코드에 근거를 담는 백엔드 개발자 전지웅

MAU 52,470명 규모의 서비스를 직접 개발하고 운영했습니다.
비전 AI와 실시간 판단 모델을 서비스에 적용한 경험도 있습니다.

wjswldnd2000@naver.com github.com/JJW0123

경력

2023.06 — 2023.12

애드캡슐소프트 · 백엔드 인턴

관리자 시스템 DB 설계와 API 개발

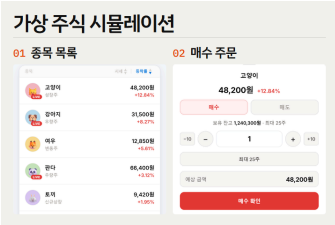
- 약 60페이지 분량의 화면정의서를 6개 도메인으로 분류
- 도메인별 데이터 관계와 제약 조건을 정의하고 ERD와 DB 정의서 작성
- 계층형 메뉴·공통코드 관리 기능과 동적 검색·필터링 구현
- 파일 업로드 검증과 HTTP 상태 코드별 JSON 오류 응답 규칙 설계

프로젝트

01 **가상 주식 시뮬레이션**

2026.04 — 2026.07 · 개인 프로젝트 · 월간 활성 이용자 52,470명

스트리머 채널을 주식처럼 거래하는 시즌제 모의 투자 서비스입니다.



02 **무인 지게차 물류 자동화**

2026.07 — 2026.08 · 6인 팀 · SSAFY 공통 프로젝트 우수상

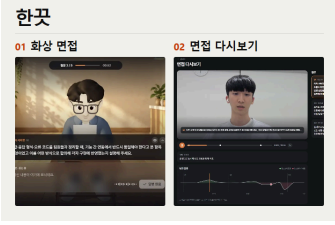
화물의 크기를 재고 선반 자리를 정해, 무인 지게차가 옮기도록 만든 시스템입니다.



03 **한끗**

2026.10 — 2026.11 · 5인 팀

AI 면접관이 답변에 실시간으로 반응하고, 피드백을 누르면 해당 장면을 다시 볼 수 있는 화상 모의면접입니다.



주력 기술

Java

중상

인턴 때 계층형 메뉴와 동적 검색을 구현했습니다. 프로젝트에서는 시세·배당 계산, 적재 위치 선정, 한끗의 피드백 타임라인을 개발했습니다.

Spring Boot

중상

매매 API와 배당 스케줄러를 개발하고 미완료 배당을 재처리하도록 했습니다. 한끗에서는 AI 분석을 작업 큐로 비동기 처리했습니다.

MySQL

중상

인턴 때 관리자 시스템의 데이터 관계와 제약 조건을 설계했습니다. 주식 서비스에서는 거래 내역 저장과 잔고 갱신을 한 트랜잭션으로 처리했습니다.

Redis

중

자주 조회하는 시세와 순위를 Redis에서 읽도록 구성했습니다. 방송 상태 조회 작업에는 만료 시간이 있는 락을 걸어 중복 실행을 막았습니다.

그 외 사용해 본 기술

인프라	Docker Compose, Nginx, Oracle Cloud VM, AWS EC2 · S3	모니터링	Prometheus, Grafana Cloud
API · 인증	Swagger, JWT	AI	Python, PyTorch, Hugging Face
협업	Git, GitHub, Jira	기타	Firebase, C#, Unity

