최원민 | 포트폴리오

About Me.

기술의 'Why'를 탐구하고 구현하는 풀스택 개발자 최원민입니다.

안녕하세요, 저는 명세서대로 구현하는 데 그치지 않고,
사용자의 경험에서 출발해 문제의 'Why'를 정의하고, 그에 맞는 최적의 기술로 'How'를 설계해왔습니다.

min5024@naver.com / 010-4232-5024

Tools & Skills.

주력 언어:

Frontend	Backend	AI	DevOps & Tools
JavaScript Vanilla JS 서비스 최적화 경험 Pub-Sub 기반 단방향 아키텍처	**FastAPI** AI 서버 개발 및 API 구성 경험	**LangChain, LangGraph** RAG 파이프라인 구현 경험 공식 문서를 활용한 개발 가능	**Git** PR/커밋 규칙 수립 경험 브랜치 충돌 해결, 복구 경험
React **TypeScript** 기획 UI 구현 경험 atomic design 컴포넌트 개발	**Django DRF** REST API 구현 경험	**Prompt Engineering** 시스템 프롬프트 작성 경험 추가 학습중	**Figma** 화면 레이아웃 설계 경험 사용자 흐름도 설계 경험
Vue.js 학습, 실습 경험			**AI IDEs** Copilot/Cursor/MCP 사내 code ai(GAUSS) 활용 경험

Education.

삼성청년SW·AI 아카데미 13기

Web Full-stack &
SW·AI 교육

2025.01 - 2025.12

- **13기 우수 수료**
- **삼성전자 네트워크사업부 연계 프로젝트 진행**
- 다수의 프로젝트 팀장 경험 (3회)
- 알고리즘 역량 및 구현 역량 함양

명지대학교

아랍지역학과,
인문ICT콘텐츠 융합전공

2019.03 - 2025.02

- 학부 프로젝트 팀장 경험 (10회)
- 융합 캡스톤 디자인 A+
- 데이터 분석/시각화 역량 (18학점 A+)

Awards.

프로젝트 우수상

삼성전자 주식회사

2025.10.02

- 삼성전자 네트워크사업부 연계 실습 프로젝트 **(1100명 중 상위 3% 우수자 선발)**
- **연간 1,600시간 업무 효율화 창출**
- **최종 임원 성과 발표회에서 우수상(3등) 수상**

성적 우수상

삼성전자 주식회사

2025.05.30

- 삼성청년 SW/AI 아카데미 800시간+ 몰입형 SW 교육 학습과정 **우수 수료**
- 알고리즘 역량, 프로그래밍 역량, AI 역량 함양

Language.

OPIc AL

ACTFL

2025.05

Algorithm.

백준 Platinum 1

solved.ac / 상위 1%

2025

삼성SW역량평가 A+ (모의)

삼성전자

2025

[Github] https://github.com/almeng-E **[Blog]** https://velog.io/@cj98123 **[BOJ]** https://solved.ac/profile/cj98123

1. 삼성전자 네트워크사업부 연계 프로젝트: Web 기반 PCB CAD 협업 플랫폼

2025.08 ~ 2025.11

🏆 **"400명의 VoC 분석으로 연간 1,600시간을 절감하고, 렌더링 성능을 98% 개선한 웹 CAD 솔루션"**

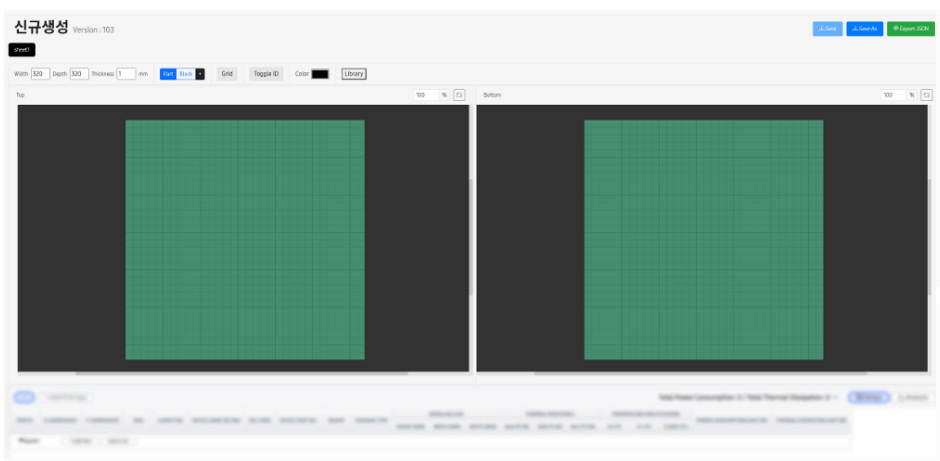
Role
팀장, FE Architect
(기여도 40% - 프론트엔드 코어 로직 및 아키텍처 설계)

