

의세계에 전생한 공학자는 자꾸 모델을 만든다

✦ 의대생 엔지니어의 의료 구조화 노트 ✦



의세계에 전생한 공학자 는 자꾸 모델을 만든다

의대생 엔지니어의 의료 구조화 노트

목차

프롤로그	6
Part 1. 왜 자꾸 모델을 만드는가	23
1. 의대생이 공학을 놓지 않는 이유.....	24
2. 공학적 사고란 무엇인가.....	30
3. 나는 왜 모델을 좋아하는가.....	36
4. 의료를 구조화한다는 것.....	41
Part 2. 구조로 공부하다	45
1. 공부를 구조로 바라본다는 것.....	45
2. 암기와 이해는 다른 층위.....	45
3. 실습은 사고의 전환점.....	48

4. 구조화의 힘	50
5. Anki는 구조 관리 도구.....	55
6. AI를 도구로 쓰는 공부	59
7. 좋은 질문은 무엇일까.....	61
8. 가르친다는 것은 구조를 다시 짓는 일이다	71
Part 3. 사고를 구현하다	84
1. 사고를 구현한다는 것.....	84
2. 구체화와 자동화.....	85
3. 집이 살아있다, HAOS.....	89
4. 작업용 개인 클라우드, Jisong Cloud 6.0.....	94
5. OpenClaw, 그리고 춘식이.....	104
6. 모델링과 확률적 판단	109
7. 감각의 객관화, VoiceGrape.....	112
8. 상태를 모델로, NeuroFrame	115
9. CDSS prototype, EstroFrame	120
Part 4. 기록은 곧바로 데이터가 되지 않는다	124
1. 감각에서 수치로, 수치에서 예측으로	124
2. 의료 텍스트는 왜 바로 데이터가 되지 않는가.....	129

3. 로컬 기반 EMR 파싱, CleanText.....	134
4. EMR 연구용 파싱, CleanEMR.....	143
5. 머신러닝 기반 당뇨 약물 추천, DiaFrame	150
Part 5. 약물은 점이 아니라 곡선이다	158
1. 다학제 약제 PK 분석, PharmaFrame.....	158
2. 수술 전 약을 끊는다는 것	169
3. 모델은 처음부터 맞지 않는다	179
Part 6. AI 시대에 살아남는 것	189
1. AI에게 일을 맡긴다는 건, 좋은 상사가 되는 일이다....	189
2. 판단은 설명될 때 완성된다	200
3. 어떤 의사가 될 것인가.....	202
4. 임상 디지털 트윈 설계자.....	210
에필로그	219
부록 (Appendix)	228
하루 30분, 1년 반. JLPT N1까지.....	228
Obsidian PKM, 연결을 관리하기	231
도메인을 산다는 것.....	241
사고 보조 장치, Antigravity IDE.....	250

사진도 구조를 보는 일이다	259
제작 노트: 의공구 Book Pipeline	265

의세계에 전생한 공학자는 자꾸 모델을 만든다
의대생 엔지니어의 의료 구조화 노트

프롤로그

의세계에 전생했다는 것



Order

Lab

Medication

Note

Vital

Previous Record

EMR

57세 female patient

type 2 diabetes mellitus

insulin follow-up

dyspnea

fever

polyuria

polydipsia

weight loss

palpitation

hand tremor

high blood glucose

eye proptosis

anion gap

urine ketone positive

직업: "의과대학생"
 특성: "공학적 사고, 게으름"
 보유 스킬:
 • "구조화 Lv.25"
 • "아마달말 Lv.10"
 • "전날 밤 베퉀치기 Lv.15"
 현재 퀘스트:
 "다음 환자의 페스피저올론저를 설명하고, 가능한 다이어그노시스와 트리트먼트 플랜을 서술하십시오."

