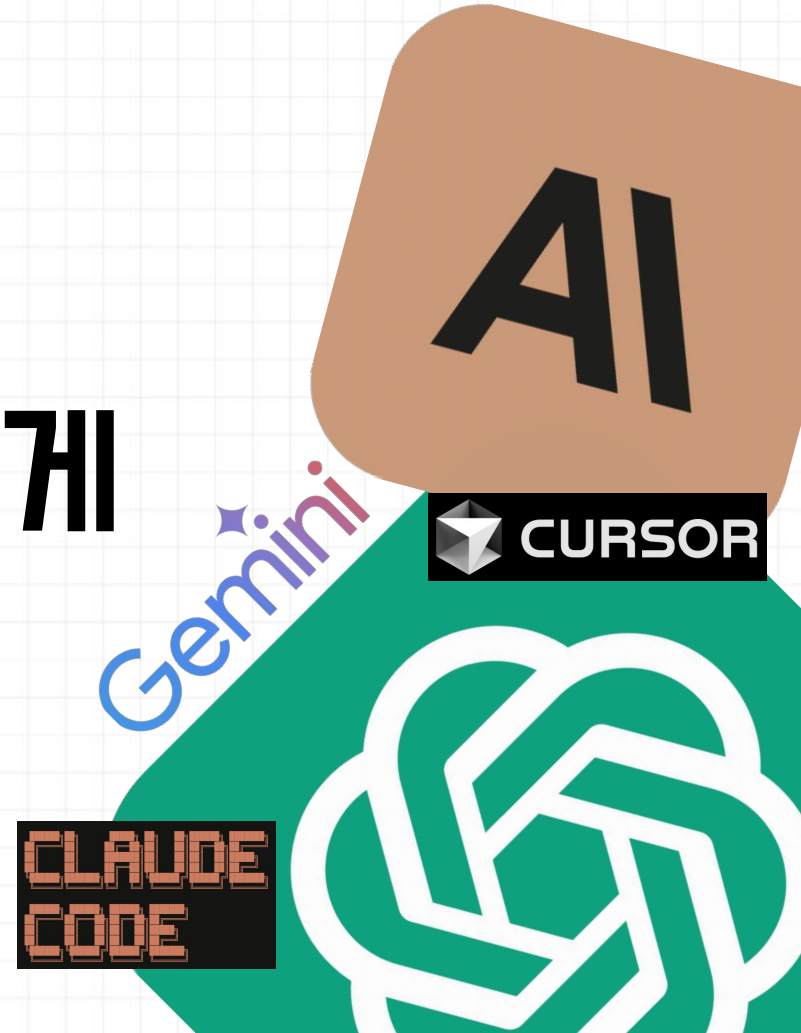
AI에게 대체될까 두려운 개발자들에게

2025 원티드 AX 컨퍼런스

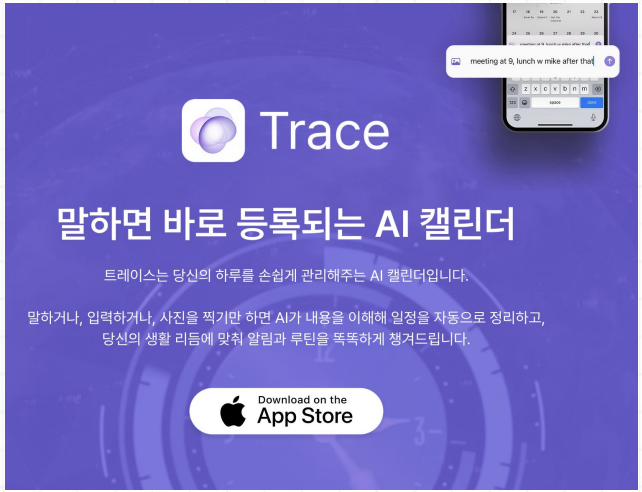
2025.10.16 by 배휘동
블로그, Facebook, LinkedIn, Thread
구독 및 1촌 요청 환영



배휘동

(전) XL8 프론트엔드 팀 리드

(현) 코르카 AI 프로덕 엔지니어 + AX 컨설턴트



개인 / 기업

개발 경력(?) 대략 15년

- LG전자 인턴
- 전산학과 대학원
- 창업 (습관형성 SNS)
- 모바일 인디게임
- 모바일 광고
- 코딩 교육
- 금융 & 소상공인
- AI 미디어 번역

2025년~: 커리어 확장

- 코치 & 교육자 & 컨설턴트
- AX 컨설턴트
- AI Product Engineer

많은 학습과 도전과 실험 예정!

Q. (원티드 담당자께 질문) 제 세션을 어떤 사람들이 듣게 될까요?
듣고 나서 어떤 변화가 생기길 기대하시나요?

A. AI의 발전으로 직무 생존을 걱정하는 개발자,
특히 5년차 이하 주니어들에게
('정답'은 아닐지라도) 현실적인 자신감이 생기면 좋겠습니다

https://www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app

This article is more than **2 years old**

ChatGPT reaches 100 million users two months after launch

Unprecedented take-up may make AI chatbot the fastest-growing consumer internet app ever, analysts say

2022.11.30: ChatGPT 첫 릴리즈

2023.02.03: "ChatGPT가 2개월만에 1억 사용자를 돌파"

"AI가 인간을 다 대체해버리는 거 아니야?"

https://www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app

This article is more than **2 years old**

ChatGPT reaches 100 million users two months after launch

Unprecedented take-up may make AI chatbot the fastest-growing consumer internet app ever, analysts say

2022.11.30: ChatGPT 첫 릴리즈

2023.02.03: "ChatGPT가 2개월만에 1억 사용자를 돌파"

"당신을 대체할 무언가는
AI 자체가 아닌 AI를 사용하는 사람이다."

ChatGPT 첫 릴리즈 이후 대략 3년간
노동 시장에 어떤 변화가 있었을까요?

이번 발표를 준비하며 좀 찾아봤습니다.

https://www.news1.kr/it-science/general-it/5785277

AI의 인간대체 시작되나…인텔 2.2만명·MS 6천명 해고 칼바람

김민석 기자

업데이트 2025.05.19 오전 11:05

가

인텔·MS·메타·구글 등 잇단 대규모 해고…IT 일자리 위기 고조
엔지니어 감원 현실화…엔트로픽 CEO "AI가 코드 90%작성"

수시로 들려오는 해고 뉴스들을 보면 AI로 인한 인력 감축이 현실화되는 걸로 보임

https://www.sedaily.com/NewsView/2GWITX3SKH

맥킨지 "5000명 자르고 AI 직원 1만 개 채용"…AI 때문에 '업계 1위'도 망할 판?

입력 2025-08-06 23:19:02 수정 2025.08.06 23:19:02 강신우 기자

https://x.com/infraa_/status/1973362080209383734

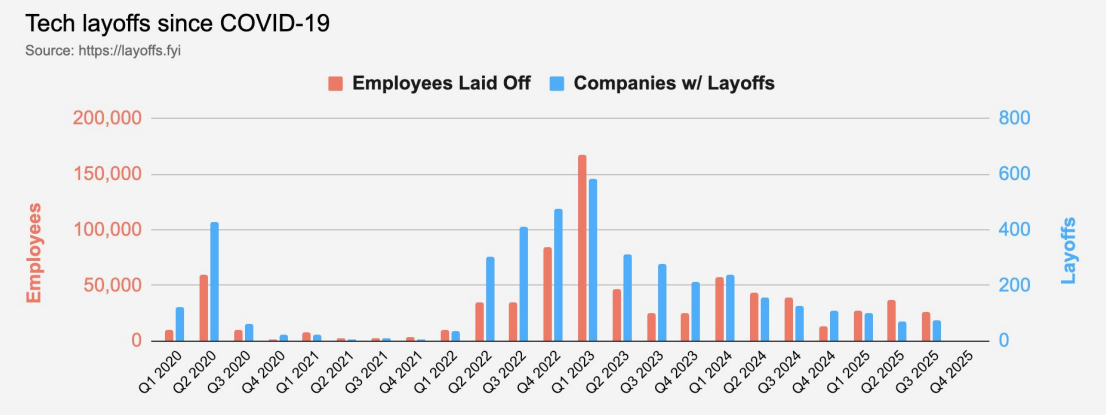
참고로 위 Job Opening 통계의
원 출처는 미국 노동통계국

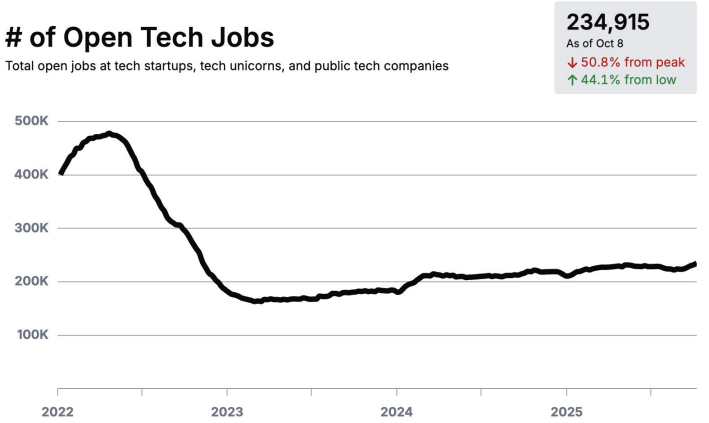
ChatGPT 릴리즈 이후
신규채용은 줄어드는
동시에 S&P 500은
고공행진 중이라는 트윗이
크게 바이럴되기도 했음

그러나...

https://layoffs.fyi/

https://www.trueup.io/layoffs





실제로는 코로나 때 이율 낮아지고 돈 풀려서 많이 뽑았다가 경제 어려워지니까 (특정 기업에서) 많이 해고했을 뿐

오히려 기술기업에서는 ChatGPT 이후 지난 3년간 조금씩 신규채용이 늘고 있음

https://budgetlab.yale.edu/research/evaluating-impact-ai-labor-market-current-state-affairs

2025년 10월 연구자료

Figure 1. Changes in the Occupational Mix Over Different Periods of Technological Change

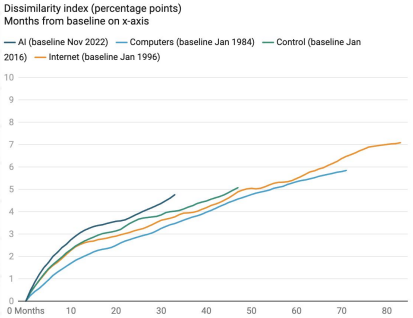
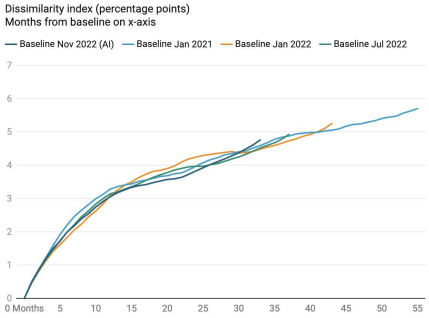


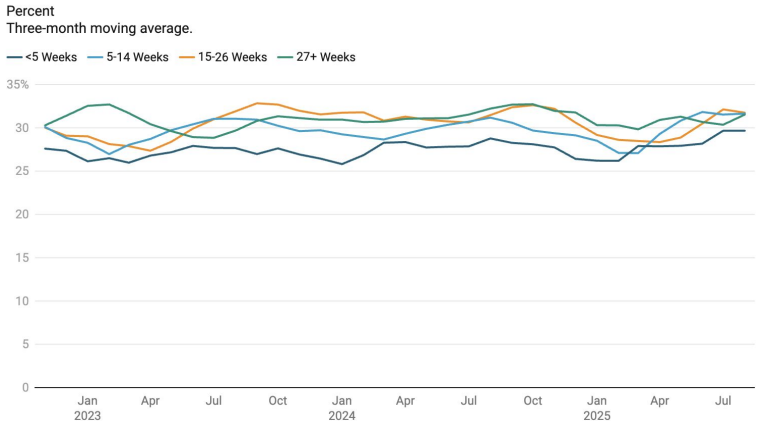
Figure 2. Changes in the Occupational Mix From Recent Baselines



Q. AI가 직업 구성 분포(전체 노동자 중 개발자가 몇%, 사무직은 몇%, ...)를 크게 바꾸고 있나? 즉, AI로 인해 사람들이 다른 일자리를 찾고 있나?

A. 아님. 컴퓨터/인터넷 도입 초기와 딱히 변화 폭 유사. 그 변화도 ChatGPT 출시(2022년 11월) 이전부터 있었던 것이라 AI 때문이라고 단정할 수 없음

Figure 12. Proportion of Tasks Exposed to AI Amongst Unemployed Workers by Duration of Unemployment



Q. 실업 기간이 긴 사람들의 AI 노출도(특정 업무의 수행 시간을 AI로 최소 50% 이상 단축할 수 있음)가 특별히 높았나? 즉, AI로 인해 실업하는 사람이 늘고 있나?

A. 아님. 실업 기간에 관계없이 AI 노출도가 25-35% 수준으로 비슷함. AI로 인한 실업 증가 증거는 아직 없음

https://ianpark.vc/p/itstheeconomystupid

주간실리콘밸리

인공지능이 벌써 주니어들을 대체했다? 문제는 경제야, 바보야

오늘도 FOMO를 조장하며 전문가인척 숟가락 얹는 호들갑쟁이들을 잡으러 주반꿀이 왔다

기업들은 아직 인공지능을 제대로 도입하지도 못했습니다

그렇다면 기업들의 인공지능 도입에 대해서 좀 더 생각해볼까요?

