

원하는 상품을 한 곳에서

BEST WISH

Project

BEST WISH

Period

25.05.29 - 25.07.07

Member

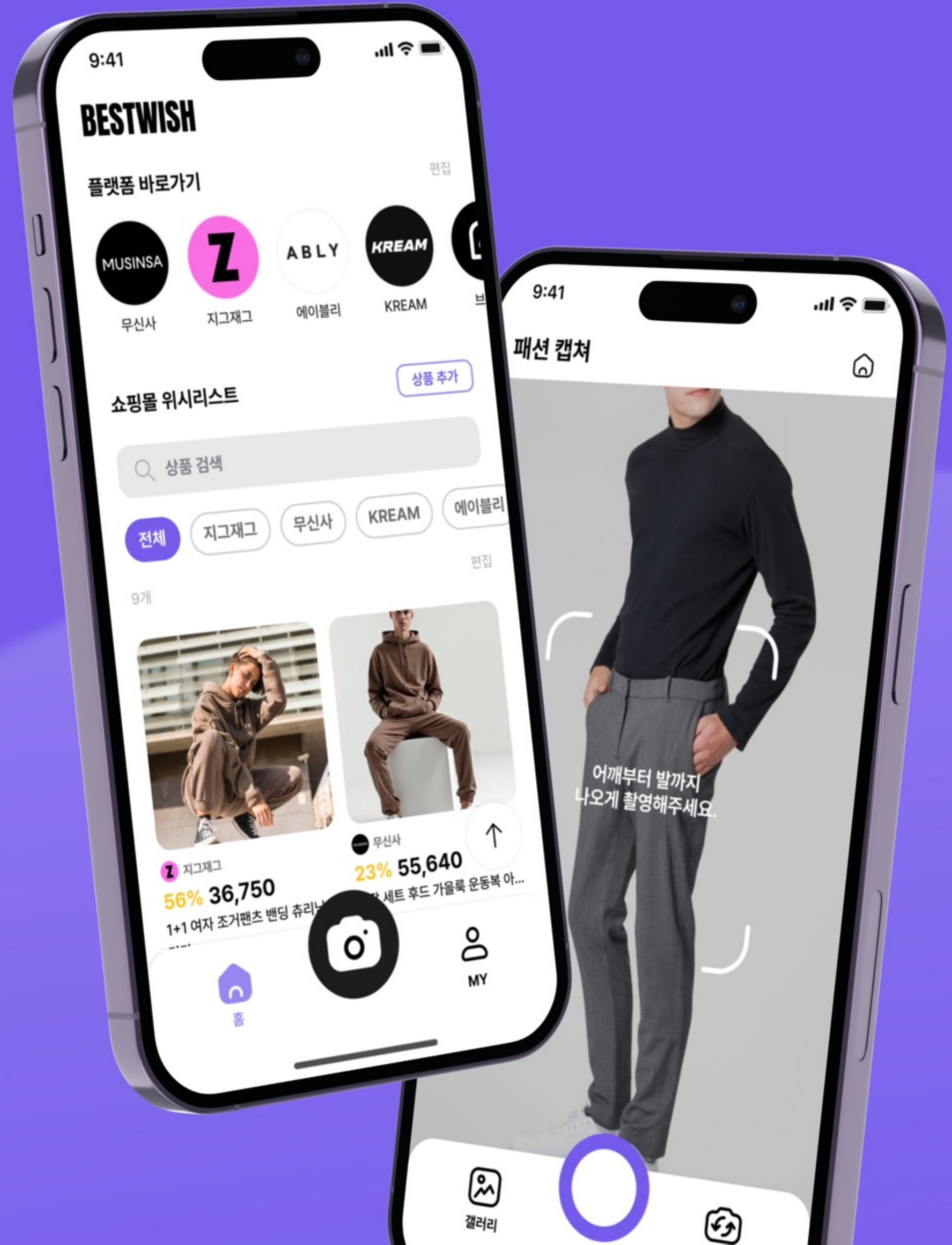
개발자 백래훈

유영웅

이세준

이수현

디자이너 박지현



시연 영상



앱 스토어



GitHub

1. 프로젝트 개요

- 프로젝트명: 베스트위시 (Best Wish)
- 프로젝트 기간: 2025.05.29 ~ 2025.07.07 (6주)
- 팀 구성: 총 5명 (iOS 개발 4명, 디자인 1명)
 - 역할: 팀 리더, iOS 앱 설계 및 개발 (홈 화면, Share Extension 구현), 프로젝트 일정 관리 및 기획 · 디자인 협업
- 프로젝트 목적:
 - 외부 쇼핑몰 상품을 링크 기반으로 저장하고 플랫폼별로 정리하여 사용자들이 상품을 효과적으로 수집, 탐색, 관리할 수 있도록 돋는 쇼핑 특화 위시리스트 관리 앱 개발

