

# 구동재

금융 앱의 백엔드를 개발하고,  
개인 게임을 Android로 출시했습니다.

학력            경북대학교 컴퓨터학부  
                  글로벌소프트웨어융합전공 학사 졸업

교육            삼성 청년 소프트웨어 AI 아카데미  
                  15기 모바일 트랙 이수

수상            삼성청년SWAI아카데미 2학기 공통프로젝트 우수상  
                  구미 1반 1등  
  
                  삼성청년SWAI아카데미 2학기 특화프로젝트 우수상  
                  구미 1반 2등

## SELECTED PROJECTS

01 / 도담            Spring Boot 백엔드 · 금융망 연동 · AI 코치

02 / 야추 온라인    개인 개발 · 실시간 게임 서버 · Android 앱 출시

[kdj8501@gmail.com](mailto:kdj8501@gmail.com) ↗

[GitHub](#) ↗

# 도담

# 01



## 청년을 위한 AI 금융 코치

계좌와 거래 데이터를 가져와 소비를 분석하고,  
목표를 세우거나 금융 상담을 받을 수 있는 앱입니다.

SSAFY 팀 프로젝트 · 2026.09.02 – 2026.09.29

---

백엔드 개발    Spring Boot · MySQL · Redis

# 도담에서 맡은 일

금융망 연동부터 소비 분석과 AI 상담까지  
앱에서 사용하는 백엔드 API를 개발했습니다.

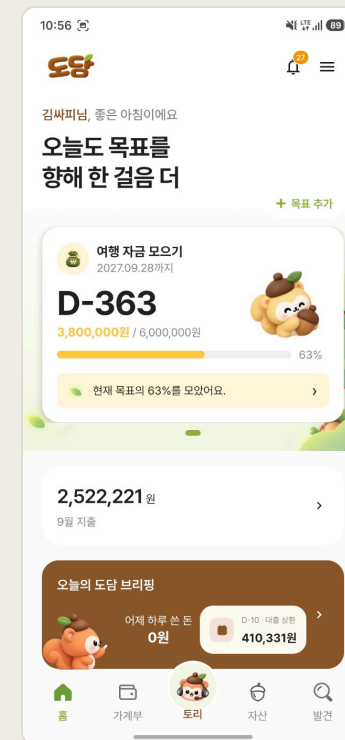
금융 데이터      SSAFY 금융망 연동, 거래 적재와 소비 집계  
목표 관리 API

인증      JWT 인증과 소셜 로그인 처리  
Refresh Token 해시 저장·회전

AI 상담      Redis 대화 세션과 호출 제한  
서버 계산 도구 연계, 응답 숫자 검증

삼성청년SWAI아카데미 2학기  
특화프로젝트 우수상

팀 수상 · 구미 1반 2등



목표와 소비 현황을 보여주는 앱 화면 · 팀 결과물

# 늦게 끝난 작업이 만료 이력을 덮지 않도록

1 / 실행 A

금융망 응답 대기

2 / 새 요청 B

A 만료 후 B 시작

3 / A의 늦은 완료

상태·갱신 시각 덮어쓰기

## 완료 처리에도 같은 잠금 규칙 적용

시작·만료·완료 시 같은 사용자 금융 연결 행을 잠급니다. 완료 시점에 RUNNING인지 확인하고, 이미 종료된 실행은 상태와 마지막 갱신 시각을 바꾸지 않도록 수정했습니다.

실행 등록과 마감은 별도 트랜잭션에서 처리합니다. 외부 금융망 호출은 상태 관리 트랜잭션 밖에서 수행합니다.

## 수정 전·후 재현 기록

| 늦은 A 완료 직후 | 수정 전      | 수정 후       |
|------------|-----------|------------|
| A 상태       | COMPLETED | FAILED 유지  |
| B 상태       | RUNNING   | RUNNING 유지 |
| 마지막 갱신 시각  | 변경됨       | 변경 없음      |

2026.09.09 기존 재검증 기록 · eecb4a7 · H2 독립 스레드 재현  
확인 범위: 실행 상태와 last\_synced\_at

# 대화는 이어가고, 호출 횟수는 제한했다

## 대화 세션

### 사용자와 대화별로 최근 맥락을 저장했습니다

사용자 ID와 대화 ID를 조합한 Redis 키에 세션 JSON을 저장하고 기본 TTL을 30분으로 설정했습니다. 사용자 문장과 서버 계산 근거 요약을 남겨 다음 질문에 활용했습니다.