- **BCG (2024):** 전 세계 기업 중 74%는 인공지능으로 실질적 가치를 창출하지 못하거나, 규모 확장(scaling)에 실패하고 있음.
- **MIT NANDA (2025):** 95%의 GenAI 파일럿이 ROI 없음 (근데 솔직히 이 보고서도 약간 과장된 면이 없지않아 있다고 생각합니다.)
- **US Census (2025):** 대기업 AI 도입률 14%→12% 하락.

AI가 노동시장에 어떤 영향을 미치고 있는지 판단하기는 아직 너무 이름

- 지나친 호들갑에 현혹되지 말되, 이 변화의 흐름에는 잘 올라타보자
- 'AI 잘 쓰는 개발자'의 가치는 지금도 높고 향후 몇년간 더 커질 거라고 봄

https://www.apple-economy.com/news/articleView.html?idxno=77285

"생성AI, 누구의 일자리도 빼앗지 않았다"

엄정원 기자 | 입력 2025.10.02 11:57 | 댓글 0

생성AI로 인한 대체율 1% 미만, "1~3차산업혁명때와 비슷"
일자리 대체 대부분은 "AI가 아니라, 경제 전반 상황 때문"
'정보·미디어 부문은 비교적 영향 커, 금융 부문도 다소 영향
예일대 연구, "장기적으로 어떤 영향 미칠지 예측하기엔 일러"

그러나 연구진은 "이러한 변화(퇴사, 이직 등)를 생성AI의 탓으로 돌려선 안 된다"고 지적했다.

즉 이러한 변화는 언뜻 보기에 생성 AI에 기인하는 것처럼 보일 수 있지만, 데이터를 분석해보면 다르다는 주장이다. 연구에 따르면 이미 이런 산업 내 추세는 챗GPT 출시 이전에 시작되었음을 시사한다. 그래서 "더 넓은 시각에서 볼 때, 정보 산업의 큰 변화는 특정 기술 발전의 결과라기보다는 산업 자체의 특성인 것으로 보인다"는 예일대 연구진의 주장이 설득력을 얻고 있다.

이에 따르면 "생성 AI가 장기적으로 어떤 영향을 미칠지 예측하기에는 아직 이르다"는게 결론이다. AI가 노동 시장에 미치는 영향은 경제 전반에 걸친 중대한 혼란보다는, '안정성'을 주로 반영한다는 논리다.

생성 AI가 변혁적이고 범용적인 기술임엔 분명하다. 그러나 이런 기술이 향후 일자리에 얼마나 큰 파괴적 영향을 미칠지는 아직 예측하기 어렵다는 주장이다. 실제 연구에서도 "기술 발전 초기 단계에서 광범위한 영향이 나타나지 않는 것은 이전 기술 혁신(1~3차산업혁명) 시기의 변화 속도와 크게 다르지 않다"는 것이다.

https://futurism.com/artificial-intelligence/yale-study-ai-job-impact

BUT WHAT ABOUT THE WEAK JOB MARKET?

New Yale Study Finds AI Has Had Essentially Zero Impact on Jobs

Excuse Me?

By **Sharon Adarlo** / Published **Oct 5, 2025 10:30 AM EDT**

https://budgetlab.yale.edu/research/evaluating-impact-ai-labor-market-current-state-affairs

(여담) 한국발 뉴스기사에는 원출처가 안나와있는 일이 참 많은데 조금만 검색해봐도 원문과 더 좋은 분석 기사가 잘 나옴. AI와 함께 읽어보면 재미있음. 쉽게 현혹당하지 않고 생각의 힘을 기르는 습관이기도 함

관심있으신 분은 [딥 리서치 링크] 참조

🙏 Disclaimer

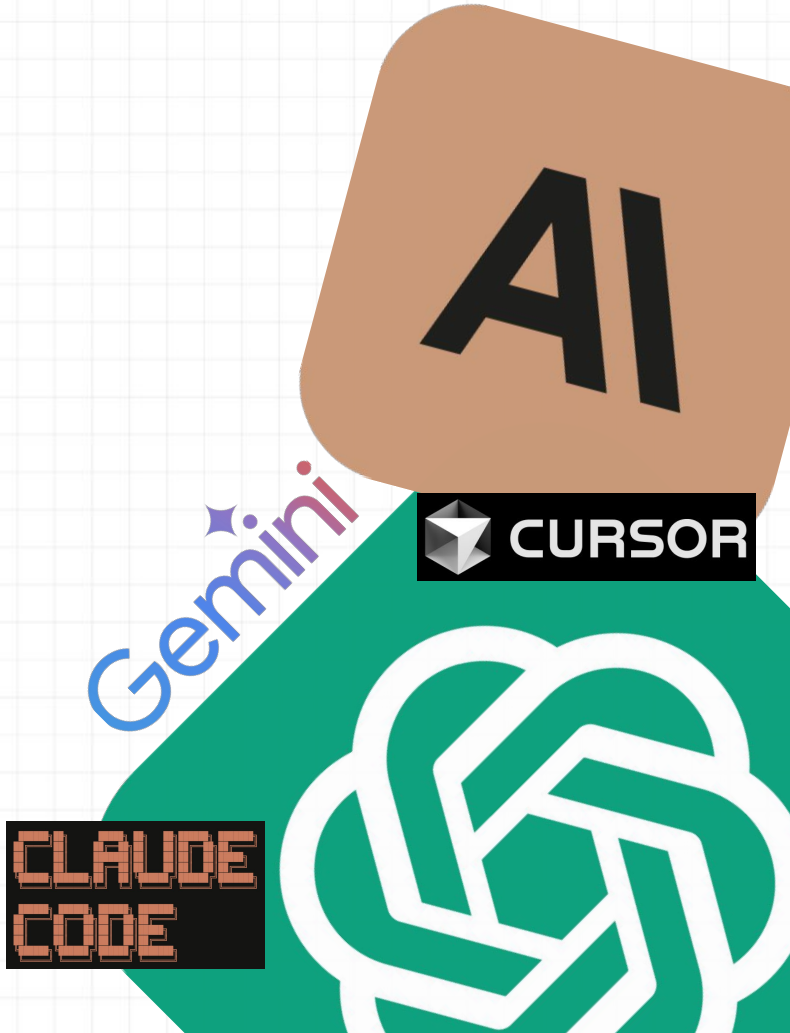
- 모든 이야기는 2025년 10월 중순 기준
  - 하루살이 신세. 그제 얻은 인사이트를 어제 실험해보고 오늘 떠들고 있음
  - 오늘 하는 이야기의 유통기한도 길어야 3개월
- 'AI 전문가', '바이브 코딩 전문가' 아님. 사실 어떤 분야의 전문가도 아님
  - 에이전트, RAG, MCP, 대규모 트래픽, 백엔드, 인프라... 모르는 것 너무 많습니다
- 글쓰기 & 공부 좋아하고, '바이브 코딩' 트렌드를 조금 일찍 접해 운좋게 이름을 조금 얻었고, 경험과 고민을 조금 누적했을 따름
- 제 이야기에 섞여있을 환각을 최대한 의심하며 들어주시길
  - 의심과 호기심은 AI 시대의 최대 덕목!

📜 오늘의 순서

1. AI의 미래와 개발자의 미래
2. 이 AI 시대에 나는 어떤 노력을 하며 살아가고 있는가
3. 앞으로의 커리어를 위해 주니어들에게 권하는 마음가짐
4. 마무리 및 Q&A

1부

AI의 미래와 개발자의 미래



대략 2024년까지만 해도
코딩에서의 AI 역할
= 코드 자동완성 도우미

내가 경험한 자동완성 도우미

- ~ 2010년대 후반: VSCode IntelliSense
- 2019년~: TabNine - 최초의 'AI' 도우미 표방
- 2022년~: GitHub Copilot
- 2023년~: ChatGPT + Copilot Chat
- 2024년~: Cursor Tab

내가 경험한 자동완성 도우미 (제대로 안써본 거 많음)

2025년: 온 세상이 코딩 에이전트다

AI-Native IDE

- Cursor
- WindSurf
- Cline
- Roo Code
- Kiro
- Trae
- Augment Code
- Zed
- VSCode

AI-Native CLI

- Claude Code
- Codex CLI
- Gemini CLI
- Qwen Code
- Grok CLI (비공식)
- Cursor CLI
- Canva CLI

0-to-1 프로토타이핑

- Lovable
- v0
- Replit
- Firebase Studio
- Bolt
- Tempo
- Emergent
- Figma Make
- Fusion

LLM Canvas

- Gemini Canvas
- ChatGPT Canvas
- Claude Artifact
- Grok Canvas

Background Agent

- Codex BA
- Google Jules
- Claude Code BA
- Cursor BA
- Copilot BA

어디까지가 '자동완성'인가?

자동완성으로 AI가 만들어내는 코드의 양이 현저히 많아짐

기존에는 이정도였는데

- 언어 엔진 및 컴파일러가 제공하는 목록에서 선택
- 탭 쳐서 받은 코드 스니펫 승인

이제는 0-to-1(그린필드) 도 되고

- 필요한 앱을 기획해서 AI와 함께 구현
- 채팅하며 생각난 앱 그 자리에서 제작

1-to-N(브라운필드) 도 됨

- 기존 코드에서 채팅으로 "여기 고쳐줘"
- 깃헙 이슈만 던지고 "해결해"

https://en.wikipedia.org/wiki/Punched_card

https://en.wikipedia.org/wiki/Machine_code

그런데 애초에 '코딩'이라는 게 뭐지?

옛날에는 펀치카드 뚫는 것도 코딩, 기계어 작성도 코딩이었음

```
    1  FDX      12:01a  23-  1
A 002000  C2 30       REP #$30
A 002002  18          CLC
A 002003  F8          SED
A 002004  A9 34 12    LDA #$1234
A 002007  69 21 43    ADC #$4321
A 00200A  8F 03 7F 01 STA $017F03
A 00200E  D8          CLD
A 00200F  E2 30       SEP #$30
A 002011  00          BRK
A 2012

r

  PB  PC   NVmxDIZC  .A   .X   .Y   SP   DP  DB
; 00 E012  00110000 0000 0000 0002 CFFF 0000 00
g 2000

BREAK

  PB  PC   NVmxDIZC  .A   .X   .Y   SP   DP  DB
; 00 2013  00110000 5555 0000 0002 CFFF 0000 00
m 7f03 7f03
>007F03 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00:UU..............
```

천공 카드로 코딩

기계어로 코딩

어셈블리로 코딩 (wasm...?)

C로 코딩

Java로 코딩

Javascript로 코딩

Typescript로 코딩

자연어로 코딩 ← 지금 여기

AI로 인해 모든 게
달라지고 있는 것 같지만,

무엇으로 코딩하느냐가 달라졌을 뿐

"개발자는 (코딩으로)
문제 해결을 하는 사람이다"
라는 본질은 같다

더 자세한 내용은 2개월 전 발표자료 참고 →

https://wiki.g15e.com/pages/Specification%20written%20in%20natural%20language

자연어로 작성된 명세서

2012-01-01 (modified: 2025-10-15)

저자: AK

자연어는 생각보다 정교하지 못하다. 자연어로 작성하는 기획서나 명세서에는 여러 한계가 있다. 이 한계로 인한 병목은 AI 에이전트 활용이 일상화되면서 더 두드러지게 드러나기 시작한다.

문제점

생각한 걸 정확히 표현하기란 어려운 일이다. 설사 정확히 표현했다고 하더라도 그에 따른 귀결이 예상과는 다른 경우가 많다.

> *"What you write down doesn't mean exactly what you think it means. And when it does, it doesn't have the consequences you expected." —Daniel Jackson, Software Abstractions: Logic, Language, and Analysis*

이건 오래 전부터 있던 문제였지만, AI 에이전트에게 자연어로 지시하는 방식의 AI-인간 상호작용 루프에서의 병목을 정확히 잡아낸 문장이라고 생각한다. (위 책은 생성 AI와 관련이 없지만 너무 찰떡같아서 인용했다.)

물론 자연어가 워낙 불명확하기 때문에 아직 '자연어 코딩'에 한계가 많고, 이를 극복하기 위한 다양한 시도가 있음

원래의 자연어 명세는 이랬다:

> *파일 시스템은 엔티티의 집합이다. 파일 엔티티에는 내용을 저장할 수 있다. 디렉토리 엔티티는 다른 파일을 담을 수 있다.*

Alloy로 작성된 현재의 명세서는 다음과 같다.

```
abstract sig Entity {}

sig Dir extends Entity { children: set Entity }

sig Content {}

sig File extends Entity { content: one Content }

one sig Root extends Dir {}

fact 루트를_제외한_모든_객체는_루트로부터_도달_가능하다 {
  Root.^children = Entity - Root
}

fact 콘텐츠는_반드시_파일에_속해야_한다 {
  all c: Content | c in File.content
}

fact 루트를_제외한_엔티티는_반드시_하나의_디렉토리에_속해야_한다 {
  all e: Entity - Root | one children.e
}

fact 순환은_존재하지_않는다 {
  no d: Dir | d in d.^
}
```

이걸 자연어로 번역해보자.

