



# 무인매장 이상행동 감지 서비스

## " 깜빡 Catch ! "

Intel 인공지능 앱크리에이터 양성과정

2025-05-08

# Contents

---

1. 팀 소개
2. 프로젝트 소개
3. 문제 및 요구 사항 정의
4. 주요 기능
5. 시연
6. 프로젝트 구성(아키텍처)
7. 협업 방식 소개
8. 개발 과정
9. 회고



깜빡 Catch !

# PART. 01

## 팀 소개



# TEAM MEMBERS



박서현

AI ENGINEER

- AI 엔지니어
- 모델 학습 및 개선



김태호

DATA ENGINEER

- 데이터 엔지니어
- 데이터 수집 및 전처리



이재봉

PM

- 프로젝트 매니저
- 프로젝트 전체 관리
- 웹 페이지 제작 및 관리



박세은

FRONTEND

- 프론트엔드 담당자
- 모바일 어플리케이션 개발



깜빡 Catch !

## PART. 02

### 프로젝트 소개



# KKAMBAK CATCH



깜빡 Catch ! - “깜빡” 놓친 사이에도 즉시 **Catch!**

깜빡!  
**Catch**

"깜빡"은 순간적인 이상행동을 포착하는 느낌을, "Catch"  
는 이를 빠르게 잡아내는 기술력을 강조 .

무인편의점 + 무인매장 CCTV 를 24시간 모니터링  
-> 이상행동을 3초 이내 탐지 + 점주에게 알림 서비스



# Persona



**박광수, 48세**  
제조업 중견기업 재무팀  
차장

## 거주지

경기 남양주시 창동 아파트

## 근무시간

09:00 - 18:00 (주 5일)

## 특이사항

월말 결산·감사 시즌에는  
잦은 야근

## 부업

무인 편의점 2곳 운영

## 서울 강남구

집 ↔ 강남 매장 : 차로 10 분

## \* 서울 마포구 홍대입구

집 ↔ 홍대 매장 : 차  
로 35 분

## Problem

- 1) 평일 주간엔 사무실 근무로 매장 모니터링 불가
- 2) 최근 무단취식·흡연·상품 파손 빈발로 복구 비용 증가
- 3) 사건 시간을 몰라 CCTV 를 들여다보며 주말 반납

## Needs

- 1) 회사나 집에서도 스마트폰으로 이상행동 즉시 알림
- 2) 두 매장 영상을 한 화면에서 동시 확인
- 3) 사건 발생 시점 저장 → 보험·경찰 제출용 증거



깜빡 Catch !

## PART. 03

### 문제 및 요구사항 정의



# 문제 정의

무인 매장 3년 연속 '범죄주의보'…절도·파손  
심각

승인 2025-04-30 17:19



김미지 인턴기자 unknown@kyeonggi.com  
[기자페이지 >](#)

무인 매장 민원 3년간 2천700여건…매년 증가  
전문가 "상인들 피해 커…형벌 강화 촉선책"



경기도 소재의 한 무인점포에 절도 범죄 관련 경고문이 부착돼 있다. 기사와 직접적 연관이 없는 사진입니다.  
B

## 수면바지 입고 '무인편의점' 찾아온 커플... '망치'로 키오스크 부수고 현금 털어갔다

최종수정일 2024.11.21. 09:18 ▾

☞ 가 ☐ ☎

### 수면바지 입고 망치 든 커플... 무인점포 키오스크 털어갔다



네이버 카페 '아프니까 사장이다'

## 두달 간 물건 훔쳐갔는데…무인점포 업주 "경찰도 못 잡아 답답"

등록 2024.08.04 00:30:00 | 수정 2024.08.04 06:52:52

☞ ☐ ☎ ☒ ☗ ☘ ☗ ☗



[서울=뉴시스] 무인점포에서 같은 사람이 계속 물건을 훔쳐 가고 있는데 처벌할 방법이 없어서 답답하다는 업주의 사연이 알려졌다. (사진= 아프니까 사장이다 갈무리) \*재판매 및 DB 금지

# 문제 정의

## 1. 무인화 확산

24 시 무인 편의점·셀프 매장 수 ↑,  
인력 감시공백 발생  
→ 범죄 발생 건수 증가

## 2. 이상행동에 의한 손실

절도·폭행·기물파손 etc  
→ 매출 손실·법적 분쟁·고객 불안

## 3. 기존 CCTV 는 사후 분석

사고 후 역추적 → 대응 지연  
→ 점주가 외부에 있으면 즉각 조치 불가

# 핵심 요구사항 정의

01

YOLO v8 경량  
모델실시간 바운딩  
박스

02

다양한 이상 행동  
클래스(4개) 분류  
- Normal  
- 전도  
- 절도  
- 파손

03

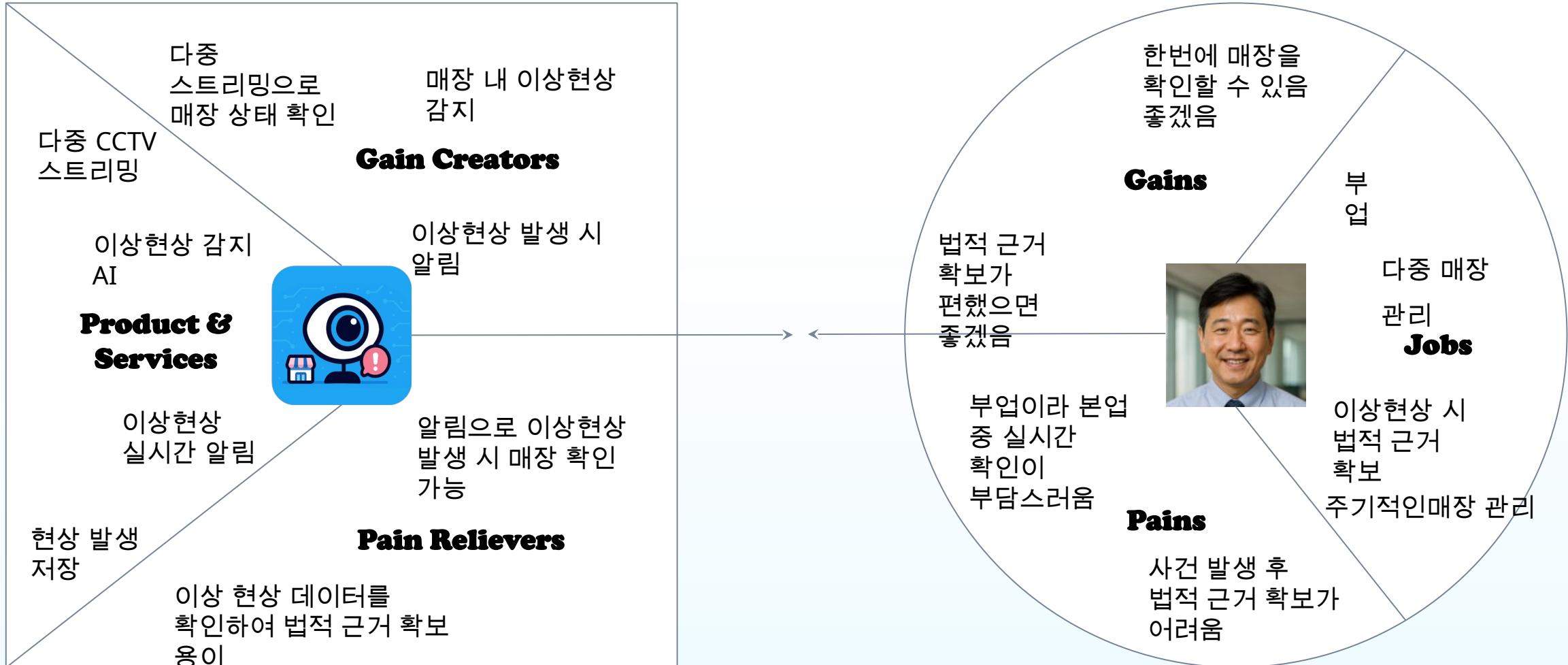
각 이상 행동  
클래스 80% 확률  
이상의 경우 발생

04

모바일  
어플리케이션 알림  
서비스  
("깜빡 Catch !")



# Value Proposition Canvas





깜빡 Catch !

