

문영균, iOS 개발자

격투 게임 정보 앱을 혼자 개발해 출시하고 운영하고 있습니다. 팀 프로젝트에서는 공통 네트워크와 채팅, 동기화를 다뤘습니다.

moongoon.cnu@gmail.com

[GitHub](#)

출시와 운영

개인 iOS 앱

비동기 데이터

채팅 · 캐싱 · 동기화

AI 활용과 검토

기능 구현 · 운영 도구

프로젝트

01 · 격투 게임 정보 앱

1인 개발·출시·운영

02 · 토닥운

iOS 팀 개발 · 미출시

03 · 레츠톡

부스트캠프 웹·모바일 9기 · iOS 4인 팀

04 · 엑소스피어랩스

macOS 개발 인턴

05 · Comi

음성 녹음·재생·평가 API 연동

격투 게임 정보 앱

철권8 기술 탐색과 캐릭터별 메모 · 2025.03-현재

1인 개발·출시·운영

게임에서 반복해서 찾는 기술 정보를 앱으로 정리했습니다. 기획부터 개발 · 배포 · 운영까지 맡았고, 출시 이후 영어 지원 요청과 데이터 · 이미지 로딩 문제를 기능과 구조 개선으로 연결했습니다.

UIKit + SwiftUI

Combine

Swift Concurrency

Core Data

Supabase

XCTest

담당 범위

- 초기 아키텍처·데이터 연동·캐싱·메모 구현
- 사용자 요청에 따른 영문 표시와 탐색 화면 개선
- AI를 활용한 영상·필터 기능과 데이터 운영 도구 개발

4K+

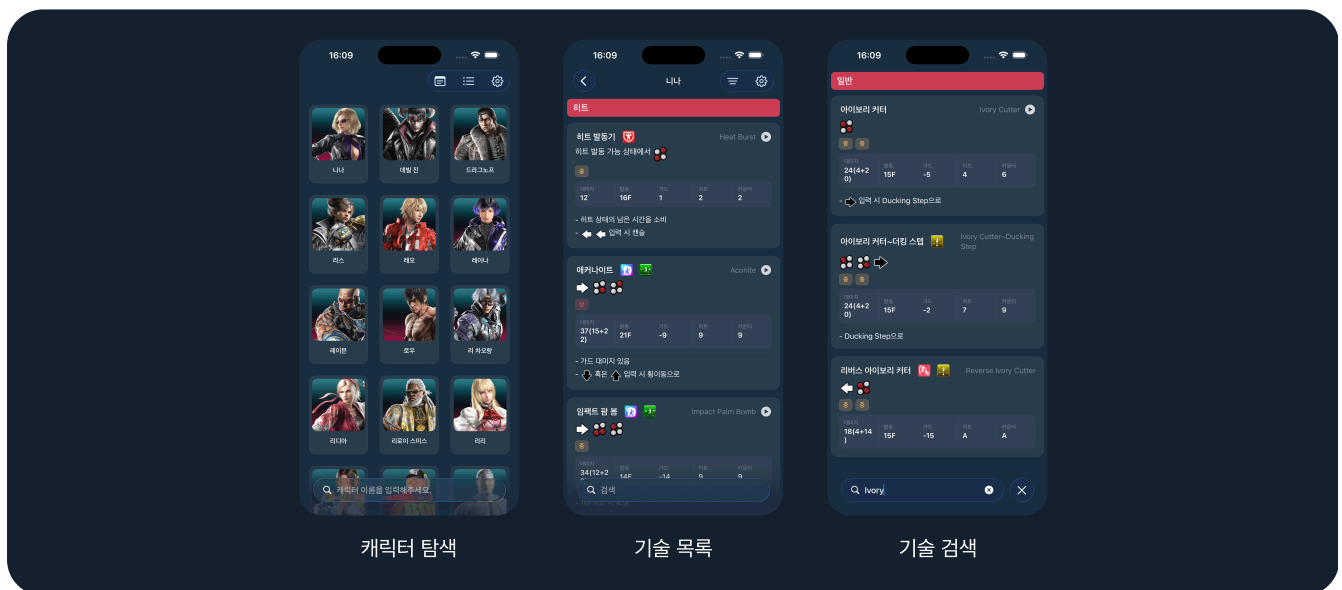
총 다운로드
2026.09 기준

890+

월간 활성 사용자
2026.09 기준

1인

기획·개발·배포·운영
개인 프로젝트



격투 게임 정보 앱의 개발 화면 · 이미지를 누르면 원본을 볼 수 있습니다.

