# 조현우 · *AI Product Engineer*

프론트엔드 개발자로 커리어를 시작해 사용자 경험과 인터페이스 설계에 대한 감각을 쌓았고, 화면 너머 제품 전반의 문제로 관심을 넓히며 지금은 프론트엔드와 백엔드를 아우르는 Product Engineer로 성장하고 있습니다.

기획·디자인·엔지니어링이 맞닿는 지점에서 일하며, "왜 이 제품을 만드는가"에 답할 수 있는 엔지니어를 지향합니다. 사용자 관점에서 출발해 비즈니스 임팩트까지 이어지는 엔드투엔드 개발을 통해 제품의 가치를 직접 만들어내는 개발자로 커리어를 확장해 가고 있습니다.

gus007dn@gmail.com · https://github.com/starz-woo · https://www.linkedin.com/in/starz-woo · https://wigtn.com/

01 / CAREER

## 경력

2025.02 - 재직 중

### (주)사운드마인드 · Manager · Full-Stack 개발

B2B 사업

#### 오디야 2025.02 - 현재

GPS 기반 자녀 위치 추적 및 안심 관리 플랫폼

- Redis caching, batch 처리, DB partitioning/indexing 전략을 적용하여 분당 3,000~5,000건 규모의 위치 데이터 환경에서 발생하던 처리 병목 및 DB 부하 문제 개선
- Activity Recognition API 및 Android 센서를 활용해 실제 이동 시점에만 고정밀 위치 추적이 활성화되도록 최적화하고 배터리 사용량 개선
- 기존 웹 기반 보호자 시스템의 접근성 한계를 개선하기 위해 React Native 기반 보호자 앱 신규 개발
- 관리자 웹 대시보드를 구축하여 사용자 상태 및 위치 데이터 운영 모니터링 환경 개선
- 오디야 서비스 안정화 및 고도화 과정에 참여하여 B2B 사업 매출 약 230% 성장에 기여

#### 모하니 2026.01 - 현재

자녀 스마트폰 사용 관리 및 부모 제어 서비스

- FCM 기반 실시간 디바이스 제어 구조를 설계하여 보호자 앱의 제어 요청이 자녀 디바이스에 즉시 반영될 수 있는 환경 구축
- Android foreground/background 상태 차이로 발생하는 이벤트 누락 문제를 해결하기 위해 상태 재동기화 및 fallback 처리 로직 구현
- 앱 사용 제한, 수면 모드, 특정 콘텐츠 차단 등 부모 제어 기능 설계 및 구현
- Redis 기반 상태 캐싱 구조를 적용하여 실시간 상태 조회 및 제어 안정성 개선
- 앱별 사용 통계 집계 및 시각화 기능을 구축하여 사용자 사용 패턴 분석 환경 구성

#### 통합 인증 시스템 2026.01 - 현재

다중 서비스 대상 인증 및 사용자 관리 시스템

- 서비스별로 분산되어 있던 인증 구조를 통합하기 위해 공통 인증 시스템 설계 및 구축
- 보호자/피보호자 구조에 맞춰 역할 기반 인증 정책 및 토큰 관리 구조 설계
- 반복 인증 요청 및 비정상 트래픽 상황에서도 안정적으로 동작할 수 있도록 인증 서버 부하 제어 구조 개선
- Blue-Green 기반 무중단 배포 환경을 구축하여 운영 중 서비스 중단 없이 배포 가능한 구조 구성
- Prometheus / Grafana / Loki 기반 메트릭·로그 통합 모니터링 환경을 구축하여 장애 원인 추적 및 운영 가시성 개선

R&D 과제

**KOCCA 한국어 말하기 평가 플랫폼** 2025.11 - 2025.12

한국어능력시험 말하기 평가 및 관리 시스템

- 연세대학교 한국어능력시험 환경에서 실제 120명의 평가자 및 20명의 채점자가 동시에 사용하는 말하기 평가 플랫폼 구축
- 다수 평가자가 동시에 접속하는 환경에서 발생할 수 있는 평가 상태 충돌 및 중복 채점 문제를 해결하기 위해 단계별 평가 workflow 및 상태 관리 구조 설계
- 평가 진행 중 네트워크 불안정 및 브라우저 이탈 상황에서도 평가 데이터 유실을 최소화할 수 있도록 임시 저장 및 상태 복구 로직 구현
- 대용량 음성 파일 처리 과정에서 발생하는 업로드·재생 지연 문제를 개선하기 위해 AWS S3 기반 저장 구조 및 Wavesurfer.js 기반 스트리밍형 음성 파형 시각화 환경 구축
- STT 기반 음성 처리 기능을 도입하여 채점 및 검수 과정의 운영 효율 개선
- 평가 루브릭 기반 결과 데이터 분석 기능을 구축하여 평가 결과 관리 및 운영 가시성 향상

---

2021.07 - 2021.08

**네오젠소프트** · 학부 연계 인턴

학부 연계 인턴십으로 기존 Android 솔루션의 현대화 작업과 백엔드 연동 실무를 경험했습니다.