교육 및 자격

교육

2026.01 – 2026.12 **삼성 청년 SW·AI 아카데미(SSAFY) 15기** 이수 중

- Java 기반 웹 개발
- AI 활용과 RAG 파이프라인 구축
- 알고리즘·CS

2025.02 **명지전문대학 전공심화 컴퓨터공학과 학사**

학점 3.94 / 4.5

자격

2024.12 **정보처리기사**

한국산업인력공단

2026.04 **TOEIC Speaking IM3**

한국TOEIC위원회

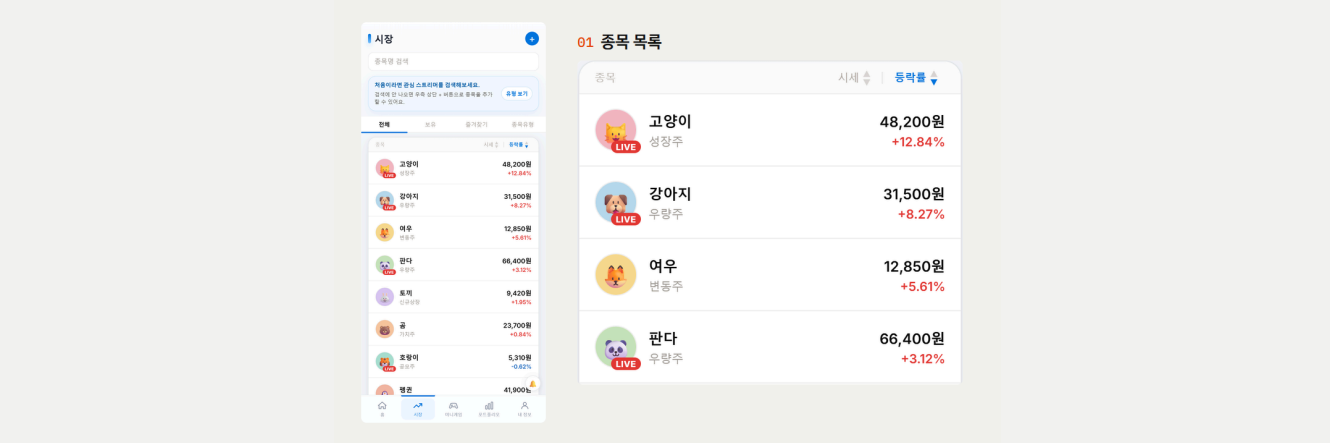
월간 활성 이용자 52,470명

가상 주식 시뮬레이션

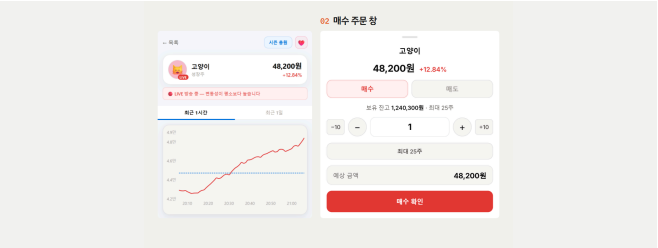
2026.04.25 — 2026.07.01 · 개인 프로젝트 · 설계·개발·배포·운영

> **52,470명**
> 월간 활성 이용자

2026.05.18–06.18 (31일간 집계)



종목 목록 · 현재가와 등락률



종목 상세 · 매수 주문 창



시즌 자산 순위표

서비스 운영은 종료됐습니다. 아래 화면에는 시연용 데이터를 사용했습니다.

개요

소개	스트리머 채널을 주식처럼 사고팔고 배당을 받는 모의 투자 서비스입니다. 시즌마다 이용자의 보유 자산을 기준으로 순위를 매깁니다.
맡은 일	거래·시세·배당 백엔드를 설계하고 구현했습니다. 인프라 구성부터 배포, 운영 중 장애 대응까지 직접 맡았습니다.
사용 기술	**Java 17 · Spring Boot · MySQL · Redis · Nginx · Prometheus · Grafana** 시세와 순위는 조회 요청이 잦아 Redis에서 가져오도록 했습니다. 매수·매도 시에는 MySQL의 한 트랜잭션에서 거래 내역을 기록하고 잔고를 갱신했습니다.
링크	GitHub · github.com/JJW0123/stock-simulation 운영 당시 주소 virtual-stock.xyz · 2026년 7월 서비스 종료

가상 주식 시뮬레이션

문제 해결

01 배당 지급을 건별로 분리

초기에는 하루치 배당을 하나의 트랜잭션으로 처리해도 1초 안에 끝났지만, 이용자가 급증하자 지급 시간이 길어졌습니다. 그 사이 매매와 배당이 같은 이용자의 잔고를 동시에 갱신하다 충돌했고, 한 건의 실패로 하루치 배당 전체가 롤백됐습니다.

이용자·종목별 배당을 별도 트랜잭션으로 처리해 한 건이 실패해도 나머지 배당은 지급할 수 있게 했습니다. 잔고에 배당금을 원자적으로 더하고, 배당일·이용자·종목에 유일 제약을 두어 중복 지급을 막았습니다.

02 날짜별 배당 배치 관리

건별 지급으로 바꾼 뒤에는 일부 지급이 실패해도 성공한 건의 이력이 남았습니다. 그날 배당이 모두 끝났는지, 재처리가 필요한지 확인할 기준이 필요했습니다.