> *파일 시스템은 엔티티의 집합이다. 엔티티는 파일 또는 디렉토리 중 하나이며 파일이면서 동시에 디렉토리인 엔티티는 존재하지 않는다. 파일에는 언제나 1개의 내용이 담기며, 파일 없이 내용만 존재할 수는 없다. 디렉토리는 비어있거나 다른 엔티티를 1개 이상 포함할 수 있다. 파일 시스템에는 반드시 단일한 루트 디렉토리가 있어야 하며, 모든 엔티티는 루트로부터 도달 가능해야 한다. 디렉토리는 자기 자신을 재귀적으로 다시 포함할 수 없다.*

그래도 우리가 인지하고, 인정해야 할 흐름은

'검증 가능한 영역'에 있는 문제는

결국 (강화학습으로) AI가 풀 수 있게 될 것이라는 사고방식

그리고 '검증 가능한 문제'로 대표적인 게 코딩 영역이라는 것

→ '코딩'의 민주화 + 교양학문화

(Easy to Learn, Hard to Master)

'문제'를 잘 가져오기만 한다면 80%는 AI가 풀어줄 것이다
그러면 무엇이 중요한가?

- 발견: 유의미한 문제를 어떻게 떠올릴까?
- 위임: 문제를 풀기에 적절한 정보, 권한, 도구를 어떻게 쥐어줄까?
- 감독: 어떤 타이밍에, 어떤 기준으로 개입하고 조정할까?
- 완료: 나머지 20%는 어떻게 마무리할까?
- 영향: 이 해결책을 어떻게 전파해서 실제로 유의미한 변화를 만들어낼까?

(AI가 발전할수록 중간 부분은 덜 중요해지겠지만 없어지지는 않을 것)

앞으로 개발자로서 어떤 커리어를 가져가면 좋을까?

1. E2E 프로덕 엔지니어: π자형 인재
2. 소프트웨어 아키텍트 & 엔지니어: 큰 그림 & 깊이 깊이
3. FDE / AX 컨설턴트: 1선에서 변화 만들기
4. UI/UX 디자이너: AI 시대의 인터페이스와 인터랙션 고민하기
5. AI 및 개발 교육자: 교양학문으로서의 'AI 코딩'을 대중에 전파하기

💼 AI 시대의 개발자 커리어 패스

1. E2E 프로덕 엔지니어: ㅠ자형 인재

도메인과 도메인을 연결해서 nich하지만 유의미한 문제를 찾고

처음부터 끝까지 (AI와 함께) 혼자 다 풀어버릴 수 있는 사람

- 발견: 유의미한 문제를 어떻게 떠올릴까?
- 위임: 문제를 풀기에 적절한 정보, 권한, 도구를 어떻게 쥐어줄까?
- 감독: 어떤 타이밍에, 어떤 기준으로 개입하고 조정할까?
- 완료: 나머지 20%는 어떻게 마무리할까?
- 영향: 이 해결책을 어떻게 전파해서 실제로 유의미한 변화를 만들어낼까?

💼 AI 시대의 개발자 커리어 패스

2. 소프트웨어 아키텍트 & 엔지니어: 큰 그림 & 깊이 깊이

위임, 감독, 완료 하나하나가 소프트웨어화/서비스화/기업화될 수 있음

큰 문제는 정해진 상태에서 세부 문제를 하나하나 잘 설계해서 풀 수 있는 사람

- 발견: 유의미한 문제를 어떻게 떠올릴까?
- 위임: 문제를 풀기에 적절한 정보, 권한, 도구를 어떻게 쥐어줄까?
- 감독: 어떤 타이밍에, 어떤 기준으로 개입하고 조정할까?
- 완료: 나머지 20%는 어떻게 마무리할까?
- 영향: 이 해결책을 어떻게 전파해서 실제로 유의미한 변화를 만들어낼까?

https://newsletter.pragmaticengineer.com/p/forward-deployed-engineers

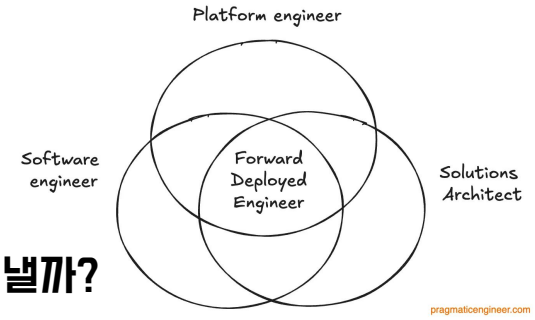
💼 AI 시대의 개발자 커리어 패스

3. FDE / AX 컨설턴트: 1선에서 변화 만들기 ← 제가 코르카에서 하고 있는 일

FDE(Forward Deployed Engineer): 팔란티어가 만들어낸 새로운 직무

개별 기업에 파견나가 AI로 풀 수 있는 문제를 발견/해결하고, 스스로 그럴 수 있는 능력을 계발

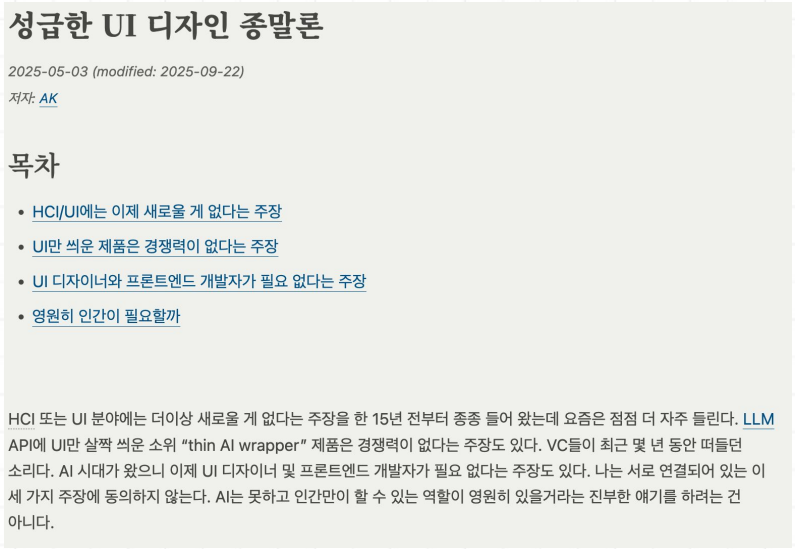
- 발견: 유의미한 문제를 어떻게 떠올릴까?
- 위임: 문제를 풀기에 적절한 정보, 권한, 도구를 어떻게 쥐어줄까?
- 감독: 어떤 타이밍에, 어떤 기준으로 개입하고 조정할까?
- 완료: 나머지 20%는 어떻게 마무리할까?
- 영향: 이 해결책을 어떻게 전파해서 실제로 유의미한 변화를 만들어낼까?



https://wiki.g15e.com/pages/Premature%20death%20of%20UI%20design

💼 AI 시대의 개발자 커리어 패스

4. UI/UX 디자이너: AI 시대의 인터페이스와 인터랙션 고민하기



성급한 UI 디자인 종말론

2025-05-03 (modified: 2025-09-22)

저자: AK

목차

- HCI/UI에는 이제 새로울 게 없다는 주장
- UI만 씌운 제품은 경쟁력이 없다는 주장
- UI 디자이너와 프론트엔드 개발자가 필요 없다는 주장
- 영원히 인간이 필요할까

HCI 또는 UI 분야에는 더이상 새로울 게 없다는 주장을 한 15년 전부터 종종 들어 왔는데 요즘은 점점 더 자주 들린다. LLM API에 UI만 살짝 씌운 소위 "thin AI wrapper" 제품은 경쟁력이 없다는 주장도 있다. VC들이 최근 몇 년 동안 떠들던 소리다. AI 시대가 왔으니 이제 UI 디자이너 및 프론트엔드 개발자가 필요 없다는 주장도 있다. 나는 서로 연결되어 있는 이 세 가지 주장에 동의하지 않는다. AI는 못하고 인간만이 할 수 있는 역할이 영원히 있을거라는 진부한 얘기를 하려는 건 아니다.

- AI가 프론트엔드 개발과 UI 디자인의 종말을 가져왔다는 주장에 동의하지 않음. 인터페이스 설계는 너무너무 중요함
- 현재 AI를 활용하는 주된 인터페이스가 챗봇, 노드 연결, CLI 3종류인데, 이게 인간에게든 AI에게든 최선은 아닐 거라고 생각 (JIT UI...? 🤔)
- 아직 정답은 없지만 개발할 때 인간뿐 아니라 AI도 아주 중요한 '주체'임을 잊지 말아야 하는 건 확실함. llms.txt, 프롬프트를 제공하는 docs, a11y 등 챙길 게 많음

https://www.youtube.com/watch?v=myNgSPXish4

💼 AI 시대의 개발자 커리어 패스

4. UI/UX 디자이너: AI 시대의 인터페이스와 인터랙션 고민하기

이 주제에 관심이 있으신 분은
노정석/최승준님의 최신
팟캐스트 추천 👉

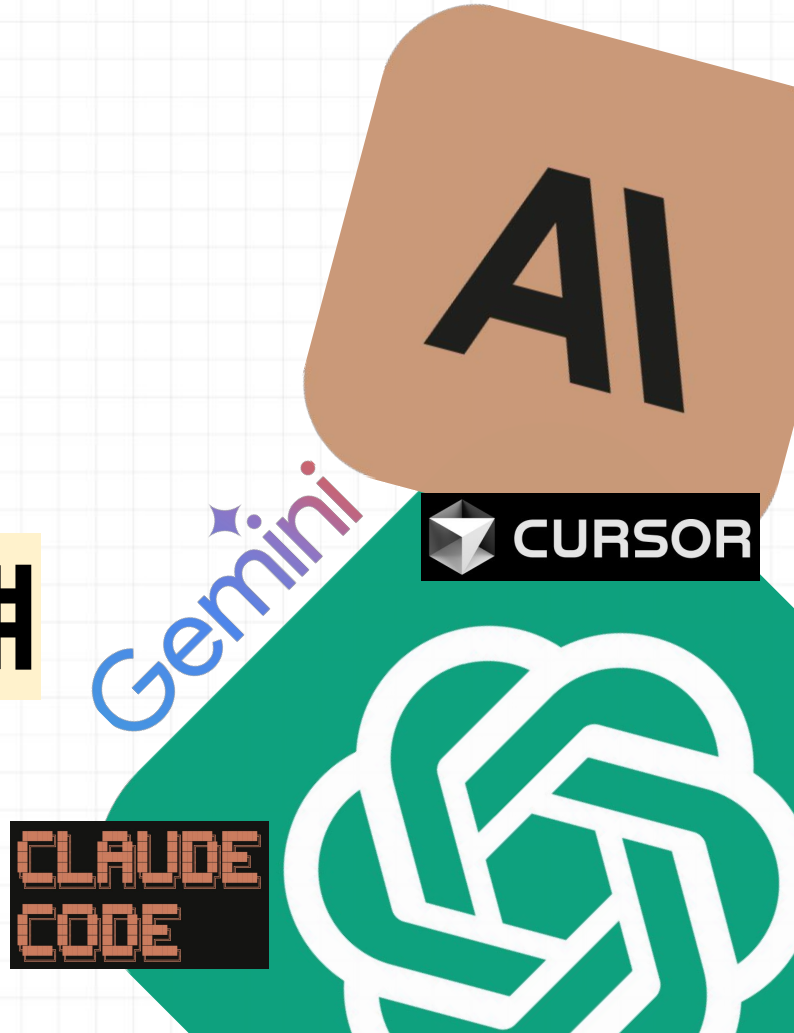
💼 AI 시대의 개발자 커리어 패스

5. AI 및 개발 교육자: 교양학문으로서의 'AI 코딩'을 대중에 전파하기

- 모든 음악 전공자가 콩쿨을 수상해서 프로 연주자가 될 수 없고, 될 필요도 없음
- '개발'이 교양학문화되는 이 시기에 교육자가 유의미하게 할 일이 엄청나게 많음
- 여전히 80% 이상의 사람들은 개발은커녕 AI도 적극적으로 활용하지 않고 있음
    - 가장 성공한 AI 제품인 ChatGPT조차 주간 활성 사용자 8억명에 불과 (세계 인구의 10%)

2부

이 AI 시대에
나는 어떤 노력을 하며
살아가고 있는가



요즘 날마다 생각, 실험, 대화하는 화두

어떻게 하면 10x 임팩트를 낼 수 있을까?

음... 이건 1x 사고방식이야.
이 일은 아무리 노력해도 상방이 1x일 뿐이겠다.
어떻게 다르게 하지?

저사람은 대략 5x로 일을 하는 느낌이군.
오, 이 팀은 10x가 될 수 있겠는데?

요즘 날마다 생각, 실험, 대화하는 화두

어떻게 하면 10x 임팩트를 낼 수 있을까?

어떻게 해야 나를 10x로 만들 수 있을까?

어떻게 해야 이분을 10x로 만들 수 있을까?

어떻게 해야 이 팀을 10x로 만들 수 있을까?

어떻게 해야 이 조직을 10x로 만들 수 있을까?

어떻게 해야 이 나라를 10x로 만들 수 있을까?

요즘 날마다 생각, 실험, 대화하는 화두

어떻게 하면 10x 임팩트를 낼 수 있을까?

기존에 하던 작업을 1/10의 시간에 한다

어떻게?

"내가 할 수 있는 모든 작업에는
훨씬 더 잘 하는 방법이 반드시 존재한다."
그런 방법을 찾지 못했다면,
아직 내가 발견과 시도를 충분히 못했을 뿐이다.

"내가 할 수 있는 모든 작업에는
훨씬 더 잘 하는 방법이 반드시 존재한다."
그런 방법을 찾지 못했다면,
아직 내가 발견과 시도를 충분히 못했을 뿐이다.