Tech Stack
`VanillaJS` `Konva.js`

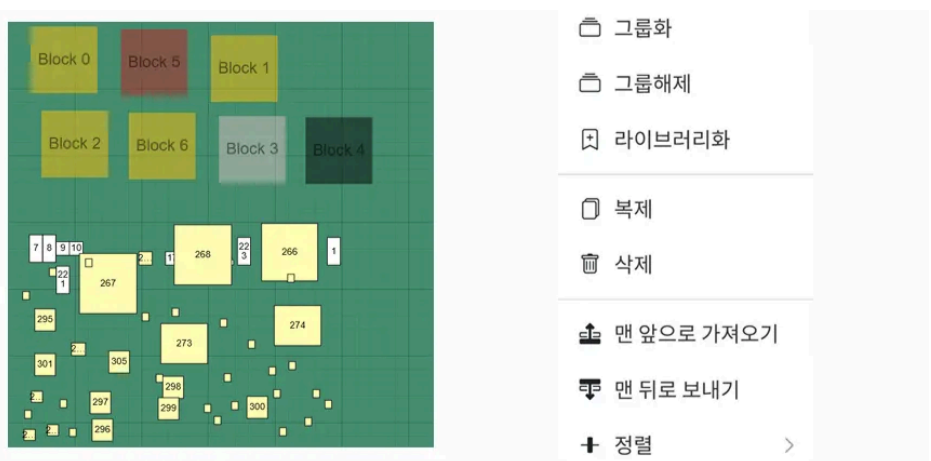
Keywords
캔버스 렌더링 최적화, 공간 분할 알고리즘, VoC 기반 기획, Agile

삼성전자 임원 발표회

우수상(3등) 수상



CAD 메인 화면 (보안으로 일부 블러처리 됨)



CAD 배치 예시 및 편의 기능

S (Situation: 문제 상황)

비즈니스 배경
삼성전자 네트워크사업부 내 PCB 설계 과정에서 여러 툴(Visio, Excel 등)이 파편화되어 있어,
데이터 불일치 및 **중복 설계 작업**으로 인한 **비효율**이 발생하고 있었습니다.

기술적 문제
웹 브라우저 환경에서 2,000개 이상의 부품(Node)을 캔버스에 렌더링하고 조작해야 했습니다.
초기 베타 버전에서는 부품 다중 선택 및 드래그 시 **780ms 이상의 렌더링 지연(Lag)**이 발생하여 실사용이 불가능한 수준이었습니다.

제약 사항
서버 DB 스키마 수정 없이 프론트엔드 단에서 'Ctrl+G 임시 그룹화' 기능과 같은 편의 기능들을 구현해야 했으며,
6주라는 한정된 기간 내에 실무에 적용 가능한 수준의 최적화가 필요했습니다.

T (Task: 역할 및 과제)

- **기획:** 현업 실무자들의 요구사항(VoC)을 수집하여 '중복 설계 제로'를 목표로 초기 명세서 너머의 **통합 협업 툴 기획**
- **개발:** 대량의 객체가 있는 캔버스 환경에서 끊김 없는(60fps) 사용자 경험을 제공하기 위해
  1. **캔버스 기능의 렌더링 성능 최적화**
  2. VanillaJS 기반 **커스텀 이벤트 아키텍처** 설계

A (Action: 해결 과정 및 기술적 의사결정)

1. 기획 및 소통: VoC 기반 의사결정

- **VoC 수집**
  1. 매 주 각 팀의 **정기 회의에 참여**하여, **시연**을 통해 개발 상황 공유 및 파트장(부장)급 임원의 VoC 수집
  2. 베타 오픈 후, 400여 명의 실무자 대상의 **공청회**를 통해 시연과 VoC 수집
- **의사결정**