배당 배치의 실행 상태와 성공·실패 건수를 날짜별로 MySQL에 별도 기록했습니다. 진행 중이거나 완료된 배당은 중복으로 시작하지 않도록 제어하고, 실패가 남은 날짜는 재처리했습니다. 전체 지급에는 수십 분이 걸렸기 때문에, 종목별 기준가는 실행 시작 시 메모리에 고정해 지급 대상과 금액을 계산했습니다.

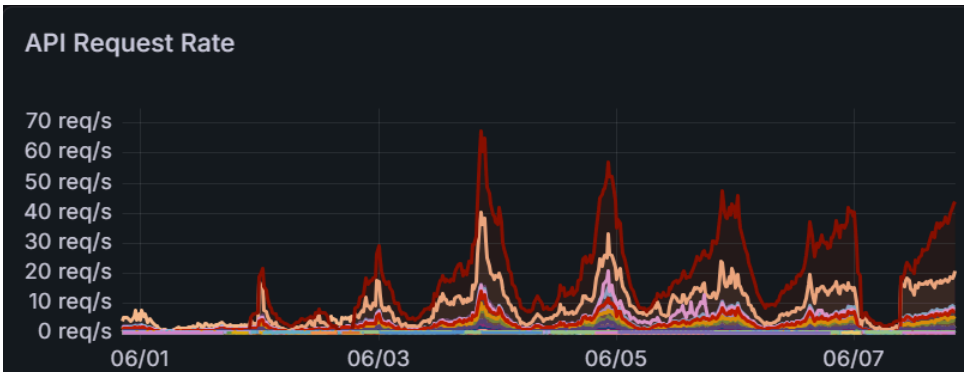
03 방송 상태 분할 조회

종목이 늘면서 외부 API로 전체 종목의 방송 상태를 확인하는 데 10분 이상 걸렸습니다.

운영 중에는 한 주기에 전체 종목의 약 10%를 조회하고, 다음 주기에는 그다음 종목부터 이어서 조회했습니다. Redis 락이 유효한 동안에는 조회 작업이 겹쳐 시작되지 않도록 했습니다. 조회에 걸린 시간과 성공·실패 건수도 기록했습니다.

회고

운영 초반에는 Grafana에서 서비스 전체의 응답 시간과 오류 건수를 봤습니다. 응답이 빠른 시세 조회에 요청이 몰리면 느린 API가 전체 지표에 묻혔습니다. 서버 오류(5xx)가 어느 API에서 발생했는지도 바로 알기 어려웠습니다. 이후 요청량과 응답 시간은 API별로, 오류 건수는 요청 경로와 응답 코드별로 나눠 확인했습니다. 운영 지표는 이상이 생겼을 때 점검 대상을 좁힐 수 있도록 구성해야 한다는 점을 배울 수 있었습니다.