GRAVES DISEASE
갑상선의 목종

TACHYCARDIA
심장의 북

TYPE 2 DIABETES
포도당의 강

의세계
의학은 신이다

임상 진료 지침서
(Clinical Guidelines)

대환자
해리슨의
내과 마도서



와아아아아아앙

존나 센
투명 드래곤이
울부짖었다.

아니,
울부짖었다기보다는
사이렌을 울렸다.



투명 드래곤이
존나 센 이유는
간단했다.

첫째, 저건
드래곤 마운틴, 즉 용산에서 다이렉트로 날아왔다.
중간 경유도 없었고,
누가 부르기 전에 이미 오고 있었다.
응급환자가 생기면 가장 먼저 도착하는 존재.
그게 바로 투명 드래곤이었다.



둘째, 잘 안 보인다.
셋째, 막을 수가 없다.
생백도, 구름도,
전장의 혼란도
저 드래곤의
긴급 이송을
가로막지 못했다.



아니, 그러니까...
저건 드래곤이라기보다
용산발
응급 이송 시스템
아닌가?

그래서 투명 드래곤은 존나 센다.



처음에는
이세계인 줄 알았다.

이머전시 룸에는
트리아지를 기다리는
환자가 밀려 있었고,



드래곤은
코아아아앙이 아니라
왜애애애애 울고 있었고,

어디선가
admission battery
계획이 날아오고 있었다.



아.

이건
이세계가 아니었다.

의세계였다.

내 앞에는
성검 대신
상태창이 떠올랐다.

EMERGENCY

CBC	
WBC	8.91 $10^9/\mu\text{L}$
RBC	4.71 $10^{12}/\mu\text{L}$
HGB	13.6 10g/dL
HCT	41.2 %
PLT	263 $10^3/\mu\text{L}$

상태창도 상태창인데,
퀘스트가 이상했다.

상태창

직업 | **의과대학생**

특성 | 공학적 사고, 게임

보유 스킬

- 구조화 Lv.25
- 야마달달 Lv.10
- 전날 밤 벼락치기 Lv.15

현재 퀘스트

다음 환자의 병태생리를 설명하고,
가능한 diagnosis와
treatment plan을 서술하십시오.

첫 퀘스트가 서술형이었다...?!

EMERGENCY

Chemistry Panel	
Na	139 mEq/L
K	4.2 mEq/L
Cl	104 mEq/L
BUN	13 mg/dL
Creatinine	0.8 mg/dL
Glucose	106 mg/dL
Ca	9.1 mg/dL

보통 첫 퀘스트라면
슬라이드 세 마리 차지하던가,

숲속 약초 채집이라든가,

마을 밖 고블린 퇴치
정도가 국룰 아닌가.

이세계가 아니라, 의세계였다.

그때 누군가의
목소리가 들렸다.

용사여, 일어나세요.
사람을 구해야 합니다.

나는 천천히
고개를 들었다.

눈앞에는 낯선 빛이
번지고 있었다.

신성한 주문,
위대한 용사처럼.

그리고 저 빛에 묻혀진
거대한 마법진.

처음에는 단순한
고대 문양처럼 보였다.

하지만 자세히 보니
그건 주문도, 빛도
세계지도도 아니었다.

Order	Lab	Note	Vital	Medication	Procedure Record	Problem List
Patient Summary Age / Sex: 19 / F Height / Weight: 163 cm / 48 kg Allergy: Penicillin Blood Type: A(+) Code Status: Full Code		Diagnosis (Active) Sepsis, unspecified Pneumonia, viral Acute hypoxemic respiratory failure		Vitals BT: 38.7 °C HR: 120 bpm BP: 96 / 58 mmHg RR: 24 / min SpO ₂ : 90 % (RA)		
Prescription Ceftriaxone 2g IV q24h Acetaminophen 1g PO q6h Norepinephrine IV 0.05 mcg/kg/min		Laboratory WBC: 16.84 10 ⁹ /μL CRP: 12.6 mg/dL Procalcitonin: 3.21 ng/mL Lactate: 2.8 mmol/L		Imaging CXR: Diffuse infiltration in both lung fields Chest CT: GGO in RLL		
ECG Monitor 						