## 호출 제한

### 카운터 증가와 만료 설정을 한 번에 처리했습니다

사용자별 고정 시간창으로 외부 AI 호출 횟수를 제한했습니다. INCR 뒤 EXPIRE 전에 처리가 끊기는 경우를 피하도록 두 연산을 Lua 스크립트에서 함께 실행했습니다.

## REDIS 장애 시

### 대체 동작도 함께 구현했습니다

|    |                     |
|----|---------------------|
| 대화 | 세션을 읽지 못하면 새 대화로 시작 |
|----|---------------------|

|       |                   |
|-------|-------------------|
| 호출 제한 | 인프로세스 카운터로 이어서 제한 |
|-------|-------------------|

|    |                                      |
|----|--------------------------------------|
| 한계 | 대체 카운터는 서버별로 동작하며<br>재시작하면 초기화될 수 있음 |
|----|--------------------------------------|

# 소비 진단에서 질문과 계산으로

소비 진단의 맥락을 대화에 전달하고, 저축액이나 소비를 바꿨을 때의 목표 달성일을 기존 서버 계산기로 구하도록 연결했습니다.

## 01 모델의 도구 요청 검사

허용한 도구와 인자인지 확인  
턴당 실행·왕복 횟수 제한

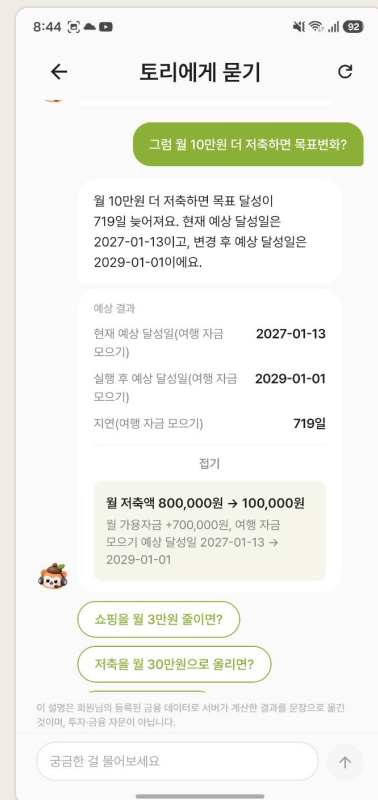
## 02 인증 사용자 범위에서 계산

본인의 목표·대출만 조회해 계산 도구 호출

## 03 계산 근거와 응답 수치 대조

검증 실패·외부 호출 실패 시 템플릿 응답

외부 전송 직전 대화 이력과 도구 결과의 민감정보 패턴도 재검사했습니다.



AI 코치 대화·계산 결과 카드 · 개발 중 화면

# 소비 내역을 진단과 유형 분석에 활용

발견한 점 · 집계 데이터

## 같은 기준으로 소비를 비교할 수 있도록

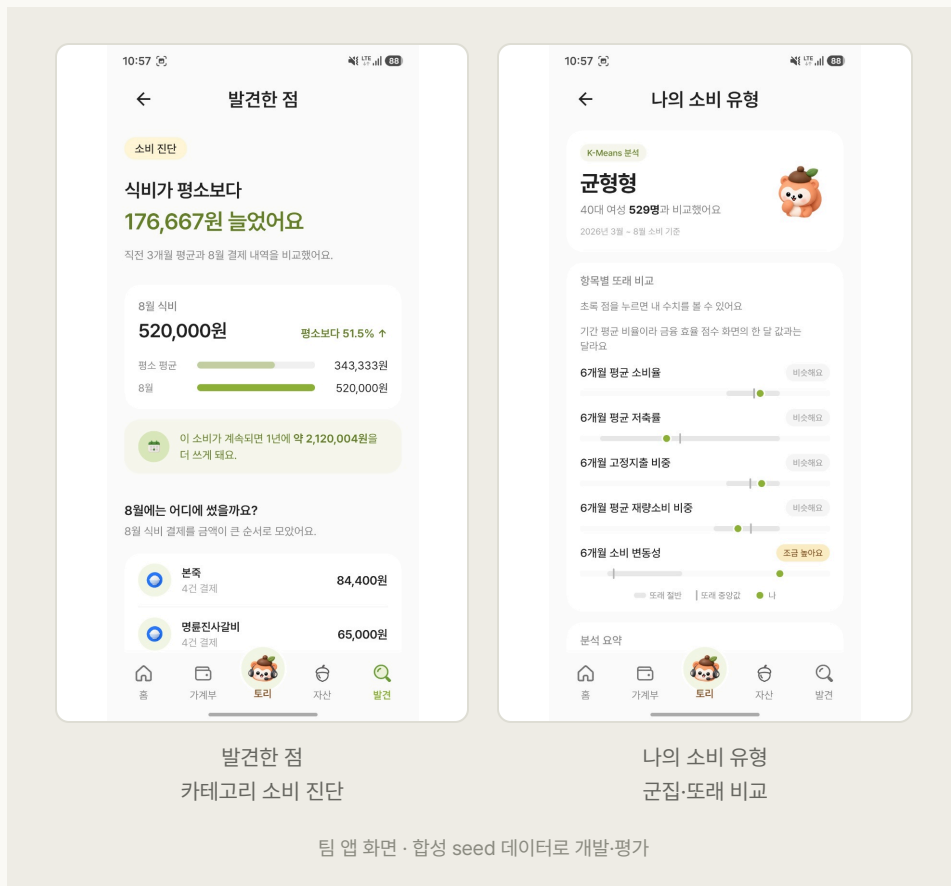
사용자·기간·카테고리별 소비를 SQL로 집계했습니다. 수입·이체·저축성 지출·숨긴 거래를 제외하는 조건을 공통화해, 소비 진단과 분석에 사용할 데이터를 제공했습니다.