- 기존 Java 기반 Android Application을 Kotlin으로 마이그레이션
- MVVM 아키텍처 패턴 학습 및 적용
- SOAP 웹 서비스와 Oracle Database 연동을 직접 구현하며 클라이언트–백엔드 통신 흐름 학습

---

EDUCATION

2016 - 2022

**광운대학교** · 컴퓨터정보공학과

학점 3.77 / 4.5

졸업 연구

**딥러닝을 통한 생체 신호 분석 및 건강 지표 모니터링 시스템**

BCG(Ballistocardiogram, 심탄도), ECG(Electrocardiogram, 심전도), PPG(Photoplethysmogram, 광혈류측정) 신호를 이용한 혈압 측정 딥러닝 모델 연구 및 설계

02 / SKILLS

## 다뤄본 도구와 기술

### LANGUAGE

JavaScript TypeScript Python Kotlin Java

### FRONTEND

React React Native Next.js HTML CSS Tailwind CSS

### BACKEND

Node.js Spring Boot FastAPI

### DATABASE

MySQL Supabase Firebase

### DEVOPS & TOOLS

Git GitHub Docker Vercel Expo

03 / AWARDS & PUBLICATIONS

## 수상 & 논문

2026

논문 채택 **ACL 2026 System Demonstrations**

WIGVO: Real-Time Bidirectional Speech Translation over Legacy PSTN Calls via Dual-Session Echo Gating

2026

대상 **Build with TRAE Seoul Hackathon 2026**

WIGENT — AI 멀티 에이전트 토론 플랫폼

2026

준우승 **Snowflake AI & Data Hackathon 2026 Korea — Final Round**

WIGTN FLAKE — 목적 기반 동네 인텔리전스

2019

Dean's List **광운대학교 컴퓨터정보공학과**

04 / AI PROJECTS

## WIGTN 팀으로 만든 AI 프로젝트

01

### WIGVO AI 실시간 PSTN 전화 통역

2026.02 - Present · 팀 개발 · AI · OpenAI Realtime API (GPT-4o) · GPT-4o-mini · Whisper

ACL 2026 System Demonstration 채택

공중전화망(PSTN) 위에서 양방향 LLM 음성 번역을 수행하는 서버사이드 릴레이. 발신자는 브라우저에서 말하거나 타이핑하고, 수신자는 일반 전화를 받기만 하면 됩니다 — 앱 설치, 통신사 연동, 전용 하드웨어가 모두 불필요합니다.

Python FastAPI Next.js 16 React 19 React Native (Expo) Twilio Supabase Docker Cloud Run

HIGHLIGHTS

- OpenAI Realtime 듀얼 세션 + 3단계 에코 게이팅(Echo Gate → RMS Energy → Silero VAD)으로 162건 PSTN 통화에서 자기 강화 번역 루프 0건 달성
- Session A E2E 중앙값 557ms (P95 1156ms) — 인터랙티브 통신 임계값 내
- 분당 비용 $0.27, 전문 전화 통역 서비스($1–3/분) 대비 5–10배 저렴
- Voice→Voice / Text→Voice / Full-Agent 3개 모드를 Strategy 패턴으로 분기 — 콜포비아·언어 장애·외국인 등 다중 페르소나 커버
- Silero VAD 비대칭 히스테리시스(시작 96ms / 종료 480ms)로 speech_stopped 레이턴시 15–72초 → 480ms 단축
- OpenAI Realtime 세션 끊김 대비 30초 링 버퍼 + Whisper 일괄 재주입 기반 복구, 10초 실패 시 Degraded 모드(STT+Chat) 폴백
- 430+ pytest 테스트(단위/통합/E2E) — Twilio 전화 자동 발신 시나리오 포함

> 기존 스트리밍 음성 번역 시스템(SeamlessM4T, Moshi 등)은 광대역 오디오와 클라이언트측 음향 에코 제거(AEC)를 전제합니다. 그러나 PSTN은 여전히 병원·식당·관공서의 주요 인바운드 채널이며 G.711 μ-law 8kHz 코덱, 80–600ms 지연, AEC 부재라는 제약을 가집니다. 재한 외국인 220만, 콜포비아 Gen-Z 약 400만 등 명확한 수요가 있음에도 소프트웨어만으로 이 문제를 해결한 시스템은 존재하지 않았습니다.
>
> WIGVO는 두 개의 독립적인 OpenAI Realtime 세션을 운영합니다. Session A는 브라우저 PCM16 오디오를 G.711로 변환해 Twilio에 송출하고, Session B는 PSTN G.711 8kHz를 받아 번역된 오디오/자막을 브라우저로 돌려보냅니다. AudioRouter가 Strategy 패턴으로 V2V·T2V·Full-Agent 파이프라인에 이벤트를 분기하며, T2V/Agent 모드의 번역은 GPT-4o-mini Chat Completion(temperature=0)에 위임해 생성적 할루시네이션을 차단합니다.
>
> 단순 단일 세션 설계에서 가장 치명적이었던 문제는 자기 강화 번역 루프였습니다. 수신자에게 재생된 TTS가 80–600ms 후 PSTN을 타고 다시 들어와 번역 사이클을 무한 반복하는 현상으로, 게이팅 없이 테스트 통화 10건 중 8건에서 발생했습니다. Echo Gate는 Session A가 TTS 스트리밍을 시작하면 동적 쿨다운(TTS 길이 + 0.3초 마진) 동안 수신 PSTN 오디오를 μ-law 무음(0xFF)으로 대체합니다. 뒤이은 RMS Energy Gate(에코 윈도우 ≥400 RMS만 통과)와 Silero VAD가 잔여 잡음을 걸러 162건 프로덕션 통화에서 에코 루프 0건을 달성했습니다.
>
> 또한 3단계 가드레일(L1 통과 / L2 반말 백그라운드 교정 / L3 욕설·유해 콘텐츠 차단 + 필러 음성)로 95%+ 케이스에서 추가 지연 0ms를 유지했습니다. 평가 결과 Session A E2E 중앙값 557ms로 인터랙티브 통신 범위 내, Session B는 STT가 87%를 차지해 중앙값 2.9초(전문 동시통역 ear-voice span 2–5초의 하한)였으며, 분당 비용은 $0.27로 전문 통역 서비스 대비 한 자릿수 저렴합니다. ACL 2026 System Demonstrations에 'WIGVO: Real-Time Bidirectional Speech Translation over Legacy PSTN Calls via Dual-Session Echo Gating'으로 채택되었습니다.