...이거 설마,

Order	Lab	Note	Vital	Medication	Procedure Record	Problem List
Patient Summary Age / Sex: 19 / F Height / Weight: 163 cm / 48 kg Allergy: Penicillin Blood Type: A(+) Code Status: Full Code		Diagnosis (Active) Sepsis, unspecified Pneumonia, viral Acute hypoxemic respiratory failure		Vitals BT: 38.7 °C HR: 120 bpm BP: 96 / 58 mmHg RR: 24 / min SpO ₂ : 90 % (RA)		
Laboratory WBC: 16.84 10 ⁹ /μL CRP: 12.6 mg/dL Procalcitonin: 3.21 ng/mL Lactate: 2.8 mmol/L Hb: 10.2 g/dL Plt: 263 10 ⁹ /μL		Medication List Ceftriaxone 2g IV q24h Acetaminophen 1g PO q6h Norepinephrine IV 0.05 mcg/kg/min		I/O Summary (24h) Intake: 1520 mL Output: 1350 mL Balance: +170 mL		

마법진이
아니라 EMR인가?

+ 환자 정보였다.

나는 천천히 고개를 들었다.

눈앞에는 낮은 빛이 번지고 있었다.

하얀 공간, 희미한 웅성거림.

그리고 내 앞에 펼쳐진 거대한 마법진.

CASE PRESENTATION

57세 여성 환자

Type 2 diabetes mellitus

Insulin follow-up

PATIENT SUMMARY			HISTORY			VITAL SIGNS		
Age / Sex	57세 / Female		• 10년 전 T2DM 진단			BP	128/74 mmHg	
Height / Weight	160 cm / 72 kg		• Insulin therapy 3년째 (Basal + Bolus)			HR	92 bpm	
BMI	28.1 kg/m ²		• 최근 1주일간 발열 및 조열 악화			RR	20 /min	
Allergy	NKDA		• 당뇨, 갈증, 전신 쇠약감 악화			BT	38.2°C	
Code Status	Full Code		• 당뇨, 갈증, 전신 쇠약감 악화			SpO ₂	96 % (RA)	

LAB SUMMARY			
항목	결과	참고치	
Glucose	432 mg/dL	70-110	
Na ⁺	131 mEq/L	135-145	
K ⁺	5.4 mEq/L	3.5-5.1	
Cl ⁻	96 mEq/L	98-107	
HCO ₃ ⁻	15 mEq/L	22-28	
Anion gap	20 mEq/L	8-16	
BUN	28 mg/dL	8-20	
Creatinine	1.2 mg/dL	0.6-1.1	
β-Hydroxybutyrate	3.2 mmol/L	0.6-4	
Serum osmolality	322 mOsm/kg	275-295	
HbA1c	10.2 %	<6.5	

SYMPTOMS			
	Dyspnea		Fever
	Polyuria		Polydipsia
	Weight loss		Palpitation
	Hand tremor		Eye proptosis 의심

CBC			
WBC	8.91	10 ³ /μL	
RBC	4.71	10 ⁶ /μL	
HGB	13.6	g/dL	
HCT	41.2	%	
PLT	263	10 ³ /μL	

Chemistry Panel			
Na	131	mEq/L	
K	5.4	mEq/L	
Cl	96	mEq/L	
HCO ₃	15	mEq/L	
BUN	28	mg/dL	
Cr	1.2	mg/dL	
Glucose	432	mg/dL	

그 아래에는 더 작은 글씨들이 하나씩 깜빡이기 시작했다.

이건 어느 나라 언어지.