메모와 사용자 데이터 · 캐릭터별 메모 작성·수정·삭제, 검색·고정과 백업·복원 흐름을 다뤘습니다. 기술 데이터 캐시 갱신과 사용자 메모 보존을 구분합니다.

탐색과 영상 · 커맨드를 아이콘으로 표시하고 기술명·별명 검색을 지원합니다. 후속 영상 재생과 필터 기능은 AI를 활용해 구현했습니다.

이 경험에서 남은 점 · 영어 지원 요청을 받으면서 처음 예상하지 못한 사용자도 앱을 쓰고 있다는 것을 알게 됐습니다. 이후에는 제가 익숙한 사용 방식 뿐 아니라 다른 언어를 쓰는 사용자가 정보를 찾고 이해하는 과정도 고려하게 됐습니다.

영어 지원에서는 한국어 원본의 커맨드와 설명을 나누고 설명을 영문으로 변환해, 영문 데이터를 따로 관리하지 않도록 했습니다. 기능을 추가할 때 데이터 구조와 표시 방식을 함께 설계하면 이후 수정·관리할 범위를 줄일 수 있다는 것을 배웠습니다.

DATA & OFFLINE

01 매번 받던 기술 데이터를, 버전이 바뀔 때만 갱신

같은 기술 목록을 다시 열어도 서버에서 전체 데이터를 받아왔습니다. 반복 로딩을 줄이면서 최신 데이터도 유지해야 했습니다.

01 화면 요청

02 Repository

03 버전 확인

04 Core Data 재사용 / 원격 갱신

제가 한 작업

- 서버 데이터 버전과 로컬 버전을 비교하고, 캐시가 비었거나 버전이 바뀌면 Supabase 데이터를 받아 Core Data에 저장했습니다.
- 데이터 출처와 갱신 판단은 Repository 안에 두고, ViewModel은 Repository Protocol에 의존하도록 분리했습니다.
- DI Container와 In-memory Core Data를 사용해 데이터 계층을 테스트할 수 있도록 구성했습니다.

달라진 점

유효한 캐시가 있으면 저장 데이터를 재사용하고 오프라인에서도 기술 정보를 볼 수 있습니다.

변경 전

변경 후

0.41s **0.002s**

데이터 확보 경로 변경 전후

원격 전체 조회와 버전 확인 후 로컬 재사용 경로를 비교한 코드 측정값입니다.

스크롤 중 끊기는 이미지 요청을 화면 밖으로 이동

UIKit 셀 안의 SwiftUI task가 셀 생명주기에 따라 취소되며 이미지 로딩이 중단됐습니다. 같은 이미지의 반복 요청도 발생했습니다.

01 이미지 요청

02 메모리 캐시

03 디스크 캐시

04 필요한 경우 원격 조회

제가 한 작업

- UIHostingConfiguration의 셀과 이미지 요청의 생명주기를 구분하고 로딩 책임을 ViewModel과 캐시 관리 흐름으로 옮겼습니다.
- 메모리 캐시, 디스크 캐시, 원격 조회 순서로 이미지를 찾도록 구성했습니다.
- 이후 Supabase 전송량 무료 한도 초과를 겪고 약 40개 캐릭터 이미지를 Asset에 포함하도록 수정했습니다.

달라진 점

메모리·디스크에 저장한 이미지를 재사용하도록 바꿨습니다. 이후 전송량 한도에 대응해 캐릭터 이미지를 앱의 Asset에 포함했습니다.

변경 전

변경 후

1.1s 0.064s

원격 다운로드와 기기 캐시 재사용 비교

캐싱 도입 전 원격 다운로드와 도입 후 기기 캐시 재사용을 코드로 측정했습니다.

