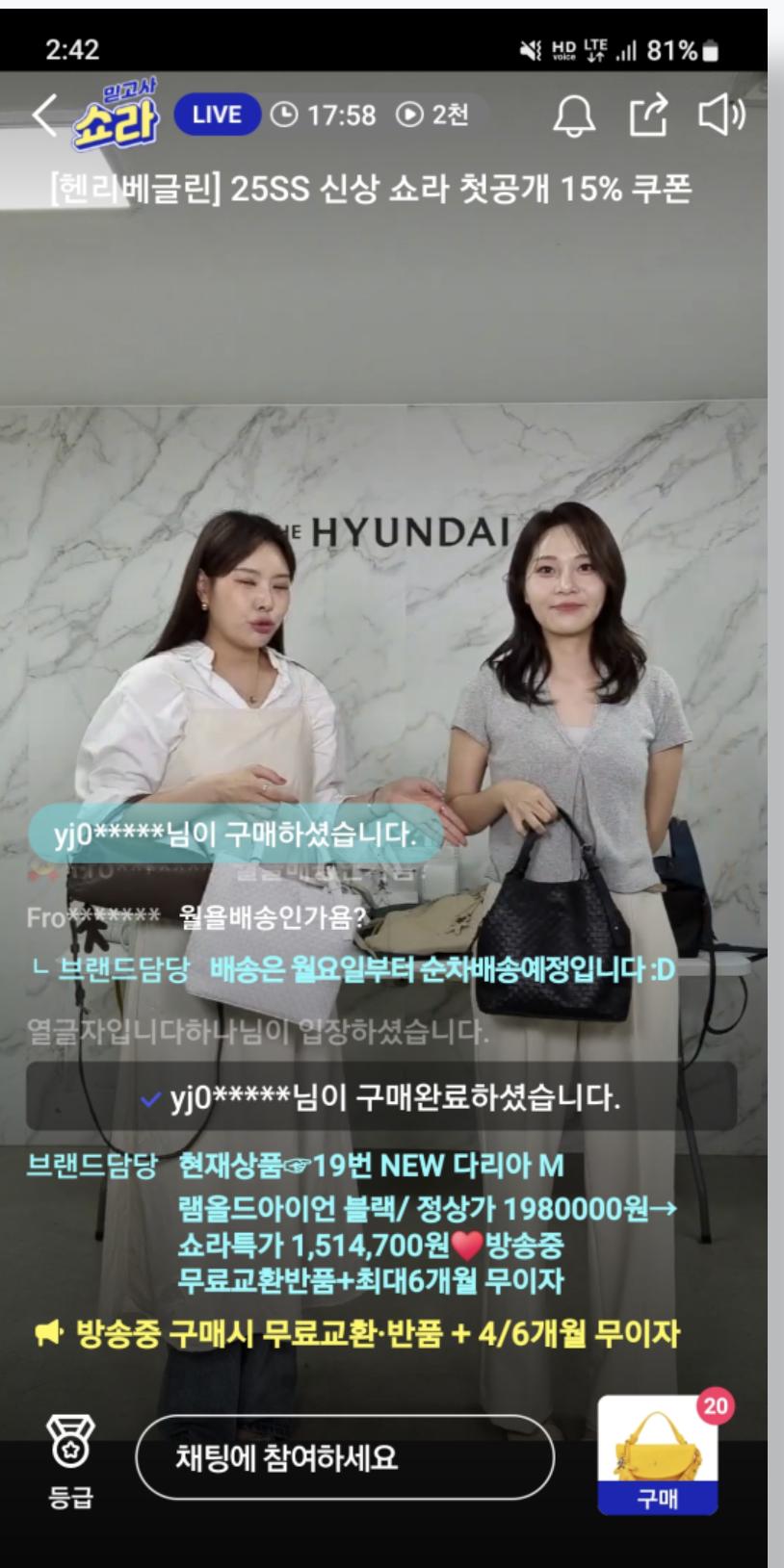
How

- 채널 전환 시 UI 초기화 완료 후 Socket 연결을 안전하게 처리하도록 순서를 명확하게 분리.
- ViewTreeObserver.OnGlobalLayoutListener**를 활용하여 UI 레이아웃이 완전히 완료된 시점을 감지하고, 처리 후 리스너를 제거하여 불필요한 반복 호출을 방지.
- Lifecycle-aware 처리: UI/Fragment Lifecycle 상태와 연계하여 Socket 연결 및 종료 시점을 관리, 리소스 누수 방지

Outcome

- UI 초기화 완료 시점과 Socket 연결 처리의 타이밍 문제를 해결하여 데이터 꼬임 현상 방지.
- 채널 전환 및 실시간 데이터 처리의 안정성 강화, 사용자 경험 향상 및 앱 완성도 강화.