**GitHub** https://github.com/wigtn/wigvo-v2 · **Live Demo** https://wigvo.run · **Demo Video** https://youtu.be/_ixVEnHJxjk

02

## WIGTN Coding Claude Code 멀티 에이전트 플러그인

2026.01 - Present · 팀 개발 · AI · Claude (Opus / Sonnet)

44 stars

기획부터 PR까지의 개발 사이클 전체를 12개의 전문 에이전트로 병렬 자동화하는 Claude Code 플러그인. /prd → /implement → /auto-commit 한 줄로 PRD 생성, 4카테고리 품질 분석, 아키텍처 결정, 팀 단위 병렬 빌드, 3-에이전트 코드 리뷰, 품질 게이트까지 처리합니다.

Claude Code Plugins | Anthropic Claude | Markdown | YAML frontmatter | Bash (hooks)

### HIGHLIGHTS

- /prd → /implement → /auto-commit 3-스테이지 파이프라인 — 순차 ~20분 → 병렬 ~6분 (3~5배 단축)
- 12개 전문 에이전트 (코디네이터 4 · 개발자 4 · 품질 4)를 동적 팀 분배해 동시 실행
- 20개 디자인 스타일 가이드 + VS(Verbalized Sampling) 기반 design-discovery 에이전트로 컨텍스트 맞춤 추천
- /auto-commit 3-에이전트 리뷰(가독성·성능·보안) → 100점 게이트, ≥80 자동 커밋 / 60–79 수정 / <60 차단, 보안 치명적은 강제 FAIL
- Project-Native Build — 코드 작성 전 Context Harvesting으로 기존 패턴을 자동 수집, generic best practice 대신 프로젝트 실제 컨벤션 준수
- team-memory-protocol 스킬로 SHARED_CONTEXT 파일 + TaskCreate + Auto Memory 3-레이어 공유 메모리 운영
- 4개 훅으로 위험 명령(rm -rf /, git push --force, DROP TABLE) 차단 + .tsx/.py/.go 파일 포맷팅·패턴 자동 검증

> Claude Code 단일 컨텍스트로 중대형 기능을 만들면 컨텍스트 오염, 패턴 일관성 부재, 순차 실행으로 인한 시간 손실이 발생합니다. Vibe coder는 PRD 없이 바로 코딩에 들어가 누락 요건과 보안 갭을 만들고, 시니어는 매번 같은 절차(요구사항 → 아키텍처 결정 → 빌드 → 리뷰 → 커밋)를 반복합니다. WIGTN Coding은 이 사이클 전체를 명시적 파이프라인 + 병렬 에이전트로 자동화합니다.
>
> /prd는 parallel-digging-coordinator를 통해 4-카테고리(완전성·실현가능성·보안·일관성) 갭 분석을 병렬 수행해 PRD.md와 PLAN.md를 생성합니다. /implement는 architecture-decision(MSA / 모놀리식 / 모듈러 모놀리스)을 결정한 뒤 design-discovery로 디자인 스타일을 선정하고, team-build-coordinator가 백엔드 · 프론트엔드 · AI 서버 · 운영 팀에 작업을 동적 분배합니다. 각 팀은 독립된 서브에이전트로 동시에 빌드합니다.
>
> /auto-commit은 parallel-review-coordinator가 3개 리뷰 에이전트(가독성·성능·보안)를 병렬 실행해 점수를 병합합니다. code-reviewer는 5개 카테고리(가독성·유지보수성·성능·테스터빌리티·베스트 프랙티스)를 100점 만점으로 평가하며, ≥80은 자동 커밋, 60–79는 수정 후 재시도, <60은 차단됩니다. 보안 치명적 이슈는 점수와 무관하게 zero-tolerance로 FAIL 처리됩니다.
>
> 핵심 원칙은 'Project-Native' — 모든 에이전트는 코드 작성 전 Context Harvesting으로 프로젝트의 실제 패턴(폴더 구조 · 네이밍 컨벤션 · 에러 핸들링 방식 · 테스트 스타일)을 스캔하고, team-memory-protocol 스킬을 통해 SHARED_CONTEXT 파일 + Claude Code TaskCreate + Auto Memory 3-레이어 공유 메모리에 기록합니다. 결과적으로 generic best practice가 아닌 이 코드베이스의 실제 컨벤션 위에서 일관된 코드가 생성됩니다.