## PART. 04

### 주요 기능



# Main Features – Website



서비스 안내     솔루션     제품 소개     요금 안내

안전성과 신뢰성을 갖춘  
**깜빡Catch**  
무인 매장 관리에 최고의 솔루션

문의하기

무인 매장 관리 최고의 선택  
**깜빡Catch입니다.**

처음 이용 고객     기존 이용 고객

uid	name	email	password	phone	pin_num	use_yn	created_at	updated_at
302fe52d-7476-4c	a123	a2@a2	a123123!	1111111111	NULL	TRUE	2025-04-29 02:04:05.551498+	2025-04-29 06:44:06.938963
64199f92-f8b1-41	test	test@test	a123123!	1012341234	NULL	TRUE	2025-04-23 08:21:11.235881+01	2025-05-02 07:38:55.107192+
8e0f14d1-a8c4-45	a123	a2@a	a123123!	1123121232	NULL	FALSE	2025-04-29 01:55:02.124218+0	2025-04-29 07:35:46.693598
a4e83ccdd-590b-4	관리자	admin	intel00	1000000000	NULL	TRUE	2025-04-29 05:21:59.153862+C	2025-04-29 06:11:42.420882+
f87c482a-a013-4c	test2	test2@test	a123123!	1012341234	NULL	TRUE	2025-04-23 08:21:46.964237+	2025-04-30 03:25:32.985036

서비스 안내     솔루션     제품 소개     요금 안내

신규 고객

이름  
이메일  
하이픈(-) 제외, 11자리 숫자 입력  
비밀번호  
비밀번호 확인

최소 8자 이상이며, 문자, 숫자, 특수문자를 포함  
입력 비밀번호와 확인 비밀번호 일치 여부

점포 등록

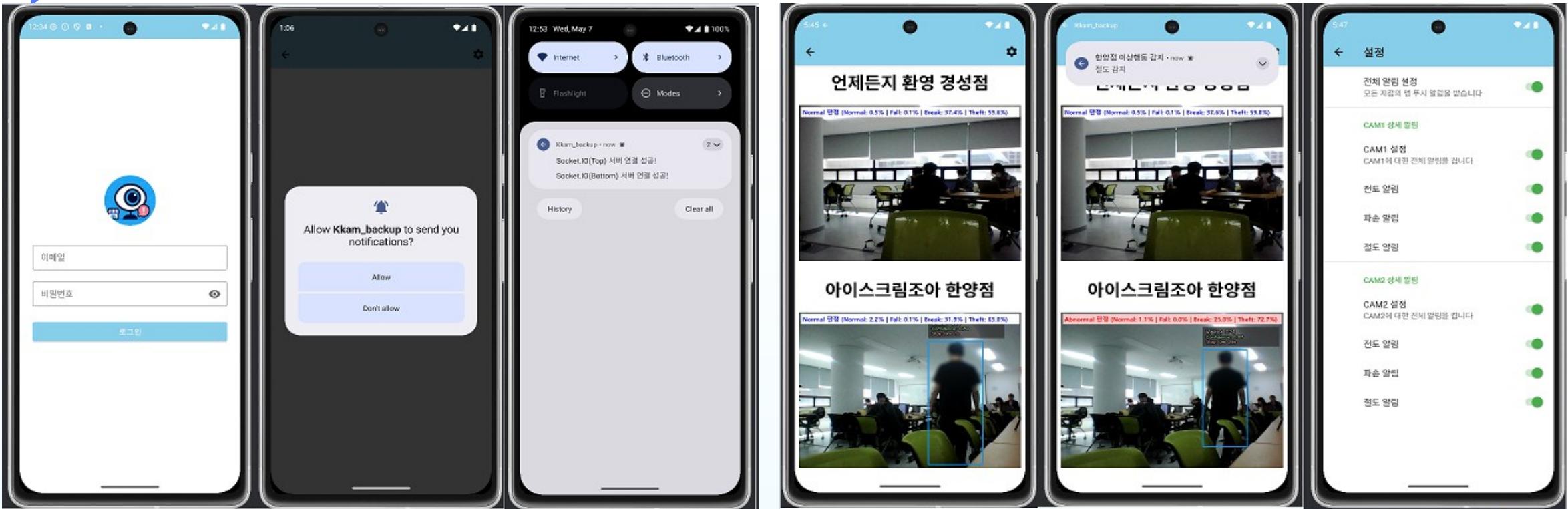
점포 이름  
점포 주소  
정렬 순서

신청하기

id	uid	store_name	sn	created_at	updated_at	address	use_yn
36224adff-bf57-43	302fe52d-74...	A-22	233	2025-04-30 01:19:28.292797+01	2025-04-30 01:19:28.292797+01	1	FALSE
3aa4a076-230f-42	f87c482a-a0...	인천도시 환경 강남점 1	1	2025-04-23 08:22:42.993164+C	2025-05-07 00:46:06.400294+	92.168.219.140	TRUE
43744860-44f4-4b	302fe52d-74...	A-A 111	22	2025-04-30 01:19:28.292797+01	2025-04-30 01:19:28.292797+01	2	FALSE
6959009b-01f5-4b	8e0f14d1-a8...	aa	1	2025-04-29 01:55:02.231384+01	2025-04-30 02:25:27.423884+C	aa	TRUE
7c8ded26-925e-4c	302fe52d-74...	1	1	2025-04-30 00:58:50.454997+C	2025-04-30 01:19:28.189194+00	0.0.2	TRUE
c0fbfb2d-830d-4c	302fe52d-74...	마이스크림조아 신사 4	4	2025-04-29 02:04:05.609884+C	2025-05-01 00:52:54.233994+C	192.168.219.170	TRUE
e9b08699-664d-4c	302fe52d-74...	o.o	3	2025-04-30 00:58:50.454997+C	2025-04-30 01:33:46.605984+C	o.232	FALSE
f5cb6978-143d-43	f87c482a-a0...	아이스크림 조아 신사 2	2	2025-05-02 01:19:29.33809+C	2025-05-02 07:39:16.392624+C	192.168.219.180	TRUE

id	uid	store_name	sn	created_at	updated_at	address	use_yn
36224adff-bf57-43	302fe52d-74...	A-22	233	2025-04-30 01:19:28.292797+01	2025-04-30 01:19:28.292797+01	1	FALSE
3aa4a076-230f-42	f87c482a-a0...	인천도시 환경 강남점 1	1	2025-04-23 08:22:42.993164+C	2025-05-07 00:46:06.400294+	92.168.219.140	TRUE
43744860-44f4-4b	302fe52d-74...	A-A 111	22	2025-04-30 01:19:28.292797+01	2025-04-30 01:19:28.292797+01	2	FALSE
6959009b-01f5-4b	8e0f14d1-a8...	aa	1	2025-04-29 01:55:02.231384+01	2025-04-30 02:25:27.423884+C	aa	TRUE
7c8ded26-925e-4c	302fe52d-74...	1	1	2025-04-30 00:58:50.454997+C	2025-04-30 01:19:28.189194+00	0.0.2	TRUE
c0fbfb2d-830d-4c	302fe52d-74...	마이스크림조아 신사 4	4	2025-04-29 02:04:05.609884+C	2025-05-01 00:52:54.233994+C	192.168.219.170	TRUE
e9b08699-664d-4c	302fe52d-74...	o.o	3	2025-04-30 00:58:50.454997+C	2025-04-30 01:33:46.605984+C	o.232	FALSE
f5cb6978-143d-43	f87c482a-a0...	아이스크림 조아 신사 2	2	2025-05-02 01:19:29.33809+C	2025-05-02 07:39:16.392624+C	192.168.219.180	TRUE

# Main Features – Mobile Application





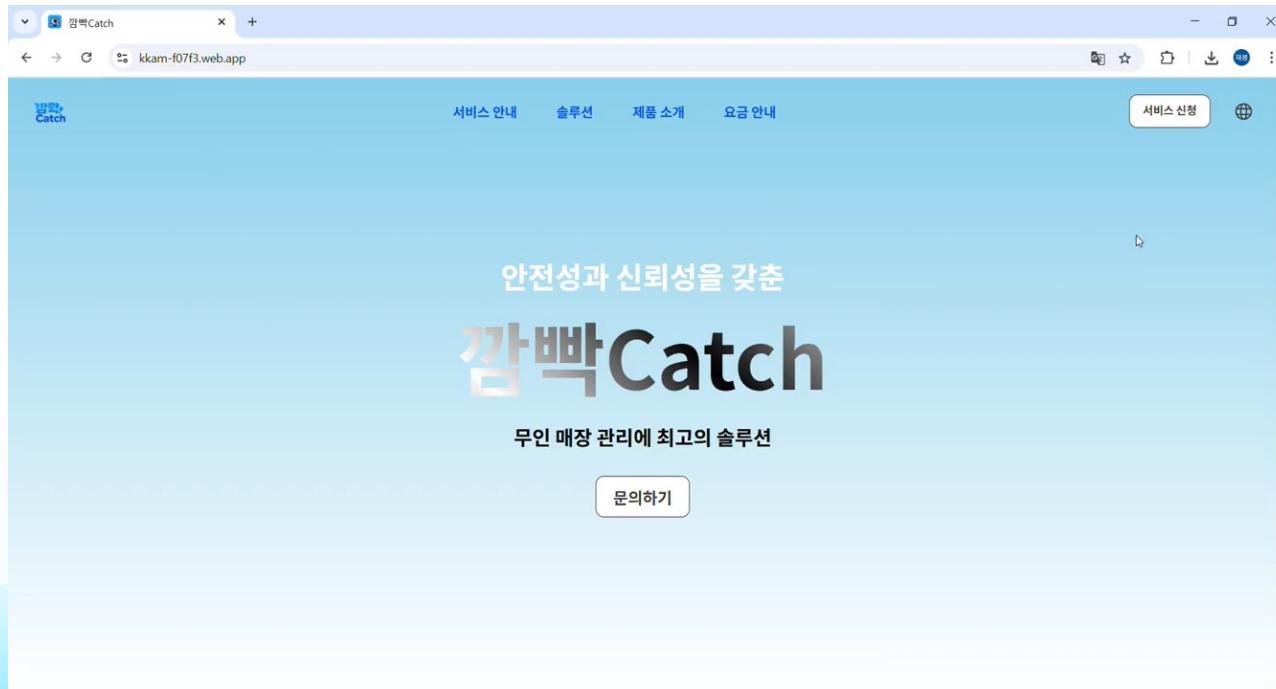
깜빡 Catch !