  요구사항을 **ICE Score(Impact, Confidence, Ease)** 기준으로 우선순위화하고,
  최대 임팩트 기능을 먼저 스프린트에 반영한 주간 정기 배포로 개발을 진행

2. 성능 최적화: Uniform Grid 공간 분할 알고리즘 도입

- **문제 분석**

  클릭, 드래그시 Konva.js 구현체의 *나이브 AABB* $O(N)$ 전체 순회 방식으로는 2,000개 객체의 충돌 감지 및 렌더링을 감당할 수 없음을 확인
- **기술 검토**
  1. *Quad-Tree:* 검색 효율은 좋으나, CAD 특성상 부품 이동이 빈번하여 트리 재구성(Re-balancing) 비용이 과다하다고 판단하여 배제
  2. *Dynamic BVH Tree* : 게임 업계에서 AABB의 최적화로 주로 채택되지만, 장기 운영 관점에서 복잡도가 크기에 오버엔지니어링이 우려되어 배제
- **해결책: Uniform Grid**

  도메인 특성상 부품 크기가 균일하고 맵 크기가 고정적이라는 점에 착안, 캔버스를 10x10 격자로 나누는 **공간 해싱(Spatial Hashing)**을 구현
  - Point Query (클릭) : $O(N) \rightarrow Amortized\ O(1), worstcase\ O(N)$
  - Range Query (드래그) : $O(N) \rightarrow O(k+M)$
    - k : 선택된 버킷 수, M : 선택된 부품 수
    - 보통 $k+M << N$

3. 아키텍처: VanillaJS 기반 이벤트 설계

- **문제 분석**

  캔버스(Konva)뿐 아니라 Table/패널/단축키 등 **페이지 기능이 확장**되면서 이벤트가 여러 뷰에 흩어져 **상태 변경 경로가 복잡**해졌고,
  그에 따라 **데이터 정합성/유지보수 리스크**가 증가
- **기술 검토**

  각 뷰가 서로 직접 참조하며 통신하면 결합도가 급증하므로, **중앙 이벤트 허브 + 단방향 데이터 흐름**이 필요하다고 판단
- **해결책: EventEmitter + Singleton EventBus**

  전역 Singleton으로 **EventBus(EventEmitter)** 를 두고, 모든 상호작용을 `emit`과 `on`으로 통일

  EventBus를 통해 **중앙 상태 모델(SSoT)**을 갱신, 각 뷰(Canvas, Table 등)는 이벤트를 구독해 **상태 변화만 반영**하도록 구성
  - 결과적으로 **단방향 통신**으로 결합도를 낮추고, 정합성과 유지보수성을 개선

R (Result: 성과 및 배운 점)

💰 **비즈니스 성과**
중복 설계 프로세스를 제거하여 **연간 1,600시간의 업무 시간을 절감**할 수 있는 기반을 마련했습니다.

📈 **성능 개선**
500개 부품 기준 렌더링 지연 시간을 **780ms → 12ms로 98% 단축**하여 쾌적한 UX를 제공했습니다.

🏅 **조직 성과**
프로젝트의 완성도를 인정받아 계약 기간이 6주 연장되었으며, 삼성전자 주식회사로부터 **프로젝트 우수상**을 수상했습니다.

✨ **배운 점**

- **VoC는 '많이 듣기'가 아니라 '우선순위화'가 핵심이다**
  - ICE로 결정 기준을 명문화해 이해관계자 합의를 빠르게 만들고, 주간 배포로 개발 속도를 올림.
- **성능 최적화는 '측정 → 병목 규명 → 가설 검증' 루프로 접근해야 한다**
  - 렌더링 지연 원인을 AABB 전체 순회로 특정하고, 공간 해싱 도입으로 복잡도를 구조적으로 낮춤.
- **도메인 제약(CAD 특성)이 자료구조 선택을 결정한다**
  - 부품 이동이 잦아 Quad-tree의 재구성 비용이 커지고, Dynamic BVH는 운영 복잡도가 높아 배제.
  - "가장 빠른 알고리즘"보다 "우리 문제에서 가장 지속 가능한 선택"이 중요함.
- **상각 시간(amortized)과 최악 시간(worst-case)을 함께 설명해야 신뢰가 생긴다**
  - 클릭/드래그 시나리오별로 기대 성능과 최악 케이스를 분리해 커뮤니케이션.
- **클라이언트 상호작용이 복잡해질수록 상태 정합성이 핵심 과제가 된다**
  - 대량 객체 캔버스(Konva) 환경에서 선택/드래그/그룹 기능이 늘어나며 상태 불일치와 사이드 이펙트가 쉽게 발생함을 체감.
  - 제약이 있는 VanillaJS 환경에서 Pub-Sub 기반 이벤트 버스를 설계·구현해 이벤트 흐름을 일원화했고,
    View 컴포넌트 결합도를 낮춰 유지보수성을 개선, 확장성을 고려함.
- **코드 보조 도구를 써도 '코드 주권'과 책임은 개발자에게 있다 (GAUSS 기반 바이브코딩)**
  - GAUSS로 구현 속도를 높였지만, 최종 코드는 내가 설계 의도·예외·성능 특성을 설명 가능한 수준으로 정제해야 했음.
  - 생성 코드를 그대로 쓰기 보다 테스트·리팩터링·문서화로 내가 소유하고 유지보수 가능한 코드로 만드는 습관을 얻음.