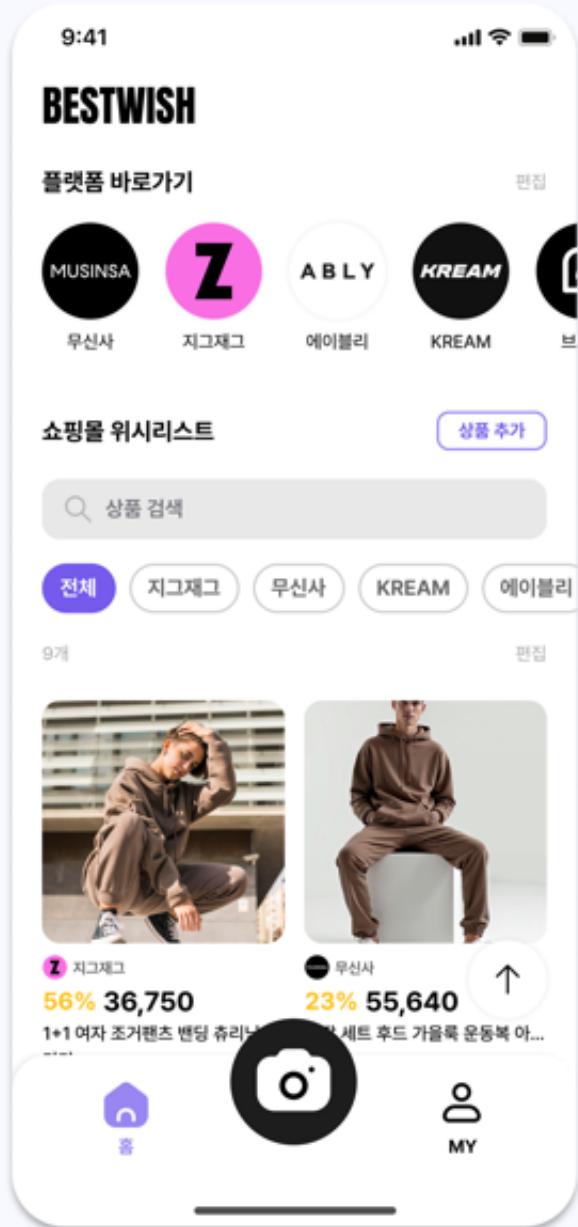
2. 서비스 소개

- 서비스 주요 기능: 소셜 로그인(Kakao, Apple) / 플랫폼 바로가기 / 위시리스트 관리 / 패션 캡쳐 / Share Extension
- 타겟 유저:
 - 다양한 쇼핑 플랫폼을 이용하다 보니, 어떤 플랫폼에 어떤 위시 아이템을 저장해뒀는지 파악하기 어렵고 귀찮았던 경험을 겪고 있던 유저들을 타겟
 - 더 나아가 직접 찍은 의류 사진이나 저장해둔 패션 이미지를 분석해, 원하는 스타일의 상품을 더 쉽게 찾아볼 수 있도록 도움