# PART. 05

## 시연 Demo



# 시연





깜빡 Catch !

# PART. 06

## 프로젝트 구성 (아키텍처)



# 구성도

---



**Video Files → Data Preprocessing → ResNet18 → LSTM → Classifier → 이상 행동 DB 저장**



## Data Preprocessing

- 동영상(3fps/sec) 프레임 별 라벨 할당 (Normal, Fall, Theft, Break)
- 영상 크기 통일 (224\*224)
- 채널 단위 Z-score 정규화



## AI 모델 구조

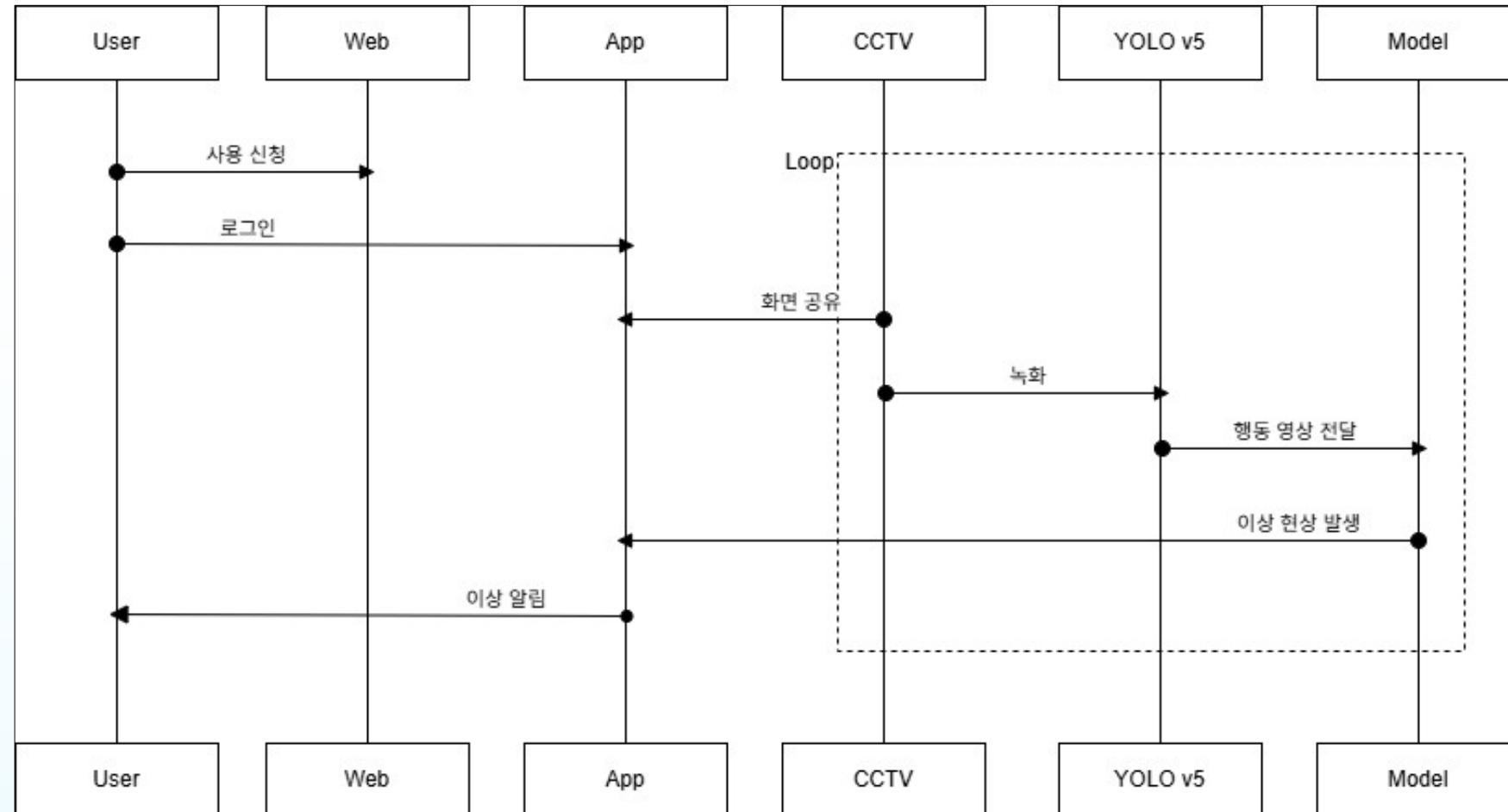
- ResNet18 + Bi-LSTM



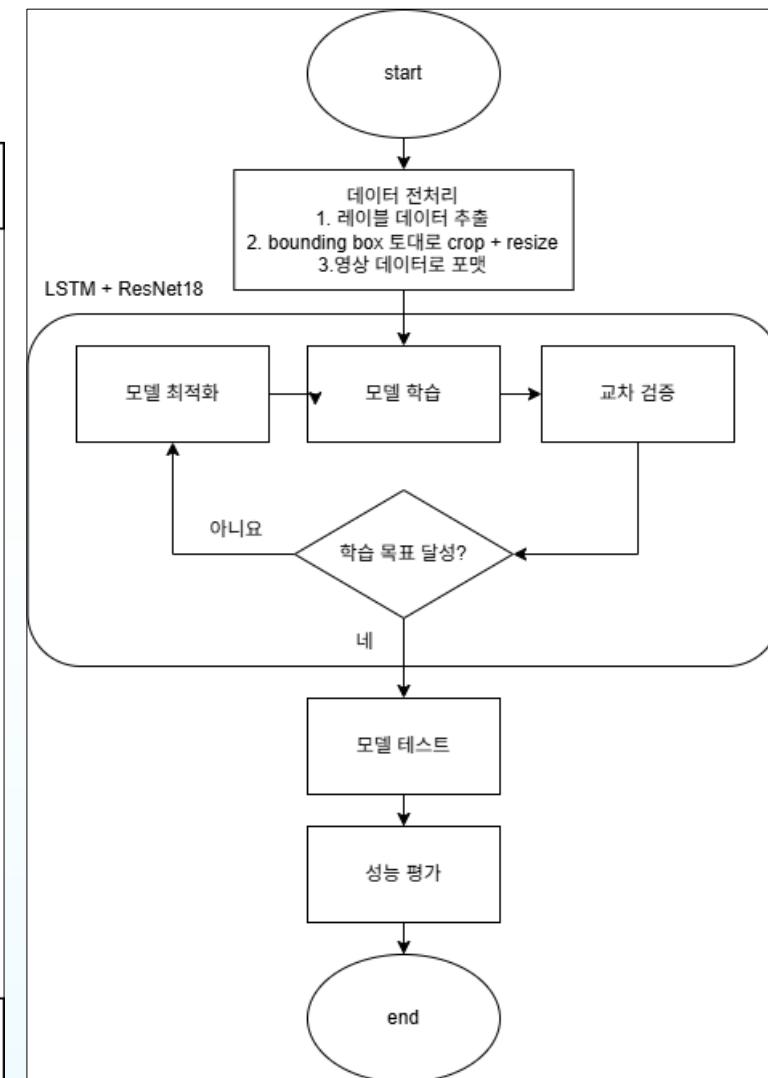
## DB 저장

- 이상행동 탐지 시 SupaBase에 분류 클래스 저장
- uid, sid, latest\_abnormal\_class

# 다이어그램



< 시퀀스 다이어그램 >



< 데이터 플로우 다이어그램 >



깜빡 Catch !

## PART. 07

### 협업방식 소개



# Collaboration Workflow



# Collaboration Tools



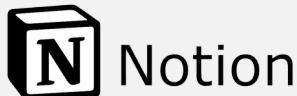
- 아이디어 회의
- 각자 진행 상황 공유
- 팀원 간 요청 사항

The image shows two screenshots of the Slack application. The left screenshot displays a project channel named 'dubnlp\_프로젝트\_아자' (dubnlp\_Project\_Aza) with a message from '박재현' (Park Jae-hyun) about a 6-day backlog. The right screenshot shows a direct message window titled '두번재\_프로젝트\_아자' (Second\_Dubnlp\_Project\_Aza) with multiple team members like '박재현', '이재봉', and '김재호' (Kim Jai-ho) discussing project details.



- 각 역할 별 코드 공유
- 코드 버전 관리

The image shows two screenshots of the GitHub application. The left screenshot shows the organization page for '깜빡Catch' (Catch) with four members: Seo-hyeon Park, woor1668, Taeho Kim, and Seeun-aicode. The right screenshot shows a list of repositories under the organization 'KKAM' (KKAM), including 'KKAM\_Android\_ReactNative', 'KKAM\_Android\_v2', 'KKAM\_AI', 'KKAM\_Hompage', 'KKAM\_Android', and 'KKAM\_Back', all updated within the last few days.