UI 상태 관리와 비동기 데이터 동기화의 중요성을 배우며, 안정적인 실시간 데이터 처리 방법을 익혔습니다.



현대홈쇼핑 쇼핑 라이브 개선 사업

2021.12 ~ 2022.03

Android Native App

문제 해결 및 조치

Case 2: ExoPlayer 이벤트 & PIP 연동 시 비동기 이벤트 처리 이슈

Why

- PIP(Picture-in-Picture) 모드 연동 과정에서 ExoPlayer의 재생 상태와 시스템 PIP 모드 전환 이벤트가 비동기적으로 발생, PIP 진입 시점에서 재생 상태가 불안정해 일부 기기에서 재생 중단, 진입 실패 현상이 발생.
- 특히, PIP 전환 중 사용자의 네트워크 상태 변화나 빠른 UI 이벤트(채널 전환, 화면 전환, 다음 방송 호출) 발생 시 ExoPlayer 상태와 PIP 이벤트 순서 불일치로 문제가 심화됨.

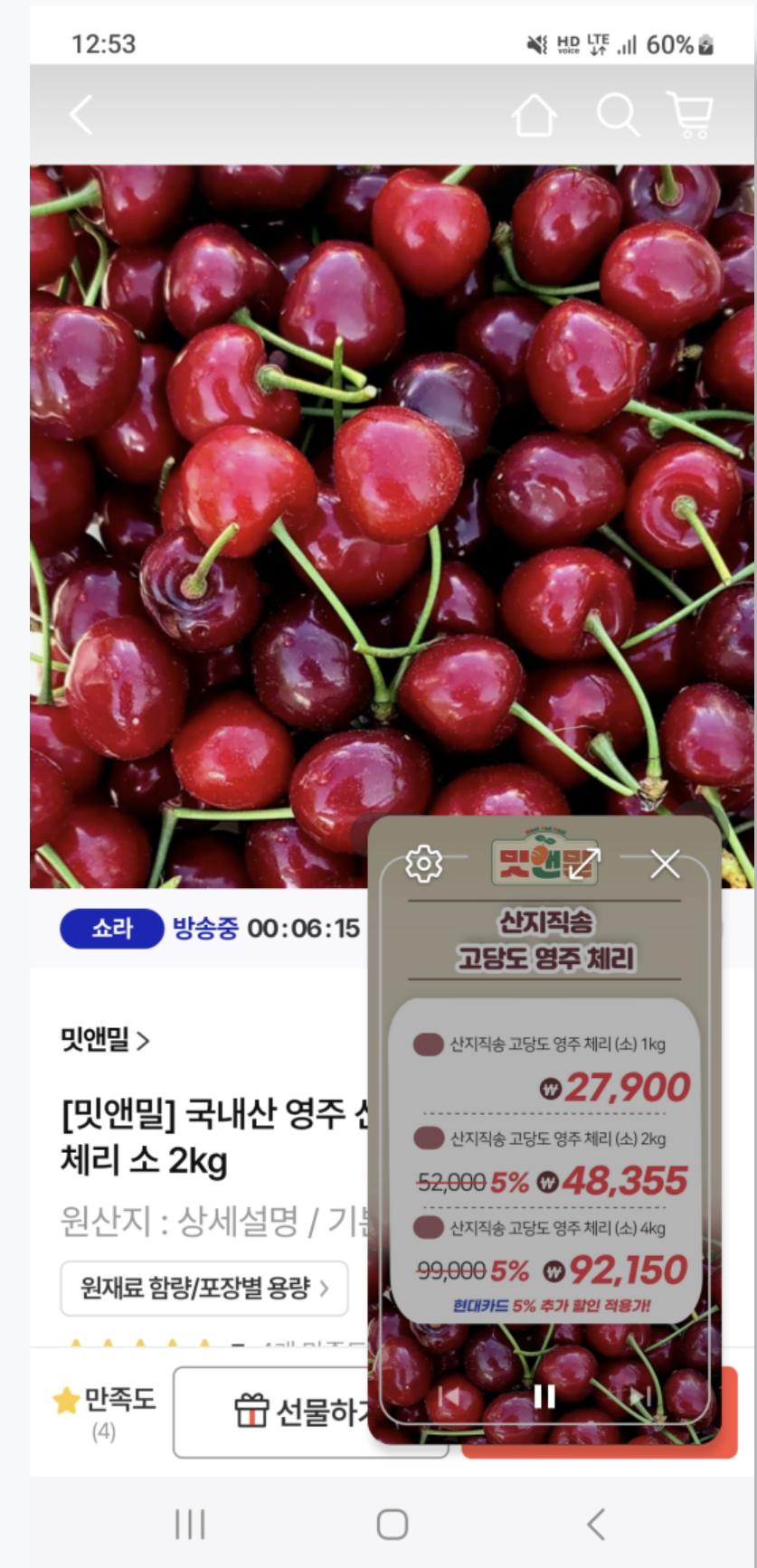
How

- PIP 진입 시 `isExoPlayerReady`와 `isPipRequested` 플래그를 관리하여 재생 상태와 시스템 PIP 이벤트를 동기화.
- PIP 전환 요청이 여러 번 들어올 경우, **요청 큐(Queue)**를 도입하여 순차적으로 처리하도록 개선. PIP 진입 및 해제 이벤트를 큐에 추가하고, `ExoPlayer 상태` 및 시스템 콜백(`onPictureInPictureModeChanged`)을 모니터링하며 이벤트 완료 후 다음 큐 작업을 진행하도록 처리.
- LifecycleObserver**를 활용하여 PIP 전환과 ExoPlayer 상태를 Fragment/Activity Lifecycle과 동기화. 재생 상태 변화를 감지하여 PIP 모드 진입 실패 시 재시도 로직 추가.

Outcome

- ExoPlayer 상태와 시스템 PIP 전환 이벤트의 비동기 처리 문제를 해결, 안정적인 PIP 진입과 재생 유지 확보.
- PIP 연동 안정성 개선으로 재생 중단 이슈 감소(약 90% 감소 추정) 및 사용자 경험 강화.
- 다양한 기기 및 네트워크 환경에서도 안정적인 PIP 전환을 유지하여 앱 완성도 강화.

비동기 이벤트 동기화 및 큐 처리 로직 구현으로 안정적인 PIP 전환과 미디어 재생 유지 역량을 강화했습니다.



LS 엠트론, 조사료 플랫폼 카우잇츠 App 개발 사업

2023.05 ~ 2023.09

Android Native App

I 개요

