진재명 Jaemyeong Jin

iOS 소프트웨어 엔지니어 · 개인 프로젝트 포트폴리오

jaemyeong@me.com · 010-3614-3752 · jaemyeong.com · github.com/jaemyeong

프로젝트 개요

아래 네 프로젝트는 새 기술을 직접 써 보며 익히려고 만든 개인 PoC 프로젝트입니다.

DailySudoku | iPhone·iPad 스도쿠 앱

2–3쪽

2026.04 시작 · App Store 출시 · dailysudoku.app

매일 새로운 퍼즐을 푸는 스도쿠 앱입니다. 손글씨로 숫자를 입력하는 기능과, 화면에 보이는 시간과 저장되는 시간을 맞춘 과정을 다룹니다.

탐구한 기술: Rust 공통 엔진(UniFFI), Core ML 온디바이스 추론, Swift 6

아홉 | iPhone·iPad 블록 퍼즐

4쪽

2020.11 출시 · 제5회 Bu:Star 챌린지 게임 공모전 최우수상 · ahob.app

게임 엔진 없이 UIKit만으로 만든 9×9 블록 퍼즐입니다. UIKit으로 구현한 게임 화면과, 접근성·테스트를 정비한 과정을 다룹니다.

탐구한 기술: UIKit 기반 게임 렌더링·애니메이션, 접근성(VoiceOver), Game Center

NutritionEngine | 음식 영양 분석 서비스

5–6쪽

2026.06 시작 · nutritionengine.io

음식 이름이나 사진을 보내면 영양 정보를 돌려주는 서비스입니다. AI가 숫자를 지어내지 못하게 한 구조와, 검색 방식을 실험으로 정한 과정을 다룹니다.

탐구한 기술: RAG, 로컬 LLM 서빙과 양자화, pgvector, MCP

StorageGame | Mac 퍼즐 게임

7쪽

2026.08 시작 · Mac App Store·itch.io 출시 · storage-game.com

플로피 디스크 한 장 크기 안에 담은 게임입니다. 이미지와 소리 파일 없이 만든 방법과, 배포 경로에 따라 구성을 나눈 과정을 다룹니다.

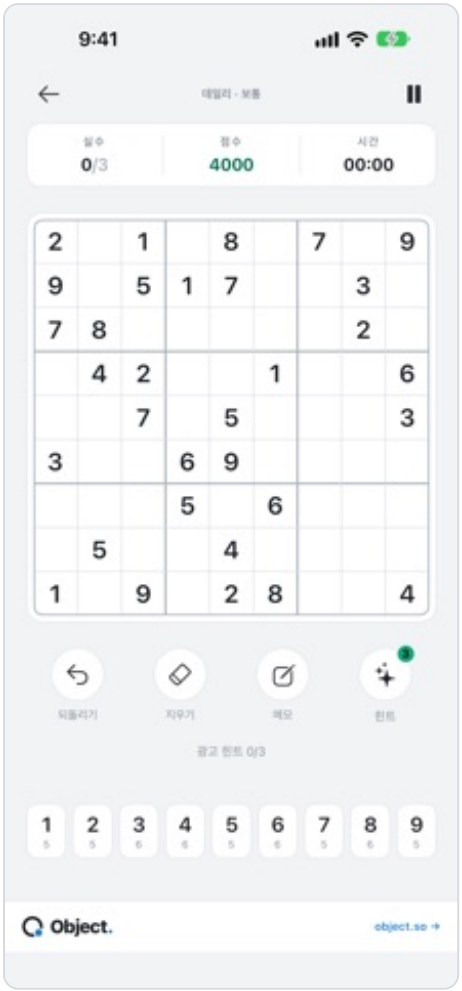
탐구한 기술: SpriteKit, 코드 기반 그래픽·합성 사운드, 바이너리 크기 최적화