- 프로젝트 일정 관리
- 일일 보고서 작성, 공유
- 팀원 별 작업 현황

The image shows two screenshots of the Notion application. The left screenshot is a calendar for April 2025, showing tasks like '3-4일 차' (3-4 day period) and '5-7일 차' (5-7 day period) with due dates ranging from April 1st to April 14th. The right screenshot shows a '작업' (Tasks) page with two lists: '사용 권한 있음' (Access Granted) and '사용 권한 없음' (Access Denied). It lists various tasks such as '홈페이지', '신청하기', '관리자페이지', '기존 사용자 페이지', '데이터 처리 전략 구상 및 개체 탐지', '모델 학습 파이프라인 구축 및 학습', 'Yolov5기반 사람 및 물 객체 탐지', '뷰어 기능 추가-얼굴 인식화, 체류시간', and '뷰어 최적화(후본 속도 개선)', each with details like assignee, status, and priority.



깜빡 Catch !

# PART. 08

## 개발 과정



# 데이터 수집 및 EDA

데이터 출처

AI Hub AI 데이터찾기 AI 허브소개 참여하기 커뮤니티 AI 개발자지원 고객지원 로그인 회원가입

데이터 찾기

#안전/환경 #환경 #안전

실내(편의점, 매장) 사람 이상행동 데이터

분야 영상이미지 유형 비디오

구축년도: 2022 | 업데이트 날짜: 2023-11 | 조회수: 23,404 | 다운로드: 1,707 | 용량: 253.17 GB

다운로드 샘플 데이터

관심데이터 등록 109

폴더 및 파일

[ROOT]

- 238-2. 실내(편의점, 매장) 사람 이상행동 데이터/
- 01-1. 정식개방데이터/
- Training/
  - 01. 원천데이터/
    - TS\_03. 이상행동\_07. 전도/
    - TS\_03. 이상행동\_08. 파손/
    - ... (이상행동별 분류)
  - 02. 라벨링데이터/
    - TL\_03. 이상행동\_07. 전도/
    - TL\_03. 이상행동\_08. 파손/
    - ... (이상행동별 XML)
- Validation/
  - 01. 원천데이터/
    - VS\_03. 이상행동\_07. 전도/
    - VS\_03. 이상행동\_08. 파손/
    - ...
  - 02. 라벨링데이터/
    - VL\_03. 이상행동\_07. 전도/
    - VL\_03. 이상행동\_08. 파손/
    - ...

항목	내용	원천 데이터 특징
영상 길이	1분 (60초)	
프레임 수	180프레임 (3fps)	
해상도	RGB, 정규화된 해상도	
포맷	.mp4 (ZIP으로 압축되어 제공)	
촬영 환경	실제 편의점/매장 내 CCTV 기반	
행동 유형	전도, 파손, 흡연, 유기, 절도, 폭행 (이상행동 6종)	

항목	설명	라벨링 데이터 구조 및 정보
VideoName	라벨이 적용된 영상 파일명	
BehaviorType	이상행동 종류 (예: 전도, 절도 등)	
StartFrame / End-Frame	행동이 시작되고 끝나는 프레임 번호 (0부터 시작)	
PersonID	이상행동을 수행한 인물 식별 ID	
Skeleton	이상행동을 수행한 인물의 2D 관절 좌표 정보	

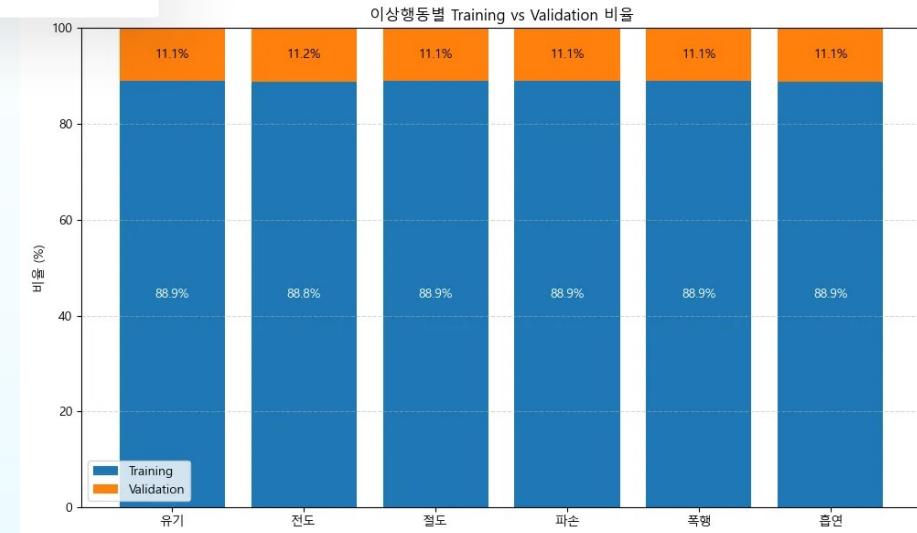
# 데이터 수집 및 EDA

이상행동 유형	설명 예시	이상행동 유형		이상행동별 영상 수량 분석				
		이상행동	Training (원천)	Training (라벨)	Validation (원천)	Validation (라벨)	총합	
전도	사람이 넘어진 상황	전도	645	645	81	81	1,452	
파손	물건을 부순 행위	파손	642	642	80	80	1,444	
흡연	실내에서의 흡연 장면	흡연	686	686	86	86	1,544	
유기	유아, 동물 등의 방치	유기	642	642	80	80	1,444	
절도	상품을 훔치는 행위	절도	641	641	80	80	1,442	
폭행	사람 간 물리적 폭력	폭행	641	641	80	80	1,442	
		합계			487	487	8,768	

## 데이터 분포 시각화



Training vs Validation  
비율



# 데이터 전처리

## 1. 경로 / 라벨 사전

```
# 레이블 맵: 클래스명 → 정수 레이블
LABEL_MAP = {
    'Normal': 0,
    '전도': 1,
    '파손': 2,
    '흡연': 3,
    '유기': 4,
    '절도': 5,
    '폭행': 6
}
```

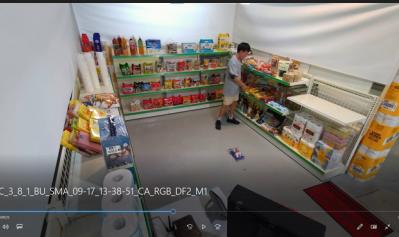
## 2. 전체 프레임 리사이즈 (224,

```
def load_frames(video_path, size=(224,224)):
    """
    MP4 전체 프레임을 (H,W,3) uint8 리스트로 반환
    """
    cap = cv2.VideoCapture(str(video_path))
    frames = []
    while True:
        ok, frame = cap.read()
        if not ok:
            break
        # resize 및 BGR→RGB
        frame = cv2.cvtColor(cv2.resize(frame, size), cv2.COLOR_BGR2RGB)
        frames.append(frame)
    cap.release()
    return frames
```

## 3. XML에서 이상행동 구간

### XML 파일 예시 (ex)파손 = bro-

```
<track id="0" label="broken_start" source="manual">
    <box frame="68" outside="0" occluded="0" keyframe="1" xtl="1011.48" ytl="182.23" xbr="1176.55" ybr="621.09" z_order="0">
        <attribute name="age group">child</attribute>
        <attribute name="gender">male</attribute>
        <attribute name="ID">1</attribute>
    </box>
    <box frame="69" outside="1" occluded="0" keyframe="1" xtl="1011.48" ytl="182.23" xbr="1176.55" ybr="621.09" z_order="0">
        <attribute name="age group">child</attribute>
        <attribute name="gender">male</attribute>
        <attribute name="ID">1</attribute>
    </box>
</track>
<track id="1" label="broken_end" source="manual">
    <box frame="142" outside="0" occluded="0" keyframe="1" xtl="685.28" ytl="217.60" xbr="842.81" ybr="385.55" z_order="0">
        <box frame="143" outside="1" occluded="0" keyframe="1" xtl="685.28" ytl="217.60" xbr="842.81" ybr="385.55" z_order="0">
    </box>
</track>
```



```
def parse_intervals(xml_path, behavior_eng):
    """
    XML에서 behavior_start ~ behavior_end 박스(outside='0') 프레임을
    전체 중 최소/최대로 묶어 [(start, end)]를 반환
    """
    tree = ET.parse(str(xml_path))
    root = tree.getroot()
    starts, ends = [], []
    for track in root.findall('track'):
        lbl = track.attrib.get('label', '')
        # 'fall_start' 또는 'fall_end' 등 정확 매칭
        if lbl == f'{behavior_eng}_start':
            for box in track.findall('box'):
                if box.attrib.get('outside') == '0':
                    starts.append(int(box.attrib['frame']))
        elif lbl == f'{behavior_eng}_end':
            for box in track.findall('box'):
                if box.attrib.get('outside') == '0':
                    ends.append(int(box.attrib['frame']))
    if not starts or not ends:
        return []
    else:
        return list(zip(starts, ends))
```