카우잇츠는 농업 현장의 조사료 거래를 효율화하고, **트랙터 작업 현황을 실시간으로 모니터링**할 수 있는 스마트 솔루션입니다. 이번 프로젝트는 팀과 협업하여 기획, 디자인, 개발, 배포까지 전 과정에 참여하여 진행하였습니다.

I 기술스택

- Kotlin / Coroutine / Flow / Hilt
- MVVM Architecture (View - DataBinding - ViewModel - Model)
- Retrofit / OkHttp / Node.js / Socket.IO / Firebase Firestore
- GPS / Location Services / Kakao Map Api

I 역할 / 기여

- Node.js, Socket.IO를 활용한 실시간 양방향 통신 서버 구축 및 클라이언트 연동
- 트랙터 위치를 실시간으로 지도에 표시하여 여러 작업자들의 현재 위치를 서로 공유할 수 있도록 구현
- 사용자가 등록한 지번 주소의 지적도를 지도 UI 위에 표시하여 시각적 이해도를 높임
- Google Play Store 배포 관리



LS 엠트론, 조사료 플랫폼 카우잇츠 App 개발 사업

2023.05 ~ 2023.09

Android Native App

문제 해결 및 조치

Case 1 : 실시간 데이터 전송 이벤트 중 데이터 무결성 위반 이슈

Why

- 실시간 위치 공유 중 트랙터가 작업 완료 처리를 위해 REST API 호출로 RDBMS 업데이트 시도
- 성공 응답을 받고 UI에서는 완료 처리 표시가 되었음,
BUT, DB에는 작업 완료 상태가 반영되지 않는 불일치 현상이 발생.

How

- 문제 원인을 분석한 결과, Socket으로 받은 위치 데이터는 클라이언트 측 UI 업데이트에 즉시 반영되지만, RDBMS 업데이트는 REST API 호출 및 서버 처리, 트랜잭션 완료까지 비동기 처리되는 구조.
- 이 과정에서:
네트워크 오류, 요청 실패, 서버 타임아웃 등으로 API 호출 실패 가능성 존재
API 호출 성공 후에도 DB 내부 충돌로 업데이트가 지연되거나 룰백(트랜잭션 처리 미흡) 가능성 존재
클라이언트에서 위치 데이터 기반 UI 업데이트 로직과 서버의 작업 완료 처리 로직 간 불일치 가능성 존재
- 개선을 위해:
API 요청 실패 및 재시도 로직 강화, 실패 시 사용자 알림 또는 로컬 상태 룰백 처리
서버-클라이언트 간 작업 완료 조건 동기화 로직 점검 및 개선
RDBMS 업데이트 트랜잭션 처리 및 커밋 확인 강화, 처리 지연 시 로깅 및 재처리 메커니즘 추가