3. UI 스크린샷

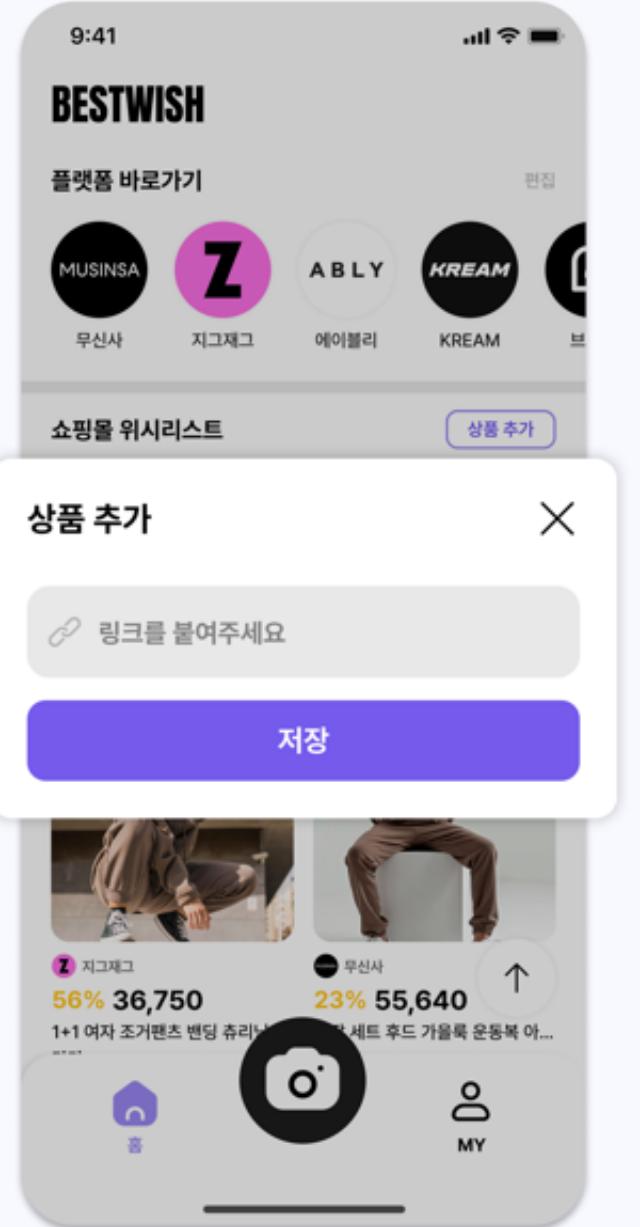
홈
모든 위치, 한 곳에

찜한 상품을 BEST WISH에서 한눈에!
다양한 쇼핑몰의 위치리스트를 한 앱에 모아보세요.



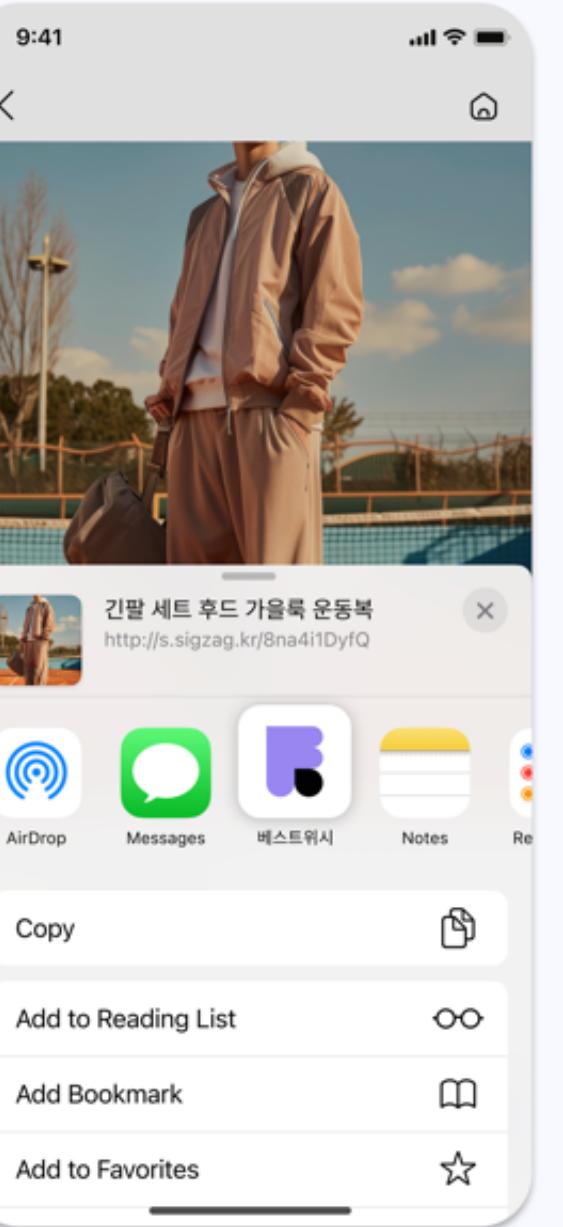
링크 저장
간편한 위치 추가

원하는 상품의 링크를 붙여넣기 한 번으로
위치리스트에 추가할 수 있어요.