# 데이터 전처리

## 4. Positive & Normal npy 생성

(XML에서 추출한 이상행동 구간 -> Positive / 이상행동 이외의 구간 -> Normal )

```
def process_video(mp4_path, xml_path, split, behavior_kor):
    frames = load_frames(mp4_path)
    N = len(frames)

    # 한글→영어 prefix
    behavior_eng = ENG_MAP.get(behavior_kor)
    if behavior_eng is None:
        return

    intervals = parse_intervals(xml_path, behavior_eng)
    if not intervals:
        print(f"이벤트 구간 없음: {mp4_path.stem}")
        return

    # Positive 저장 폴더
    pos_X_dir = OUT_ROOT/split/behavior_kor/'x'
    pos_y_dir = OUT_ROOT/split/behavior_kor/'y'
    pos_X_dir.mkdir(parents=True, exist_ok=True)
    pos_y_dir.mkdir(parents=True, exist_ok=True)

    # Negative 저장 폴더
    neg_X_dir = OUT_ROOT/split/'Normal'/'x'
    neg_y_dir = OUT_ROOT/split/'Normal'/'y'
    neg_X_dir.mkdir(parents=True, exist_ok=True)
    neg_y_dir.mkdir(parents=True, exist_ok=True)
```

```
# Positive 클립 생성
for idx, (s, e) in enumerate(intervals):
    clip = np.stack(frames[s:e+1])
    X_name = f"{mp4_path.stem}_{idx}_X.npy"
    y_name = f"{mp4_path.stem}_{idx}_y.npy"
    np.save(pos_X_dir/X_name, clip)
    np.save(pos_y_dir/y_name, np.array(LABEL_MAP[behavior_kor], dtype=np.uint8))

# Normal 클립 생성
mask = np.zeros(N, dtype=bool)
for s, e in intervals:
    mask[s:e+1] = True
normals = [f for m, f in zip(mask, frames) if not m]
if normals:
    clip = np.stack(normals)
    X_name = f"{mp4_path.stem}_N_X.npy"
    y_name = f"{mp4_path.stem}_N_y.npy"
    np.save(neg_X_dir/X_name, clip)
    np.save(neg_y_dir/y_name, np.array(LABEL_MAP['Normal'], dtype=np.uint8))
```

# 데이터 전처리

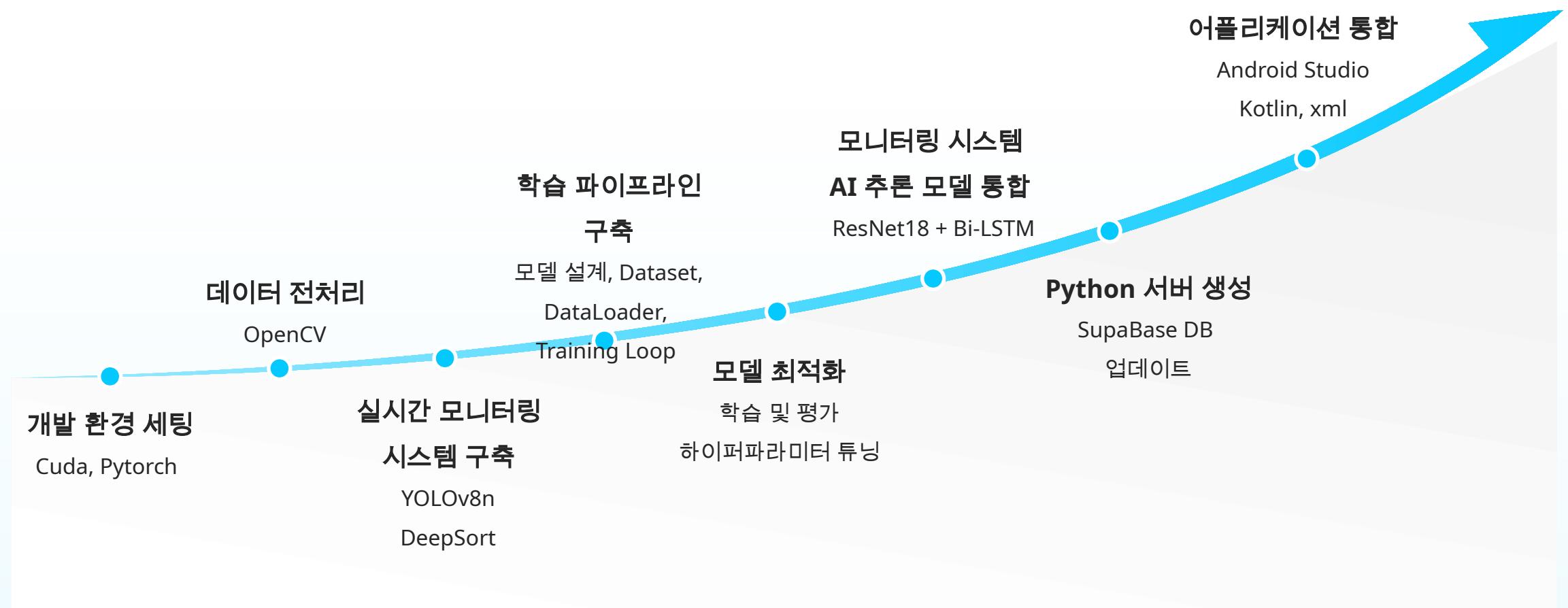
저장된 \_X.npy / \_y.npy 파일 구조

파일	dtype	shape 예시	안에 든 내용
..._E0_X.npy ..._E1_X.npy ...	uint8	(T, 224, 224, 3) • T = 이상행동 구간 프레임 수 (예: 75)	- 시퀀스 전체 RGB 프레임 - 채널 순 서: H × W × C (OpenCV /BGR→RGB 변환 후) - 픽셀 값 0~255
..._N_X.npy	uint8	(TB <sup>uff</sup> , 224, 224, 3) • TB <sup>uff</sup> = 영상에서 이상행동을 제외한 프레임 수	- 같은 영상의 “정상” 구간만 모은 클립
대응 *_y.npy	uint8	()(스칼라)	- 클래스 ID 하나만 저장 • 0 : Normal • 1 : 전도 • 2 : 파손 • 3 : 흡연 • 4 : 유기 • 5 : 절도 • 6 : 폭행

최종 전처리된 npy 데이터 폴더

구조			
PC > 로컬 디스크 (D:) > 238-2.실내(편의점, 매장) 사람 이상행동 데이터 > processed_npy > train >			
이름	수정한 날짜	유형	크기
Normal	2025-04-20 오후 6:56	파일 폴더	
유기	2025-04-21 오후 5:00	파일 폴더	
전도	2025-04-20 오후 6:56	파일 폴더	
절도	2025-04-20 오후 8:28	파일 폴더	
파손	2025-04-20 오후 7:20	파일 폴더	
폭행	2025-04-20 오후 8:50	파일 폴더	
흡연	2025-04-20 오후 7:43	파일 폴더	
PC > 로컬 디스크 (D:) > 238-2.실내(편의점, 매장) 사람 이상행동 데이터 > processed_npy > train > 유기 >			
이름	수정한 날짜	유형	크기
X	2025-04-23 오전 10:59	파일 폴더	
y	2025-04-23 오전 10:59	파일 폴더	
PC > 로컬 디스크 (D:) > 238-2.실내(편의점, 매장) 사람 이상행동 데이터 > processed_npy > train > 유기 > X			
이름	수정한 날짜	유형	크기
C_3_11_1_BU_DYA_07-31_15-51-35_CA_RGB_DF2_M1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	5,440KB
C_3_11_1_BU_DYA_07-31_15-51-37_CB_RGB_DF2_M1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	6,028KB
C_3_11_1_BU_DYA_07-31_15-51-37_CC_RGB_DF2_M1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	6,175KB
C_3_11_1_BU_DYA_08-06-14-09-50_CC_RGB_DF2_M1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	5,440KB
C_3_11_1_BU_DYA_08-06-14-09-51_CB_RGB_DF2_M1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	5,440KB
C_3_11_1_BU_DYA_08-06-14-09-52_CA_RGB_DF2_M1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	5,146KB
C_3_11_1_BU_SMA_08-14-14-04-30_CA_RGB_DF2_M1_F1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	12,349KB
C_3_11_1_BU_SMA_08-14-14-04-30_CB_RGB_DF2_M1_F1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	12,496KB
C_3_11_1_BU_SMA_08-14-14-04-30_CC_RGB_DF2_M1_F1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	12,055KB
C_3_11_1_BU_SMA_08-14-14-04-30_CD_RGB_DF2_M1_F1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	12,202KB
C_3_11_1_BU_SMA_08-28-14-11-46_CC_RGB_DF2_M1_F1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	15,142KB
C_3_11_1_BU_SMA_08-28-14-11-46_CD_RGB_DF2_M1_F1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	14,848KB
C_3_11_1_BU_SMA_08-28-14-11-47_CA_RGB_DF2_M1_F1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	15,142KB
C_3_11_1_BU_SMB_08-28-16-10-40_CA_RGB_DF2_M1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	15,877KB
C_3_11_1_BU_SMB_08-28-16-10-40_CB_RGB_DF2_M1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	15,730KB
C_3_11_1_BU_SMB_08-28-16-10-40_CC_RGB_DF2_M1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	16,024KB
C_3_11_1_BU_SMB_08-28-16-10-40_CD_RGB_DF2_M1_E0_X.npy	2025-04-20 오후 8:13	NPY 파일	16,318KB