Outcome

- 실시간 위치 업데이트와 RDBMS 상태 동기화 간 비동기 처리 문제의 원인 파악 및 개선 경험 확보
- Deadlock 및 Lock Timeout 등 RDBMS 내부 충돌을 해결하기 위해 트랜잭션 처리 안정성 및 재시도 로직의 필요성을 깊이 이해

데이터 무결성의 중요성과 신뢰성 확보의 필요성을 깊이 깨달았습니다.



IDAEYE 영유아 수면 자세 감지 App 개발 및 유지보수

2025.01 ~ 2025.05

Android Native App

I 개요

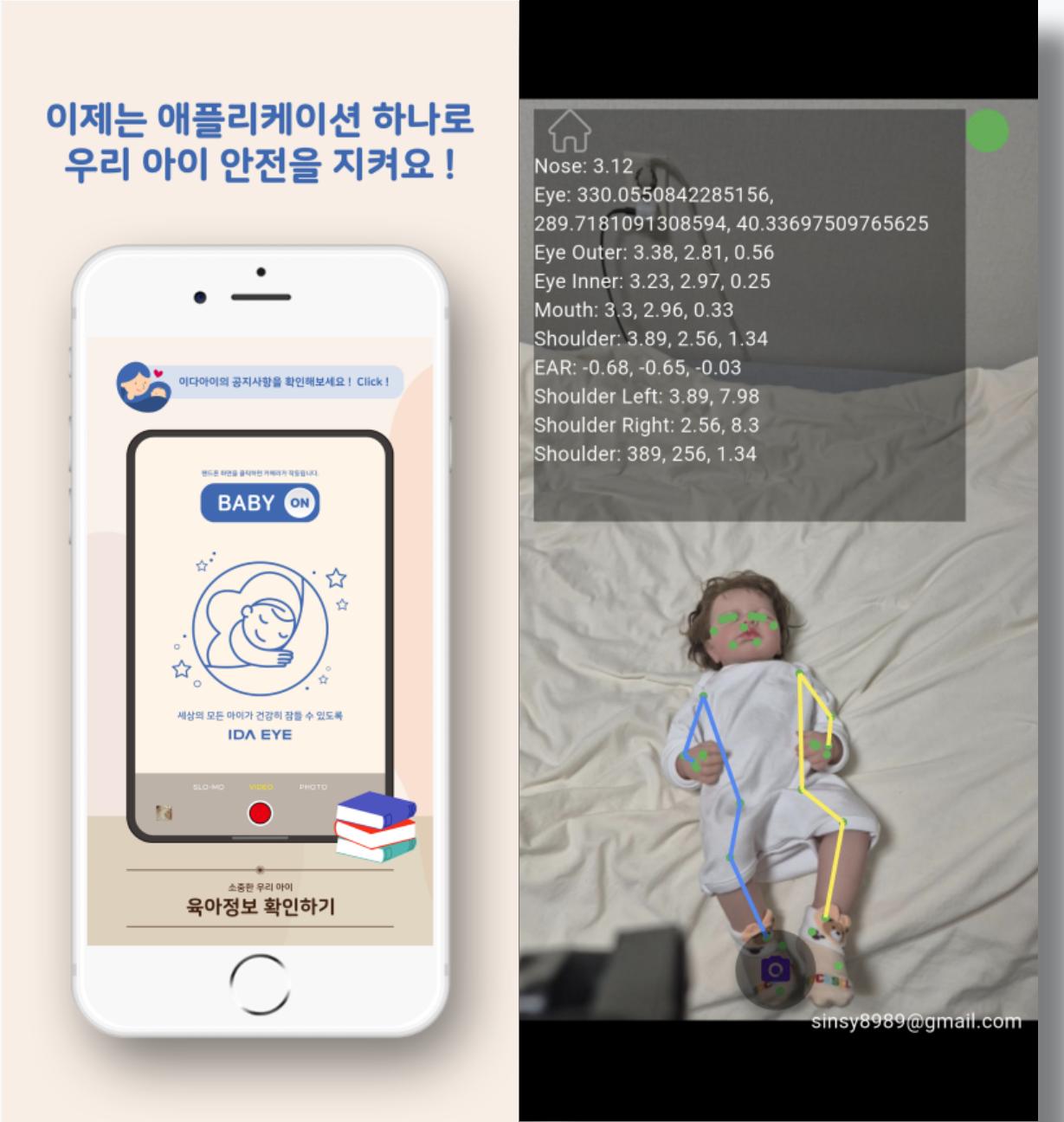
IDAEYE는 아기의 수면 자세를 실시간으로 모니터링하고, 뒤집힘 등의 위험한 자세 발생 시 즉시 알람이 울리는 앱입니다.
기획, 디자인, 개발, 배포까지 전 과정에 참여하여 진행하였습니다.