**GitHub** https://github.com/wigtn/wigtn-plugins-with-claude-code

03

## WIGENT Agent Arena — AI 토론 → 랜딩 페이지 자동 생성

2026.03.28 · 팀 개발 · AI · GPT-4o

Build with TRAE Seoul Hackathon 2026 대상

주제만 던지면 PM 에이전트가 도메인 전문가들을 자동 소환해 멀티턴 토론을 진행하고, 결론이 나면 그 토론 내용으로 풀 랜딩 페이지 HTML까지 자동 생성하는 멀티 에이전트 디베이트 플랫폼. Build with TRAE Seoul Hackathon 2026 대상 수상작.

Next.js 16 (App Router) | React 19 | TypeScript 5.9 | Tailwind CSS v4 | Framer Motion | Archiver | OpenAI GPT-4o

### HIGHLIGHTS

- 주제 기반 에이전트 동적 스폰 — PM 에이전트(앵커) + 주제별 도메인 전문가(마케터·테크 리드·UX 리서처 등)를 매 세션 자동 소환, 라운드마다 은퇴·교체로 토론이 진화하는 만큼 멤버 구성도 진화
- Slack 스타일 실시간 토론 UI — 타이핑 인디케이터, 입장·퇴장 이벤트, 라운드 사이 분석 단계까지 채팅 메타포로 자연스럽게 시각화
- 토론 → 랜딩 페이지 트랜지션 — 결론 라운드의 구조화된 결과를 입력으로 GPT-4o가 풀 HTML 랜딩 페이지를 생성, 채팅 뷰가 애니메이션으로 페이지로 스왑
- 7+ 멀티 에이전트 디자인 패턴 — Orchestrator / Specialist / Persistent Anchor (PM) / Spawning / Retirement / Multi-turn Debate / Result Synthesis를 한 제품 안에서 시연
- 토론 히스토리 브라우징 + Archiver 기반 ZIP 익스포트로 생성된 랜딩 페이지를 그대로 다운로드 가능
- 프리셋 토픽으로 빠른 시작 + 자유 입력 토픽 모두 지원, 단일 페이지 앱(`page.tsx`)에서 입력 → 토론 → 랜딩까지 한 번에 흐름

> Build with TRAE Seoul Hackathon 2026 대상 수상작. '에이전트끼리 토론하면 더 나은 결과가 나오는가'라는 가설을 가장 직관적인 형태로 시연하는 데모를 목표로 했습니다 — 사용자는 토픽 한 줄만 던지면, AI들이 토론하는 모습을 실시간으로 보고, 결론이 곧바로 동작하는 랜딩 페이지로 변신하는 경험을 합니다.
>
> 에이전트 구성은 두 축 — 항상 존재하는 PM 에이전트(앵커 역할로 토픽에서 벗어나지 않게 가드레일)와, 주제에 따라 매번 다르게 자동 생성되는 도메인 전문가들. 라운드가 끝날 때마다 Orchestrator가 토론 흐름을 분석해 더 이상 적합하지 않은 에이전트를 핸드오프 메시지와 함께 은퇴시키고, 새로운 전문가를 스폰합니다. 결과적으로 같은 토픽이라도 토론이 진행되면서 멤버 구성이 진화합니다.
>
> UI는 Slack을 메타포로 — 채팅 타이핑 인디케이터, 멤버 입장/퇴장 이벤트, 라운드 사이의 분석 단계까지 실시간 채팅 경험으로 자연스럽게 표현했습니다. 마지막 라운드에서는 Result Synthesis 패턴으로 토론 결과를 구조화된 데이터로 정리하고, 이 데이터를 GPT-4o에 다시 넘겨 풀 HTML 랜딩 페이지를 생성합니다. 채팅 뷰는 Framer Motion 애니메이션으로 페이지로 매끄럽게 스왑됩니다.
>
> 한 제품 안에서 7개 이상의 멀티 에이전트 디자인 패턴(Orchestrator·Specialist·Persistent Anchor·Spawning·Retirement·Multi-turn Debate·Result Synthesis)을 모두 시연한다는 점이 심사에서 좋게 평가됐습니다. 결과물은 토론 히스토리에 자동 저장되어 다시 열람할 수 있고, Archiver로 ZIP 다운로드해 그대로 배포 가능한 정적 페이지로 내보낼 수 있습니다.