2. AI 인터랙티브 그림책 플랫폼: PicTale

2025.07 ~ 2025.08

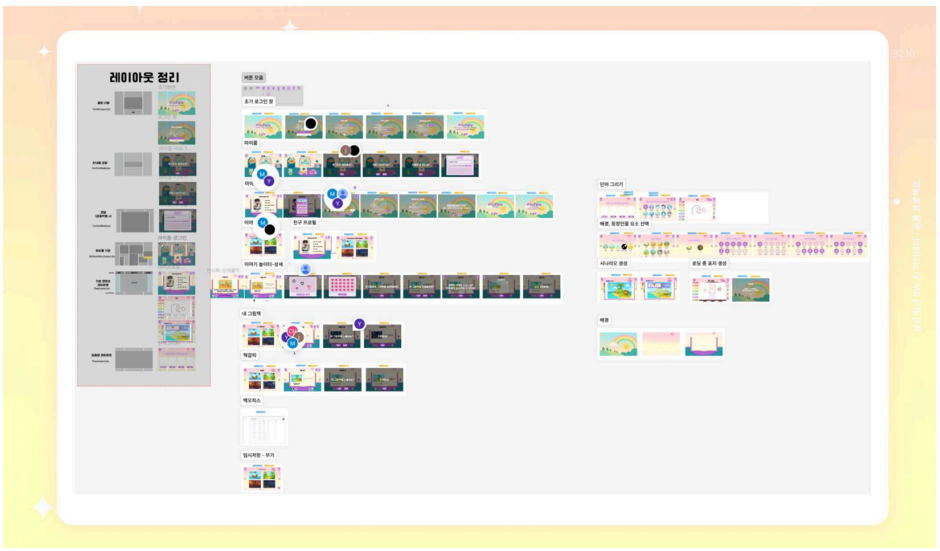
📚 **"유아 인지 특성을 고려한 UX 설계와 AI 파이프라인 최적화로, '글을 몰라도 작가가 되는' 경험 제공"**

Role
팀장, 기획, FE, AI
(기여도 50% - UX/UI 총괄, AI 파이프라인 최적화)

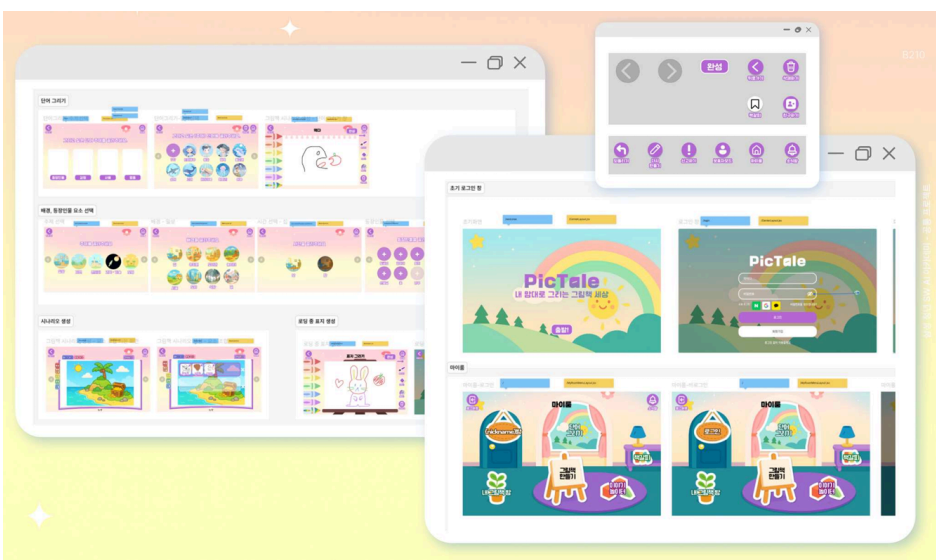
Tech Stack
`React` `TypeScript` `LangChain` `Atomic Design`