I 기술스택

- Kotlin / Coroutine / Flow / Hilt
- MVVM Architecture (View - DataBinding - ViewModel - Model)
- Retrofit / OkHttp / Node.js / Express / React
- Google ML Kit

I 역할 / 기여

- Google ML Kit을 활용한 신체(Body) 객체 인식 및 위험 자세 감지 기능 구현
- AI 모델 추론 데이터(자세, 움직임 등)와 UI 상태를 연동하여 실시간으로 사용자 알림 제공
- Node.js(Express)와 React 기반의 관리자 페이지 구현
- Jetpack Compose와 Kotlin Flow 기반의 단방향 상태 흐름(Unidirectional Data Flow)을 적용
- 실제 영유아 대상 반복 테스트 통해 조건 최적화 및 예외 케이스 보완



IDAEYE 영유아 수면 자세 감지 App 개발 및 유지보수

2025.01 ~ 2025.05

Android Native App

I 문제 해결 및 조치

Case 1 : 2D 카메라 환경 얼굴 기울기 데이터 감지 한계 극복

Why

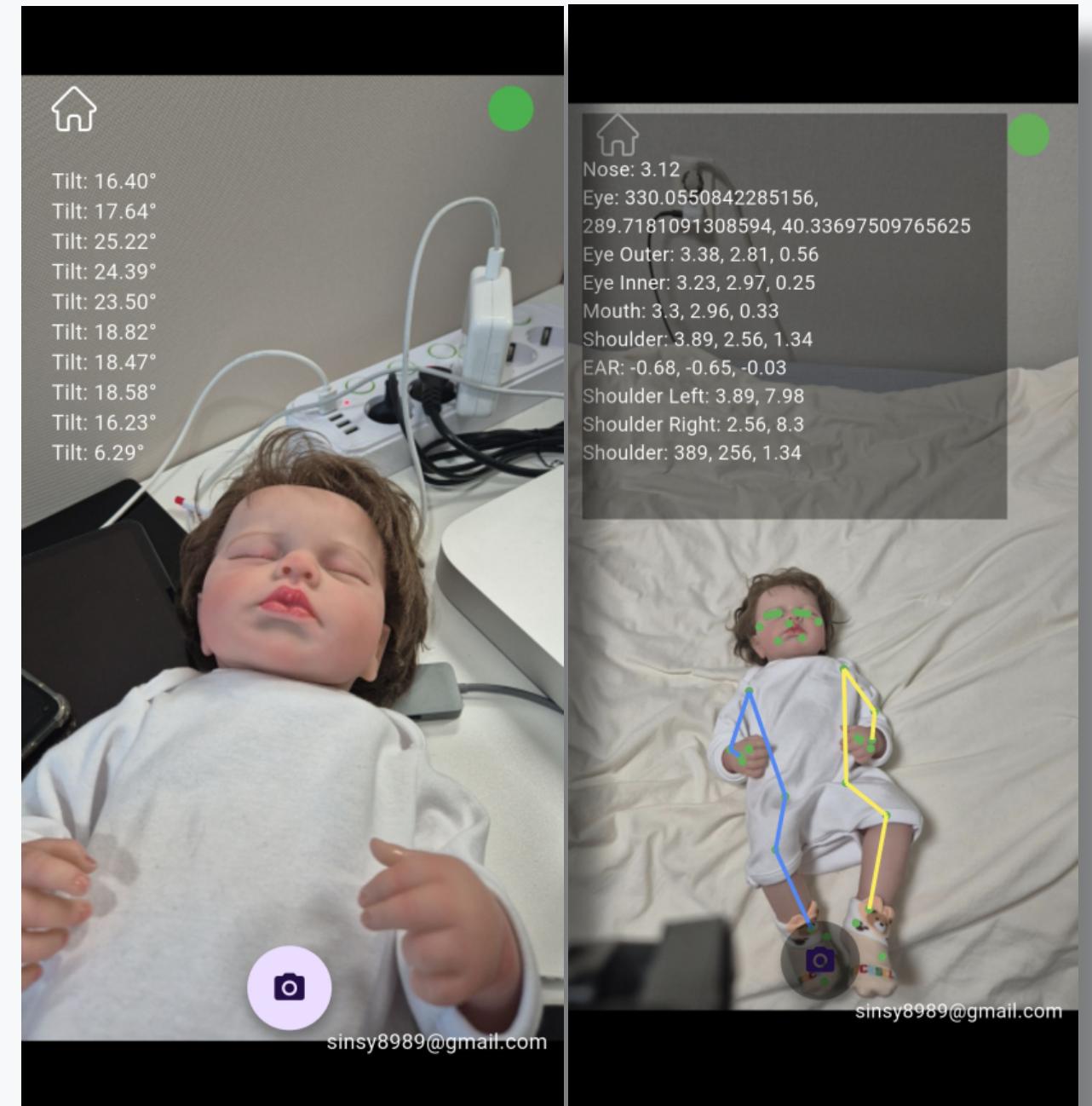
- 얼굴이 바닥을 향한 영유아의 위험 수면 자세를 감지해야 했으나,
2D 카메라 환경에서 눈·코·입 미검출 시 Face LOST 반환 → 얼굴 각도 기반 감지 한계 발생