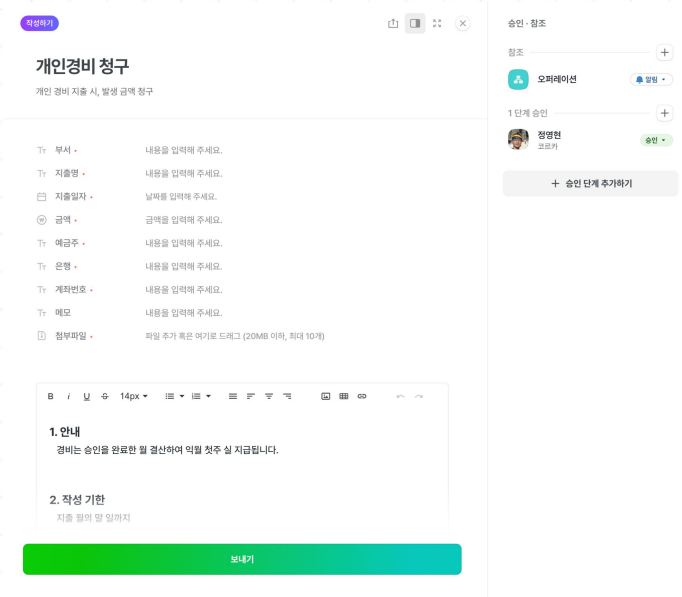
작업 결과물을 어떻게
더 낫게 할까?

작업 과정을 어떻게 더
쉽고 깔끔하게 할까?

작업 자체를 어떻게 덜
지루하고 더 재밌게 할까?

모든 작업에는 더 잘 하는 방법이 있다

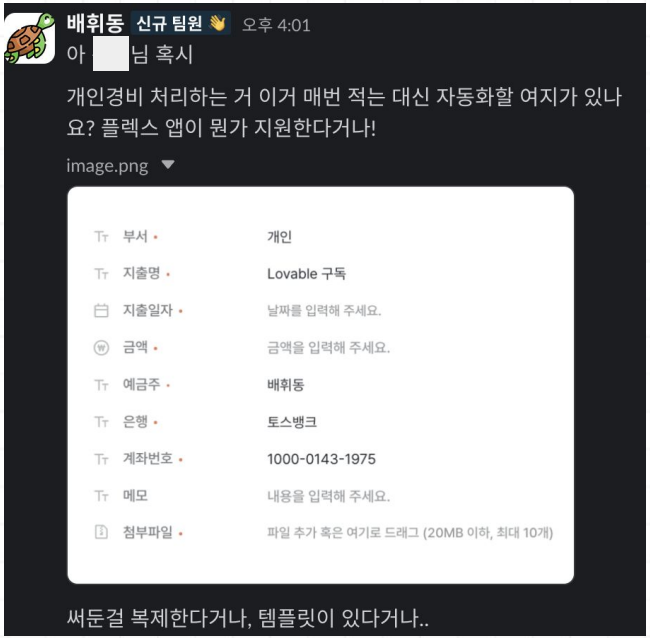
사례: 개인경비 청구



- 코르카 입사 전 구독했던 AI 서비스들을 Flex로 청구해야 하는 상황
- 2개월간 구독했던 툴도 너무 많은데, 각각 적으려니 항목이 너무 많고 귀찮음
  → 더 나은 방법이 없을까? 🤔 난 Flex는 처음인데...
- 1차 시도: 잠깐 이것저것 눌러봤는데 템플릿/복제 같은 기능이 없어 보임 (소요시간 1분)

모든 작업에는 더 잘 하는 방법이 있다

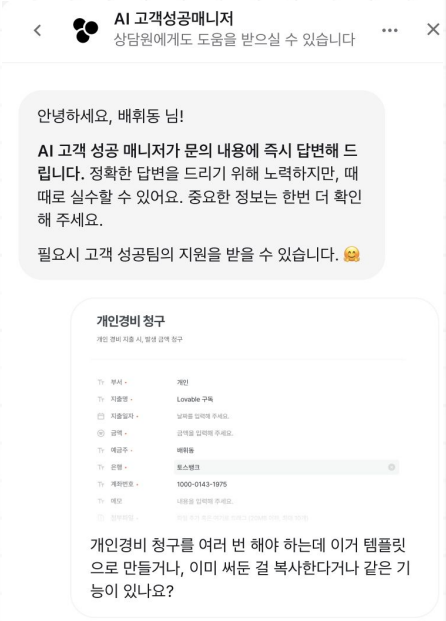
사례: 개인경비 청구



2차 시도: 나보다 Flex를 더 잘 알만한 사람 (= 나를 Flex에 온보딩해준 인사팀 동료)에게 물어봄 (소요시간 1분)

모든 작업에는 더 잘 하는 방법이 있다

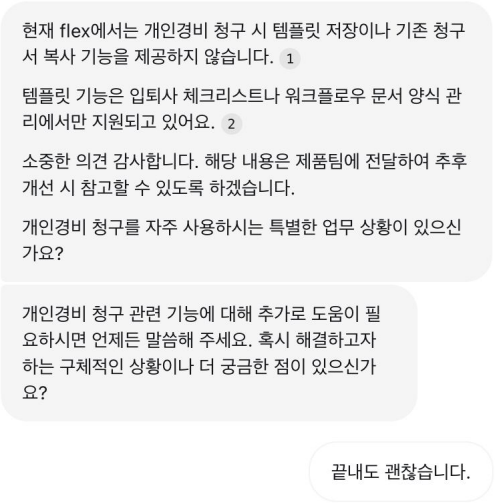
사례: 개인경비 청구



(동료에게 즉답이 오지 않아) 3차 AI도

- 요즘 웬만한 SaaS에는 도움요청/피드백 전달을 위한 창구가 존재하고, 대개 AI 기능이 붙어있음 (= 즉답한다)
- Flex도 AI Support가 있었음
- 문의 결과: '그런 기능 없음'

(소요시간 1분)



모든 작업에는 더 잘 하는 방법이 있다

사례: 개인경비 청구

4차 시도: (의심) 그런데 내가 '개인경비 청구'라고 한 말을 AI가 제대로 알아들은 건 맞을까?

워크플로우는 복제 가능하다고 했는데, 개인경비 청구도 코르카에서는 워크플로우임

→ 출처 문서 읽기 (소요시간 3분)

양식 분류 및 양식 관리

최소 4달 전에 업데이트됨

목차

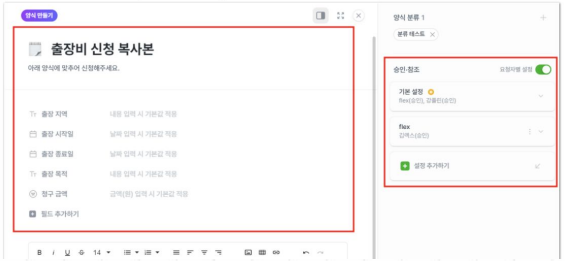
양식을 카테고리별로 분류할 수 있어요. 양식 분류를 통해 특정 구성원에게만 조직만 사용, 편집 권한을 부여하여 운영할 수 있어요.
혹은 지금 당장 사용하지는 않지만, 미리 만들어두거나 삭제하기 애매한 양식은 [권한자만 보기] 를 통해 숨길 수도 있어요.

양식 분류와 양식 관리는 **[워크플로우 양식 관리]** 권한을 가진 관리자만 설정이 가능해요!

양식 분류 추가

- 좌측 메뉴 **[워크플로우]** → 우측 상단 **[문서 양식 관리]** 클릭, 상단의 **[분류 수정]** 을 클릭해주세요

- 원본 양식의 **필드값과, 기본 결재선, 요청자별 승인 설정이 그대로 복사**돼요.

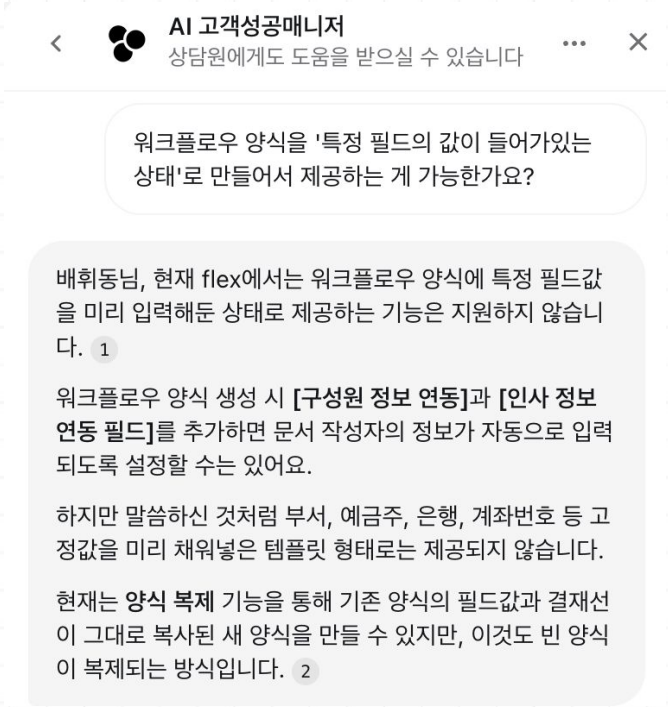


모든 작업에는 더 잘 하는 방법이 있다

사례: 개인경비 청구

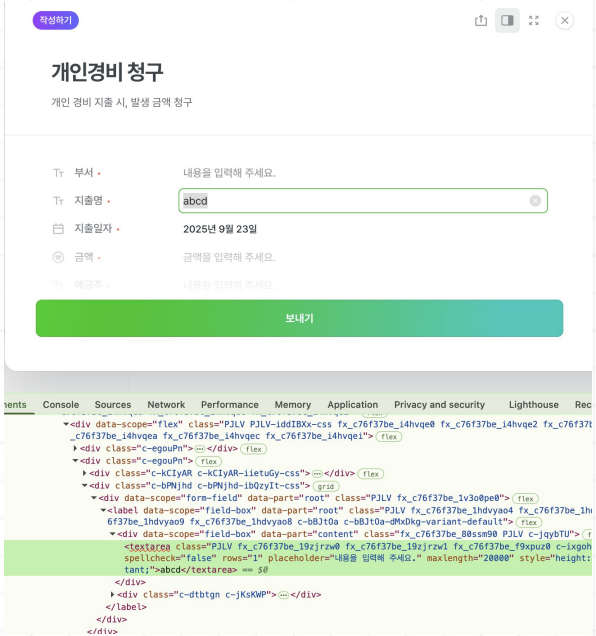
출처 읽어보니 일단 Flex에서 관리자 권한이 필요하고, '필드'는 복제되지만 '값은 복제되지 않는 것으로 판단

그래서 5차 시도: AI에게 재문의해서 내 이해가 맞는지 확인 (소요시간 2분)



모든 작업에는 더 잘 하는 방법이 있다

사례: 개인경비 청구



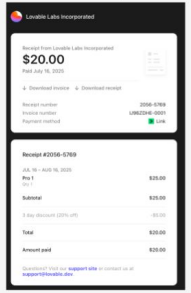
6차 시도: 자동화 스크립트 제작 가능성 탐색

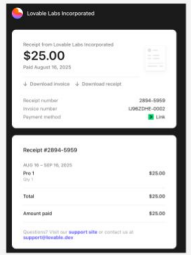
- '요소 검사'로 HTML 구조 파악, 임시 ID 부여하여 값 조작 등
- 권한 문제, 브라우저 조작 문제 등 까다로운 게 있고, '파일 첨부'가 필요해서 100% 자동화는 안되더라도 내 일을 어느정도는 덜어줄 가능성이 있다고 봤음
- 슬슬 바이브 코딩 해볼까 하다가... (여기까지 소요시간 5분)
- 🤔 이거 정말 하나하나 보내야 하나? 내가 이거 하는 목적이 뭐였지? 개인 경비를 어떻게든 돌려받는 거 아닌가?