Keywords
UX 개선, 시장 분석, RAG Pipeline, Vibe Coding

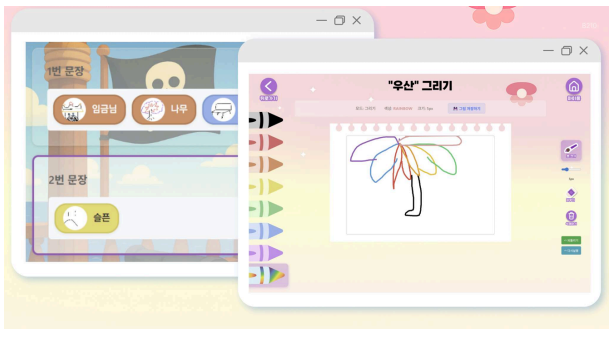
우수 프로젝트 선정 및 아시아창업엑스포(FLY ASIA) 2025, BEXCO 포스터 전시



Figma 사용 유아용 레이아웃 기획



유아 친화적인 컬러로 화면 구성



블록 선택 + 드로잉 중심 UI/UX

클릭으로 생성하는 그림책

유저의 그림을 기반으로 완성된 그림책

S (Situation: 문제 상황)

팀 방향성 혼란
초기 서비스 정체성이 교육 앱 vs 놀이 앱 vs 단순 생성 앱 사이에서 흔들리며, 핵심 가치와 구현 우선순위가 불명확했습니다.

시장 분석
기존 AI 동화 서비스들은 텍스트 입력 중심이거나 성인 UI를 그대로 사용해, 실사용자인 3~6세 유아가 주체적으로 사용하기 어려웠습니다.

기술적 문제
삽화의 캐릭터/스타일 연속성을 확보하는 데에 문제가 있었습니다.
Img2Img와 Text2Img 방식의 속도·비용·연속성 트레이드오프 문제가 있었습니다.

T (Task: 역할 및 과제)

- **기획:** 경쟁사/시장 조사 기반으로 차별점과 컨셉(니치)을 정의하고 팀 목표를 정려
- **FE:** UX 설계, 텍스트 입력 없는(Non-typing) 인터페이스와 유아의 신체/인지 특성을 반영한 UI 설계 및 구현
- **AI :** 삽화 연속성을 유지하면서 지연/비용을 낮추는 효율적인 생성 파이프라인 설계, 캡셔닝 메타데이터 레이어 추가

A (Action: 해결 과정 및 기술적 의사결정)

1. 기획 및 소통: 시장조사 기반 아이디어/방향성 수립

- **시장 조사**

  국내외 유사 서비스(ex. Gemini Storybook, StoryWizard, Bedtime Stories 등)를 설치·분석

- **아이디어/방향성 수립**

  “텍스트 시작 + 성인 UI”가 시장의 공통 한계임을 정리

  놀이 + 드로잉 + 공유 라는 키워드로 서비스 컨셉 제안

  확정된 컨셉을 구현 과제로 분해해 (Non-typing UX / 연속성 / 속도) 개발 우선순위를 명확화

2. FE: 논문 기반 UX 설계 + Vibe Coding으로 개발 효율화

- **UX 설계**

  1. Fitts's Law와 아동 터치 인터랙션 연구(*Anthony et al.*)을 근거로, 유아의 조작 정확도를 높이기 위해
     버튼/핵심 인터랙션 요소를 성인 권장 규격**(48px)의 2배인 96px**으로 **설계**
  2. 유아용 UI는 기성 컴포넌트 라이브러리로 일관되게 맞추기 어려워 **Atomic Design**으로 버튼/모달 등 기초 컴포넌트 직접 구성