LOCALIZATION

03 영문 DB를 따로 만들지 않고 영어 지원 요청에 대응

한국어 기술 데이터 안에는 게임 커맨드와 설명이 섞여 있었습니다. 영문 DB를 추가하면 두 원본을 함께 관리해야 했습니다.

01 한국어 원본

02 커맨드 / 설명 분리

03 변환 규칙

04 영문 표시

제가 한 작업

- 커맨드 토큰과 설명을 분리하고, 사전과 정규식 규칙으로 설명을 영문 표시로 변환했습니다.
- 커맨드 표기는 유지하고 반복 변환 결과는 캐시했습니다.
- XCTest로 주요 변환 규칙을 확인하고 한국어·영어 UI와 탐색 기능을 연결했습니다.

달라진 점

한국어 원천 데이터를 유지하면서 영어 표시를 지원했습니다. 도입 전후 월평균 해외 다운로드는 45건에서 138건으로 관찰했습니다.

App Store Connect에서 확인한 영어 지원 도입 전후의 월평균 해외 다운로드입니다.

AI-ASSISTED OPERATIONS

04 CSV·터미널 작업을 눈으로 확인하는 운영 도구로 전환

기술 데이터를 CSV로 수정하고 명령으로 DB에 반영하는 작업, 영상 이름과 기술을 대조하는 작업이 번거로웠습니다.

01 CSV 편집 / 영상 정리

02 후보 대조·수정

03 변경 미리보기

04 DB 반영

제가 한 작업

- 기술 수정·검증·반영을 브라우저에서 처리하도록 AI를 활용해 로컬 대시보드를 만들었습니다.
- 영상 녹화와 이름 OCR, 다운스케일링을 자동화하는 작업에도 AI를 활용했습니다.
- R2 영상 목록과 기술 데이터를 대조하고, 이름 매칭 후보 확인·수동 수정·DB 반영을 화면에서 처리하도록 했습니다.