**GitHub** https://github.com/wigtn/wigent

04

## WIGTN FLAKE 목적 기반 동네 인텔리전스

2026.03 - 2026.04 · 팀 개발 (4인) · AI · Snowflake Cortex (claude-4-sonnet) · GPT-4o

Snowflake AI & Data Hackathon 2026 Korea Final Round 준우승

사용자가 '카페 창업 / 렌탈 가전 타겟 / 광고판 입지 / 부동산 투자 / 상권 이상 감지' 같은 목적을 선택하면, 목적에 맞는 AI 전문가 에이전트가 동적으로 소환되어 Snowflake Cortex 기반으로 부동산 시세 × 유동인구 × 카드매출 × 통신계약 4개 데이터셋을 교차 분석하고, Top 3 동네 추천 + 이상 시그널 감지 + 6개월 예측 + 실행 액션을 자동 생성합니다.

Next.js 16 | React 19 (Compiler) | TypeScript 5.9 | Tailwind CSS 4 | Framer Motion | Vega-Lite | snowflake-sdk | Snowflake Cortex | OpenAI SDK

### HIGHLIGHTS

- Purpose-first UX — 토픽이 아니라 '무엇을 하고 싶은지'에서 시작하는 5개 프리셋(창업·렌탈·광고·투자·이상감지) + 자유 입력. 목적 컨텍스트가 토론·랭킹·리포트 전 단계에 주입됨
- 목적 기반 에이전트 동적 생성 — 고정 멤버는 PM 진행자 + 데이터 분석가뿐. 선택된 목적에 맞춰 도메인 전문가와 Cortex 분석가(Forecast / Insight / Sentiment / News / Anomaly)가 매 세션 다르게 소환되어 같은 데이터도 다른 관점으로 해석
- Snowflake Cortex 11개 기능 자연스러운 통합 — Cortex Agent · Analyst×4 · LLM(claude-4-sonnet) · FORECAST · ANOMALY_DETECTION · AI_SENTIMENT · AI_CLASSIFY · data_to_chart · Dynamic Tables×2 · Python UDF×2 · Semantic Model YAML×4
- ANOMALY_DETECTION 주연 승격 — 랭킹 결과에 '지금 이상 시그널 발견' 자동 주입으로 데모 클라이맥스 설계, 카페 매출 +18% 급성장·국지적 -12% 하락 등 실데이터 시연
- 3-Tier 폴백 아키텍처 — Cortex Agent → Cortex Analyst 직접 호출 → GPT-4o Function Calling(4 도구) → GPT-4o 순수 추론. trial 제약·토큰 붕괴를 모든 계층에서 그레이스풀 처리
- 하이브리드 모델 전략 — 데이터 분석은 Cortex Agent, 리포트 생성은 Cortex LLM(17.3s/1657자 벤치), 토론 페르소나는 GPT-4o. `hasGarbageTokens()` 스트리밍 검사로 Cortex 토큰 붕괴 시 즉시 GPT-4o 폴백
- 4개 데이터셋 전수 활용 — SPH(SKT 유동인구+신한카드+KCB 자산) / RichGo(아파트 시세 2012~2024) / NextTrade(주식 실시간) / 아정당(통신·GA4·콜센터). 각각 Semantic Model YAML로 Cortex Analyst 연결
- Slack 스타일 SSE 실시간 토론방 → Top 3 랭킹 카드 + Vega-Lite FORECAST 차트 + 이상 시그널 배지 + 목적별 액션 체크리스트로 자연스러운 의사결정 흐름

> Snowflake AI & Data Hackathon 2026 Tech Track 출품작. 일반적인 '데이터 대시보드'가 아니라, 의사결정자(소상공인·B2B 마케터·프랜차이즈·개인 투자자·상권 운영자)가 자기 목적을 직접 선택하면 '어느 동네가 그 목적에 맞는지'를 데이터로 답해 주는 도구를 목표로 했습니다. 같은 쿼리라도 목적이 다르면 해석이 달라진다는 점이 핵심 가설.
>
> 오케스트레이션은 GPT-4o 진행자(PM 에이전트)가 목적 컨텍스트를 받아 고정 멤버(데이터 분석가) + 동적 전문가(목적 도메인 + Cortex Forecast / Insight / Sentiment / News / Anomaly)를 소환하고, Slack 스타일 토론방에서 SSE로 발언이 스트리밍됩니다. Cortex Agent가 Semantic Model 4개를 활용해 자연어→SQL을 수행하고, FORECAST는 Top 3 대상 6개월 예측, ANOMALY_DETECTION은 랭킹 결과에 이상 시그널을 자동 주입합니다.
>
> 신뢰성은 3-Tier 폴백으로 확보 — Tier 1 Cortex Agent(Analyst×4 + data_to_chart), trial 제약 시 Cortex Analyst 직접 호출로 하강. Tier 2 GPT-4o Function Calling(execute_snowflake_sql / web_search Tavily / real_estate_transaction 국토부 / statistical_analysis), Tier 3 GPT-4o 순수 추론. 또한 Cortex LLM이 가비지 토큰(`<|reserved_special_token|>`, `Dünnschicht` 등)을 흘릴 경우 `hasGarbageTokens()`가 스트리밍 중 감지해 즉시 GPT-4o로 전환합니다.
>
> 데이터 측면은 SPH·RichGo·NextTrade·아정당 4개 데이터셋을 모두 Semantic Model YAML로 정의해 Cortex Analyst가 단일 자연어 입력으로 4 도메인을 교차 조회하도록 했고, Dynamic Tables(`DT_DISTRICT_HEALTH`, `DT_DISTRICT_DNA`)와 Python UDF(디커플링 지수·DNA 유사도·목적 적합도)로 의사결정 점수를 계산합니다. 결과적으로 단순 추천이 아니라 '왜 1위인가'를 데이터로 논쟁하는 토론 + 이상 시그널이 결합된 데모로 Final Round 준우승을 수상했습니다.