모든 작업에는 더 잘 하는 방법이 있다

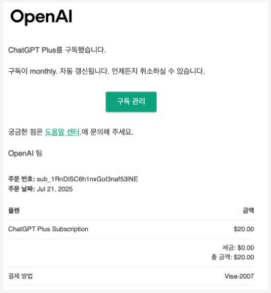
사례: 개인경비 청구

Lovable 7/17 $20, 8/17 $25

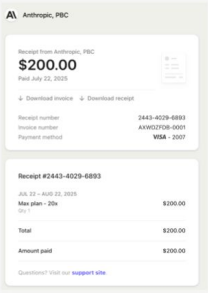


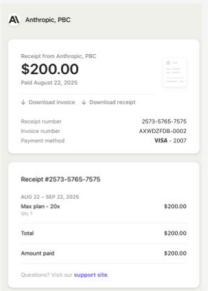


ChatGPT 7/21 $20, 8/21 $20



Claude 7/22 $200, 8/23 $200





Cursor 7/23 $20, 8/23 $20

7차 시도: 각 서비스 청구서 기록을 (청구하려고 이메일 온 거 따로 그룹핑해뒀던 것들) 캡처해서 하나의 PDF 파일로 만들어서 Flex로 청구

승인 안되면 그때 가서 자동화 다시 고민하자 → 근데 그냥 승인됨

모든 작업에는 더 잘 하는 방법이 있다

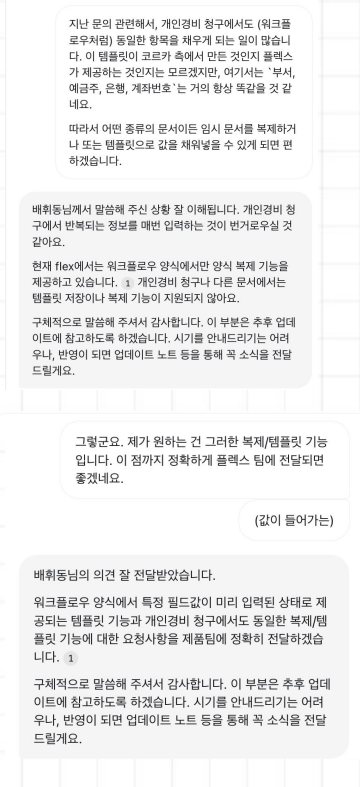
사례: 개인경비 청구

이 과정에서 내가 얻은 것들

지식: 2025년 9월 시점에는, 'Flex 워크플로우 문서 작성'이라는 작업 내에 존재하는 '동일 값 복사' 라는 반복행위를 줄일 방법이 Flex 내부 기능으로는 존재하지 않는다 (하지만 내가 문의 넣었으니까 나중에는 생길 수도 있다)

기술: 다음에 Flex에서, 또는 다른 유사 서비스에 대한 자동화 스크립트를 짜게 된다면 어디부터 어떻게 시작할지 감을 조금 잡았다

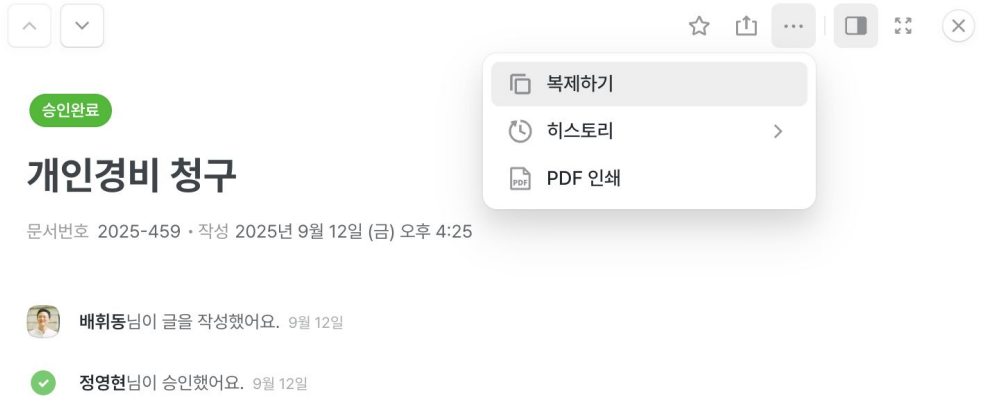
태도: '모든 작업에는 더 잘 하는 방법이 있다'는 마음가짐에 대한 확신 수준이 또 한번 올랐다



모든 작업에는 더 잘 하는 방법이 있다

사례: 개인경비 청구

반전: 사실 내부 기능이 있었음



코르카 내부 특강 후 한 분이 알려주심. "되는데요?" (다른 분은 아무도 몰랐음)

즉 AI의 환각에 또 걸려들었던 것

물론 몰랐어서 오히려 잘 된 케이스 😄

"내가 할 수 있는 모든 작업에는
훨씬 더 잘 하는 방법이 반드시 존재한다."
그런 방법을 찾지 못했다면,
아직 내가 발견과 시도를 충분히 못했을 뿐이다.

작업 결과물을 어떻게
더 낫게 할까?

작업 과정을 어떻게 더
쉽고 깔끔하게 할까?

작업 자체를 어떻게 덜
지루하고 더 재밌게 할까?

🤔 여기에 AI를 접목한다면...?

"내가 컴퓨터로 할 수 있는 모든 작업에는
AI의 도움을 받아 훨씬 더 잘 하는 방법이 반드시 존재한다."
그런 방법을 찾지 못했다면,
아직 내가 발견과 시도를 충분히 못했을 뿐이다.

- 인간이 (컴퓨터로) 할 수 있는 일은 반드시 AI도/프로그램도 할 수 있다
- 내 작업을 대신하든, 내 작업을 도와주든 AI에게 도움받을 건 얼마든지 있다
- 이미 AI는 충분히 잘 하는데 내가 충분히 잘 못 쓰고있을 뿐이다

→ 이런 믿음을 가진다면 어떤 시도를 해볼 수 있을까?

"내가 컴퓨터로 할 수 있는 모든 작업에는
AI의 도움을 받아 훨씬 더 잘 하는 방법이 반드시 존재한다."
그런 방법을 찾지 못했다면,
아직 내가 발견과 시도를 충분히 못했을 뿐이다.

- 내가 일상적으로 어떤 반복 작업을 하고, 무엇이 불편한지 발견하기
- 내 작업의 단계를 분석/분해해서 어디를 조금만 개선해도 확 편해질지 찾기
- 그 단계를 다시 쪼개서 어디를 어떻게 개선할지 설계하기 (인간 / AI / 코드 협업)
- AI와 함께 세부 단계를 구현/테스트/수정하기 + 반복
- 또는, 아예 전체를 잘 가이드해서 브라우저/컴퓨터-유즈 에이전트에게 맡기기

그런데...

이걸로 충분할까?

요즘 날마다 생각, 실험, 대화하는 화두

어떻게 하면 10x 임팩트를 낼 수 있을까?

기존에 하던 작업을 1/10의 시간에 한다

이걸로 충분한가? 아낀 시간에는 뭘 하지?

요즘 날마다 생각, 실험, 대화하는 화두

어떻게 하면 10x 임팩트를 낼 수 있을까?

기존에 하던 작업을 1/10의 시간에 한다

이걸로 충분한가? 아낀 시간에는 뭘 하지?

🤔 비슷한 작업을 더 많이 한다?

이런 작업을 계속 찾을 수 있을까? 이것만으로 정말 만족스러울까?

요즘 날마다 생각, 실험, 대화하는 화두

어떻게 하면 10x 임팩트를 낼 수 있을까?

기존에 하던 작업을 1/10의 시간에 한다

이걸로 충분한가? 아낀 시간에는 뭘 하지?

🤔 비슷한 작업을 더 많이 한다?

이런 작업을 계속 찾을 수 있을까? 이것만으로 정말 만족스러울까?

👊 10x, 100x 임팩트가 날 만한 새로운 일을 꾸준히 시도한다

핵심은 시스템화, 추상화, 패턴화

👊 AI도 1: AX 가속화 컨설팅

(1x) 내가 매번 직접 교육하는 대신

(10x) 교육자를 양성하는 교육을 해보자

AX 가속화 컨설팅

저는 AI 시대에 기업들에게 가장 중요한 것 중 하나가 AI Transformation이 전사적으로, 자발적으로 일어나서 개개인이 10x 인재가 되는 것이라고 생각합니다.

현재 제게 특강/워크숍 형태의 AI 교육 요청이 굉장히 많이 들어옵니다. 하지만 이런 교육이 정말로 해당 기업에 유의미한 변화를 미쳤는지는 체감하기 어려우며, 제가 변화에 대한 영향력을 행사할 수 있는 범위도 한정적입니다. 워크숍 방식이라면 '변화'는 좀 더 잘 일어날 수 있겠지만, 참여자에 한정지어 변화가 일어난다는 단점이 있습니다. 즉 이런 교육을 진행하는 것은 제 시간과 노력을 투입하는 정도에 비례해서 스케일이 나오는, **1x 임팩트**를 만듭니다.

그래서 임팩트를 **10x 또는 그 이상**으로 내기 위한 방안으로 고안한 것이 **'기업 내에서 `AX Day`를 운영할 수 있는 `AX 슈퍼스타`를 양성하는 컨설팅'**입니다. (현재 코르카 내부에서도 진행 중입니다)

아직 구상/논의 단계에 있습니다만 좀 더 디테일하게 적어보면:

- `AX Day` 는 팀 단위로 진행됩니다. 반나절 또는 한나절동안 원래 하던 업무를 하는 대신, 본인 업무 및 팀 업무에서 반복 작업과 불편함을 발견하여 직접 개선하는 날로 삼습니다.
- 목표는 그날 안에 본인 스스로 개선점을 정리하고, 그걸 일부라도 해결하는 Ver.0을 직접 만드는 경험을 하는 것입니다. (Ver.0 예시: 리팩토링, 테스트 코드 추가, 바이브 코딩으로 어드민 앱/슬랙 봇 만들기, FAQ 문서 작성, 다른 도구 도입 등)
- 이 과정을 팀 내부에서 시작하기 어려우니, 그 팀에게 퍼실리테이션과 교육을 할 수 있는 `AX 슈퍼스타` 를 양성하는 교육을 하고, `AX Day`를 운영할 수 있는 플레이북을 제공합니다. 이 교육 자체도 `AX Day` 와 동일하게 (플레이북을 이용하여) 진행하되, 제가 슈퍼스타 역할을 합니다.
- 컨설팅에는 `AX 슈퍼스타` 를 모집/선발하기 위한 과정 논의, 기업에 맞게 플레이북을 커스터마이즈하는 과정이 포함될 수 있습니다.

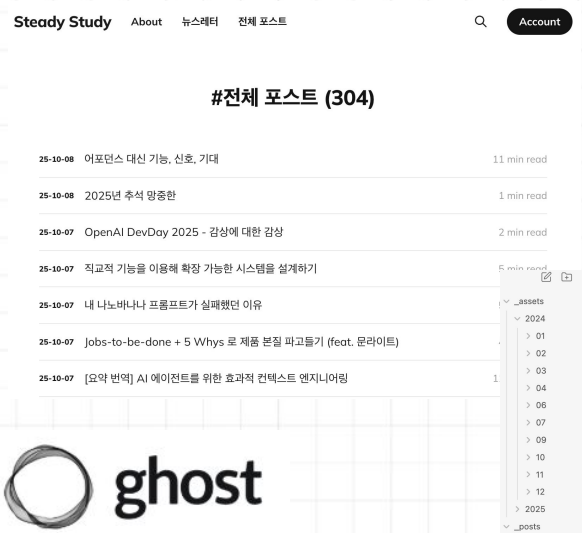
이런 과정을 통해 내부 구성원에게 효능감이 생기고 전사적 AX 마인드셋이 생기면 AI Transformation이 가속화되리라고 생각합니다. `AX Day`에 충분히 효과를 느낀다면 주기적으로, 슈퍼스타의 도움 없이도 팀 내에서 진행할 수 있을 것이고요.

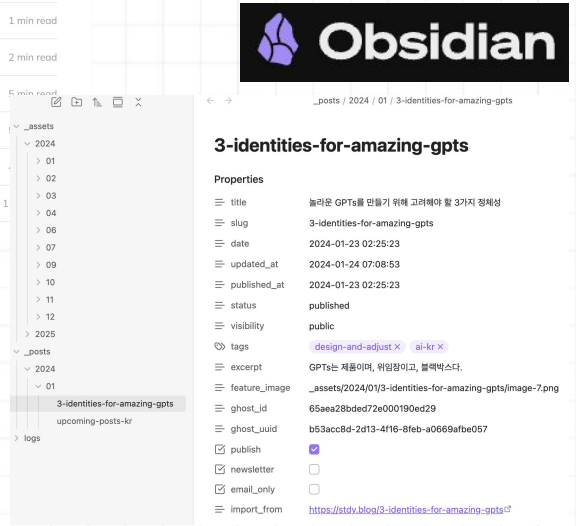
https://www.stdy.blog/bulding-personal-llms-txt/

👊 시도 2: Ghost-to-Obsidian 프로젝트

(1x) 매번 컨텍스트를 일일이 입력하는 대신

(10x) 컨텍스트를 더 쉽게, 더 잘 입력할 수 있는 환경을 만들자



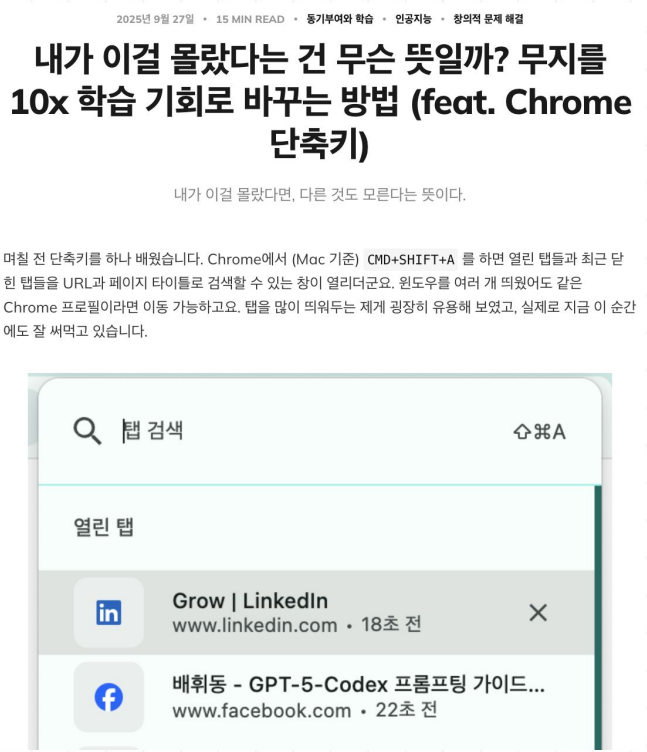


https://www.stdy.blog/how-to-create-10x-learning-opportunity-from-any-ignorance/

👊 시도 3: 무지를 10x 학습 기회로

~~바꾸거~~ 단축키 하나를 새로 배우는 대신

(10x) 내가 주로 어떤 단축키를
모르는지, 그걸 왜 몰랐는지 탐구한다.
즉 내 '무지의 패턴'을 발견하고,
'패턴을 발견하는 패턴'을 발견한다



2025년 9월 27일 • 15 MIN READ • 동기부여와 학습 • 인공지능 • 창의적 문제 해결

내가 이걸 몰랐다는 건 무슨 뜻일까? 무지를 10x 학습 기회로 바꾸는 방법 (feat. Chrome 단축키)

내가 이걸 몰랐다면, 다른 것도 모른다는 뜻이다.