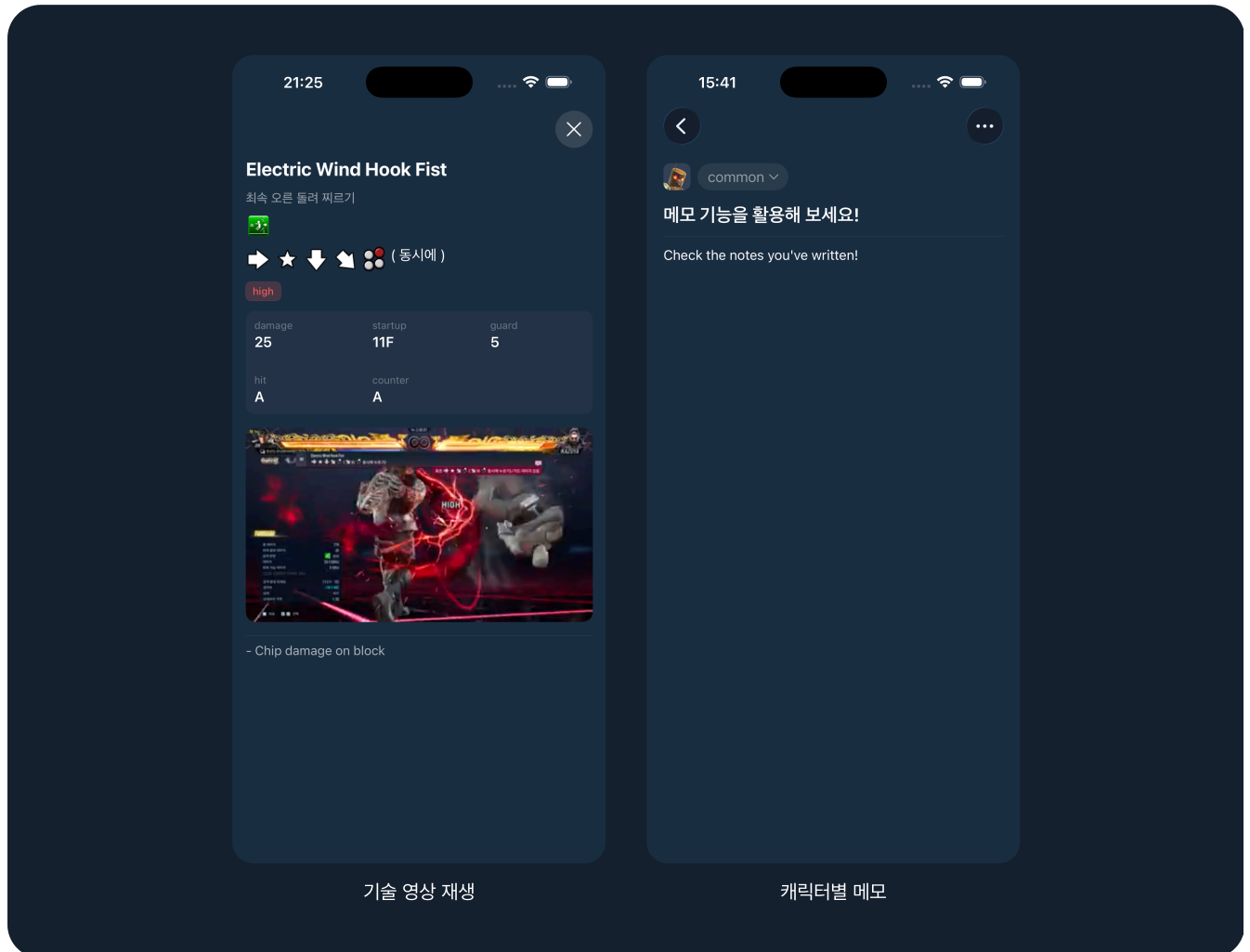
달라진 점

기술 수정과 반영, 이름 매칭 오류 확인 등 운영 작업을 시각화해 더 간단하게 처리하고 있습니다.

영상 파일은 Cloudflare R2에, 기술과 영상의 연결 정보는 Supabase DB에 저장합니다.

영상으로 확인하고, 메모로 기록하기

기술 상세 화면에서 동작 영상을 확인하고, 캐릭터별 메모로 학습 내용을 기록합니다.



격투 게임 정보 앱의 개발 화면 · 이미지를 누르면 원본을 볼 수 있습니다.

토닥운

사주 정보를 바탕으로 대화하는 AI 챗봇 · 2026.05-현재

iOS 팀 개발 · 미출시

YAPP 팀에서 공통 네트워크 계층과 AI 채팅, 마이페이지 개발에 참여했습니다. AI 보조로 구현한 변경을 검토하고, 디자인 원본과 화면 결과를 확인하는 방식도 함께 다뤘습니다.

SwiftUI

TCA

Tuist

Swift Concurrency

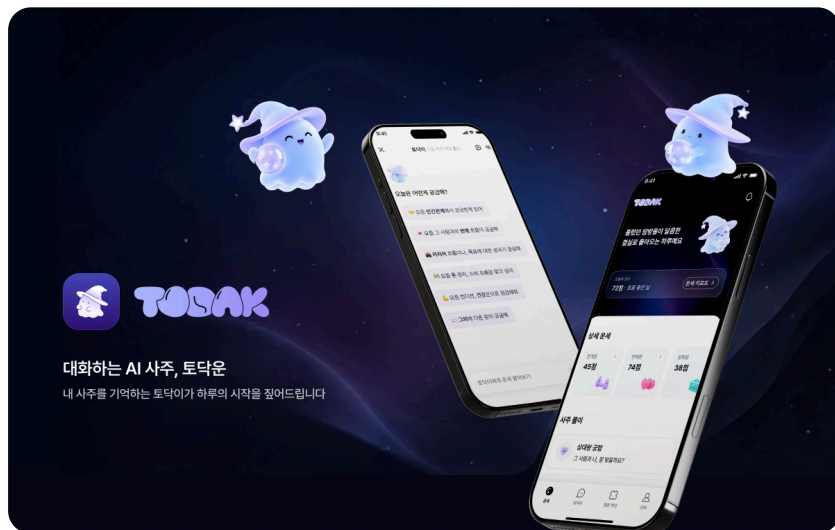
URLSession

Swift Testing

담당 범위

- NetworkCore와 기능별 API Client, SSE 채팅 흐름
- 마이페이지와 인증·가입 만료의 사용자 흐름
- 팀 코드리뷰 전 사전 검토 스킴 작성, 디자인 원본 확인·수정

2026년 10월 기준, App Store 정책 거절로 아직 출시하지 못했습니다.