API별 초당 요청 수 · Grafana

SSAFY 공통 프로젝트 우수상

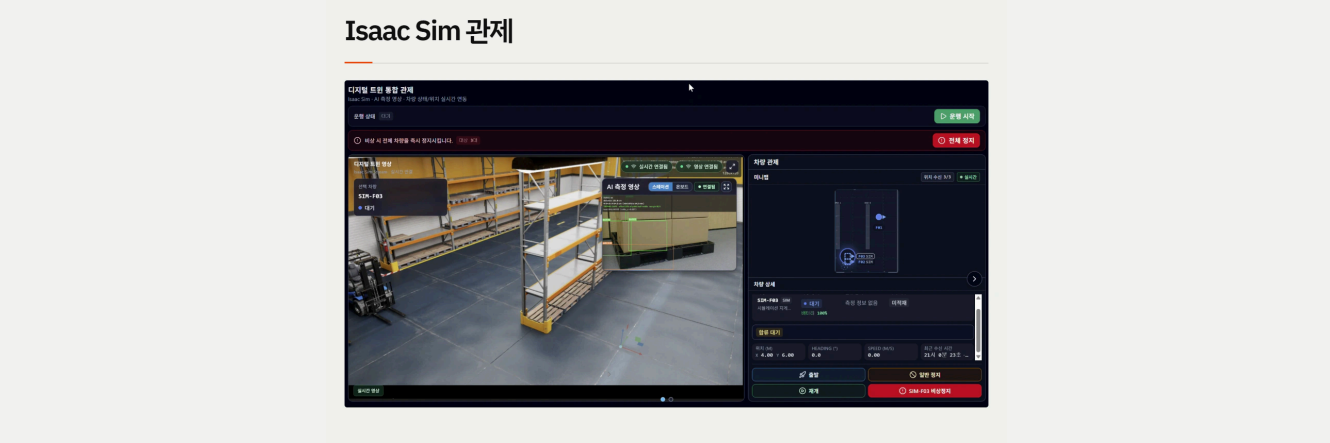
무인 지게차 물류 자동화

2026.07.14 — 2026.08.12 · 6인 팀 · AI·백엔드 개발

98.98%
화물·파렛트 검출 성능

2.30mm
화물 치수 최대 오차

두 수치는 고정된 실험 환경에서 실물 화물로 검증한 결과입니다. 검출 성능은 독립 평가셋의 mAP@0.5이며, 치수 오차는 1/10 모형 기준으로 환산했습니다.



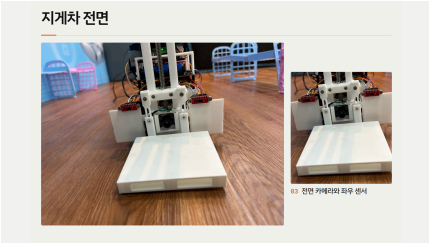
Isaac Sim 관제 화면 · 팀 시연 영상 캡처



화물 치수 측정 · 실험 화면



무인 지게차와 파렛트 · 축소 모형



지게차 전면 · 카메라와 포크

개요

소개	지게차에 달린 카메라로 입고된 화물을 촬영합니다. 카메라 영상과 거리 측정값으로 화물의 높이와 너비를 계산하고, 화물이 파렛트(화물 받침대) 위에서 한쪽으로 치우쳤는지도 확인합니다. 서버가 화물 크기에 맞는 빈 선반을 고르면 지게차가 그곳으로 화물을 옮깁니다. 운반과 적재는 실물의 1/10 크기로 만든 모형에서 검증했습니다.
맡은 일	카메라 보정, 화물 치수 계산, 오차 검증을 맡았습니다. AI 담당 팀원과 화물·파렛트 이미지를 라벨링하고 검출 모델을 학습시켰습니다. 서버 담당 팀원과는 측정 요청 처리, 작업 상태 관리, 적재 위치 추천을 구현했습니다.
사용 기술	**Python · PyTorch · RTMDet / MMDetection · ONNX Runtime · OpenCV · Java 21 · Spring Boot · MySQL**
링크	팀 GitHub · github.com/bjs0306b/fast-autonomous-forklift

무인 지게차 물류 자동화

문제 해결

01 플라스틱 파렛트 검출 개선

기존 모델은 실험 이미지 12장 모두에서 플라스틱 파렛트를 검출하지 못했습니다. 공개 학습 데이터는 목재 파렛트 위주였습니다.

AI 담당 팀원과 자체 촬영 이미지 289장에서 화물과 파렛트의 바운딩 박스를 라벨링해 기존 모델을 추가 학습했습니다. 학습에 사용한 이미지 중 12장을 골라 개선 전후를 비교했습니다. 추가 학습 후에는 11장에서, 저조도 이미지와 합성 데이터를 보강한 뒤에는 12장 모두에서 파렛트를 검출했습니다.

02 적재 형태를 반영한 독립 평가셋 구축

플라스틱 파렛트 검출을 개선한 뒤, 적재 형태가 달라져도 화물과 파렛트를 안정적으로 검출하는지 확인해야 했습니다. 모델이 놓친 객체까지 평가에 반영할 수 있도록 예측과 독립된 라벨을 만들었습니다.

별도로 촬영한 이미지에서 상자 수가 다른 30장을 골라 화물과 파렛트의 바운딩 박스와 클래스를 직접 라벨링했습니다. 이 평가셋을 기준으로 최종 모델의 mAP@0.5를 산출했습니다.

03 실물 상자로 치수 오차 검증

측정한 치수는 화물이 들어갈 선반을 고르는 데 쓰였습니다. 오차가 허용 범위에 드는지 확인해야 했지만, 축소 모형에서는 몇 mm 차이를 자로 구분하기 어려웠습니다.

실물 상자 3종을 거리와 방향을 바꿔 촬영했습니다. 카메라와 거리 센서로 계산한 치수를 자로 잰 값과 비교했습니다. 실측값과의 차이를 1/10 모형 기준으로 환산했습니다.

→ 53장 모두 허용 오차 4mm 이내 · 평균 오차 0.66mm (1/10 모형 기준)

04 동시 측정 요청의 세션 충돌 방지

서버는 화물의 측정 결과를 받아 선반 자리를 정합니다. 측정 요청이 동시에 들어와도 한 번에 하나의 측정 세션만 열도록 제어해야 했습니다.