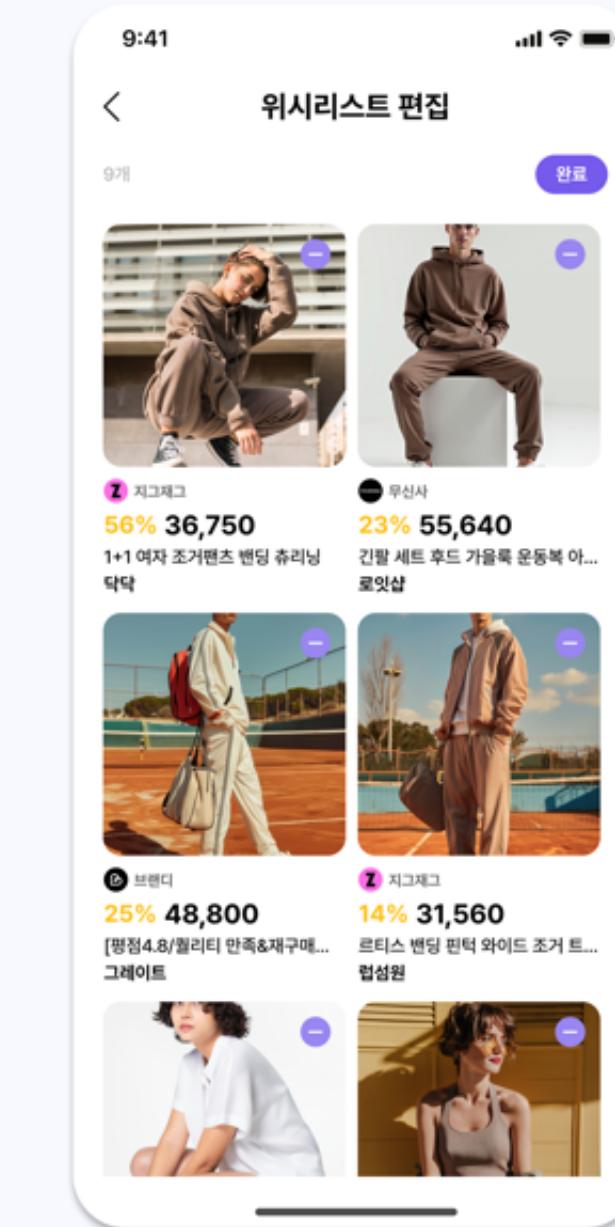
공유 간편 저장
공유만 하면 저장 끝!

링크 복사 없이, 공유만으로
간편 저장이 가능해요!



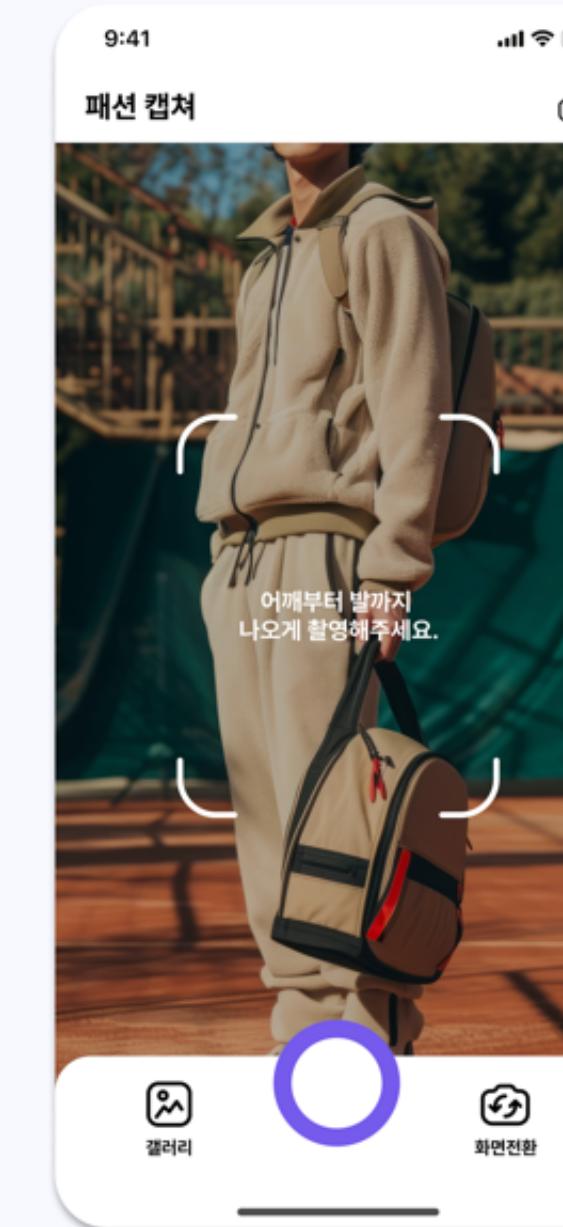
상품 편집
원하는 건 남기고, 정리는 쉽게!

위치리스트도 미니멀하게.
터치 한 번으로 상품 삭제, 편집까지 손쉽게 끝!