며칠 전 단축키를 하나 배웠습니다. Chrome에서 (Mac 기준) CMD+SHIFT+A 를 하면 열린 탭들과 최근 닫힌 탭들을 URL과 페이지 타이틀로 검색할 수 있는 창이 열리더군요. 윈도우를 여러 개 띄웠어도 같은 Chrome 프로필이라면 이동 가능하고요. 탭을 많이 띄워두는 제게 굉장히 유용해 보였고, 실제로 지금 이 순간에도 잘 써먹고 있습니다.

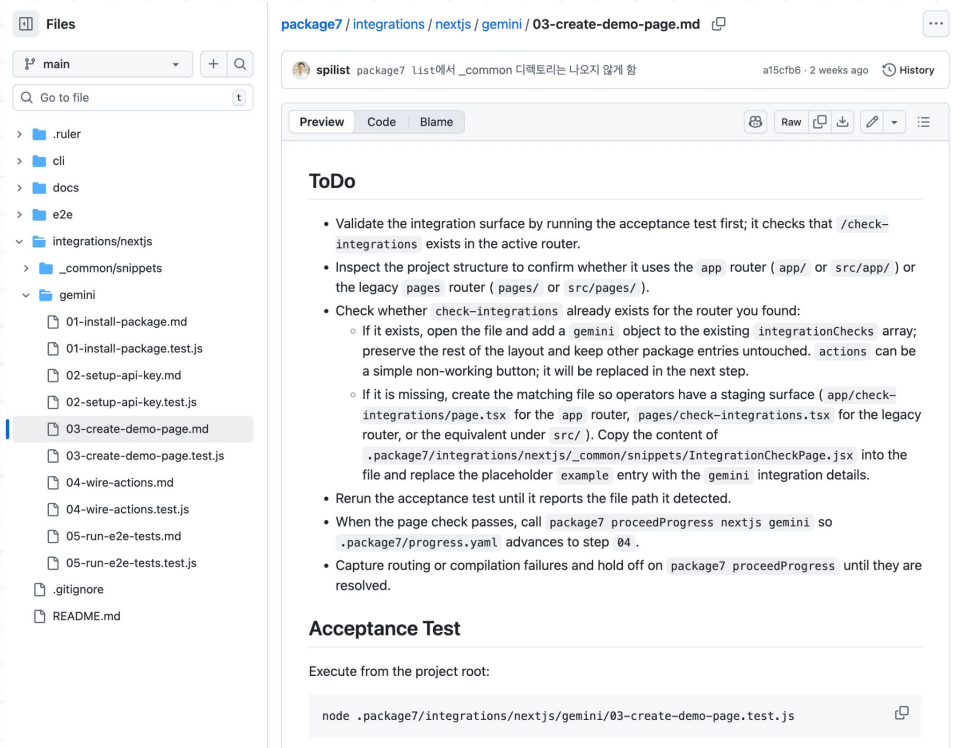
(에이전틱 워크플로우 설계에 대해서는 후술)

👊 시도 4: Package7 프로젝트

(1x) SDK, API를 직접 찾아 코딩해서 연동하는 대신

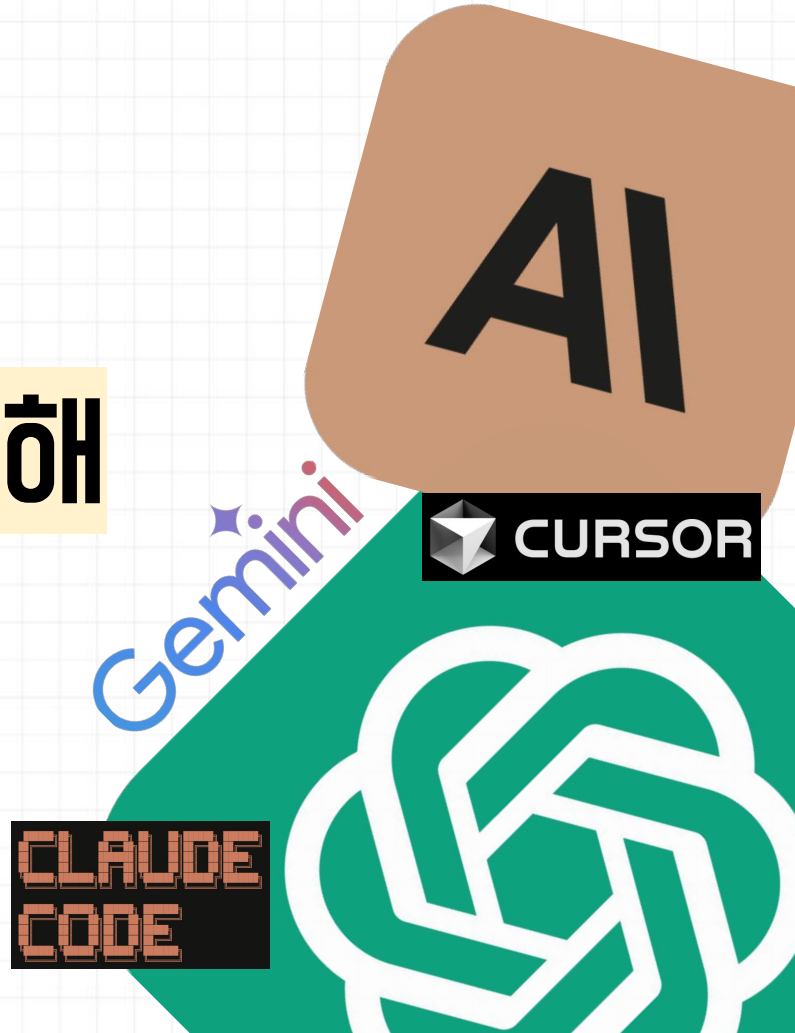
(10x) 어떤 SDK든 프롬프트로 연동할 수 있는 오픈소스를 만들자

(100x) 이 프롬프트들을 구현하고, 검증하는 전체 과정을 에이전틱 워크플로우로 설계하자



3부

앞으로의 커리어를 위해 주니어들에게 권하는 마음가짐



가장 먼저 가져야 할 마음가짐:

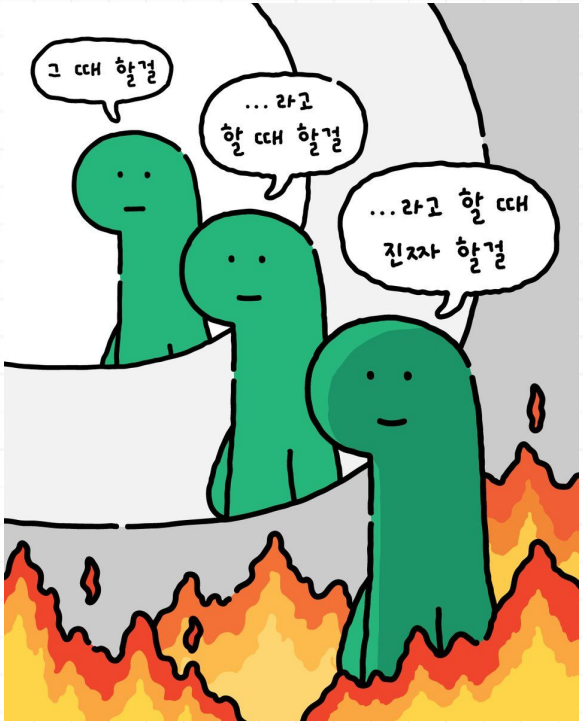
FOMO 이겨내기

(Fear Of Missing Out)

https://x.com/jogumanstudio/status/1967513735498461574

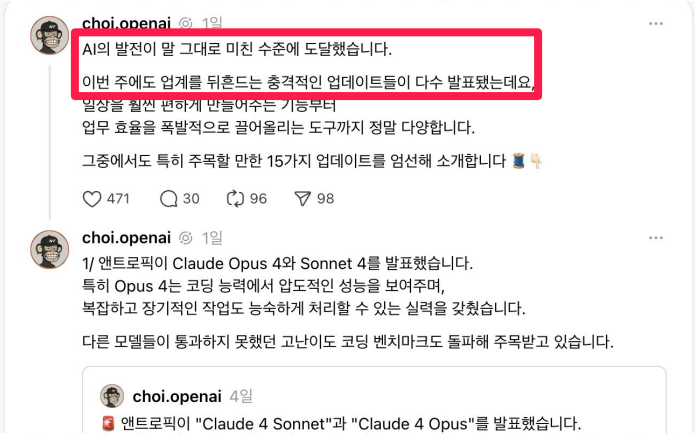
😱 지금 당장 안 따라가면 큰일나!

- 사실 정말로 당장 안 했을 때 큰일나는 일은 별로 많지 않음
- 늦게 출발할수록 어쩌면 더 고효율로 따라갈 수 있음
- 다만 따라가는 사이에 있는 그 잠깐의 갭으로 인한 비즈니스 가치는 놓치게 될 수 있음은 분명히 인지하는 게 좋음

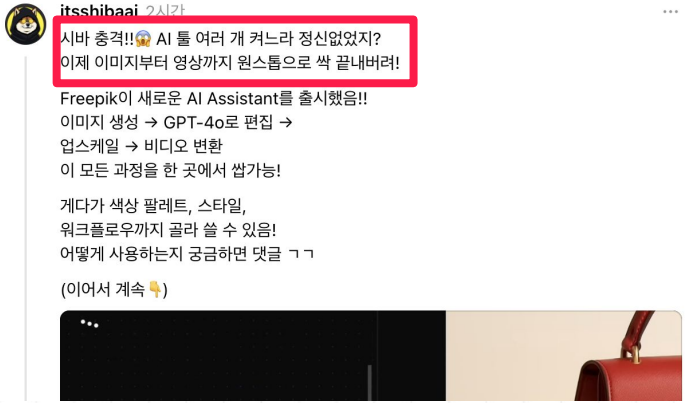


호들갑이 많지만 항상 반반의 마음을 가지자

→ 신기하다, 하지만 별것 아니니 나도 할 수 있다



https://www.threads.com/@choi.openai/post/DKEn2j2vdMB?xmt=AQF0gJO2ZhN7iNvMRw3mUMr0-eNwn00wIaBXi6V3zfBS8Q



https://www.threads.com/@itsshibaai/post/DKIjFH0PiQZ?xmt=AQF0HZpIWqZGIWiSI-LonOI7xDA_ERGQ0lv29FQ5mJcD9g

지금부터 '라고 할 때 할걸'을 위해
어디에 집중하는 게 좋을까?

https://fortelabs.com/blog/boundary-intelligence-why-what-you-can-access-matters-more-than-what-you-know/

Boundary Intelligence: Why What You Can Access Matters More Than What You Know

BY TIAGO FORTE

최근 인상깊게 읽었던 글:

Boundary Intelligence

by Tiago Forte

Let me step ba asing from that piece).

Our definition

For most of human history, intelligence was mainly defined as “strong memory.” Information was so hard to access in the first place that your ability to recall specific facts and details from memory was considered the highest mark of intelligence.

https://fortelabs.com/blog/boundary-intelligence-why-what-you-can-access-matters-more-than-what-you-know/

AI 시대에 어떤 지능이 중요할까?

- 지능의 정의는 시대에 따라 바뀌어왔음
  - 인간에게 무엇이 유의미한가? 무엇이 어려운가(비싼가)?
- 인류 역사에서 아주 오랫동안 지능 = '강한 기억력' (storage)
- 중요한 정보에 접근하기가 너무 어려웠기 때문에 특정한 사실을 기억하고, 세부사항을 떠올리는 능력이 곧 강한 지능의 증거였음

https://fortelabs.com/blog/boundary-intelligence-why-what-you-can-access-matters-more-than-what-you-know/

AI 시대에 어떤 지능이 중요할까?

- 책이 보편화되면서부터는 단순한 기억력의 중요성은 점차 감소
- 문서화된 정보를 처리하고 분석하는 능력이 더 중요해짐
  - 여러 문서에서 정보를 조합해 새 아이디어를 만들어내는 능력 포함
- 디지털 시대에서 이 흐름은 계속 강화됨
  - 얼마나 빠르게, 얼마나 좋은 인사이트를 뽑아내는가?
- Storage 비용은 싸졌지만 Computation은 여전히 비싼 상태

https://fortelabs.com/blog/boundary-intelligence-why-what-you-can-access-matters-more-than-what-you-know/

AI 시대에 어떤 지능이 중요할까?

- AI 시대가 오면서, AI가 대용량 정보를 처리하고 해석하는 능력이 보편적 인간의 수준을 훨씬 뛰어넘게 됨
  - Storage도 싸고 Computation도 싸짐
- 인간지능이 유의미해지기 위해 또다른 정의가 필요해진 시점

  → Boundary Intelligence
- 적절한 데이터를 적절한 위치 및 시점에 불러올 수 있는 능력

https://fortelabs.com/blog/boundary-intelligence-why-what-you-can-access-matters-more-than-what-you-know/

AI 시대에 어떤 지능이 중요할까?