DailySudoku · iPhone·iPad 앱

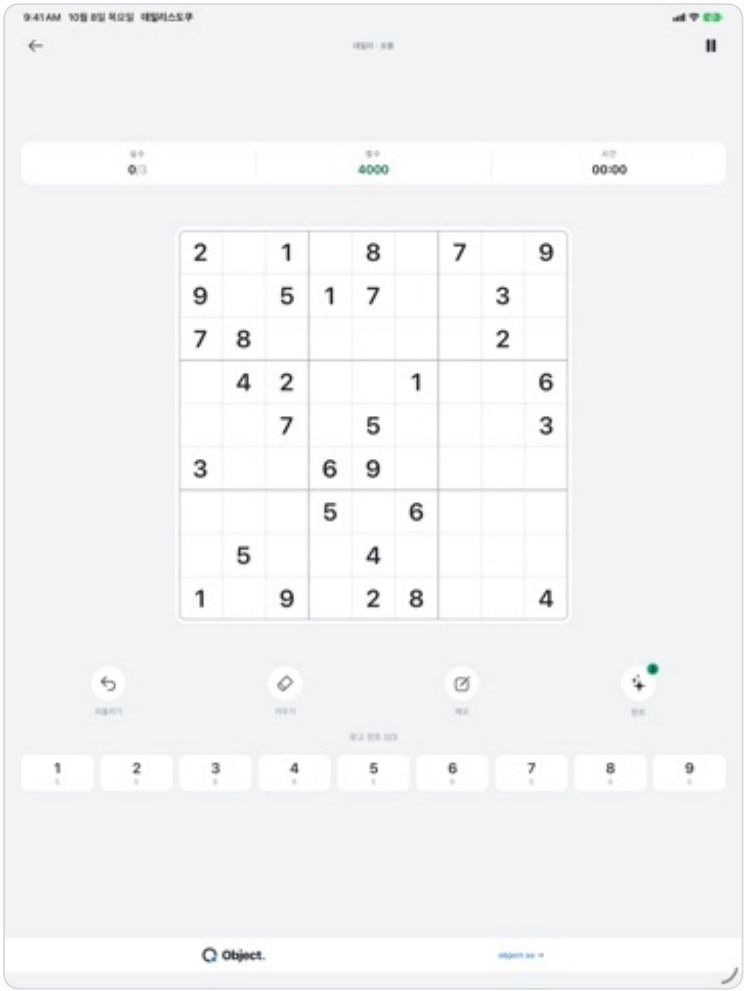
매일 새로운 퍼즐을 풀고 기록을 이어 가는 스도쿠 앱

2026.04 시작 · App Store 출시 · dailysudoku.app

DailySudoku는 날짜와 난이도를 고르면 그날의 퍼즐이 주어지고, 숫자와 메모로 풀어 가는 스도쿠 앱입니다. 게임 규칙처럼 어느 기기에서나 똑같이 동작해야 하는 부분을 한 곳에서 관리하는 구조를 직접 검증해 보고 싶어서 시작했고, 기획부터 설계·구현·출시까지 직접 진행했습니다.



iPhone 게임 화면



iPad 게임 화면

DailySudoku · iPhone·iPad 앱 (계속)

매일 새로운 퍼즐을 풀고 기록을 이어 가는 스도쿠 앱

손글씨로 숫자 입력

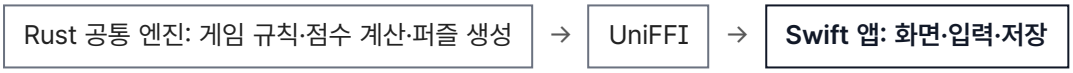
Apple Pencil로 칸에 숫자를 쓰면 바로 입력됩니다. 서버를 거치지 않고 기기 안에서 글씨를 알아보도록, 작은 인식 모델을 앱에 넣었습니다. (Core ML)

- 모델을 학습시킨 환경과 앱에서 같은 글씨를 같은 숫자로 인식하는지 비교하는 테스트를 만들어, 입력을 다듬는 과정의 차이로 결과가 달라지는 일을 막았습니다.

화면에 보이는 시간과 저장되는 시간 맞추기

점수는 경과 시간에 따라 매초 바뀝니다. 매초 게임 상태 전체를 다시 계산하는 대신, 점수에 필요한 숫자 네 개(기본 점수·경과 시간·실수·힌트)만으로 표시 점수를 계산하도록 바꿨습니다. 그러면 화면의 시간과 저장되는 시간이 어긋날 수 있어서, 숫자를 입력하거나 저장·완료하기 직전에 누적 시간을 게임 상태에 반영하도록 했습니다.