How

- Body Detector를 연동하고 어깨, 손, 발의 위치 데이터를 함께 분석하여
얼굴 인식 실패 시에도 자세를 유추할 수 있도록 보완 알고리즘 설계 및 테스트 진행

Outcome

- 반복 테스트를 통해 신뢰도 높은 감지 정확도를 확보하였으며,
내부 테스트 사용자들의 승인을 받아 정식 기능으로 적용됨



기술의 한계를 보완하는 설계와 반복 테스트의 중요성을 체감하였습니다.

개인 프로젝트 | 오독오독, 독서 기록 App

2025.01 ~ 현재

Android Native App

Github URL: <https://github.com/stevey-sy/odok-compose>

개요

오독오독은 사용자가 읽은 책의 정보를 등록하고 독서 일지를 기록할 수 있도록 설계된 모바일 애플리케이션입니다.

이 프로젝트는 안드로이드 최신 아키텍처와 모던한 개발 요소(Kotlin & Jetpack Compose)를 직접 학습하고 연구해보기 위한 개인 프로젝트로, Kotlin & XML과 Kotlin & Jetpack Compose 버전을 각각 개발하며 두 방식의 효율성과 유지보수성, 개발 편의성을 비교하고 있습니다.

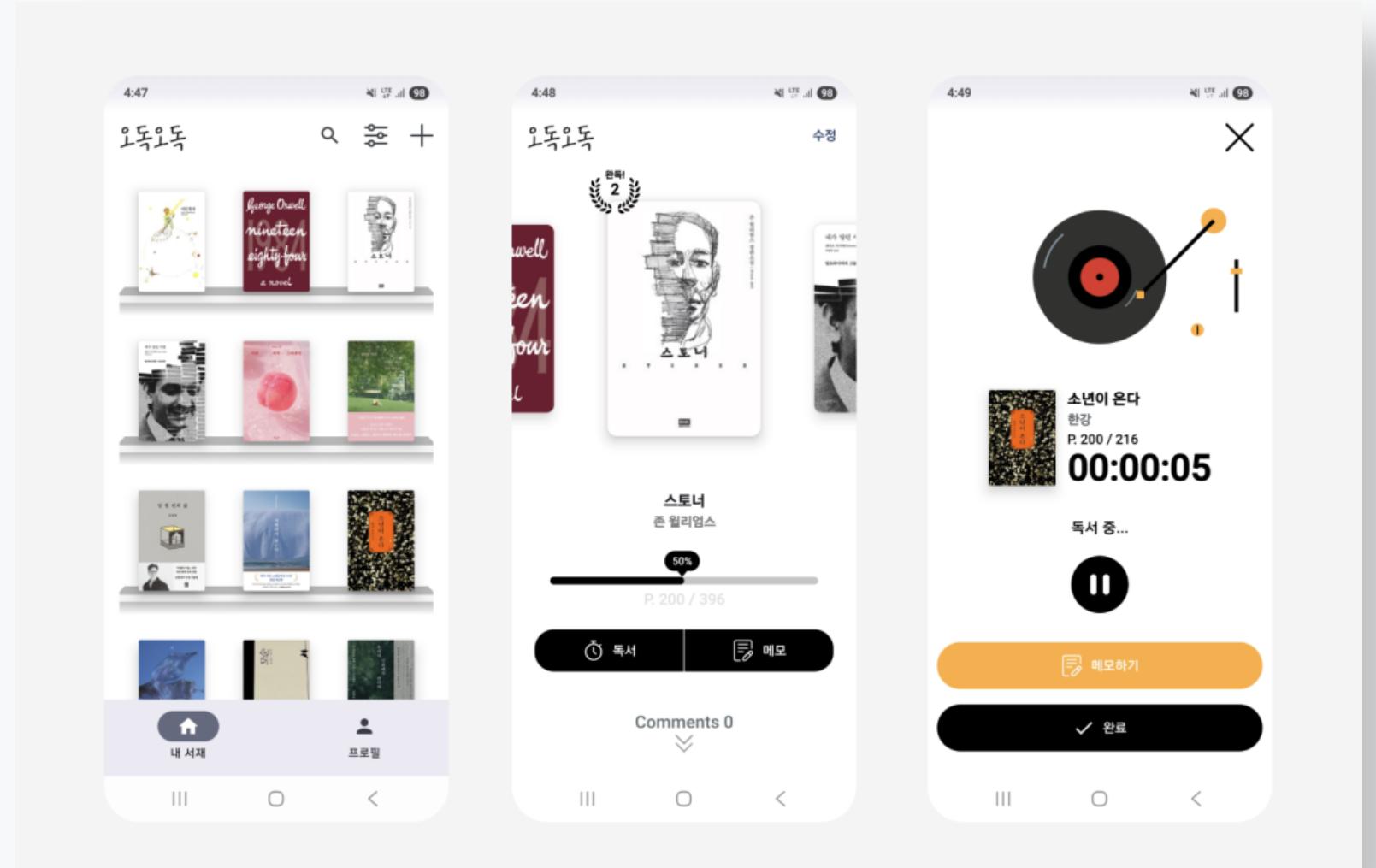
퇴근 후 매일 조금씩이라도 개발하며 학습을 이어가며, 최신 기술 트렌드에 맞춘 안드로이드 앱 개발 역량을 꾸준히 성장시키고 있습니다.