Internal Intelligence vs Boundary Intelligence

- 잘 읽기보다 무엇을 읽을지 선택하기가 더 중요해짐
- 토론 잘하기보다 언제 누구에게 이야기할지 결정하기가 더 중요해짐
- 생각 잘하기보다 현명하게 주의집중하기가 더 중요해짐
- 개인 생산성 키우기보다 전문가 팀을 조율하는 능력 키우기가 더 중요해짐

좌측은 점점 더(또는 이미?) AI가 더 잘 하니 인간은 우측에 더 집중해야 함

→ 이렇게 하기 위해 (FOMO 이겨내기 외에) 필요한 마음가짐은?

AI 시대의 최대 덕목:

(건강한) 의심과 호기심

AI가 한 말도, 사람이 한 말도 한번쯤 검증해보기

궁금한 게 생기면 바로바로 딥 리서치 돌려보기

계속 질문하기: 이거 더 낫게 할 방법 없나?
이거 나보다 더 잘 하는 사람 누구지?

👥 가장 중요한 인사이트는 사람으로부터 나옴

고수를 만났을 때 어떻게 더 효과적으로 배울 수 있을까?

→ 그 사람에게 교육 전문성이 없다면 내가 유도해야 함 (from 김창준)

- "어떤 신호로부터 그 패턴을 인식하셨나요? 어디에 주목하셨어요?"
- "저는 문제가 A라고 봐서 B를 했고, 결과가 C일줄 알았는데 D가 됐어요. 당신이라면 이 상황에서 어떻게 했을까요? 왜 그렇게 하셨을까요? 그러면 결과가 어떻게 달라졌을까요?"
- "A 상황에서 B하면 어떤 일이 왜 벌어질까요? C요? 그걸 제가 어떻게 알 수 있었을까요?"
- "제가 이거 연습하면서 / 이 교육에서 무엇에 집중해야 하나요?"

https://www.stdy.blog/what-is-happening-inside-experts-brain/

모두 AI에게 써도 좋은 방법들입니다!

👥 가장 중요한 인사이트는 사람으로부터 나옴

뛰어난 사수를 만났을 때 어떻게 더 효과적으로 배울 수 있을까?

→ 그 사람의 지식/기술/경험뿐 아니라 그걸 만드는 방법 자체를 배워야 함

2025년 3월 9일 • 13 MIN READ • 주니어를 위한 고민 • 전문성과 인지작업분석 • 뉴스레터

전문가 머릿속에서는 대체 무슨 일이 벌어지고 있을까

전문가에게 상황 인식의 신호, 행동 전략, 시뮬레이션 방법을 배우면 압축성장이 가능합니다. LLM에게도 아주 잘 먹힙니다.

💡 글 최하단에 프리미엄 구독자 분들을 위해, 이 글의 유용한 정보를 활용하는 LLM 프롬프트를 넣어두었습니다.

미국의 심리학자 개리 클라인(Gary Klein)은 소방관, 응급실 간호사, 비행기 조종사 등 급박한 현장에서 전문성을 발휘해야 하는 사람들을 수십년간 연구한 '**전문성 전문가**'입니다.

클라인은 전문가들에 대한 심층 인터뷰와 관찰을 통해, 문제 상황을 만났을 때 그들의 머릿속에서 무슨 일이 벌어지는지를 모델링하여 인식-행동 촉발 모델(RPD: Reconition-Primed Decision Model)이라고 이름붙였는데요. 1986년 최초 발표된 이후 여러 연구를 통해 실제로 여러 분야의 전문가들이 이렇게 행동하고 있음이 밝혀졌습니다.

코드 리뷰에서 배우기

예를 들어 주니어들이 코드리뷰에서 "변수명 XX가 너무 짧네요. 변수명은 이해할 수 있을 정도로 긴 게 좋습니다. YY로 바꿀까요?" 같은 코멘트를 받으면 단순히 '넵' 하고 넘어가기 쉬운데요. 여기서도 RPD를 적용하여 질문함으로써 시니어를 괴롭히면 압축성장이 가능합니다.

- 혹시 변수명이 짧아도 괜찮은 상황은 없을까요?
  - 그럼 여기서는 변수명이 너무 짧다는 걸 어떻게 판단하셨어요?
- 변수명 XX를 그대로 가져간다면 이후 어떤 문제가 생길 것 같으신가요?
- YY라는 변수명은 어떻게 떠올리셨어요?
- 나중에 YY가 부적절해질 만한 상황이 있다면 뭘까요?
  - 그땐 변수명을 어떻게 바꾸실 것 같으세요?

물론 너무 괴롭히면 시니어가 코드리뷰를 안해줄 수도 있으니 적당히 해야겠지만요. 😃

대신, 불평하지 않는 LLM을 괴롭히는 것도 좋습니다. 모든 좋은 질문법과 마찬가지로, LLM에게 RPD식 질문을 하면 굉장히 풍부한 정보를 얻을 수 있을 거예요.

👥 좋은 사람들과 관계 맺는 게 곧 100x 기회

- 올해 우연한 기회로 새로 만나서, 대화하고, 식사하며 교류하고 있는 분들이 10명 넘음. 그분들께 굉장히 많이 배우고 자극 받고 있음
- 강규영님: 너무 멋진 글 쓰신 거 보고 반해서 연락 → 식사 → 다른 분들과도 함께 보고 싶어서 그룹대화방 만듦 → 매주 페어작업 중
- 최수민(커서맛피아)님: Vooster 개발하시는 과정이 흥미로워보여서 연락 → 식사 → 자주 만남 → 매주 페어작업 중
- 이렇게 깊은 관계를 맺는 건 어렵더라도...

📧 그냥 연락만 해봐도 이미 상위 1%

- 지금까지 수천명 앞에서 발표, 교육해봤고 그중 많은 곳에서 편하게 메일 보내라고 했으나 실제로 이메일 보내는 사람은 극소수. 기억상 20명 미만
- 정중하고 진정성 있게 질문하고 요청하면 (애초에 이런 사람이 극소수라서) 높은 확률로 긍정적 응답이 온다
- 거절당하거나 답이 오지 않아도 전혀 손해가 없음. 질문하는 과정에서 내 생각도 정리되기 때문
- 그러니까 여러분도 편하게 이메일로 연락해보세요 (저에게, 다른 사람에게)

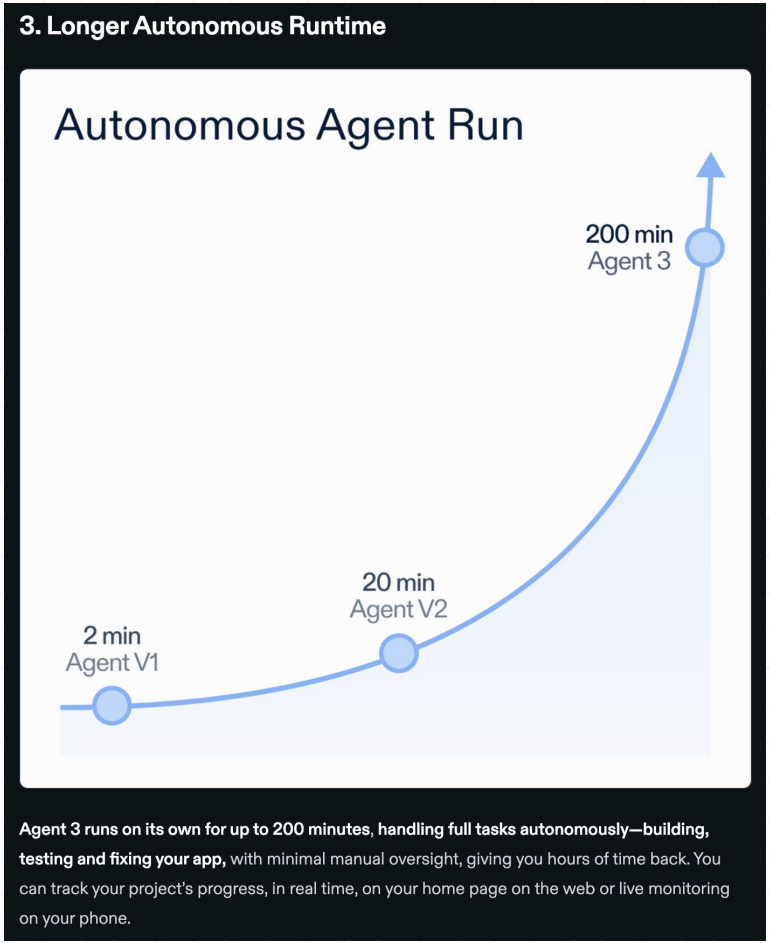
너무 '마음가짐' 얘기만 했는데
집중해서 훈련할 만한 기술을 하나 꼽는다면?

→ Designing Agentic Workflow

현 시점 개발자에게(또는 누구에게나)
가장 가치있는 기술 중 하나라고 생각

https://blog.replit.com/introducing-agent-3-our-most-autonomous-agent-yet

https://www.anthropic.com/news/claude-sonnet-4-5



Frontier intelligence

Claude Sonnet 4.5 is state-of-the-art on the SWE-bench Verified evaluation, which measures real-world software coding abilities. Practically speaking, we've observed it maintaining focus for more than 30 hours on complex, multi-step tasks.

새 AI 모델과 에이전트 홍보에서 점점 더 자주 등장하는 종류의 문구

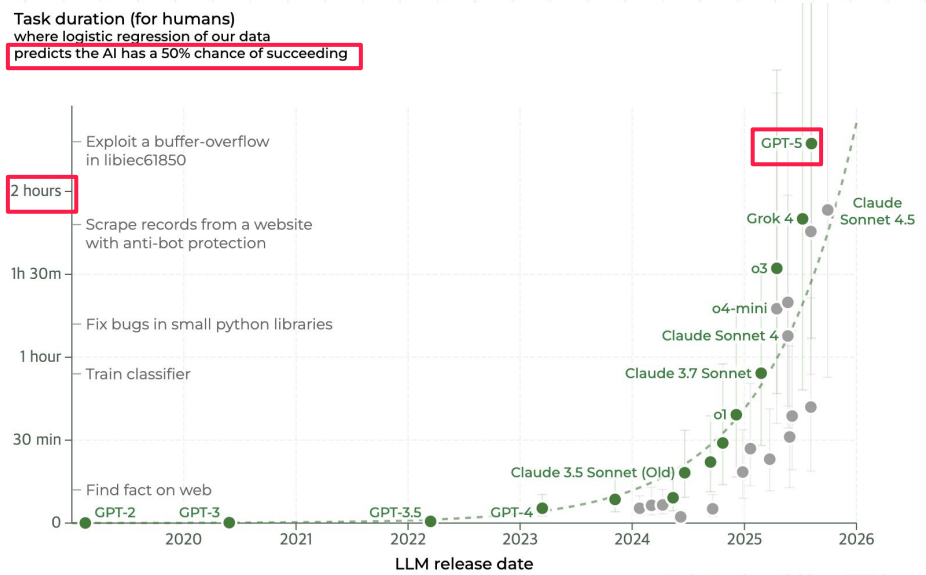
"우리 아이는 오랫동안 혼자 놀아요"

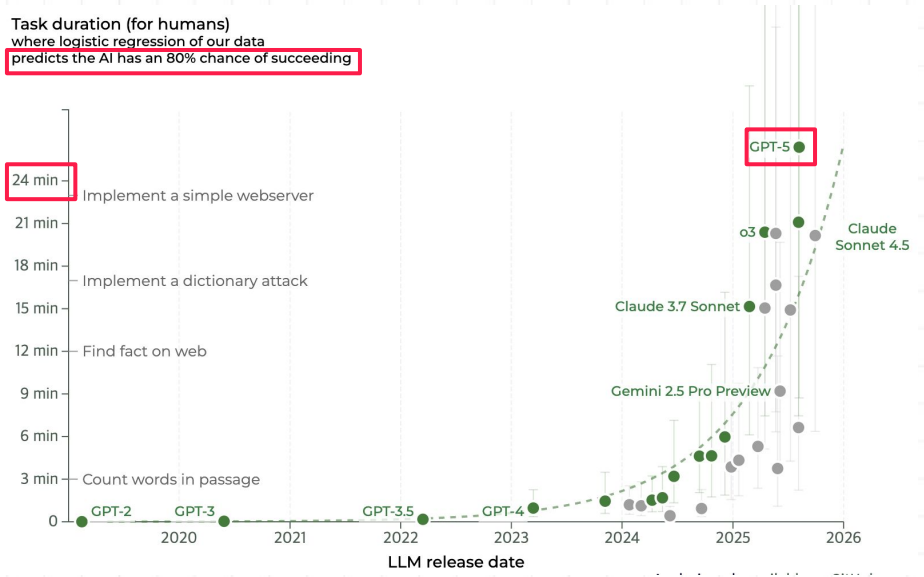
에이전트의 정의?

"An LLM agent runs tools in a loop to achieve a goal"

→ 목표를 성취하기 위해 여러 도구를 사용하며 오랫동안 동작하는 존재

https://metr.org/blog/2025-03-19-measuring-ai-ability-to-complete-long-tasks/





METR 벤치마크: (인간 작업 시간 기준) AI가 얼마나 긴 작업을 일정 비율로 성공시킬 수 있을까?

→ GPT-5는 50% 성공률로 2시간 17분짜리 작업, 80% 성공률로 26분짜리 작업을 할 수 있음

그리고 이 작업 시간이 약 7개월마다 2배씩 늘고 있음 (2025년 3월 Sonnet 3.7은 50%로 54분짜리 작업)

→ 이미 AI는 충분히 발전했는데, 우리가 역량이 딸려서 길게 작업을 못 시키고 있다

https://simonwillison.net/2025/Sep/30/designing-agentic-loops/

Agentic Workflow 설계를 연습하자

When to design an agentic loop

Not every problem responds well to this pattern of working. The thing to look out for here are problems with **clear success criteria** where finding a good solution is likely to involve (potentially slightly tedious) **trial and error**.