나의 소비 유형 · K-Means

## 학습 결과를 Spring API에서 활용했습니다

Python에서 표준화 값과 군집 중심을 JSON으로 내보내고, Spring에서 로딩해 소비 유형을 배정했습니다. 군집 내 위치와 나이대·성별 비교 정보를 확장하고, 군집 이탈 신호를 기존 진단·추천에 연결했습니다.

나이·성별은 비교 기준이며 군집 입력 특징이 아닙니다.  
모델·입력 조건 미충족 시 규칙 기반 분석으로 대체합니다.



# 야추 온라인

## 02

### 직접 만들고 출시한 주사위 게임

개인적으로 시작한 주사위 게임에 배운 기술을 적용하며  
데스크톱 게임을 Android 앱으로 발전시켰습니다.

개인 프로젝트 · Google Play 누적 다운로드 1,000+

---

Flutter · Node.js    기획·개발·출시·운영

# 개인적으로 만든 게임을 Google Play에 출시

주사위 5개로 점수를 겨루는  
1:1 실시간 멀티플레이 게임

시작  
개인 프로젝트로 Java Swing 게임 구현  
컴퓨터망에서 배운 내용을 적용해 서버 구축

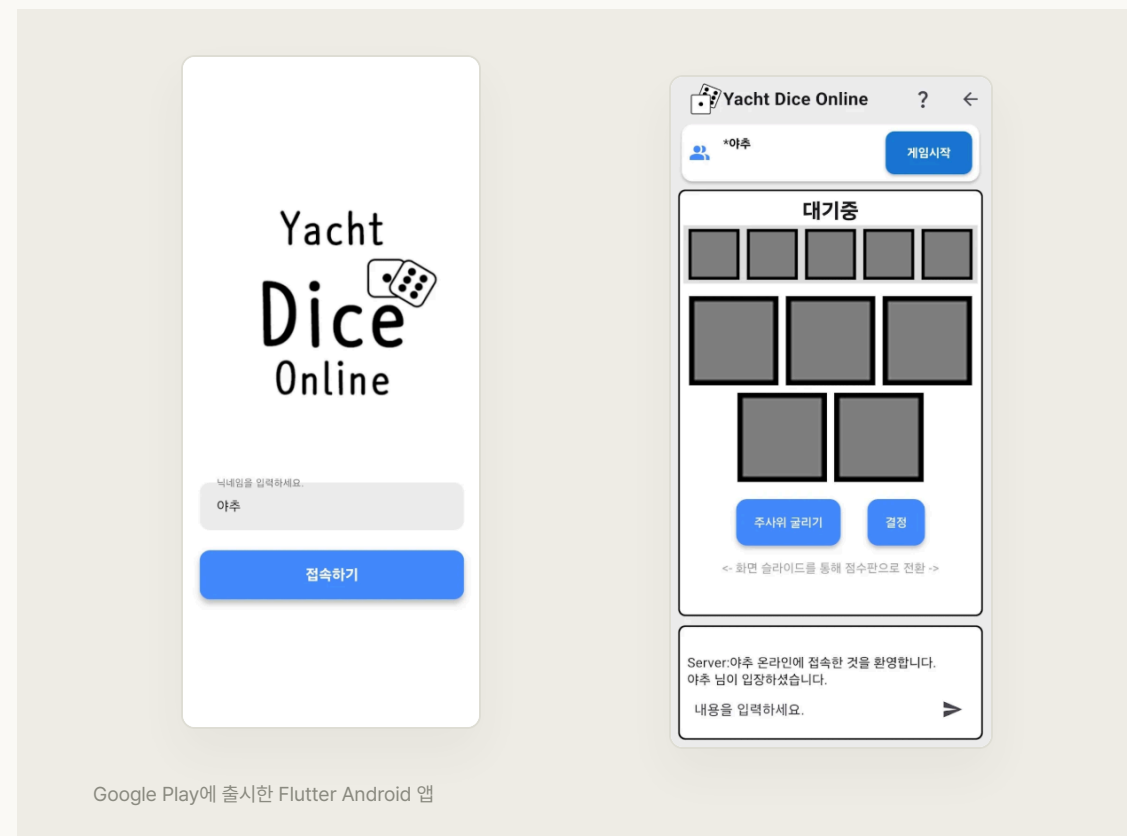
Android  
모바일앱 개발을 배우며 Kotlin으로 포팅  
이후 Flutter·Node.js로 리팩토링