# 모델 개발 단계



# AI 모델 학습 전략 및 세팅

## 학습 전략

- Multi-label classification (4-class)
- BCE + sigmoid: 각 클래스에 대한 독립적인 2진 분류(binary classification) 수행
- Validation loss 기반 best model 가중치 업데이트

## 하이퍼파라미터 세팅

```
# 학습 파라미터
parser.add_argument('--epochs', default=50, help='number of training epochs')
parser.add_argument('--batch-size', default=4, help='batch size per GPU')
parser.add_argument('--lr', default=1e-4, help='learning rate')

# 데이터셋 파라미터
parser.add_argument('--window-size', default=2, help='time window size for clip (sec)')
parser.add_argument('--num-workers', default=4, help='number of DataLoader workers')
parser.add_argument('--fps', default=3, help='number of frames per clip (3fps)')

# 모델 파라미터
parser.add_argument('--hidden-dim', default=256, help='LSTM hidden dimension')
parser.add_argument('--num-layers', default=1, help='number of LSTM layers')
parser.add_argument('--num-classes', default=4, help='number of anomaly classes; default = auto from dataset')
parser.add_argument('--bidirectional', default=True, help='use bidirectional LSTM')
parser.add_argument('--freeze-backbone', default=False, help='freeze ResNet backbone weights')
```

# AI 모델 개발 주요 코드

## 데이터 전처리

Torch.utils.data - Dataset 및 z-score 정규화

```
# Normalize each frame manually
mean = np.array([0.485, 0.456, 0.406])
std = np.array([0.229, 0.224, 0.225])

frames = []
for frame in x_data:
    frame = np.transpose(frame, (1, 2, 0)) # (H, W, C)
    frame = cv2.resize(frame, (224, 224))
    frame = (frame / 255.0 - mean) / std
    frame = torch.tensor(frame, dtype=torch.float32).permute(2, 0, 1)
    frames.append(frame)

x_tensor = torch.stack(frames) # (seq_len, C, H, W)
return x_tensor, label
```

## 모델 설계

VAModel: ResNet18 + Bi-LSTM

Layer (type:depth-idx)	Output Shape	Param #
VAModel	[4, 8]	--
ResNet18: 1-1	[64, 512]	--
Sequential: 2-1	[64, 512, 1, 1]	--
Conv2d: 3-1	[64, 64, 112, 112]	9,408
BatchNorm2d: 3-2	[64, 64, 112, 112]	128
ReLU: 3-3	[64, 64, 112, 112]	--
MaxPool2d: 3-4	[64, 64, 56, 56]	--
Sequential: 3-5	[64, 64, 56, 56]	147,968
Sequential: 3-6	[64, 128, 28, 28]	525,568
Sequential: 3-7	[64, 256, 14, 14]	2,099,712
Sequential: 3-8	[64, 512, 7, 7]	8,393,728
AdaptiveAvgPool2d: 3-9	[64, 512, 1, 1]	--
LSTM: 1-2	[4, 256]	--
LSTM: 2-2	[4, 16, 256]	788,480
Linear: 1-3	[4, 8]	2,056
Total params:	11,967,048	
Trainable params:	11,967,048	
Non-trainable params:	0	
Total mult-adds (G):	116.12	
Input size (MB):	38.54	
Forward/backward pass size (MB):	2543.45	
Params size (MB):	47.87	
Estimated Total Size (MB):	2629.86	

# AI 모델 개발 주요 코드

## 학습 코드

- Trainer class 생성
- Validation loss 기반 model 가중치 저장

```
class Trainer:  
    def __init__(self, model: nn.Module,  
                 train_loader: DataLoader,  
                 val_loader: DataLoader = None,  
                 criterion: nn.Module = nn.BCEWithLogitsLoss(),  
                 optimizer: torch.optim.Optimizer = None,  
                 scheduler=None,  
                 device: torch.device = None,  
                 save_path: str = 'BCE_4_best_model.pth'):  
  
        self.model = model  
        self.train_loader = train_loader  
        self.val_loader = val_loader  
        self.criterion = criterion  
        self.optimizer = optimizer  
        self.scheduler = scheduler  
        self.device = device  
        self.save_path = save_path  
  
    def fit(self, epochs: int):  
        for epoch in range(1, epochs + 1):  
            start = time.time()  
            train_loss, train_acc = self.train_epoch()  
            val_loss, val_acc = self.validate_epoch()  
  
            # 스케줄러 업데이트  
            if self.scheduler:  
                self.scheduler.step(val_loss)  
  
            # 베스트 모델 저장  
            if val_loss < self.best_val_loss:  
                self.best_val_loss = val_loss  
                torch.save(self.model.state_dict(), self.save_path)  
                print(f"Best model saved at epoch {epoch} (val_loss: {val_loss:.4f})")  
  
            elapsed = time.time() - start  
            print(f"Epoch {epoch}/{epochs} | Train Loss: {train_loss:.4f} | Val Loss: {val_loss:.4f} "  
                  f" | Train Acc: {train_acc:.4f} | Val Acc: {val_acc:.4f} | Time: {elapsed:.1f}s")  
  
            # 로그 저장  
            with open("training_log.csv", "a") as f:  
                f.write(f"{epoch},{train_loss:.4f},{val_loss:.4f},{train_acc:.4f},{val_acc:.4f}\n")
```

## 모니터링 시스템 설계

- YOLOv8n 기반 object detection
- DeepSort 기반 tracking
- 얼굴 블러링 - 익명화

```
# YOLOv8 모델 로드 (사람 탐지)  
yolo_person = YOLO('yolov8n.pt') # yolov8n/s/m/l/x 중 선택  
  
# DeepSORT 트래커  
tracker = DeepSort(max_age=30, n_init=2)  
  
# 얼굴 블러링 함수  
def anonymize_face_simple(frame, x1, y1, x2, y2, method='pixelate'):  
    h, w = frame.shape[:2]  
    x1, y1 = max(0, x1), max(0, y1)  
    x2, y2 = min(w, x2), min(h, y2)  
    if x2 <= x1 or y2 <= y1:  
        return frame  
  
    face = frame[y1:y2, x1:x2]  
    if face.size == 0:  
        return frame  
  
    if method == 'blur':  
        face = cv2.GaussianBlur(face, (15, 15), 10)  
    elif method == 'pixelate':  
        temp = cv2.resize(face, (10, 10), interpolation=cv2.INTER_LINEAR)  
        face = cv2.resize(temp, (x2 - x1, y2 - y1), interpolation=cv2.INTER_NEAREST)  
  
    frame[y1:y2, x1:x2] = face  
    return frame
```