- **바이브 코딩**

  Figma MCP와 GitHub Copilot을 활용한 **Vibe Coding**을 도입해 반복 구현 시간을 단축 (약 50%, PR 리드타임 기준)
  확보된 시간을 UX 디테일(애니메이션/피드백) 개선에 재투자

3. AI 파이프라인 최적화: 메타데이터 캡셔닝 레이어로 연속성 확보 + 비용/속도 최적화

- **문제 분석**

  Img2Img는 연속성에 유리하지만 **지연이 커서** 유아 UX에 부적합
  반대로 Text2Img는 빠르지만 **연속성이 불분명**

- **기술 검토**

  Imagen3 모델 기준, 두 방법의 토큰 비용 동일

  1. 직렬 생성/Img2Img : 페이지를 1→2→3… 순서로 생성하면 컨텍스트 전달이 쉬우나 총 지연이 커짐
  2. 병렬 생성/Text2Img : 묶음(1~4, 5~8)으로 생성하면 속도에 유리하나, 캐릭터/스타일 연속성이 깨질 수 있음

- **해결책:** 캡셔닝 레이어 추가, 파이프라인 재설계 (생성 단계가 아닌, **드로잉 단계**에서 전처리 레이어 추가)

  사용자 그림의 캐릭터 특징/스타일/핵심 요소(색, 표정)를 **캡션 메타데이터**로 추출해 추후 생성 단계에서 **컨텍스트로 주입**

  - Text2Img, 병렬 비동기 호출로 전환하여 **생성 속도 ~75% 단축**
  - 병렬 생성에 동일한 메타데이터 컨텍스트를 공유해 **연속성 저하를 완화**

R (Result: 성과 및 배운 점)

✅ **차별된 서비스 기획**
유사 AI 그림책 서비스 3개를 분석해 **텍스트 시작·성인 UI의 한계**를 정의하고, **유아 중심 Text-free 그림책**으로 차별화 기획을 수립했습니다.

📈 **성능 개선**
Img2Img 중심 파이프라인을 **캡셔닝 + Text2Img** 기반으로 재설계해, **10페이지 생성 기준 대기시간을 약 75% 단축**했습니다.

🏅 **조직 성과**
프로젝트 가치를 인정받아 **BEXCO '아시아창업엑스포 2025' SSAFY관에 전시**되었습니다.