팀에서 제작한 서비스 소개 이미지

CHAT DATA FLOW

01 서버 응답

02 NetworkCore /
SSE

03 기능별 Client

04 TCA 상태

05 채팅 화면

이 경험에서 남은 점 · AI를 활용한 구현에서도 요구사항 확인과 결과 검토는 남습니다. 디자인 원본, 오류 조건과 리뷰 지적을 각각 확인할 수 있는 기준으로 두고 개발했습니다.

SSE STREAMING

01 서버 응답이 와도 채팅에 나타나지 않던 문제

URLSession line stream에서 SSE 이벤트를 나누는 빈 줄이 유실돼 응답 처리가 누락됐습니다.

01 응답 bytes

02 줄·빈 줄 복원

03 SSE 이벤트

04 채팅 상태 반영

제가 한 작업

- AI 보조로 raw byte에서 줄과 빈 줄 경계를 유지하도록 파싱 방식을 보완했습니다.
- 공통 SSE 이벤트와 채팅 도메인 이벤트를 분리하고 응답 시작·본문 누적·완료·오류를 화면 상태에 연결했습니다.
- 파서·전송 계층·기능별 매핑을 나눠 줄바꿈 형식, 잘못된 입력과 취소 조건을 테스트로 다뤘습니다.

달라진 점

이벤트 경계를 유지하는 처리 흐름을 마련하고, 응답과 오류를 채팅 화면 상태로 연결했습니다.

대화 이력의 조회·상세·삭제와 메시지 전송 기능도 구현했습니다.

NETWORK & SESSION

02 공통 통신 처리와 기능별 데이터 처리를 구분

기능별 API를 화면에 직접 연결하면 요청·오류 처리 기준이 분산되고 독립적인 확인이 어려워집니다. 인증 갱신 실패 후 로그인 복귀도 연결해야 했습니다.

01 기능 Client

02 공통 HTTP 처리

03 주입된 Transport

04 서버

제가 한 작업

- 요청 생성, HTTP 처리와 오류 타입을 NetworkCore에 두고 실제 실행은 주입 가능한 Transport로 분리했습니다.
- 기능별 Client가 API 경로와 DTO 변환, 기능 오류 매핑을 맡도록 했습니다.
- 인증 갱신 API의 401을 세션 만료로 식별해 토큰 정리와 로그인 전환을 연결하고, 동시 요청의 중복 종료를 테스트로 다뤘습니다.

달라진 점

공통 통신과 기능 데이터의 역할을 나누고, 실제 네트워크 없이 오류·취소·응답 매핑을 확인할 수 있는 구조로 구성했습니다.

회원가입 중 token 만료는 별도로 구분해 만료 안내·입력 초기화·로그인 복귀를 TCA 상태 전이로 다뤘습니다.

AI REVIEW WORKFLOW

03 팀 코드리뷰 전에 먼저 문제를 찾는 사전 리뷰 스킬

AI로 만든 변경도 검토해야 할 범위와 위험이 다릅니다. 팀 코드리뷰에 넘기기 전에 먼저 문제를 찾는 과정이 필요했습니다.

01 변경과 검증 근거

02 역할별 검토

03 지적 재확인·수정

04 팀 코드리뷰

제가 한 작업

- 서버에이전트가 변경을 먼저 검토하도록 self-review 스킬을 작성했습니다.
- 변경 규모에 맞는 검토 수준과 역할을 정하고, 지적 내용과 근거를 확인해 수용·보류를 구분했습니다.
- SSE 변경에서 리뷰 피드백을 경계 테스트와 README 보강에 반영하고 재검증하는 흐름을 사용했습니다.

달라진 점

팀 코드리뷰 전 1차 검토와 수정, 검증 근거를 남기는 절차를 마련했습니다.