**GitHub** https://github.com/wigtn/wigtn-for-snowflake · **Demo Video** https://www.youtube.com/watch?v=BHwb6zbUZuE

05

## TimeLens AI 문화유산 컴패니언

2026.03 - Present · 팀 개발 · AI · Gemini Live API · Gemini 2.5 Flash · Gemini 3 Pro Image

카메라를 유물에 비추면 Gemini Live 큐레이터가 음성으로 역사적 맥락을 설명하고, 손상된 유물의 복원 이미지와 일러스트 방문 다이어리까지 실시간 생성하는 박물관 컴패니언 앱. Gemini Live Agent Challenge 출품작.

Next.js 15 | React 19 | TypeScript 5 | Tailwind CSS 4 | Google ADK | @google/genai | Firebase Firestore | Google Places API | Cloud Run (Seoul) | Docker

HIGHLIGHTS

- Gemini Live API(`gemini-2.5-flash-native-audio`) WebSocket 세션으로 PCM16 16kHz 마이크 + 1fps 카메라 프레임을 동시 스트리밍 — 별도 인텐트 분류기 없이 모델이 4개 Function Declaration으로 의도 라우팅
- 듀얼 파이프라인 — Live(스트리밍 음성·영상) + REST(이미지 생성·외부 API) 분리로 저지연과 무거운 작업의 책임 분리
- `recognize_artifact`만 Live 세션 내부에서 Google Search Grounding과 함께 처리, `generate_restoration` / `discover_nearby` / `create_diary`는 REST 엔드포인트로 위임
- Google GenAI SDK + ADK 이중 경로 — WebSocket 가용 시 Live 메인, 폴백은 ADK `LlmAgent` 5개(orchestrator + curator/restoration/discovery/diary)로 텍스트 에이전트 오케스트레이션
- Firebase Auth(익명) + Firestore에 세션·다이어리 저장, Google Places API(New)로 GPS 기반 박물관 검색 및 주변 문화유산 추천
- Cloud Run(asia-northeast3 서울) + GitHub Actions 자동 배포, Ephemeral Token 발급 라우트로 클라이언트 키 노출 없이 Gemini 세션 생성

> TimeLens는 박물관 방문이라는 짧은 문맥 안에서 음성 대화·시각 인식·이미지 생성·기록 보존을 하나의 흐름으로 연결합니다. 사용자는 박물관을 선택해 세션을 열고(Search Grounding으로 오늘의 전시까지 반영된 인사를 받음), 유물을 카메라로 비추며 자연스럽게 질문하면 AI 큐레이터가 시대·문명·맥락을 설명하고, '원래 모습 보여줘'로 복원 이미지를, '다이어리 만들어줘'로 일러스트 방문 기록을 즉석에서 만들어 줍니다.
>
> 핵심 아키텍처는 듀얼 파이프라인입니다. 파이프라인 1은 앱 사용 내내 열려 있는 Gemini Live WebSocket 세션 — 마이크 PCM16/16kHz 오디오와 카메라 JPEG/1fps 프레임이 동시 송출되고, 모델이 실시간 음성 응답과 Function Call을 반환합니다. 파이프라인 2는 이미지 생성·Places 호출 같은 무거운 작업을 처리하는 Next.js API 라우트로, Function Call이 이 라우트를 트리거합니다. 인텐트 분류기를 별도로 두지 않고 모델의 도구 선택 능력에 라우팅을 위임한 것이 핵심 결정.
>
> Function Declaration은 4개 — `recognize_artifact`만 카메라 프레임이 이미 스트리밍 중이라는 점을 활용해 Live 세션 안에서 Google Search Grounding과 함께 처리하고(REST 호출 없음), 나머지 3개(`generate_restoration` → Gemini 2.5 Flash Image, `discover_nearby` → Places API, `create_diary` → Gemini 3 Pro Image + Firestore)는 REST로 위임됩니다. WebSocket을 사용할 수 없는 환경에서는 `@google/adk` 기반의 텍스트 에이전트 오케스트레이션(orchestrator + 4 전문가)로 폴백되며, 두 경로 모두 동일한 백엔드 API를 공유합니다.
>
> 프로덕션은 Cloud Run 서울 리전에 GitHub Actions로 자동 배포되고, 클라이언트는 직접 Gemini 키를 들고 있지 않습니다 — `POST /api/session`이 Ephemeral Token을 발급해 보안과 비용 통제를 동시에 잡습니다. Firebase Firestore에는 세션·방문 기록·공유 가능한 다이어리가 저장되며, Firebase 키 없이도 앱은 정상 동작하도록 그레이스풀 디그레이드 처리됐습니다.