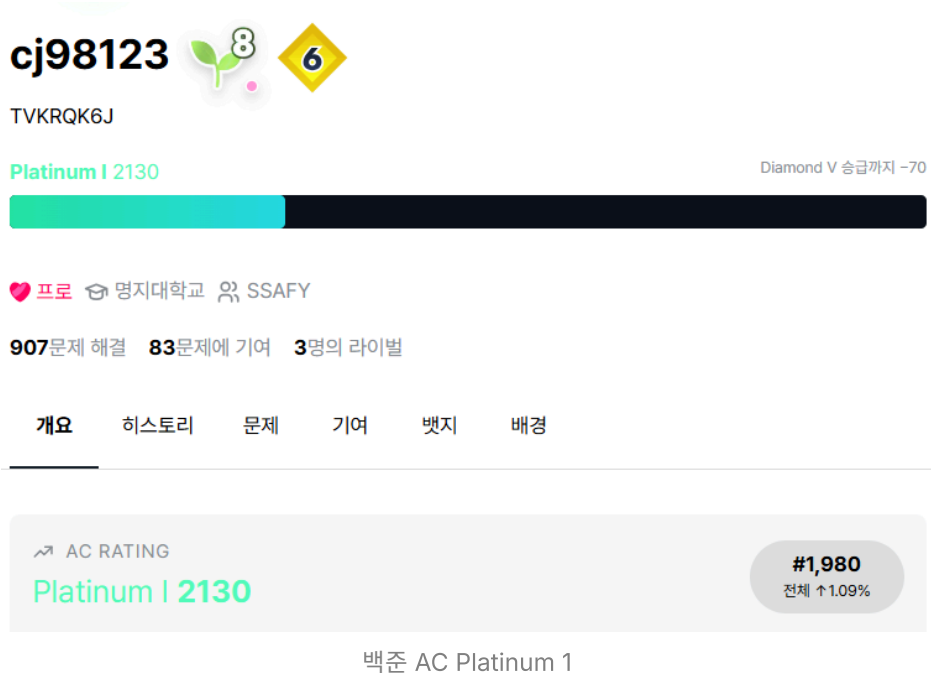
✨ **배운 점**

- **방향을 잃은 팀은 '의견'이 아니라 '근거'로 이끌어야 한다**
  - 시장/경쟁사 분석을 매트릭스로 정리해 "교육이 아닌 놀이, 텍스트가 아닌 드로잉"으로 컨셉을 확정,
    이를 구현 과제(Non-typing UX·연속성·속도)로 분해해 팀 목표를 정렬함.
- **좋은 제품은 혼자 만드는 게 아니라, 피드백 루프에서 완성된다**
  - 컨설턴트·팀원·외부 팀 피드백을 반복적으로 반영하며 UX/플로우를 개선,
    "공유 가능한 산출물(시연/정리/우선순위)"이 개발 속도를 높인다는 걸 체감.
- **프론트엔드 개발에서 TypeScript는 '생산성'이 아니라 '품질/디버깅 비용'의 문제다**
  - 타입이 인터페이스 역할을 하면서 런타임 버그와 디버깅 시간을 줄였고, 협업 시 변경 영향 범위를 예측하기 용이.
- **멀티스택 협업에서는 '경계(Contract)'를 분리하는 설계가 핵심이다**
  - Java 메인 서버가 비대해지는 문제를 경험하며, AI 처리는 FastAPI 같은 별도 서비스로 분리해
    책임을 나누는 이유(배포/스케일/개발 속도)를 이해.
- **AI 적용은 쉽지만, '성능·비용·품질' 트레이드오프를 설계하는 건 어렵다**
  - Img2Img의 비용/지연과 연속성 요구 사이에서, 캡셔닝 메타데이터 기반 파이프라인과 같은
    균형점을 찾는 과정이 'AI 제품화'의 핵심임을 이해.
- **팀장 역할은 개발뿐 아니라 '일정/갈등 관리'까지 포함한다**
  - 기여가 적은 팀원으로 일정이 흔들렸을 때, 개인 비난이 아니라
    팀 목표·약속·할 일을 재합의해 실행력을 회복시키는 것이 리더의 역할임을 체감.

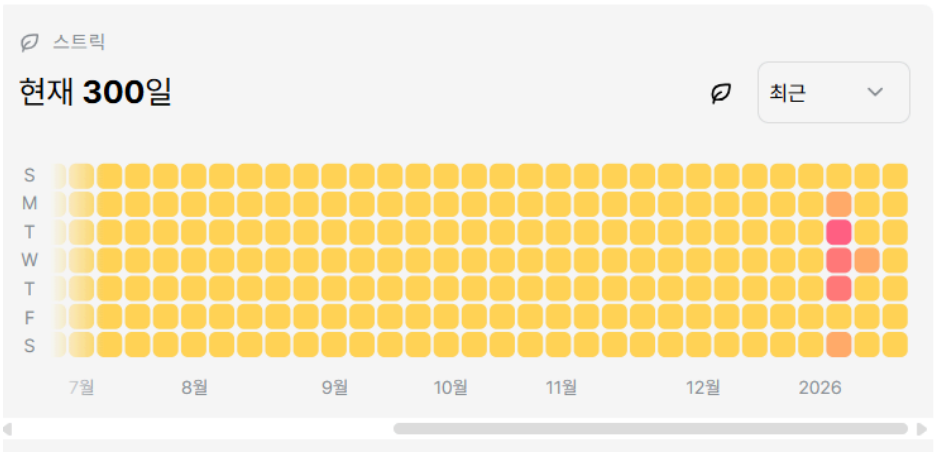
3. DP 알고리즘 스터디 주최 경험

2025.03 ~ 2025.04

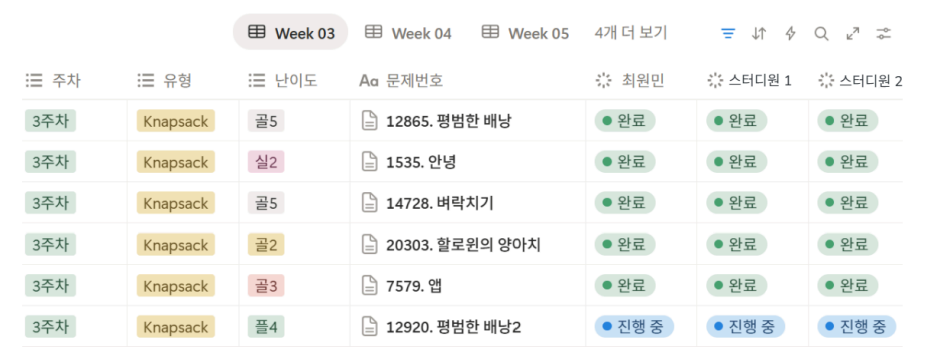
💻 **"공유를 통한 학습 실천"**



백준 AC Platinum 1



300일 연속 문제 풀이



주차별 DP 문제 풀이

S (Situation: 문제 상황)