ALERTS

- ⚠️ 혈당 급격 상승
- ⚠️ 케톤체 상승
- ⚠️ 고삼투압 경향
- ⚠️ 탈수 위험
- ⚠️ 감염 가능성

추가 검사 필요

ABGA

- Ketone (Serum/Urine)
- Lactate
- CXR
- Urinalysis & Culture
- TSH, Free T4
- 전해질 재확인

CURRENT MEDICATION

- Insulin glargine 24 U qHS
- Insulin aspart 8 U TID AC
- Metformin 1000 mg BID
- Atorvastatin 10 mg qHS
- Lisinopril 10 mg qD

GURRRR-AFFIRMING HORMONE THERAPY

- ⚡ Estradiol valerate 20 mg IM q2 weeks
- ⚡ Spironolactone 100 mg qD
- ⚡ Progesterone 100 mg qHS

★ Last inj. 6 days ago (다음 주기 예정일 D+1)

TIMELINE				
D-7	D-5	D-3	D-1	Today
식욕 저하, 당뇨 시작	피로감 심화, 갈증 악화	복통, 메스꺼움, 호흡 기복됨	의식 혼미 내원 전 발열	응급실 도착

DKA? HHS?

Precipitating factor?

감염?

인슐린 누락?

탈수?

약물 상호작용?

호르몬 주사 관련 이상 반응?

문제가 아니라 보스전 브리핑 아닌가?

이거 설마... 마법진이 아니라

케이스 프레젠테이션인가.



붉은 세계 | 당대사 위기

최종 보스 : 마왕 다이아바타스
(당대사 혼란의 제왕)

DKA (당노성 케톤산증) **HHS** (고삼투성 고혈당 상태)

주요 특징

- 고혈당 악화
- 탈수 및 전해질 이상
- 산중 / 고삼투성
- 의식 저하 위험

주요 단서

- 감염, 약물, 탈수
- 인슐린 부족
- 스트레스 상황 등

푸른 세계 | 감상선 폭주

군단장 : 그레이브스
(감상선 폭주의 지휘관)

심한 빈맥 발한 / 재중감소 안구돌출 손떨림 고열 / 불안

주요 특징

- 감상선 호르몬 과다
- 대사 항진
- 고감신경 항진
- 심혈관계 부담 증가

주요 단서

- 그레이브스병
- 스트레스, 감염 등
- 약물 / 감상선 수술 등





+ 내과 진료의 3대 마도: 당대사 위기의 핵심 치치 +

Fluid

수액 요법
말수 교정, 순환 개선

DKA

Insulin

인슐린 요법 · 교혈당 교정, 세포 정상 인식, 갈증

Potassium correction

K⁺ Mg²⁺ Ca²⁺

전해질 교정
갈증 보충, 마그네슘/칼슘 관리

HHS

바디 마도서의 가르침에 따라
fluid, insulin, potassium correction의 상중 요법으로
급성 전장을 안정화하십시오.

DKA와 HHS의 안개를 걷어내고,
precipitating factor를 찾아야 합니다.

2025 Clinical Practice Guidelines for Diabetes Management in Korea
- 대한당뇨병학회 권고 -

진단

당파형체소, 공복혈당
경구포도당부하검사 등

혈당 조절 목표

개인화된 목표 설정
저혈당 최소화

생활습관 관리

식이, 운동, 계중 관리
갑연, 절주

치료 알고리즘

환자 중심의 단계적
약물 치료 전략

+ 근거 기반, 환자 중심, 안전 우선 +

권장 모니터링 오더 세트 (예시)

- CBC, BMP, ABGA
- Serum osmolality / Ketone
- V/S q1h / EKG
- I/O check / UA

주요 약물 선택지

고혈당 조정

- Regular insulin (IV)
- Glargine / Detemir (SC)

전해질 보충

- KCl, MgSO₄, Ondansetron
- 보충 약물 PPT DVT prophylaxis

GURRER-AFFIRMING HORMONE THERAPY

- Estradiol valerate 20 mg b.i.d 42 weeks
- Spironolactone 100 mg qd
- Progesterone 100 mg qHS

* Last inj. 6 days ago (다음 주기 예정일 D+1)

그만큼 중요하다는 뜻입니다.
급성 전장이 안정되면,
마왕 다이아베타스 타입1류를 계속으로 의식을 바꿔 또 다시 진료해야 합니다.