사전 검토 스킬은 Draft PR 이후 또는 Ready for review 전에 실행하도록 구성했습니다.

UI & COLLABORATION

04 디자인 원본과 실제 화면을 확인하며 마이페이지 구현

AI가 읽은 디자인 요소가 실제 화면에서도 타이포그래피와 간격을 유지하는지 확인해야 했습니다.

01 디자이너의 Figma

02 원본 요소 읽기

03 화면 구현

04 디버그 도구 확인·수정

제가 한 작업

- 디자이너가 만든 Figma 요소를 TalkToFigma MCP로 읽게 하고 원본 기준에 맞춰 AI 보조 개발을 진행했습니다.
- 팀원이 만든 SwiftUI UI 디버그 도구로 요소의 타이포그래피·간격을 확인하고 차이를 수정했습니다.
- 공통 버튼의 타이포 차이를 디자이너에게 확인하고 수정했습니다.
- 회원 탈퇴 문구에서 예상과 다른 UI 문제가 생겨 PM에게 문구 수정을 요청했습니다.

달라진 점

디자인 원본과 구현 결과를 확인·수정하는 작업에 참여하고 기능 화면을 연결했습니다.

레츠톡

AI와 대화하며 작성하는 회고 서비스 · 2024.10.28-2024.12.06

부스트캠프 웹·모바일 9기 · iOS 4인 팀

대화형 회고 앱에서 Core Data와 CloudKit 연동, 채팅 UI를 담당했습니다. 동기화 완료 시점과 화면 갱신을 연결하고, 실제 데이터 구현을 기다리지 않고 화면을 개발할 수 있도록 Mock을 사용했습니다.

SwiftUI

UIKit

Core Data

CloudKit

Combine

Swift Concurrency

담당 범위

- Core Data·CloudKit 연동과 import 완료 후 목록 갱신
- 채팅 전송 대기·실패·재전송 UI
- 합의된 Protocol에 맞춘 Mock 구현과 화면 선행 개발



프로젝트 README에 보존된 당시 서비스 화면

이 경험에서 남은 점 · 동기화 완료와 앱 시작은 같은 시점이 아닙니다. 데이터가 준비되는 이벤트를 화면 갱신에 연결하고, 협업에서는 인터페이스 합의와 Mock을 통해 구현 완료 순서의 영향을 줄였습니다.

CLOUDKIT SYNC

01 iCloud 동기화가 끝나면 최신 회고를 다시 표시

앱 시작 때 목록을 조회하면 iCloud import가 아직 끝나지 않아 최신 회고가 화면에 빠질 수 있었습니다.

01 앱 실행·로컬 조회

02 iCloud import 완료

03 성공 이벤트

04 재조회·화면 갱신

제가 한 작업

- NSPersistentCloudKitContainer.Event를 구독해 import 완료 시점을 확인했습니다.
- import 성공 후 데이터를 다시 조회하도록 연결해 목록이 갱신되게 했습니다.

달라진 점

앱을 재실행하지 않아도 동기화된 최신 회고를 화면에 반영하도록 수정했습니다.

MOCK & CHAT UI

02 실제 Manager를 기다리지 않고 채팅 화면 개발

데이터를 다루는 Manager 구현이 끝나기 전에도 채팅 전송·응답 대기·실패 상태를 개발하고 확인해야 했습니다.

01 합의된 Protocol

02 Mock 응답

03 채팅 상태 UI

04 실제 구현 연결

제가 한 작업

- 팀에서 합의한 Manager Protocol에 맞춰 Mock 객체를 먼저 구현했습니다.
- 응답 대기 중 전송 버튼 상태를 관리하고 실패 후 재전송할 수 있는 화면을 구현했습니다.
- 설정 데이터의 변경을 단일 진입점으로 모으는 작업에도 참여했습니다.

달라진 점

실제 Manager 완성 전에 화면 개발을 진행하고 채팅의 대기·실패 흐름을 확인할 수 있었습니다.