출시  
기획과 앱·서버 개발을 혼자 담당  
AdMob 적용 후 Google Play 출시

1,000+

Google Play 누적 다운로드  
프로젝트 기록 기준

[Google Play에서 보기 ↗](#)



Google Play에 출시한 Flutter Android 앱

# 두 화면이 같은 게임 상태를 보도록

## 서버의 역할

### 주사위와 점수는 Node.js 서버가 확정했습니다

Flutter 앱은 굴리기·홀드·점수 칸 선택을 WebSocket으로 요청합니다. 서버는 본인 차례인지, 굴린 횟수가 3회 이내인지, 선택한 점수 칸이 비어 있는지 확인한 뒤 상태를 변경합니다.

## 상태 전달

### 점수 계산과 차례 전환을 마친 뒤 함께 전송했습니다

서버의 주사위로 점수와 보너스를 계산하고 다음 차례를 준비합니다. 확정된 주사위·점수판·굴린 횟수·차례를 방 참여자에게 전달하고, 계정과 승수는 SQLite에 저장했습니다.

## FLUTTER 앱

### 서버 응답을 화면에 반영

WebSocket 서비스에서 메시지를 구분해 화면 콜백에 전달했습니다. 스트림 종료와 오류도 연결 종료 콜백으로 처리했습니다.

---

기기 비율에 따라 점수판이 잘리는 문제는  
LayoutBuilder와 MediaQuery로 화면 배치를  
조정했습니다.

# 이전 연결이 새 세션을 지우지 않도록

같은 계정이 새 소켓으로 접속한 뒤, 이전 소켓의 종료 이벤트가 도착할 수 있습니다. 종료 이벤트에서 계정 ID만 보고 삭제하면 새 세션까지 사라질 수 있어 소켓을 비교했습니다.

```
if (activeSessions.get(ws.playerId) === ws) {  
    activeSessions.delete(ws.playerId);  
}
```

종료된 소켓이 현재 등록된 소켓과 같을 때만 세션을 삭제합니다.

## 연결 정리

### 종료 처리는 한 번만

|       |                         |
|-------|-------------------------|
| 중복 방지 | cleanedUp 플래그로 반복 정리 방지 |
|-------|-------------------------|

|     |                                  |
|-----|----------------------------------|
| 무응답 | Ping-Pong으로 감지한 뒤<br>공통 종료 처리 실행 |
|-----|----------------------------------|

|       |                                  |
|-------|----------------------------------|
| 이탈 정책 | 게임 중 이탈하면 진행 중단<br>참여자가 없는 방은 삭제 |
|-------|----------------------------------|

# 개발할 때는 보이지 않던 문제들

---

## GOOGLE PLAY

### 스토어에서 받은 앱만 로그인이 실패했습니다

로컬에서는 되던 Google 로그인이 배포본에서 실패했습니다. 업로드 키와 Play 앱 서명 키가 다르다는 점을 확인하고, 배포용 인증 정보를 등록해 해결했습니다.

## RASPBERRY PI

### 서버 로그 밖에서 원인을 찾았습니다

서버가 불규칙하게 멈추거나 재부팅됐습니다. 네트워크와 앱 로그에 이어 커널 로그를 확인하면서 UnderVoltage 경고를 찾았습니다. 정격 전원 어댑터로 교체한 뒤 정지 문제가 해결됐습니다.

---

구동재    [kdj8501@gmail.com](mailto:kdj8501@gmail.com)    [github.com/kdj8501](https://github.com/kdj8501) ↗