제책이 너무 길어요.
모든 세트 이름이 전수록 센 건가요?

성검 = 수술

용사여.

이건 내과 의학의
대원작 해리스의
남긴 내과 마도서가
담긴 책입니다.

DKA와 HHS의
안개를 걷어내고,

precipitating
factor를 찾아야
합니다.



EMR CARE



내분비 환자인데
GS 콜했어?
너 미쳤어?

General Surgery
(60대, 베테랑 서진)

아앗...
(아무튼 강한편
되는 거 아닌가?)

결국 중요한 건, 환자를 살리는 지혜와 판단!





선택지는 보이지 않았다.

그저 이해해야 이세계에서 살았다.

다음 환자의 병태생리를 설명하고, 가능한 diagnosis와 treatment plan을 서술하십시오.



그래도 나는 목소리를 살렸다.

언제가는 용사를 부르는 목소리가 아니었다.

학생.

DKA HHS

Graves disease



학과 지금 몇 시예요?

문제, 지문 읽는 중에 자는 게 맞습니까?

문제 다 풀고 자는 거예요?

학생, 일어나세요.



나는 눈을 깜빡였다.

눈앞에는 더 이상
마왕도, 용사도,
기승창 같은 것도,
EMR 마법진도,
대형 상태창도
보이지 않았다.



조용한 강의실.

책상 위의 시험지.

종이 넘기는 소리.

분필 굴리는 소리.

그리고 바로,
지금이라는
교수님의 말.

네, 학생.



나는 내분비학 시험을 치고 있었다.

아, 맞다.

나는 2년 전에 의세계에 전생했었다.

물론 실제로 전생한 것은 아니다. 하지만 체감상으로는 그랬다. 공학을 공부하던 사람이 어느 날 의학이라는 낯선 세계에 들어왔고, 그

곳에는 전혀 다른 언어와 규칙과 시험과 문화가 있었다.

처음에는 모든 것이 외계어 같았다.

당화혈색소, 사구체여과율, 소변 알부민-크레아티닌 비, 체질량지수, 공복혈당, C-펩타이드. 처음 보는 수치들은 어디에 놓아야 할지 알 수 없는 조각처럼 보였다. 당뇨병, 갑상샘질환, 부신, 뇌하수체, 호르몬 축. 질병 이름과 기관 이름은 하나씩 외우는 것만으로도 벅찼다.

약물 이름도 마찬가지였다.

메트포르민, SGLT2 억제제, GLP-1 수용체 작용제, 인슐린, 레보티록신. 처음에는 주문처럼 보였던 이름들이었다. 어떤 약을 언제 쓰고, 어떤 환자에게 피하고, 어떤 부작용을 조심해야 하는지 하나씩 배워야 했다.

그러나 의학의 문제는 단순히 이름을 외우는 데서 끝나지 않았다.

어떤 환자에게 어떤 약을 쓸 것인가. 검사 수치가 애매할 때 어느 쪽으로 해석할 것인가. 연속혈당측정기라는 도구를 언제 사용할 것인가. 수술 전 약을 언제 중단할 것인가. 인공지능이 추천한 답을 어디까지 믿을 것인가. 불확실한 상황에서 무엇을 설명하고, 무엇을 책임질 것인가.

처음에는 그저 살아남기 위해 문제를 풀었다. 외우고, 정리하고, 야마를 찾고, 시험 범위를 줄이려 했다. 시험을 통과해야 했고, 실습을 버텨야 했고, 낯선 언어를 조금이라도 빨리 익혀야 했다.

하지만 시간이 지나면서 알게 되었다.