# AI 모델 개발 주요 코드

## 모니터링 시스템 설계

- YOLOv8n 기반 object detection
- DeepSort 기반 tracking
- Info box 생성 (객체 id, 체류 시간)
- 이상 상태 DB 저장

```
def detect_and_draw(frame):
    start_time = time.time()

    # YOLO 감지
    results_person = yolo_person(frame, conf=0.8, verbose=False)[0] # 첫 결과
    boxes = results_person.bboxes
    dets_person = boxes.xyxy.cpu().numpy()
    confs = boxes.conf.cpu().numpy()
    clss = boxes.cls.cpu().numpy()

    # 사람 감지된 박스 리스트 구성
    detections = []
    for i in range(len(dets_person)):
        if int(clss[i]) == 0: # class 0: person
            x1, y1, x2, y2 = map(int, dets_person[i])
            conf = confs[i]
            detections.append(([x1, y1, x2 - x1, y2 - y1], conf, 'person'))

    # DeepSORT 추적 업데이트
    tracks = tracker.update_tracks(detections, frame=frame)
    h, w = frame.shape[:2]

    for track in tracks:
        if not track.is_confirmed():
            continue

        track_id = int(track.track_id)
        l, t, r, b = map(int, track.to_ltrb())
        current_time = time.time()
```



```
# 반투명 박스 그리기
overlay = frame.copy()
cv2.rectangle(overlay, (info_box_x, info_box_y),
             (info_box_x + box_width, info_box_y + box_height),
             (50, 50, 50), -1)
alpha = 0.8
cv2.addWeighted(overlay, alpha, frame, 1 - alpha, 0, frame)

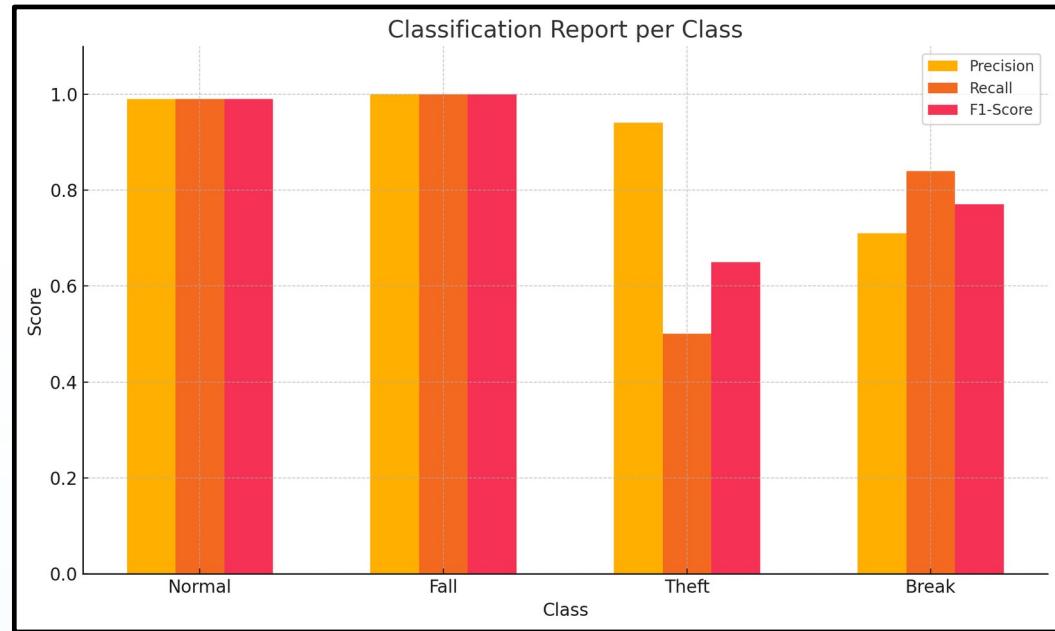
# 텍스트 그리기 - 가독성을 위해 밝은 글씨와 검은 외곽선 추가
draw_text_with_outline(frame, f'Visitor {track_id:03d}', (info_box_x + 10, info_box_y + 20),
                       cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)
draw_text_with_outline(frame, f'Confidence: {score:.2f}', (info_box_x + 10, info_box_y + 35),
                       cv2.FONT_HERSHEY_SIMPLEX, 0.4, (200, 255, 200), 1)
draw_text_with_outline(frame, f'Stay: {duration_min}m {duration_sec}s',
                       (info_box_x + 10, info_box_y + 50),
                       cv2.FONT_HERSHEY_SIMPLEX, 0.4, (200, 200, 255), 1)
```

```
# Supabase 기록
if status == "Abnormal" 판정":
    abnormal_class = np.argmax([fall_prob, break_prob, theft_prob]) + 1
    latest_abnormal_class = int(abnormal_class)
    sb.table('abnormal').insert({
        'uid': store_uid,
        'sid': store_sid,
        'class': latest_abnormal_class,
    }).execute()

else:
    status = "Normal" 판정"
    latest_abnormal_class = 0
```

# 모델 학습 결과

F1 Score	: 0.8539
Precision	: 0.9097
Recall	: 0.8346
precision recall f1-score support	
Normal	0.99 0.99 0.99 384
Fall	1.00 1.00 1.00 64
Theft	0.94 0.50 0.65 64
Break	0.71 0.84 0.77 64
micro avg	0.95 0.92 0.94 576
macro avg	0.91 0.83 0.85 576
weighted avg	0.95 0.92 0.93 576
samples avg	0.92 0.92 0.92 576



## 특이 사항

기존의 이상 행동 class <유기, 전도, 절도, 파손, 폭행, 흡연> 중

**유기 class, 흡연 class**는 모델이 normal로 판단하는 경향이 강하여 최종 모델에서 제외.

**폭행 class**는 파손 class와 혼동하는 경향이 강하여 두 개의 클래스 중 더 중요하다고 생각되는 파손 class를 남기고 폭행 class를 최종 모델에서 제외.

# 웹 사이트

서비스 가입 후기

## 고객님들의 솔직한 후기



고객 행동까지 분석해줘요.  
매장 내 비정상적인 동선은 손이나 도난을 예방할 수 있다는데 눈뜨고



### 깜빡Catch만의 합리적인 요금

유형	기본 요금	프리미엄 요금	프로 요금	엔터프라이즈 요금
업*성	월 5,900원	월 9,900원	월 19,900원	월 49,000원
설명	기본 분석 보고서	기본 분석 보고서 + 전화 지원	기본 분석 보고서 + 전화 지원 + 전담 고객 매니저	기본 분석 보고서 + 전화 지원 + 전담 고객 매니저 + 맞춤형 통합

신청하기

신청하기

#### 사용자 목록

아이디	이메일	이름	승인 여부	생성일	수정일
64199f92-f8b1-41e2-80aa-05dad032d77c	test@test	test	승인	2025-04-23	2025-05-02
f87c482a-a013-4646-8948-f13bc3765f6a	test2@test	test2	승인	2025-04-23	2025-04-30
8e01f4d1-a8c4-499b-b10c-4a82dfdbc795	a2@a	a123	미승인	2025-04-29	2025-04-29
302fe52d-7476-4db4-a0b3-690425a72813	a2@a2	a123	승인	2025-04-29	2025-04-29
a4e83cdd-590b-4863-adfc-49ade0b43025	admin	관리자	승인	2025-04-29	2025-04-29

#### 가게 목록

가게 ID	가게명	주소	승인 여부	생성일	수정일
f3cbb878-f43d-431b-ab20-e8a6dba0e1db	아이스크림 조아 흥대점	192.168.219.180	승인	2025-05-02	2025-05-07
91c7a011-801e-4754-b487-6000fce55d02	언제든지 경성점	192.168.219.154	승인	2025-05-07	2025-05-07
3aa4a976-230f-4244-a5a3-59e391a62eb8	언제든지 환영 강남점	192.168.219.140	미승인	2025-04-23	2025-05-07

Q. 해당 솔루션을 어떻게 홍보할 것인가?

- A. 프로젝트 소개용 웹 사이트 개발

Q. 비즈니스적으로 어떻게 수익을 창출할 것인가?

- A. 구독제 분할 서비스로 사용자 상황에 맞게 결제

Q. 주소를 알면 아무나 사용 가능한가?

- A. 관리자의 승인 시스템으로 보안 강화

# 모바일 어플리케이션

```
26     class MainActivity : AppCompatActivity() {
27         private fun initSocketIO() {
28             // Top 소켓
29             mSocketTop = IO.socket("http://", opts).apply {
30                 connect()
31                 on(Socket.EVENT_CONNECT) {
32                     Log.d(TAG, "Top Socket connected")
33                     runOnUiThread {
34                         NotificationHelper.showHeadUp(
35                             context = this@MainActivity,
36                             id: 9998,
37                             title: "Socket.IO(Top)",
38                             message: "서버 연결 성공!"
39                         )
40                     }
41                 }
42             }
43             on(Socket.EVENT_DISCONNECT) {
44                 Log.d(TAG, "Top Socket disconnected")
45             }
46         }
47     }
48
49     on(S)
50     class LoginPinActivity : AppCompatActivity() {
51         companion object {
52             private const val CORRECT_EMAIL = "email@domain.com"
53             private const val CORRECT_PASSWORD = "password123"
54         }
55
56         override fun onCreate(savedInstanceState: Bundle?) {
57             super.onCreate(savedInstanceState)
58             // activity_login.xml 레이아웃 사용
59             setContentView(R.layout.activity_login)
60
61             etEmail = findViewById(R.id.etEmail)
62             etPassword = findViewById(R.id.etPassword)
63             btnLogin = findViewById(R.id.btnLogin) // R.id.btnLogin 과 일치시킴
64
65             btnLogin.setOnClickListener {
66                 val email = etEmail.text.toString().trim()
67                 val password = etPassword.text.toString()
68
69                 when {
70                     email.isEmpty() || password.isEmpty() -> {
71                         Toast.makeText(context = this, text = "이메일과 비밀번호를 모두 입력하세요", Toast.LENGTH_SHORT).show()
72                     }
73                     email != CORRECT_EMAIL || password != CORRECT_PASSWORD -> {
74                         Toast.makeText(context = this, text = "이메일 또는 비밀번호가 올바르지 않습니다", Toast.LENGTH_SHORT).show()
75                     }
76                     else -> {
77                         // 로그인 성공 시 MainActivity로 이동
78                         startActivity(Intent(packageContext = this, MainActivity::class.java))
79                         finish()
80                     }
81                 }
82             }
83         }
84     }
85
86     build.gradle.kts (app)
87     AndroidManifest.xml
88
89
90
91
92
93
94
95
96 }
```

## 1. 주요 기능 & 화면 구성

- 로그인 화면: 이메일, 비밀번호로 로그인
- 메인 페이지: 실시간 영상 피드 UI
- 이상행동 알림: 알림 발생 조건 · 알림 UI 샘플
- 설정 페이지: 사용자 환경설정 항목

## 2. 기술 스택

- 언어 & 프레임워크: Kotlin, Android SDK
- 라이브러리: Socket.IO(네트워크), Engine.IO WebSocket 트랜스포트(이미지 로딩), Android Notification Channel/API (알림), MediaPlayer (사운드 재생)

## 3. 주요 구현 내용

- 웹캠(카메라) 연결: 권한 요청 → WebView 연결
- 이상 행동 알림 처리: Android Notification Channel 소리 설정, 핸즈업 알림 처리



깜빡 Catch !