- 처음 만든 테스트는 화면의 시간만 확인해서, 저장 직전 반영을 빼도 통과했습니다. 저장된 시간을 확인하도록 테스트를 고쳤습니다.
- 게임 규칙과 점수 계산은 Rust 공통 엔진에 두고, 앱은 화면·입력·저장을 맡도록 역할을 나눴습니다. (Rust 공통 엔진을 Swift 앱에 연결)



기술 요약

- Rust 공통 엔진을 UniFFI로 Swift 앱에 연결했습니다. 같은 날짜·난이도에서는 플랫폼과 무관하게 같은 퍼즐을 생성합니다.
- 비트마스크·MRV 기반 유일해 솔버와 SplitMix64 난수 생성기를 직접 구현했습니다.
- 손글씨 인식은 MNIST CNN(약 11만 파라미터, fp16 약 215KB)을 Core ML로 기기에서 추론합니다.
- Swift 6 언어 모드를 적용했고, GitHub Actions·Xcode Cloud·fastlane으로 배포를 자동화했습니다.

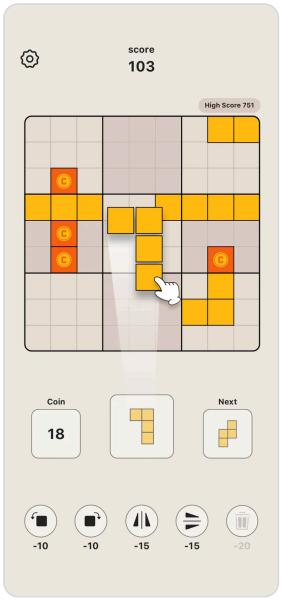
사용 기술: Swift · UIKit · Core ML · Rust(UniFFI)

아홉(Ahob) · iPhone·iPad 게임

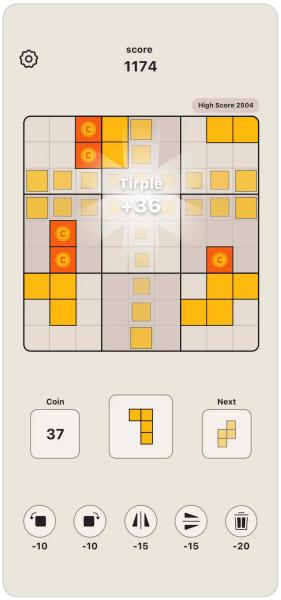
게임 엔진 없이 UIKit만으로 만든 9×9 블록 퍼즐

2020.11 출시 · 제5회 Bu:Star 챌린지 게임 공모전 최우수상 · ahob.app

블록을 놓아 가로줄, 세로줄, 3×3 구역 가운데 하나를 채우면 지워지는 퍼즐 게임입니다. 게임 엔진 없이 UIKit만으로 인터랙티브 게임을 구현하며 렌더링과 애니메이션을 깊이 이해하고, 기획부터 출시까지의 전 과정을 직접 경험해 보기 위해 시작했습니다.



게임 화면



콤보

UIKit으로 구현한 게임

- 게임 엔진 없이 UIKit으로 보드, 블록 드래그, 애니메이션을 구현했습니다. 게임 플레이는 오프라인에서 동작합니다.
- Game Center 순위표 3종, 광고 제거 인앱 결제, UMP 동의 흐름, 40개 로케일, 왼손 모드·다크 모드를 지원합니다.

접근성과 테스트

- 퇴사 후 진행한 2026년 개편 작업에서 디자인 캔버스와 실제 화면 스크린샷 38장을 1:1로 대조했습니다.
- 점검 범위에서 VoiceOver 이중 노출과 44pt 미만 탭 대상을 0건으로 만들었고, 테스트를 374개에서 509개로 늘렸습니다.

알고리즘으로 검증한 스테이지

- 캠페인 스테이지 9개의 최소 명령 수를 BFS 탐색으로 증명했습니다.