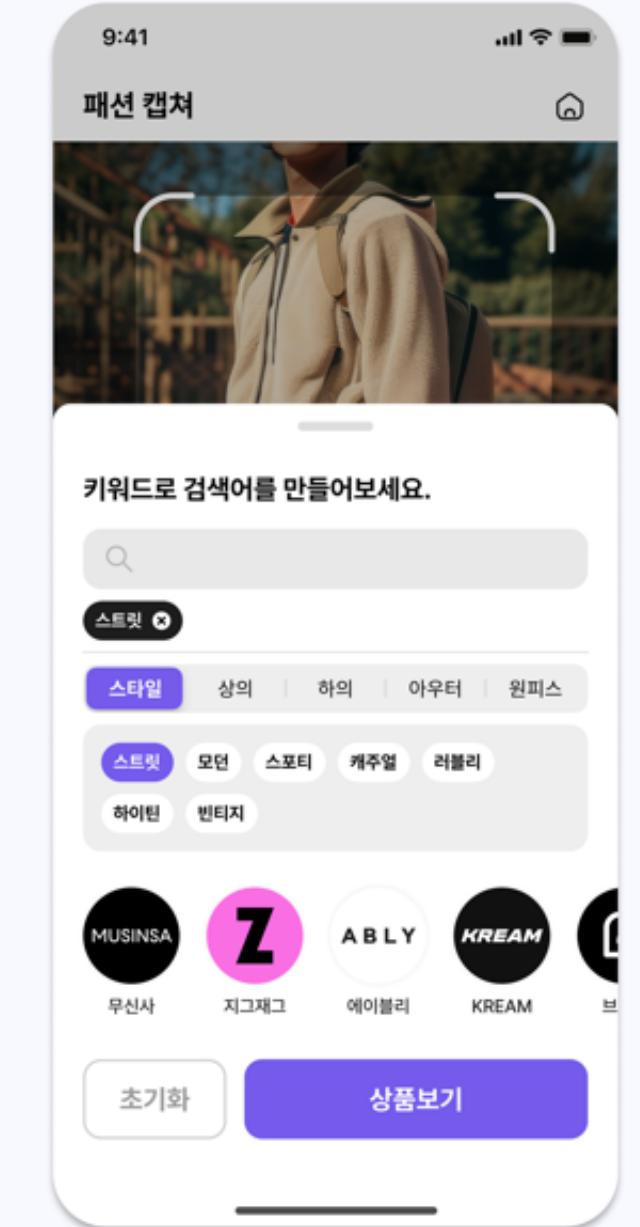
패션 캡처
실시간 패션 캡처

지금 입은 스타일, 바로 저장!
전신을 찍어 스타일링 상품을 빠르게 서치해요



키워드 검색어
어울리는 키워드 추천까지

AI가 스타일을 분석해 키워드를 추천!
외부 쇼핑몰로 바로 이동 가능해요.