**GitHub** https://github.com/wigtn/wigtn-timelens · **Demo Video** https://youtu.be/ITaMtVO5jFg

06

## WIGSS Style Shaper — Visual AI Code Refactoring

2026.03 - Present · 팀 개발 · AI · GPT-4o (function calling) + Direct Tailwind Mapping

실제로 떠 있는 dev 서버 위에 항시 연결된 AI 에이전트를 띄워, UI 컴포넌트를 드래그·리사이즈하면 소스의 Tailwind 클래스가 결정론적으로 수정되는 비주얼 리팩토링 도구. `npx wigss --port 3000` 한 줄로 실행되는 npm 패키지(Trae.ai 해커톤 2026 출품작).

TypeScript | Next.js (App Router) | ws (WebSocket) | OpenAI GPT-4o | Zustand | Tailwind CSS | commander (CLI) | Node fs | pnpm Workspaces | Vitest

### HIGHLIGHTS

- 5가지 자율 에이전트 행동 — 컴포넌트 자동 인식 / 디자인 제안(신뢰도 점수) / 실시간 편집 피드백 / 채팅 상담 / 코드 리팩토링 — 도구가 아닌 항시 연결 에이전트로 동작
- WebSocket 기반 양방향 스트리밍(`ws-server.ts` + mutex 큐) — 드래그/리사이즈 이벤트가 실시간으로 GPT-4o에 흘러가고 'X px 차이납니다, 맞출까요?' 같은 능동적 제안이 다시 흘러옴
- AI 없는 결정론적 리팩토링 — Save 시 `fullClassName` 기반 줄 단위 매칭 + Tailwind 토큰 매핑(TW_MAP)으로 diff 생성, fs로 직접 적용. 코드 변경의 안정성과 예측 가능성을 AI 외부에 둠
- iframe-overlay 비주얼 에디터 — dev 서버를 iframe으로 감싸고 오버레이에서 컴포넌트 박스를 렌더, 30fps 스로틀로 라이브 미리보기 + Undo/Redo 50단계 히스토리
- 보안 가드 — origin 검증, rate limit, 경로 순회 공격 방지, diff 검증(className/style만, JS 변경 거부), `.bak` 백업 자동 생성으로 사용자 코드 보호
- Zero-install 배포 — `npx wigss@latest --port 3000` 한 줄로 즉시 실행, `--demo` 플래그로 기존 프로젝트 없이도 체험 가능, 키 미제공 시 대화형 입력
- npm 패키지 공식 배포 (`wigss`) + Apache 2.0 라이선스, Vitest 단위 테스트 포함

> Trae.ai 해커톤 2026 ('에이전트' 주제) 출품작. 출발점은 코딩 에이전트(Cursor/Claude Code/Trae)로 화면을 대충 만드는 건 잘 되지만 '디자인을 다듬는 단계'는 여전히 고통이라는 관찰이었습니다 — '이 카드를 좀 더 크게'를 정확히 말로 옮기기 어렵고, CSS 수정 → 새로고침 → 확인 루프는 느립니다. WIGSS는 Figma 없이, CSS 수동 수정 없이, 그냥 화면에서 드래그하면 소스가 바뀌는 경험을 목표로 합니다.
>
> 핵심 아키텍처는 두 개의 구분 — '제안과 대화'는 AI(GPT-4o function calling)에 맡기되, '실제 코드 변경'은 결정론적 매핑(`refactor-client.ts`)으로 처리합니다. 드래그 후 Save를 누르면 `fullClassName`을 키로 소스 줄을 매칭하고 Tailwind 토큰 변환표(TW_MAP)로 px↔클래스를 양방향 변환해 CodeDiff[]를 생성합니다. 이 분리 덕분에 AI가 환각해도 사용자 파일이 망가지지 않습니다.
>
> 통신 계층은 항시 연결 WebSocket(`ws` 포트 4001) + mutex 큐로 동시성 안전 — 일반 AI 도구가 'request → response' 모델이라면 WIGSS는 항시 연결 + 능동적 개입에 가깝습니다. 드래그/리사이즈 이벤트가 30fps로 스로틀되어 흐르고, GPT-4o가 'X px 차이납니다, 맞출까요?' 같은 정렬 보정 제안을 실시간으로 돌려줍니다. Editor/Agent 두 개의 Zustand 스토어로 상태가 분리되어 있고, 50단계 Undo/Redo 히스토리를 지원합니다.
>
> 운영 안전성은 다층 가드 — origin 검증, rate limit, 경로 순회 공격 방지, diff 검증(className/style만 허용·JS 변경 거부), `.bak` 자동 백업으로 사용자 코드 변경을 안전 범위 안에 가둡니다. 배포는 `npx wigss --port 3000` 한 줄 실행을 목표로 commander 기반 CLI 진입점과 Next.js instrumentation으로 부팅 시 WS 서버 + 에이전트를 자동 기동합니다. `--demo`로 데모 페이지 즉시 체험 가능, npm 패키지로 정식 배포됐습니다.