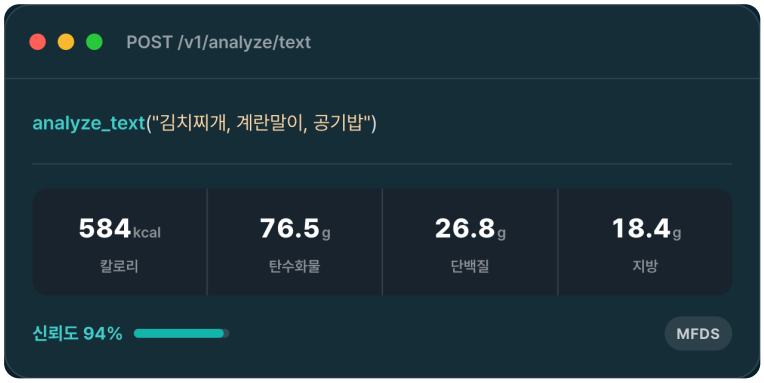
사용 기술: Swift · UIKit · Game Center · StoreKit

NutritionEngine · 음식 영양 분석 서비스

음식 이름이나 사진을 보내면 출처가 있는 영양 정보를 돌려주는 서비스

2026.06 시작 · nutritionengine.io

NutritionEngine은 다른 앱이나 AI 도구가 음식의 영양 정보를 조회할 때 쓰는 서비스입니다. 음식을 설명하는 문장이나 사진을 보내면 영양 성분을 돌려줍니다. AI가 그럴듯한 숫자를 지어내는 문제를 구조로 막을 수 있는지 직접 확인해 보고 싶어서 만들었습니다.



분석 요청과 결과 예시

AI의 역할과 데이터의 역할 나누기

AI(언어 모델)는 음식이 무엇인지, 양이 얼마인지, 어떤 검색어로 찾을지만 정합니다. 영양 성분 수치는 공개 식품 데이터에서 찾아온 값만 사용합니다. AI가 내놓는 결과 양식에는 영양 수치를 적을 칸 자체를 두지 않았습니다.

음식 문장·사진 → AI: 음식·분량·검색 계획 → 식품 데이터 검색 → 출처·값 확인 → **영양 결과**

- 식품 데이터는 식품의약품안전처(MFDS), USDA, Open Food Facts를 정규화해 연결했고, BGE-M3 임베딩과 pgvector HNSW 인덱스로 검색합니다.
- 언어 모델은 양자화된 로컬 모델(Gemma 4 12B QAT)을 직접 서빙해 연결했습니다. (LM Studio, Python·FastAPI)

중간에 값이 바뀌지 않았는지 확인

찾아온 값이 이후 처리 단계에서 바뀌거나 출처가 빠질 수 있습니다. 그래서 값을 찾아온 직후의 사본을 따로 보관해 두고, 최종 결과를 내보내기 전에 비교합니다.

- 리뷰 과정에서 이 비교를 빠져나갈 수 있는 경로를 찾아 보완했습니다. 예를 들어 Python은 참(True)과 숫자 1을 같다고 판정해서, 값이 True로 바뀌어도 비교를 통과할 수 있었습니다. 값과 함께 종류(자료형)도 비교하도록 고쳤습니다.

검색 방식을 실험으로 정하고 AI 도구에 구현·검토를 맡긴 과정

제공 방식

- REST API(앱에서 호출), MCP 서버(AI 도구에서 호출), 콘솔(분석 결과와 사용량 조회)로 제공합니다.
- LLM 응답 캐시를 2계층으로 두었고, Idempotency-Key 기반으로 같은 요청이 다시 시도되어도 사용량이 두 번 기록되지 않게 했습니다.

검색 방식 비교 실험

사용자는 음식 이름을 띄어쓰기 없이 쓰거나, 일부만 쓰거나, 틀리게 씁니다. 뜻이 비슷한 것을 찾는 검색(dense)과, 여기에 글자가 일치하는 검색을 합친 방식(hybrid) 가운데 무엇을 기본으로 할지 실험으로 정했습니다.

개발 데이터에서 식품 이름 200개를 고르고, 띄어쓰기를 빼거나 넣은 경우, 오타, 앞부분이나 일부만 입력한 경우를 합쳐 904개의 검색어를 만들어 두 방식을 비교했습니다.