스터디 주최 배경
코딩테스트 핵심 유형인 DP는 진입장벽이 높았고, SSAFY 정규 과정에서 충분히 다뤄지지 않아 **동료들 사이에서 학습 포기가 발생**했습니다.

개인적 어려움
저 역시 DP의 핵심인 **상태 정의·점화식 설계·Top-down/Bottom-up 선택**이 익숙하지 않아,
시행착오를 줄일 **체계적인 학습 구조와 피드백 루프**가 필요했습니다.

스터디 진행 중 문제
난이도가 올라갈수록 이탈이 발생하며 **주차별 문제 풀이 참여율이 저하**되었습니다.

T (Task: 역할 및 과제)

- **기획 (커리큘럼):** 주차별 유형 로드맵 설계 및 문제 큐레이션
- **운영 (스터디장):** 참여율 유지/개선, 학습 문화 정착, 팀원 동기부여
- **피드백 루프:** 코드 리뷰 + 상호 코드 피드백 구조 설계 및 운영

A (Action: 해결 과정 및 기술적 의사결정)

1. 기획: AI 활용 커리큘럼 설계

- Perplexity/ChatGPT를 활용해 DP 학습 경로를 설계하고, 난이도 곡선을 고려해 주차별로 문제를 큐레이션
- **1차원 DP → 2차원 DP → 동전 → 배낭 → TSP** 순으로 단계적 커리큘럼 구성

2. 스터디 운영: 운영 방식 고도화

- **초기(토론 중심):** 각자 풀이 공유·토론 방식으로 운영
- **전환(강의 + Q&A):** 난이도 상승으로 이탈이 발생→ 매주 대표 문제를 미리 풀고 다음 내용을 중심으로 정리해 **강의형태로 운영** 전환
  - 상태 정의를 **DAG 관점**으로 정리
  - **Top-down vs Bottom-up 선택 기준** 제시
  - 자주 틀리는 포인트(초기값/전이/중복 계산) 체크리스트화

3. 공유 시스템 구축: Notion + 코드리뷰 문화 정착

- Notion에 주차별 문제/풀이/핵심 패턴(상태·전이·복잡도)을 정리해 **복습 가능한 지식베이스** 정리
- 오프라인 코드리뷰 피드백 루프를 통해 동료들과 코드 리뷰 문화 확보

R (Result: 성과 및 배운 점)

👥 **스터디 성과**
스터디 인원이 **4명 → 9명**으로 증가했고, 주차별 문제 풀이 참여율을 **30% → 80%**로 개선했습니다.

📈 **개인적 성장**
DP 문제 접근이 체계화되었고, 공유+소통 기반 학습 태도를 유지하며 최근에는 블로그 정리로 이를 이어가고 있습니다.

✨ **배운 점**

- **공유를 통한 성장은 빠르고 강력하다.**
  - 코드 리뷰와 토론을 통해 서로의 사고 과정 비교
    혼자 학습 할 때보다 훨씬 빠르게 성장함을 체감.
- **DP의 핵심은 '상태 정의'다.**
  - 점화식보다 먼저, 문제의 '상태'를 정의하는 것이 우선
    Top-down/Bottom-up은 DAG 상태 정의 이후의 일
- **팀원 독려는 '의지'가 아니라 '운영 설계'로 해결된다.**
  - 참여율 저하는 개인 의지 문제가 아니라,
    난이도 곡선·준비 부담·피드백 구조 같은 운영 구조의 문제일 수 있음을 이해.
- **알고리즘 PS는 추상화된 훈련도구다.**
  - 문제를 구조화하고, 가설을 세우고, 검증하는 과정에서 엔지니어링의 본질적인 매력을 느낌.

- 끝 -