**GitHub** https://github.com/wigtn/wigss · **npm** https://www.npmjs.com/package/wigss

07

## WIGEX 해외여행 가계부 (개발 중)

2026.03 - In Development · 팀 개발 · AI · wigtn-coding (Claude Code) · OCR API

해외 여행 중 지출을 18개 통화 실시간 환율로 즉시 기록하고, 영수증을 OCR로 자동 입력하며, 오프라인에서도 동작하는 가계부 앱. Mobile · API · Admin · Shared 4개 워크스페이스를 Monorepo + MSA 패턴으로 묶어 동일 저장소에서 통합 관리합니다.

React Native 0.81 | Expo 54 | Expo Router 6 | NestJS 10 | Next.js 15 | React 19 | TypeScript 5.9 | Prisma 6 | PostgreSQL 16 | AWS SQS | Zustand 5 | Tailwind CSS | pnpm Workspaces | Turbo | Docker

### HIGHLIGHTS

- Monorepo + MSA 패턴 — apps/mobile (Expo), apps/api (NestJS), apps/admin (Next.js), packages/shared(공용 타입·상수) 4 워크스페이스를 pnpm Workspaces + Turbo로 빌드 그래프 관리
- 오프라인 우선 모바일 — expo-sqlite 로컬 DB에 지출을 즉시 기록하고, /sync/push & /sync/pull 엔드포인트로 서버와 양방향 동기화 (네트워크 복구 시 자동 머지)
- 영수증 AI OCR — apps/api의 ai 모듈이 영수증 사진을 받아 항목·금액·통화를 자동 추출, AWS SQS로 비동기 큐잉해 모바일 응답 지연 방지
- 18개 통화 실시간 환율 환산 — exchange-rate 모듈에서 환율 캐싱, 식비·교통·쇼핑·숙박·관광·기타 6 카테고리로 통계·캘린더·파이차트 자동 집계
- JWT 기반 Passport.js 인증 + class-validator 입력 검증 + Swagger(OpenAPI) 문서 자동 생성, expo-secure-store로 모바일 토큰 안전 저장
- 관리자 대시보드 — Next.js 15 + Radix UI + Recharts로 회원/AI 사용량/API 트래픽/시스템 상태/설정 5개 패널 운영
- Docker Compose 환경 분리(local/dev/prod) + GitHub Actions CI/CD로 모노레포 단일 파이프라인 자동 배포

> 현재 활발히 개발 중인 프로젝트입니다. 해외 여행 중 가장 마찰이 큰 두 지점 — '환율 계산이 귀찮아서 기록을 미루다 까먹는' 경험과 '영수증을 한 장씩 수동 입력하는 피로' — 를 모바일 환경에서 한 번에 해소하는 것이 목표입니다. 18개 통화 실시간 환율과 영수증 OCR 자동 입력으로 기록 마찰을 최소화하고, 오프라인 우선 설계로 비행기·해외 데이터 환경에서도 안정적으로 동작하도록 만들고 있습니다.
>
> 코드 구조는 의도적으로 Monorepo + MSA 패턴 — 모놀리식이 아니라 각 서비스(Mobile/API/Admin)가 독립적으로 배포 가능한 MSA이지만, packages/shared로 공용 타입과 상수를 통합 관리해 'Single Source of Truth + Atomic Change' 두 이점을 동시에 노립니다. 모바일에서 정의한 Expense 타입을 API와 Admin이 그대로 import하므로 한 번의 PR로 3개 앱이 동기화됩니다.
>
> 백엔드는 NestJS 10 + Prisma 6 + PostgreSQL 16로 도메인 모듈(auth · user · trip · expense · wallet · exchange-rate · ai · sync · queue · health)을 분리했고, AI OCR과 같은 무거운 작업은 AWS SQS로 비동기 큐잉해 모바일 응답 시간을 지키도록 설계했습니다. 모바일은 Expo Router 6 파일 기반 라우팅 + Zustand 5 상태 관리 + expo-sqlite 로컬 DB로 오프라인 캐시 + 동기화 흐름을 단순화했습니다.
>
> 운영 측면에서는 Docker Compose 환경 분리(local/dev/prod)와 GitHub Actions로 단일 파이프라인을 구성, Swagger 자동 문서화와 Admin 대시보드(회원·AI 사용량·트래픽·시스템 상태)로 관측성을 확보합니다. 현재 핵심 흐름(여행 생성 → 지출 기록 → OCR → 통계)이 동작하는 단계이며, 다국어 / 결제 연동 / 영수증 자동 분류 정확도 개선 등이 로드맵에 있습니다.

**GitHub** https://github.com/wigtn/wigex

05 / CONTACT

## 새로운 기회나 협업에 관심이 있으시다면 편하게 연락주세요.

| | |
|---|---|
| EMAIL | gus007dn@gmail.com ↗ |
| GITHUB | github.com/starz-woo ↗ |
| LINKEDIN | linkedin.com/in/starz-woo ↗ |
| TEAM | wigtn.com ↗ |

© 2026 조현우. All rights reserved.

Built with Next.js · Tailwind CSS · 맨 위로 ↑