지표	뜻 기반(dense)	혼합(hybrid)
1순위가 정답인 비율	53.2%	54.8%
3순위 안에 정답이 있는 비율	66.3%	69.7%
20순위 안에 정답이 있는 비율	77.4%	81.6%

전체 수치는 혼합 방식이 높았습니다. 하지만 띄어쓰기를 뺀 검색어에서는 1순위가 정답인 비율이 뜻 기반 83.6%, 혼합 74.5%로 혼합 방식이 오히려 낮았습니다. 평균만 보고 바꾸면 흔한 입력에서 결과가 나빠진다고 판단해, 기본값은 그대로 두고 혼합 방식은 선택 설정으로 제공했습니다.

- 검색 단계만 평가한 결과입니다. (개발 데이터 317,483행, BGE-M3 임베딩, RRF 결합)

AI 도구에 구현과 검토 나누어 맡기기

콘솔 화면의 "불러오는 중, 결과 없음, 오류, 권한별 표시" 상태를 테스트로 고정하는 작업을
AI 도구와 진행했습니다.

- 먼저 화면 상태를 표로 정리하고, 고칠 수 있는 파일과 건드리면 안 되는 범위, 완료 조건을 글로 정했습니다.
- 구현은 Codex에 맡겨 상태 테스트 32개를 추가했고, 같은 변경을 Claude가 따로 검토하게 했습니다.
- 도구가 끝났다고 보고한 뒤 전체 테스트를 직접 실행해 확인했습니다. (47개 파일·356개 테스트 통과)

배운 점

AI가 만든 결과가 맞는지 확인하는 것만으로는 부족했습니다. AI가 바꿀 수 있는 값의 범위를 구조로 제한하고, 확인 절차를 빠져나갈 수 있는 경로를 테스트로 막는 편이 더 믿을 수 있었습니다.

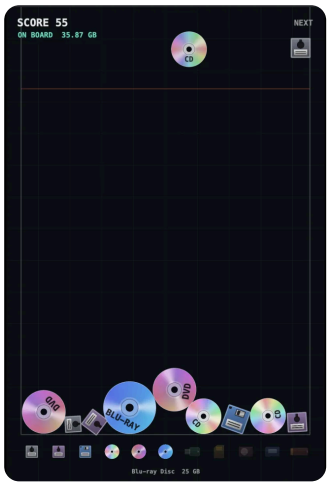
사용 기술: Python · FastAPI · PostgreSQL · pgvector · TypeScript · 로컬 LLM

StorageGame · Mac 게임

저장 매체를 합쳐 더 큰 저장 매체를 만드는 퍼즐 게임

2026.08 시작 · Mac App Store·itch.io 출시 · storage-game.com

키보드로 블록을 놓고, 같은 저장 매체끼리 합쳐 더 큰 용량의 매체로 키워 가는 Mac용 게임입니다. 앱 전체를 플로피 디스크 한 장 크기(1.44MB) 안에 담는다는 제약을 두고, 그 안에서 그림·소리·통신을 어떻게 만들지 탐구했습니다.



게임 플레이



메뉴·병합 단계

이미지와 소리 파일 없이 만들기

용량을 줄이기 위해 앱 아이콘을 제외한 이미지와 소리 파일을 넣지 않았습니다. 그림은 코드로 그리고, 효과음은 앱이 시작될 때 계산해서 만들어 둔 뒤 게임 중에 재사용합니다. (SpriteKit·AppKit, 사인파 합성)

배포 경로에 따라 다른 구성

온라인 기능에 흔히 쓰는 외부 라이브러리는 용량이 커서, 직접 배포하는 버전에는 필요한 통신 기능만 직접 구현했습니다. Mac App Store 버전은 조건에 따라 라이브러리를 포함하도록 나눴습니다. (URLSession 기반 REST·SSE)

- 크기는 서명까지 끝난 앱을 기준으로 재고, 앱 자체와 배포용 설치 파일(DMG)을 구분해 관리했습니다. 앱 번들의 실측 크기는 775,426바이트로, 한도(1,474,560바이트)의 약 53%입니다.
- 빌드, 설치 파일 생성, 스토어 스크린샷 생성을 도구로 만들어 출시 작업을 반복할 수 있게 했습니다.
- Apple Silicon·Intel 유니버설 바이너리로 빌드했습니다.

사용 기술: SpriteKit · AppKit · Objective-C++ · C++ · URLSession