# PART. 09

회고



# 주요 성과



## 1. 실시간 이상 감지 기술 내재화

- 영상 스트리밍 기반의 실시간 처리 파이프라인 구축에 성공
- 특정 이상행동(예: 쓰러짐, 침입, 절도 등)을 딜레이 없이 탐지 가능
- Latency-accuracy trade-off를 고려한 모델 설계 및 경량화 적용

## 2. 무인 환경에 적합한 안정성과 운영 효율 확보

- 사람이 없는 매장에서도 24시간 자동 감시 가능
- 알림 기반 이벤트 감지로 점주의 즉각적인 대응 가능성 향상
- 전통적인 CCTV 대비 능동적 모니터링 시스템으로 전환

## 3. 상업적 확장 가능성

- 추후 도입 가능한 기능(혼잡도 분석, 재고 예측 등)을 고려한 모듈화된 구조 설계
- 다양한 업종(편의점, 무인카페, 무인피트니스 등)으로의 확장성 확보
- 고객 입장 수 / 패턴 분석 기능과 결합 시, 운영 전략 수립에 기여 가능

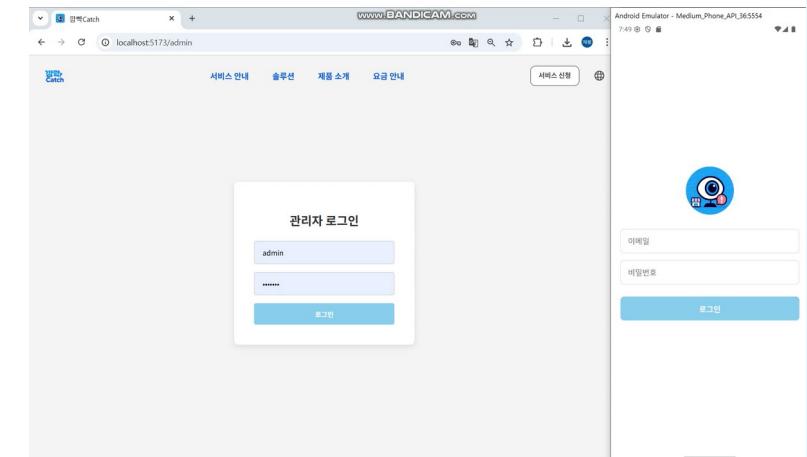
## 4. 실제 서비스 개발 경험 축적

- 실시간 서비스 환경에서의 영상 처리, 시스템 병목 분석 경험
- 사용자 중심 설계를 위한 UX 요소(알림 방식, 대시보드 시각화 등) 고려
- PoC 수준을 넘는 프로덕션 수준의 프로토타입 구축 경험

# 개선 과제 / Known Issue



- 시간/요일별 고객 방문 패턴 분석 미비
  - 특정 요일이나 시간대에 따른 고객 수를 정량적으로 분석하는 기능 구현할 시간이 부족
  - 이를 통해 혼잡 시간대 예측 및 운영 최적화 가능성을 높일 수 있음.
- 점주를 위한 통합 관리 시스템 확장 필요
  - 단순한 이상탐지를 넘어서 혼잡도 모니터링, 재고 관리 기능 등으로 확장할 필요가 있음.
- DB 와 앱 연동 개선 필요
  - DB 와 앱이 연동이 되지 않아 이상 행동이 발생한 시간과 지점을 확인 할 수 있게 개선 필요.



# 회고

## PM 이재봉

프로젝트 시작 전에 호환성을 고려하지 않고 언어를 선택한 탓에, 프로젝트 후반부에 호환성 문제를 발견하여 추가적인 고도화 작업이 필요하게 되었습니다. 다음 프로젝트에서는 호환성까지 꼼꼼히 검토하여 보다 안정적으로 진행할 수 있길 바랍니다.

## DATA ENGINEER 김태호

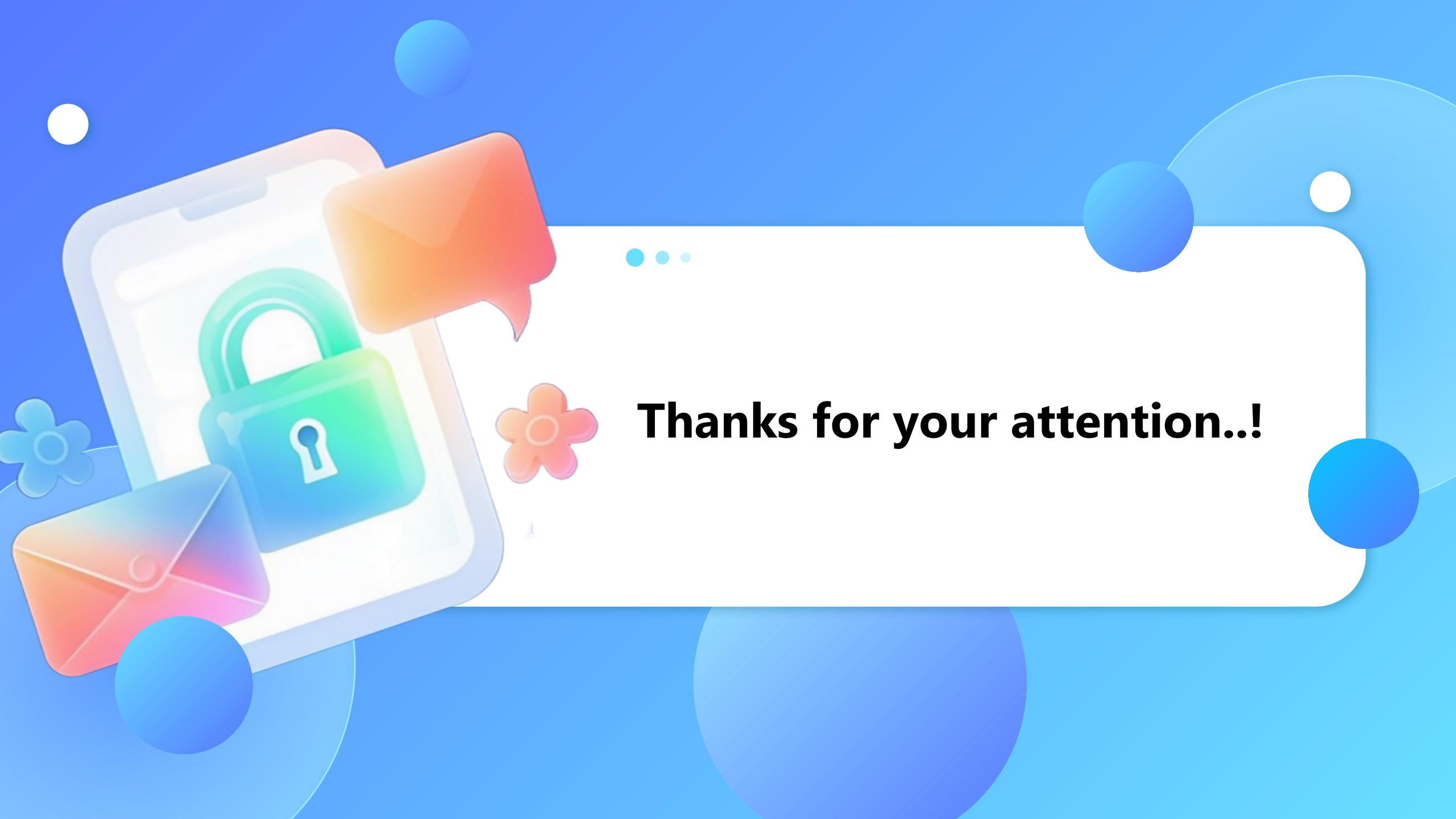
처음 모델 학습 방식을 선택한 후, 여러 가지 과정을 통해서 모델의 성능을 테스트 해 본 결과, 어떤 방식으로 모델을 바꿔도 성능이 크게 향상되지 않는 과정을 겪었습니다. 결국 모델의 성능을 결정하는 중요한 부분은 데이터 임을 다시 한 번 크게 깨달았고 데이터 엔지니어 역할의 중요성에 대해서 몸소 느낄 수 있는 소중한 과정이었습니다.

## AI ENGINEER 박서현

이번 프로젝트를 통해 \*\*latency(지연 시간)\*\*와 정확도 사이의 트레이드오프를 실제로 고려해보며, 사용자 중심의 설계 방향성과 서비스 현실화를 함께 고민해볼 수 있는 의미 있는 경험이었습니다.

## FRONTEND 박세은

안드로이드 스튜디오와 UI 작업이 모두 처음이라 어려움이 많았지만, 팀원분들의 도움 덕분에 하나하나 배워가며 진행할 수 있었습니다. 처음 계획했던 모든 내용을 완전히 실행하지는 못했지만, 핵심 기능 구현에는 성공해 나름대로 만족스럽게 마무리할 수 있었습니다.



**Thanks for your attention..!**