엑소스피어랩스

macOS 고객센터 앱과 정책 변경 알림 연동 · 2023.12-2024.04

macOS 개발 인턴

기존 보안 프로그램의 로그와 정책 기능을 별도 고객센터 앱에서 사용할 수 있도록 연결했습니다. Electron에서 받은 정책 변경 알림을 native 영역으로 전달하는 프로토타입도 구현했습니다.

Swift

SwiftUI

NSXPCConnection

Electron

FCM

Vapor

담당 범위

- 고객센터 앱의 로그 조회·필터링·추출과 정책 on/off 연동
- Electron의 공통 알림 수신·전달 로직과 macOS local API 수신부

MACOS SUPPORT

01 기존 로그·정책 기능을 고객지원 앱에서 사용

고객 문제를 확인하고 조치하려면 메인 앱에 있는 로그와 정책 기능을 지원 담당자가 사용할 수 있어야 했습니다.

01 고객지원 앱

02 공통 인터페이스 / XPC

03 기존 로그·정책 기능

04 결과 표시

제가 한 작업

- SwiftUI 기반 고객지원 앱에서 NSXPCConnection을 통한 연결을 사용했습니다.
- 기존 로그를 조회·필터링·추출하고 정책을 켜고 끄는 흐름을 화면에 연결했습니다.

달라진 점

기존 기능을 고객지원용 앱에서 확인하고 조작할 수 있도록 구현했습니다.

PLATFORM INTEGRATION

02 FCM 알림을 Electron에서 native API로 전달

정책 변경 알림을 받아 각 OS의 회사 프로그램에 전달하는 공통 흐름을 검증해야 했습니다.

01 FCM 알림

02 Electron 공통 로직

03 OS별 local API

04 macOS 정책 변경 연동

제가 한 작업

- Electron에서 FCM 정책 변경 알림을 수신하고 OS별 local API로 전달하는 공통 로직을 구현했습니다.
- macOS는 Vapor 수신부를 구현하고, Windows 경계는 더미 local API 호출로 확인했습니다.

달라진 점

Electron 수신·전달과 macOS 수신부를 연결하는 프로토타입을 만들었습니다.

Comi

음성으로 대화하고 발음 평가를 받는 졸업 프로젝트 · 2024

음성 녹음·재생·평가 API 연동

서버에서 받은 AI 음성을 재생하고 사용자 음성을 녹음하는 흐름, Azure API로 발음 평가를 받아 다음 대화에 전달하는 작업을 진행했습니다.

Swift

AVAudioSession

AVAudioRecorder

AVAudioPlayer

Azure Speech SDK

담당 범위

- 서버 음성 수신·재생과 사용자 음성 녹음
- 발음 평가 결과·인식 문자열과 다음 대화 요청 연결

AUDIO CONVERSATION

01 AI 음성 재생

02 사용자 녹음

03 인식·발음 평가

04 다음 대화 요청

05 · Comi / 주요 사례

AUDIO & EXTERNAL API

01 음성 입력과 평가 결과를 대화 흐름에 연결

화면의 대화뿐 아니라 AI 음성 재생과 사용자 녹음, 외부 발음 평가 결과를 순서대로 연결해야 했습니다.

제가 한 작업

- AVAudioSession과 Recorder-Player를 사용해 녹음과 재생을 다뤘습니다.
- 서버 음성을 받아 WAV 저장·재생 흐름을 연결했습니다.
- Azure Speech SDK의 발음 평가 결과를 모델에 저장하고 인식 문자열·점수를 다음 대화 요청에 전달했습니다.

달라진 점

AI 음성 → 사용자 녹음 → 평가·다음 요청으로 이어지는 대화 기능을 구현했습니다.

발음 평가 결과에는 정확도·유창성 등의 점수와 인식 문자열이 포함됩니다.

BACKGROUND

학습과 경력의 기반

교육

- 충남대학교 컴퓨터융합학부 · 2019.03-2025.02
- 네이버 부스트캠프 웹·모바일 9기 · 2024.07-2024.12
- CNU SW아카데미 2기 iOS · 2023.01-2023.04

moongoon.cnu@gmail.com

[GitHub · MoonGoon72](#)

자격 · 어학

- 정보처리기사 · 2025.12
- TOEIC Speaking IH · 2025.06