기술스택

- Kotlin / Jetpack Compose / Coroutine / Flow / Hilt
- Multi-Module / MVVM (View - DataBinding - ViewModel - Model) / Single Activity Architecture
- Room / Retrofit / OkHttp / Google ML Kit / Paging 3 / Shared Elements Transition
- Aladin Open Api

역할 / 기여

- 사용자가 책장에서 책을 고르는 듯한 UI 연출을 위해 **Shared Elements Transition** 효과를 적용하여 자연스럽고 몰입감 있는 화면 전환 구현
- **core 모듈에 database, network, designsystem, domain을 분리 설계하고, Repository Pattern을 적용**하여 코드 재사용성과 유지보수성 강화
- **Single Activity Architecture**를 적용하여 화면 전환과 Navigation을 단일 Activity에서 관리, 앱의 구조를 단순화하고 안정성을 강화



개인 프로젝트 | 오독오독, 독서 기록 App

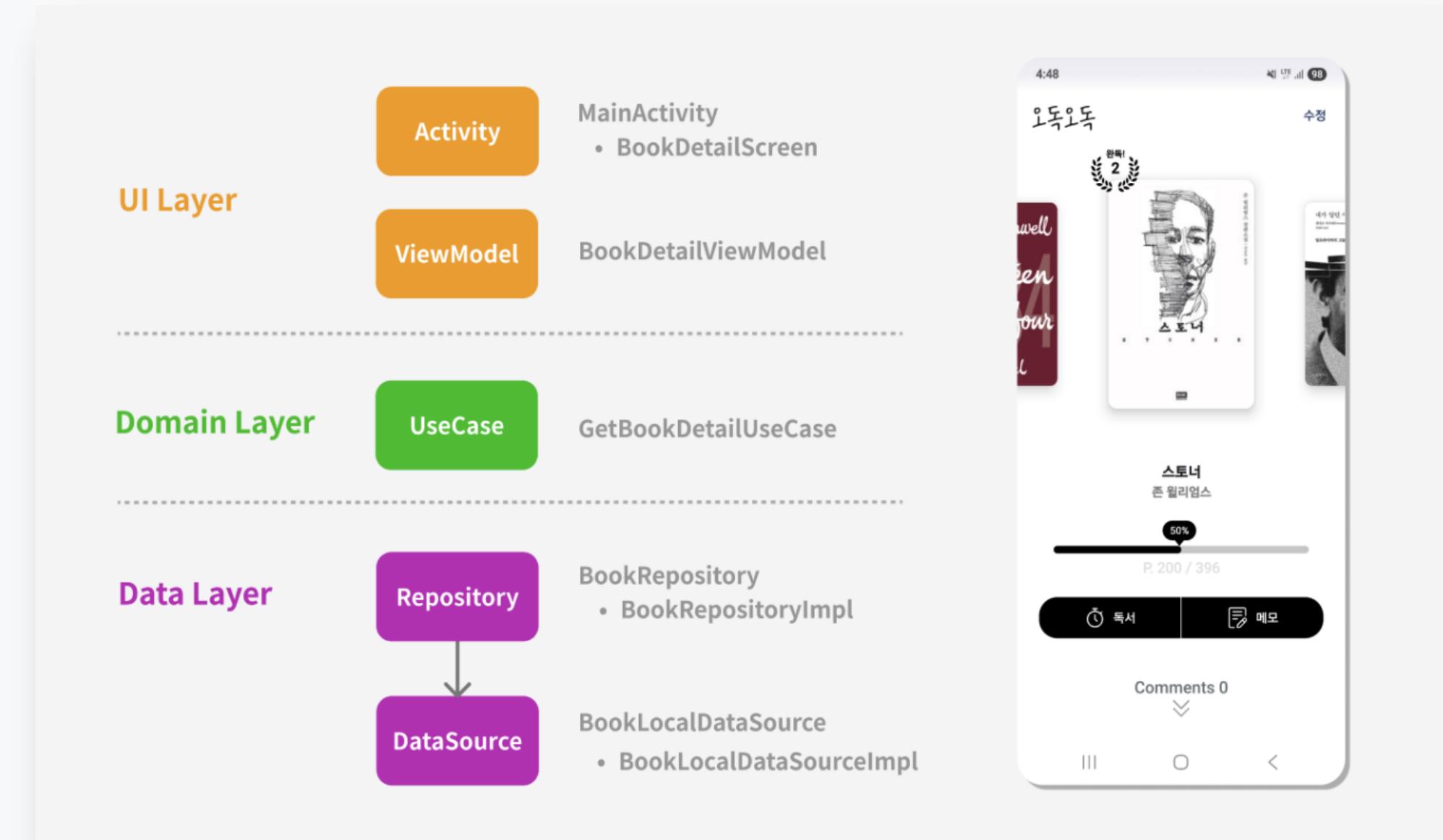
2025.01 ~ 현재

Android Native App

Github URL: <https://github.com/stevey-sy/odok-compose>

Architecture

- 오독오독은 UI, Domain, Data 계층으로 구성된 Clean Architecture를 따릅니다.
- **UI Layer:** 사용자 인터페이스와 ViewModel을 포함하며, 화면 상태와 사용자 이벤트를 처리합니다.
 - BookDetailViewModel이 화면 상태를 관리하며 사용자 인터랙션을 처리합니다.
- **Domain Layer:** 앱의 비즈니스 로직을 담당하는 계층으로, UseCase를 통해 로직을 수행합니다.
 - GetBookDetailUseCase가 비즈니스 로직을 담당하고, 데이터를 요청하기 위해 Repository에 의존합니다.