이 세계의 문제는 단순히 정답을 맞히는 문제가 아니었다.

문제를 이해해야 했다. 조건을 나누어야 했다. 변수를 찾아야 했다. 가정을 세워야 했다. 어디까지 알고, 어디서부터 모르는지 표시해야 했다. 그리고 그 판단을 다른 사람에게 설명할 수 있어야 했다.

그러니까 결국, 모델을 만들어야 했다.

나는 토목공학도였다. 구조물을 이해하려면 하중과 반력, 경계 조건과 안전율을 보아야 한다고 배웠다. 하중이 어디에 걸리는지, 힘이 어떤 방향으로 전달되는지, 어떤 부재가 어느 정도의 응력을 받는지 계산해야 했다. 구조물은 겉모양만 보고 이해할 수 없었다. 그 안에서 힘이 어떻게 흐르는지 보아야 했다.

의학에서도 비슷한 감각을 느꼈다.

환자의 증상은 단순한 문장이 아니었다. 검사 수치는 단독으로 존재하는 숫자가 아니었다. 약물은 한 번 처방되고 끝나는 점이 아니라, 시간에 따라 효과와 위험이 변하는 곡선이었다. 의료 기록은 처음부터 연구용 데이터가 아니었고, 인공지능의 추천은 그 자체로 판단이 아니었다.

각각은 따로 떨어져 있는 것처럼 보였지만, 사실은 서로 연결되어 있었다.

증상과 병력, 검사 수치와 약물, 시간과 반응, 가이드라인과 예외, 환자의 선호와 의사의 판단. 이 모든 것들이 하나의 구조 안에서 움직였다. 그리고 그 구조를 보지 못하면, 의학은 끝없는 암기 목록처럼 느껴졌다.

그래서 나는 자꾸 구조를 그리기 시작했다.

공부를 할 때도 그랬다. 암기해야 할 사실을 그냥 쌓기보다, 무엇이

원인이고 무엇이 결과인지 보고 싶었다. 어떤 개념이 어떤 개념을 지탱하는지, 어떤 수치가 어떤 판단으로 이어지는지, 어떤 기준이 어떤 예외를 갖는지 정리하고 싶었다.

프로젝트를 만들 때도 그랬다. 반복되는 생각을 코드로 옮기고 싶었다. 감각을 수치로 바꾸고, 수치를 관계로 묶고, 관계를 모델로 만들고 싶었다. 목소리의 변화를 음성 지표로 보고, 하루의 상태를 에너지 곡선으로 보고, 호르몬 농도를 시간축 위에서 보고 싶었다.

의료 데이터를 볼 때도 그랬다. 진료 기록은 그 자체로 깔끔한 데이터가 아니었다. 환자를 보기 위해 쓰인 기록에는 생략과 관습과 맥락이 섞여 있었다. 그것을 연구 가능한 형태로 바꾸려면 단순히 복사하고 붙여넣는 것만으로는 부족했다. 어떤 정보를 남기고, 어떤 정보를 지우고, 어떤 변수를 정의하고, 어떤 구조로 정리할 것인지 생각해야 했다.

처음에는 이 모든 것이 서로 다른 일처럼 보였다.

공부는 공부였고, 개발은 개발이었고, 의료 데이터는 의료 데이터였고, 인공지능은 인공지능이었다. 하지만 시간이 지나면서 하나의 흐름이 보이기 시작했다.

감각을 수치로. 수치를 모델로. 모델을 예측으로. 예측을 설명 가능한 판단으로.

이 책은 그 흐름에 대한 기록이다.

공학을 전공한 의대생이 의학이라는 낯선 세계에 들어와, 공부하고, 실습하고, 개발하고, 의료 데이터를 바라보고, 인공지능과 모델링을 고민하면서 남긴 기록이다. 단순한 공부법도 아니고, 프로젝트 소개

도 아니고, 의료 인공지능에 대한 낙관론도 아니다.