활성 측정 세션이 있는지 확인하는 조건과 새 세션 ID를 기록하는 동작을 하나의 MySQL UPDATE 문에 담았습니다. 갱신에 성공한 작업만 측정할 수 있도록 했습니다. 서버는 활성 세션 ID와 일치하는 결과만 저장했습니다. 제한 시간(기본 60초) 안에 결과가 오지 않으면 해당 작업을 실패로 처리하고 다음 작업이 측정을 시작할 수 있게 했습니다.

회고

선반을 추천하려면 화물과 파렛트의 높이에 포크를 올릴 때 필요한 여유 높이까지 합산해야 했습니다. 측정값에 파렛트 높이를 포함하면 서버가 같은 높이를 한 번 더 더해 실제로 적재할 수 있는 선반도 후보에서 빠질 수 있었습니다. 서버 담당 팀원과 측정 API의 cargoHeight를 화물 자체의 높이(m)로 정의하고, 파렛트 높이와 여유 높이는 서버가 적재 가능 여부를 판단할 때 합산하도록 정했습니다. 파렛트 규격이나 적재 여유 기준이 바뀌어도 같은 실측값을 바탕으로 서버 설정에서 적재 기준을 조정할 수 있습니다.

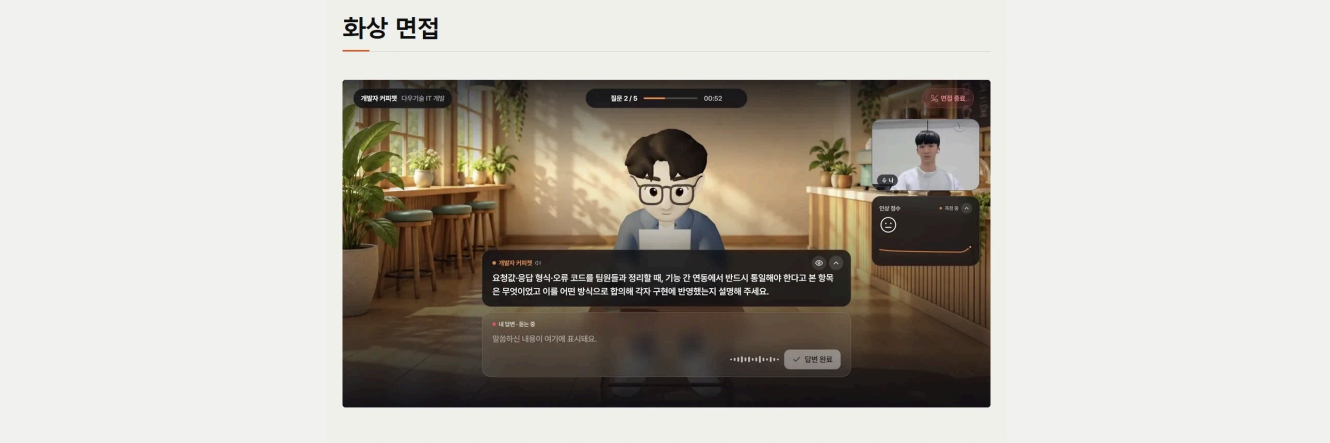
한꿋

2026.10 — 2026.11 · 5인 팀 · 백엔드·AI 개발

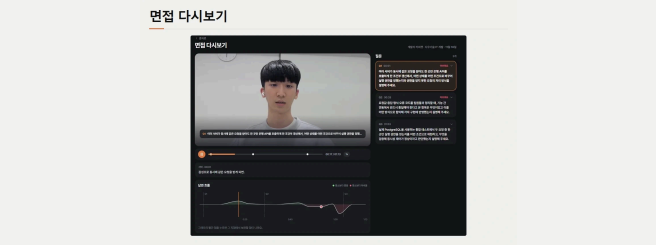
10.8% → 3.2%

면접 에이전트 CPU 사용률

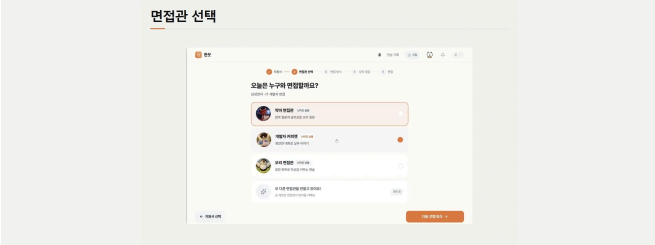
로컬 LiveKit 방에서 지원자의 답변을 수신할 때 측정한 에이전트 프로세스의 CPU 사용률입니다. 오디오 프레임 길이를 10ms에서 100ms로 바꾼 뒤, 측정 범위가 8.5–10.8%에서 3.2–3.4%로 낮아졌습니다.