- **Data Layer:** 실제 데이터의 저장소로, Repository와 DataSource를 통해 로컬 또는 네트워크 데이터를 제공합니다.
 - Data Layer에서는 BookRepository와 BookLocalDataSource가 데이터 요청을 처리하며, 구체적인 구현(BookRepositoryImpl, BookLocalDataSourceImpl)은 interface를 기반으로 분리되어 있습니다.
- 모든 데이터 흐름은 UI → Domain → Data 방향으로 진행되며, "단방향 의존성(unidirectional dependency)"을 지향합니다.



* 위 이미지는 오독오독 앱의 **책 상세 화면(BookDetailScreen)**이 렌더링되기까지의 구조를 Clean Architecture 기반의 3계층 아키텍처로 시각화한 것입니다.

개인 프로젝트 | 오독오독, 독서 기록 App

2025.01 ~ 현재

Android Native App

Github URL: <https://github.com/stevey-sy/odok-compose>

App Modularization

- 오독오독 앱은 유지보수성과 확장성을 고려해 멀티 모듈 아키텍처로 구성하였습니다.
- AppModule을 중심으로 기능별로 분리된 Feature Module과, 공통 로직을 담당하는 Core Module로 나누어 각 책임을 명확히 분리했습니다.
- **AppModule:** 앱의 진입점이자 모든 모듈을 통합하는 역할
- **Feature Module:** mylibrary, search, timer, memo, profile 등 주요 화면 단위로 분리되어 독립적인 개발 및 테스트가 가능
- **Core Module:**
 - design, ui: 공통 UI 요소 및 디자인 시스템을 관리
 - domain, repository, model: 비즈니스 로직과 데이터 모델 정의
 - datasource, database, network: 외부 데이터 접근 계층으로, 내부 구현을 추상화하여 테스트 용이성 확보