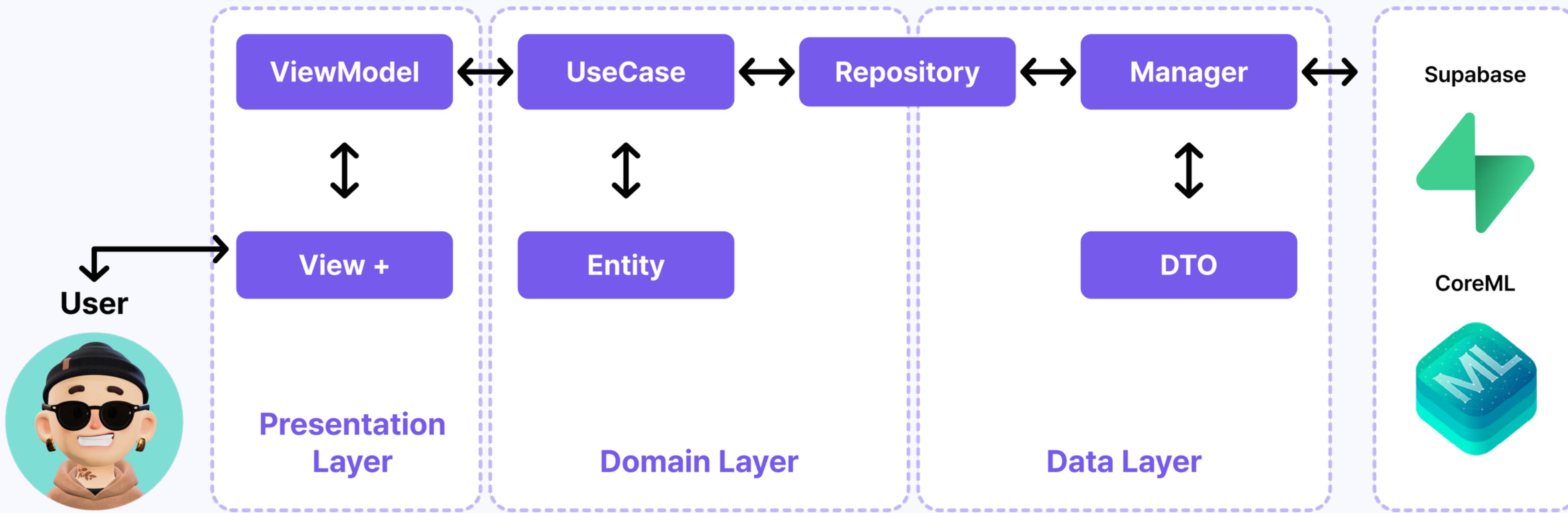


4. 기술 스택 및 아키텍처

- UIKit + Swift Concurrency:
복잡한 UI와 비동기 흐름을 직관적으로 구현
- RxSwift + RxDataSources:
이벤트 및 데이터 흐름을 선언적으로 관리
- Clean Architecture-MVVM:
유지보수성과 테스트 용이성을 고려한 구조 설계
- Action-State 기반 ViewModel:
예측 가능한 상태 전이 및 테스트 코드 작성 용이
- CoreML:
사용자 이미지를 기반으로 의류 키워드 분석
- Share Extension:
외부 플랫폼에서 상품을 앱에 즉시 저장 가능
- Supabase:
사용자 계정 및 상품 데이터를 실시간으로 관리



4. 기술 스택 및 아키텍쳐



- 사용자 요청은 **View + ViewModel**이 처리하고, 비즈니스 로직은 **UseCase**와 **Entity**가 담긴 **Domain Layer**에서 담당합니다.
- **Repository**는 **Data Layer**의 **Manager**를 통해 **Supabase**, **CoreML** 등 외부 의존성과 연결되며, **DTO**를 **Entity**로 변환해 도메인과 통신합니다.
- 레이어 간 의존성 방향을 명확히 하여 테스트 용이성과 유지보수성을 높였습니다.

5. 핵심 기능 구현

- 외부 쇼핑몰에서 공유된 상품 정보를 앱에 저장하는 기능으로, 무신사·지그재그·에이블리 3개 플랫폼을 MVP 범위로 설정하여 구현했습니다.



상품 추가 기능 (Share Extension)

- 플랫폼별 HTML 구조 차이를 통합적으로 처리하기 위해 **ProductSyncManager**를 설계하고, 각 플랫폼에 대응하는 **Fetcher** 클래스를 통해 상품 정보를 추출한 후 **ProductDTO**로 변환하였습니다.
- Share Extension** 모듈을 도입하여, 사용자는 앱 외부에서도 공유 버튼을 통해 상품을 바로 저장할 수 있도록 구현하였으며 동일한 상품 저장 흐름을 앱 내와 **Share Extension** 양쪽에서 일관되게 동작하도록 설계하였습니다.
- 사용자가 업로드한 패션 이미지를 분석해 의류 키워드를 도출하고, 이를 통해 플랫폼별 검색 결과로 이동할 수 있는 기능입니다.



패션 캡쳐 기능 (CoreML)

- 초기에 OpenAI 모델 활용을 검토했으나, 한국어 및 **.mlmodel** 미지원으로 자체 모델 학습으로 전환하였습니다.
- 학습 가능한 데이터셋 수집 및 가공을 통해 이미지-라벨 데이터를 Python 스크립트로 전처리하여 **CoreML** 학습에 적합하게 구성하였습니다.
- 결과적으로 자체 학습한 **.mlmodel**을 통해 평균 50~60% 정확도의 키워드 예측 초기 모델을 구축할 수 있었으며 이 기능을 통해 사용자에게 이미지를 기반으로 추천 키워드 → 플랫폼 검색 흐름을 자연스럽게 제공할 수 있었습니다.

6. 트러블 슈팅

6-1. RxDataSources 재바인딩에 대한 무한 루프 발생

Problem

- Compositional Layout을 사용하는 이미지 분석 화면에서 RxDataSources가 지속적으로 재바인딩되며 무한 루프 발생
- 헤더의 SegmentControl 값 변경 → 컬렉션 뷰가 끝없이 reloadData (스크롤 점프, FPS 하락)
- 데이터가 실제로 바뀌지 않아도 동일 섹션 배열이 반복 방출
- 결과적으로 메모리 사용량 급증·앱 응답 지연 발생

Impact

- 사용자 입장에서 SegmentControl을 눌렀을 때 UI가 즉각적으로 반응하지 않거나 버벅임이 발생할 수 있음
- 무한 루프가 발생하면서 메모리 누수가 누적될 수 있으며, 이로 인해 앱 Crash가 발생할 가능성이 높아짐
- 동일한 연산과 UI 업데이트가 반복적으로 수행되면서 불필요하게 CPU/GPU를 점유하여 배터리 소모를 높일 수 있음
- 앱이 멈추거나 강제 종료되는 현상이 반복되면서 사용자 이탈 및 부정적 리뷰로 연결될 수 있음