화상 면접 · 실시간 리액션과 인상 점수



면접 다시보기 · 타임라인 피드백



면접관 선택 · 개발자 커피챗

개요

소개	자기소개서와 지원 직무를 입력하면 AI 면접관이 그 내용을 바탕으로 질문합니다. 화상 면접에서는 답변에 맞춰 면접관이 실시간으로 반응합니다. 면접을 마치면 답변별 피드백을 확인하고, 다시보기에서 보완할 장면으로 바로 이동할 수 있습니다.
맡은 일	지원자의 답변에 실시간으로 반응하는 가상 면접관을 제안하고 구현했습니다. LiveKit 면접방 전환과 음성 인식·합성 연동을 맡았고, Jev의 웹캠 분석 결과를 다시보기 피드백에 연결했습니다.
사용 기술	**Java 21 · Spring Boot · Python · FastAPI · WebRTC · LiveKit · Jev · Soniox · ElevenLabs**

한끗

문제 해결

01 실시간 리액션을 위한 모델 선정

가상 면접관이 답변 도중 실시간으로 반응하려면 면접 한 번에 모델을 여러 차례 호출해야 했습니다.

응답 지연과 반복 호출 비용을 고려해 문장 생성 없이 선택지별 확률을 한 번에 반환하는 System One 모델 Jev를 채택했습니다.

판단에 사용할 답변도 정확히 받아 적어야 했습니다. Whisper, Google STT, RTZR을 시험하면서 한국어 문장에 섞인 기술 용어가 잘못 인식되는 경우를 확인했습니다. 용어 사전을 미리 전달할 수 있는 Soniox를 발화 테스트로 검증하고, 사전의 용어와 표기를 보완한 뒤 인식 결과를 Jev에 연결했습니다.

02 실시간 음성 처리와 녹화를 LiveKit으로 통합

초기 설계에서는 브라우저가 STT와 TTS 서비스에 각각 연결하고, 녹화 파일은 S3에 직접 업로드하는 방식이었습니다. 하지만 실시간 음성 분석과 다시보기를 함께 제공하려면 AI 에이전트와 녹화 서비스가 같은 면접의 미디어 트랙을 사용할 수 있어야 했습니다.

브라우저와 AI 에이전트를 같은 LiveKit 면접방에 연결해 음성 처리와 녹화를 방 단위로 묶었습니다. 백엔드는 입장 토큰을 발급하고 에이전트를 해당 방에 배정했습니다. 에이전트는 마이크 트랙을 구독해 Soniox로 보내고, 질문 음성은 오디오 트랙으로, Jev의 리액션은 LiveKit RPC로 전달했습니다.

Egress는 같은 방에서 지원자의 영상·음성과 면접관 음성을 녹화했습니다. 이 구조에서 지원자의 마이크 트랙을 실시간 음성 인식과 다시보기에 함께 사용했습니다.

03 웹캠 분석 추가와 다시보기 시점 동기화

Jev의 이미지 입력을 활용해 면접 중 웹캠 프레임을 분석하고 인상 점수를 표시했습니다. 이미지 분석을 추가하면서도 실시간 리액션을 독립적으로 처리할 수 있어야 했습니다.

실시간 리액션과 웹캠 분석 요청을 별도의 HTTP 세션으로 분리했습니다. 웹캠 분석 결과에 촬영 시점을 함께 남기고, 이를 녹화 영상의 재생 위치로 변환했습니다. 피드백을 누르면 해당 장면으로 바로 이동하도록 연결했습니다.

04 오디오 프레임 처리 빈도 조정

오디오 처리 단계별로 CPU 사용률을 측정해 LiveKit에서 PCM 프레임을 수신하는 구간에 부하가 집중된 것을 확인했습니다.

마이크 수신과 TTS 송출의 프레임 길이를 10ms에서 100ms로 늘렸습니다. 두 경로 모두 SDK와 에이전트 사이의 프레임 전달 횟수가 초당 100회에서 10회로 줄었습니다.