Any time you find yourself thinking "ugh, I'm going to have to try a lot of variations here" is a strong signal that an agentic loop might be worth trying!

A few examples:

- **Debugging**: a test is failing and you need to investigate the root cause. Coding agents that can already run your tests can likely do this without any extra setup.
- **Performance optimization**: this SQL query is too slow, would adding an index help? Have your agent benchmark the query and then add and drop indexes (in an isolated development environment!) to measure their impact.
- **Upgrading dependencies**: you've fallen behind on a bunch of dependency upgrades? If your test suite is solid an agentic loop can upgrade them all for you and make any minor updates needed to reflect breaking changes. Make sure a copy of the relevant release notes is available, or that the agent knows where to find them itself.
- **Optimizing container sizes**: Docker container feeling uncomfortably large? Have your agent try different base images and iterate on the Dockerfile to try to shrink it, while keeping the tests passing.

A common theme in all of these is **automated tests**. The value you can get from coding agents and other LLM coding tools is massively amplified by a good, cleanly passing test suite. Thankfully LLMs are great for accelerating the process of putting one of those together, if you don't have one yet.

언제 에이전틱 워크플로우를 설계해야 할까?

→ 명확한 성공 기준을 가지고 시행착오를 통해 성공시킬 수 있는 작업.

(디버깅. 성능 최적화. 의존성 업데이트 등)

이런 작업을 어떻게 떠올릴까?

→ 내가 직접 하기 귀찮은 일을 메타인지한다

이런 작업을 어떻게 시킬까?

→ 내가 직접 할 때 어떤 도구를 조합해 어떻게 하는지 알려주고. 내가 잘 했는지 명확하게 확인하는 방법 알려준다 (CLI 도구 + 자동화 테스트 등)

→ 꼭 필요한 수준의 권한만 주고. 안전하게 오랫동안 돌 수 있는 환경을 마련해준다 (Docker. 리모트 서버 등)

Agentic Workflow 설계 사례: Package7

```
### Task 1: 새로운 <테크 스택>*<라이브러리> 조합 추가

여기서 해야 할 일은 이런 것들이야. (여기서는 테크 스택이 웹 UI라고 가정)
- 웹에서 특정 테크 스택의 LTS 버전 - 너무 실험적이지 않은, 안정적으로 동작하지 않는 버전을 찾는다.
- 웹에서 특정 라이브러리의 LTS 버전을 찾는다.
- 해당 <테크 스택> * <라이브러리> 조합에서:
  - 안정적으로 그 라이브러리 버전을 그 테크 스택에 install하기 위한 코드 스니펫을 찾는다
  - 만약 그 라이브러리를 사용하기 위한 API Key가 필요하다면 발급받는 스텝바이스텝 가이드를 찾는다
    - 가능하면 대신 발급받는다...? 가능할까?
  - 그 테크 스택에서 demo page를 만들기 위한 코드 스니펫을 찾는다
  - 그 demo page에 ['이 라이브러리를 사용할 완료가 됐음'을 검증할 수 있게 하는 버튼]을 구현하는 코드 스니펫을 찾는다
  - 위 demo page에서 '해당 라이브러리 연동이 완료됐음을 검증할 수 있는 e2e 테스트'를 돌리기 위한 라이브러리를 찾는다
- 위에서 찾은 정보들을 조합하여 nextjs:gemini와 유사하게 md 파일과 테스트 파일을 구현한다
  - 이 과정에서 테크 스택에 common하게 쓰일 수 있는 snippet이 있다면 따로 빼둔다

### Task 2: <테크 스택>*<라이브러리> 조합을 연동하는 프롬프트가 실제로 동작하는지 확인

준비: package7 내에 e2e/nextjs/template 에 nextjs starter 복제해놓고, docker 환경 준비해둠

코딩 에이전트에게 이런 명령을 내린다
- docker 환경에서 템플릿을 e2e/nextjs/gemini 에 복제하라
- 이 안에서 package7 nextjs gemini 호출하라
- 여기서 나오는 프롬프트대로 순차적으로 실행해서 step completed 될 때까지 해봐라
- 이 과정애서 실패하는 단계가 있다면 롤백하고, 성공시킬 수 있도록 프롬프트를 수정한 뒤 재시도하라. 단, Acceptance Test는 수정하면 안 된다
- 위 과정에서 수정된 프롬프트처럼 각 source doc이 처음부터 나오게 했으려면 prompter.md 를 어떻게 고쳐야 하는가? 그대로 수정하라
```

👊 시도 4: Package7 - 어떤 SDK든 프롬프트로 연동할 수 있게 하자

이 프로젝트를 제대로 검증하고 확장하기 위해 필요한 2가지 long-running task 설계를 LLM 딥 리서치로 수행. 이 딥 리서치 자체도 long-running task. (클로드 기준 1시간+)

현재 설계한대로 실험해보는 중. 어느정도 성과 나오면 오픈소스로 공개 예정

https://github.com/humanlayer/advanced-context-engineering-for-coding-agents/blob/main/ace-fca.md

https://www.youtube.com/watch?v=Og0mtAbDrDY

Agentic Workflow 설계를 위한 추천 자료들

Getting AI to Work in Complex Codebases

It seems pretty well-accepted that AI coding tools struggle with real production codebases. The Stanford study on AI's impact on developer productivity found:

1. A lot of the "extra code" shipped by AI tools ends up just reworking the slop that was shipped last week.
2. Coding agents are great for new projects or small changes, but in large established codebases, they can often make developers *less* productive.

The common response is somewhere between the pessimist "this will never work" and the more measured "maybe someday when there are smarter models."

After several months of tinkering, I've found that **you can get really far with today's models if you embrace core context engineering principles**.

This isn't another "10x your productivity" pitch. I tend to be pretty measured when it comes to interfacing with the ai hype machine. But we've stumbled into workflows that leave me with considerable optimism for what's possible. We've gotten claude code to handle 300k LOC Rust codebases, ship a week's worth of work in a day, and maintain code quality that passes expert review. We use a family of techniques I call "frequent intentional compaction" - deliberately structuring how you feed context to the AI throughout the development process.

I am now fully convinced that AI for coding is not just for toys and prototypes, but rather a deeply technical engineering craft.

Video Version: If you prefer video, this post is based on a talk given at Y Combinator on August 20th

One More Thing: 기-승-전-프롬프트

의도적으로 개발 방식을 '코드'보다는

'프롬프트'를 중심으로 가져가보자

- 코드 리뷰보다 프롬프트 리뷰
- 페어 코딩보다 페어 프롬프팅
- 코드 커밋보다 프롬프트 커밋

좌측도 중요하지만 우측에 더 큰 가치를 둔다

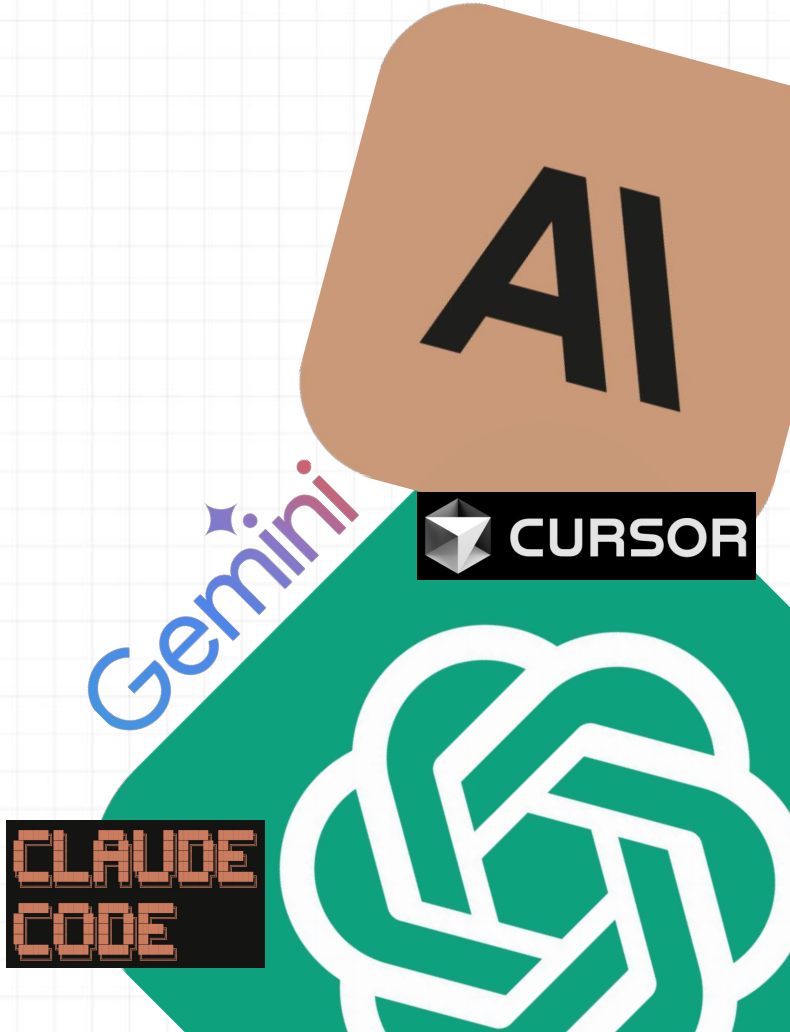
추천 자료들 👉

https://docs.google.com/presentation/d/1xWmcBbSCKgQfN1WOdvG-_uF9AeQwWIm7TdNArQyAMII

https://www.stdy.blog/adding-i18n-robustness-to-legacy-projects-with-ai/

4부

마무리 및 Q&A



Q. 제 세션을 어떤 사람들이 듣게 될까요?
듣고 나서 어떤 변화가 생기길 기대하시나요?

A. AI의 발전으로 직무 생존을 걱정하는 개발자,
특히 5년차 이하 주니어들에게
('정답'은 아닐지라도) 현실적인 자신감이 생기면 좋겠다

요약

- AI가 (아직) 인간을 대체하고 있지는 않다. 그러나 뭔가 큰 게 오고 있는 건 맞다
- FOMO를 이겨내자. 별거 아니다. '라고 할 때 하면' 된다
- Boundary Intelligence에. 일단 How보다는 What과 Why에 집중하자
- 기존에 하던 작업을 1/10의 시간에 하기 위해 노력하고, AI와 Agent를 이용해 10x 이상의 임팩트가 날 만한 새로운 일을 꾸준히 찾아서 시도하고, 공유하자
  - 에이전틱 워크플로우 설계. 즉 AI에게 더 길게 위임하기 위해 고민하고 훈련하자
  - 의도적으로 개발 방식을 '코드'보다는 '프롬프트'를 중심으로 가져가보자
- 건강한 의심과 호기심을 가지고 나만의 커리어 패스를 찾아가보자
  - E2E 프로덕 엔지니어, 소프트웨어 아키텍트&엔지니어, FDE / AX 컨설턴트, UI/UX 디자이너, 교육자 등
- 좋은 사람들과 관계 맺는 게 곧 100x 기회임을 인지하고, 연락하자

Q&A

Q. 이전에는 대규모 시스템을 구축하기 위해 필요한 기술 스택 등 비교적 학습해야할 대상이 명확했던 것 같습니다. 그러나 AI가 발전하며 무엇을 학습해야할 지 그 대상이 모호해진 느낌이 듭니다. 구체적으로 무엇을 어떻게 학습해야한다고 생각하시는지 궁금합니다.

A. 프롬프트 중심 개발 + 에이전틱 워크플로우 설계를 위한 기술 훈련에 집중하길 추천드립니다. AI에게 시킬 일 발견, 더 명확히 요구사항 설명, 제대로 수행했는지 확인할 수 있는 테스트 만들고, 적절한 환경/권한/도구 쥐어주는 등의 기술 훈련

그리고 건강한 의심과 호기심에 기반한 메타인지 높이기. 글쓰기, 다학제적 학습, 개인/팀/조직에 변화 만들기, 인지심리학, 사람들로부터 관심을 끄는 기술 등

Q. 백엔드 개발자(자바 4년)에서 AI 엔지니어로 전환했는데, SI 경력 특성상 아키텍처를 제대로 설계해본 경험이 부족합니다. AI 프로덕트는 고려할 변수가 훨씬 많은데 자원은 한정적이다 보니 어디서부터 어떻게 설계해야 할지 막막합니다. 코르카에서 AI 프로덕트의 아키텍처를 설계하실 때 우선순위를 어떻게 정하시고, 제한된 리소스 안에서 기술 부채를 최소화하는 방법이 있을까요?

A. 어떻게 해야 이 코드베이스에서 AI에게 병렬로 일 시키는 게 더 잘 될까? 에 집중합니다. 레이어별로 경계가 뚜렷하게 설계하고(병렬로 일할 때 conflict 안 나게), 암묵지를 명시지로 다 적어놓고(예를 들어 DB 스키마에 comment 달아서 스키마만 봐도 AI가 구조와 의도를 이해할 수 있게), 구현 패턴을 따라할 수 있는 아주 좋은 사례 한두개는 신중하게 직접 작성하고, 린터와 자동화 테스트를 빡세게 만들고 등등.

강규영님의 훌륭한 글(AI 시대의 소스코드 품질)을 읽어보시길 추천드립니다.