Cause

- didSelectedSegmentControl 액션이 값 비교 없이 그대로 스트림으로 전달
→ 같은 카테고리도 중복 방출
- RxDataSources는 items가 변경되면서 diff 계산 수행
→ reload를 수행하므로, 불필요한 diff가 반복 실행
- BehaviorRelay<[AnalysisSectionModel]>가 섹션 전체를 보유해 작은 변화에도 전체 배열 교체 발생
- ViewModel 내부에 중복 이벤트 차단 로직이 없어 세그먼트 값이 반복될 때도 changelItem 메서드 호출
- 테스트 단계에서 UI 이벤트(debounce, filter)를 고려하지 않아 뒤늦게 무한 루프 발견

Solution

1. ViewModel에 private var previousCategory = "" 프로퍼티를 추가해 마지막 선택 값 캐싱
2. setItems(category:) 메서드 내에 아래 코드를 추가하여 같은 카테고리면 조기 리턴하도록 수정

```
// 이전 카테고리와 비교해서 같은 값이 방출되면 무시
// RxDataSource에서 이벤트를 무한으로 방출하는 문제를 해결하기 위함
guard category != previousCategory else { return }
previousCategory = category
```

3. 중복 방출이 차단되면서 RxDataSources 재바인딩 루프 해소, 성능 하락과 프리즈 현상이 모두 개선됨

6. 트러블 슈팅

6-2. 통합 에러 처리 구조를 통한 스트림 안정성과 가독성 향상

Problem

- 에러 처리의 **코드 중복과 사용자 친화적 에러 메시지 부족**
 - 각 계층(Manager, Repository, ViewModel)에서 도메인에 정의된 에러 타입을 전달 · 처리
 - ViewModel에서 처리해야 할 **Error 타입 증가**
 - Rx 스트림마다 onError를 별도로 구독, Alert를 띄우는 코드가 반복되어 중복 코드 발생
 - 에러 발생 시 Rx 스크립트가 끊겨 의도치 않은 UI 이벤트 무시 상황 발생
 - 사용자 친화적인 에러 메시지 부족
- 에러 메시지 관리의 **일관성 부족**
 - Alert Title, Message를 ViewController에서 직접 관리해야 하므로, 일관성 유지가 어려움

Impact

- 에러 처리 방식이 화면마다 제각각으로 구현되어 **코드 패턴화가 심화되고**, 새로운 개발자가 코드를 파악하거나 수정할 때 **유지보수 난이도가 크게 상승함**
- 동일한 에러임에도 화면마다 서로 다른 메시지를 표시하여 사용자 경험의 일관성이 깨지고, 사용자 신뢰도가 저하될 우려가 있음
- 에러 전파와 처리 로직이 통일되지 않아 에러 발생 원인을 추적하거나 재현하기 어려워지고, 디버깅과 테스트 과정에서 불필요한 시간과 비용이 증가함

Cause

- Rx 스트림의 **onError** 사용
 - 데이터 스트림에서 에러 발생 시 onError로 스트림이 종료되어, 재시도나 지속적인 에러 관찰 불가
- 에러 메시지와 **Alert 관리의 분산**
 - Alert Title, Message 등 UI 요소를 각 ViewController에서 직접 관리해야 하므로, 일관성 저하
- 사용자에게 보여주는 에러 메시지와 **디버깅** 에러 메시지가 **분리되지 않음**

Solution

1. **AppError**로 **에러 타입 통합**
 - **Presentation Layer**에서는 하나의 에러 타입만을 전달 받아 **일관된 에러 처리**
2. **AppError**는 **LocalizedError 프로토콜 채택**
 - 사용자 친화적인 에러 메시지, 디버깅 에러 메시지 구분
3. **AppError**의 각 세부 에러 타입마다 **Alert Title** 정의
 - 일관된 **에러 Alert 생성**
4. **ViewModel**에서 에러를 **Observable 상태로 관리**
 - ViewController에서는 **하나의 에러 스트림만 구독**
 - 지속적 에러 관찰, 재시도 가능 및 **코드 중복 개선**

```
/// 앱 에러 프로토콜 - errorDescription, debugDescription
public protocol AppErrorProtocol: LocalizedError {
    var errorDescription: String? { get }
    var debugDescription: String { get }
}

/// 앱 에러 열거형
public enum AppError: AppErrorProtocol {
    case supabaseError(AppErrorProtocol)
    case mappingError(AppErrorProtocol)
    case authError(AppErrorProtocol)
    case keyChainError(AppErrorProtocol)
    case coreMLError(AppErrorProtocol)
    case unknown(Error)
}
```

[사용자 친화적 에러 메시지 / 디버깅 메시지 구분]