이 책에서 말하는 구조화는 정답을 자동으로 만드는 일이 아니다.

의료를 구조화한다는 것은 불확실성을 없애겠다는 뜻이 아니다. 환자를 숫자로 줄이겠다는 뜻도 아니다. 오히려 불확실성을 더 정직하게 드러내고, 판단의 근거를 더 분명히 하고, 환자에게 설명 가능한 형태로 바꾸려는 시도에 가깝다.

모델은 현실을 완벽하게 복제하지 못한다. 모든 모델은 단순화이고, 모든 단순화는 무언가를 버린다. 특히 의료에서는 이 사실을 잊으면 안 된다.

환자는 데이터 포인트가 아니다. 검사 수치는 환자의 전체 상태를 대신하지 못한다. 인공지능의 추천은 의사의 판단을 대체하지 못한다. 시스템은 책임을 대신하지 않는다.

그렇지만 모델은 필요하다.

모델은 정답이 아니라 가설을 제시한다. 좋은 가설은 현실을 조금 더 선명하게 만든다. 무엇을 알고 있는지, 무엇을 모르는지, 어떤 변수가 중요한지, 어떤 가정을 하고 있는지, 어디서부터 예측이 흔들릴 수 있는지를 보여준다.

인공지능도 마찬가지다. 인공지능은 후보를 보여줄 수 있다. 하지만 그 후보가 안전한지, 환자의 맥락에 맞는지, 실제 판단으로 옮길 수 있는지는 다시 검토되어야 한다. 중요한 것은 인공지능이 답을 내는 것이 아니라, 그 답을 검증하고 설명할 수 있는 구조를 만드는 일이다.

결국 마지막에 남는 것은 의사다.

환자 앞에서 설명하는 사람. 불확실성을 숨기지 않고 말하는 사람.
모델이 놓친 맥락을 다시 보는 사람. 인공지능이 제시한 답을 그대로 믿지 않고 검토하는 사람. 최종적인 판단의 책임을 지는 사람.

나는 그런 의사가 되고 싶다.

모든 것을 아는 의사가 되고 싶은 것은 아니다. 그럴 수도 없고, 그래서 안 된다고 생각한다. 대신 내가 알고 있는 것과 모르는 것을 구분할 수 있는 사람이고 싶다. 불확실성을 없애는 척하지 않고, 필요한 만큼 구조화해 설명할 수 있는 사람이고 싶다. 모델을 쓰되 모델을 믿어버리지 않고, 인공지능을 쓰되 책임을 넘기지 않는 사람이고 싶다.

그리고 환자 앞에서, 결국 내가 왜 그렇게 판단했는지 설명할 수 있는 사람이고 싶다.

이 책은 완성된 결론이 아니다.

의세계에 적응해가는 한 공학도의 기록이고, 의사가 되어가는 과정에서 남긴 초판의 지도에 가깝다. 시간이 지나면 일부 생각은 바뀔 것이고, 어떤 모델은 버려질 것이고, 더 나은 질문이 생길 것이다.

하지만 지금 이 시점에서 남겨두고 싶은 말은 분명하다.

현실은 복잡했고, 문제는 애매했고, 마왕은 성실하게 문제를 냈다.
그래서 나는 자꾸 모델을 만들기 시작했다.

Part 1. 왜 자꾸 모델을 만드는가

1. 의대생이 공학을 놓지 않는 이유

의대에 들어오면 이전 전공은 종종 과거가 됩니다. 입학 전에는 무엇을 공부했는지, 어떤 분야에 있었는지, 어떤 방식으로 생각했는지는 생각보다 빠르게 뒤로 밀려납니다. 의대 공부는 양이 많습니다.