6. 트러블 슈팅

6-3. Supabase 소셜 로그인 관련 메모리 누수

Problem

- OAuth를 관리하는 ASWebAutenticationSession 인스턴스가 내부 `_referenceToSelf` 프로퍼티로 자기 자신을 강하게 참조하는 현상 발생

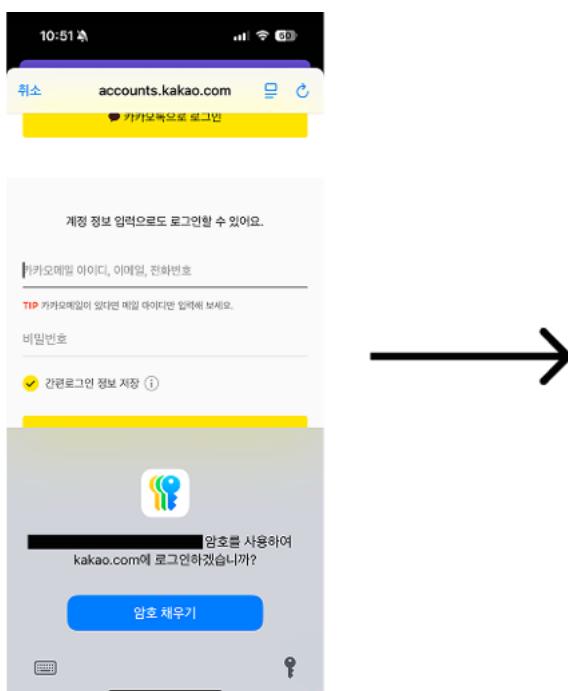


Impact

- 지속적인 메모리 누수로 인해 앱의 메모리 사용량이 점진적으로 증가하여, 장시간 사용 시 앱이 느려지거나 얘기치 않게 종료될 수 있음
- 로그인 세션이 명확히 종료되지 않아 인증 흐름 중 예외 상황이 발생할 수 있으며, 이는 사용자에게 반복 로그인 실패나 동작 불능 상태로 인식될 수 있음
- 불필요한 세션 인스턴스가 계속 유지되면 앱이 백그라운드 상태에서도 리소스를 점유하게 되어, 배터리 소모 증가 또는 앱의 백그라운드 상태 유지 제약으로 이어질 수 있음

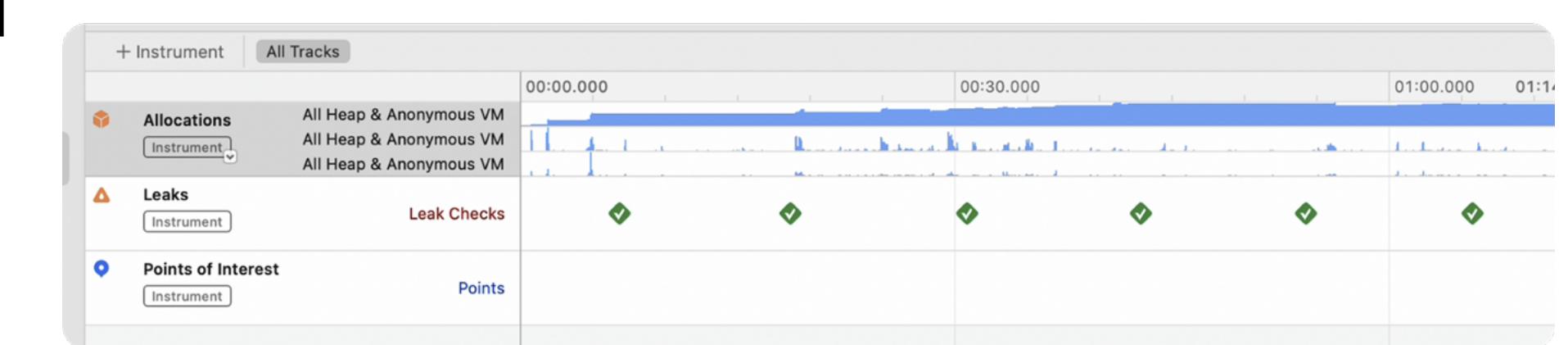
Cause

- 카카오톡 로그인 → 필수/선택 등의 제출 화면에서 SFAuthenticationSession, ASWebAuthenticationSession 메모리 누수 발생



Solution

- 카카오톡 로그인을 위한 ASWebAuthenticationSession 분리
- Supabase.auth.signInWithOAuth 호출 시 전달한 클로저 내부에서 반환된 ASWebAuthenticationSession 인스턴스를 카카오 세션 프로퍼티에 할당
- do-catch를 통해 로그인 취소시 cancel()로 ASWebAuthenticationSession 종료, 참조 해제





시연 영상



앱 스토어



GitHub

Thank You