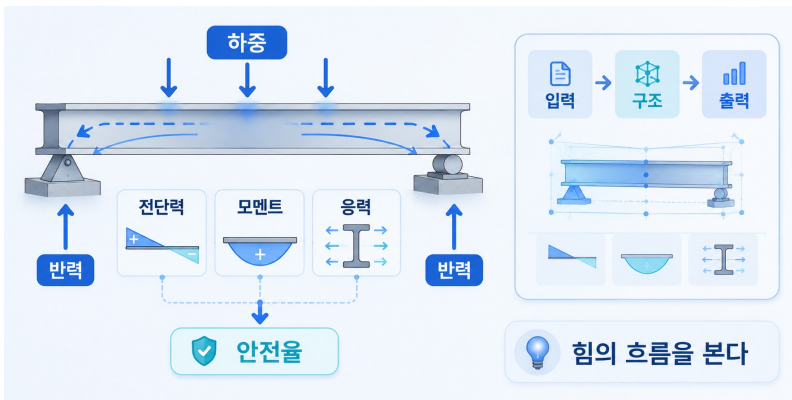
외워야 할 것도 많고, 따라가야 할 일정도 많습니다.

해부학, 생리학, 병리학, 약리학을 지나 내과, 외과, 소아과, 산부인과, 정신과를 공부하다 보면 이전 전공은 잠시 접어두게 됩니다.

저도 그랬습니다.

공학을 전공했지만, 의대에 들어온 뒤에는 한동안 의학을 따라가는 것만으로도 충분히 바빴습니다. 하지만 시간이 지나면서 알게 되었습니다. 공학은 완전히 사라진 것이 아니었습니다. 전공 지식은 흐려졌지만, 생각하는 방식은 남아 있었습니다.

1) 공학은 문제를 구조로 보는 방식이다



제가 공부했던 공학의 중심에는 구조가 있었습니다. 구조역학에서는

힘을 계산합니다. 어디에 하중이 걸리는지 보고, 그 하중이 어떤 방향으로 전달되는지 봅니다.

힘은 모멘트를 만들고, 모멘트는 부재 안의 응력으로 이어집니다. 응력이 재료가 견딜 수 있는 범위를 넘으면 구조물은 안전하지 않습니다. 그래서 구조를 설계할 때는 단순히 모양을 그리는 것이 아니라 힘의 흐름을 계산해야 합니다.

하중. 반력. 전단력. 모멘트. 응력. 변형. 안전율. 각각은 따로 떨어진 숫자가 아닙니다. 하나의 구조 안에서 연결되어 있습니다. 공학은 이렇게 문제를 나누고, 관계를 계산하고, 제약 안에서 가능한 설계를 찾는 일에 가깝습니다.

입력이 무엇인지. 출력이 무엇인지. 조건은 무엇인지. 제약은 어디에 있는지. 오차는 어디서 생기는지. 시스템은 어떤 피드백을 받는지. 공학에서는 문제가 주어지면 먼저 전체 구조를 봅니다. 현실은 복잡하지만, 그 복잡함을 그대로 붙잡고 있으면 아무것도 설계할 수 없습니다. 그래서 나눕니다. 단순화합니다. 모델을 만듭니다. 그리고 그 모델이 어디까지 유효한지 확인합니다.

이 사고방식은 의학에서도 계속 남았습니다.

2) 의학도 입력과 출력 사이의 판단이다

처음 의학을 공부할 때는 의학이 거대한 지식 목록처럼 보였습니다. 질병 이름. 진단 기준. 검사 수치. 치료 알고리즘. 약물 용량. 금기와 부작용.

끝없이 외워야 하는 항목들의 집합처럼 느껴졌습니다. 하지만 실습을 돌고, 환자를 보고, 교수님들의 판단을 옆에서 보면서 조금 다르게 보이기 시작했습니다. 의학도 결국 입력과 출력 사이의 판단이었습니다.

환자의 증상과 병력. 신체진찰 소견. 검사 결과. 기저질환과 복용약. 나이와 생활 맥락. 환자의 선호와 치료 목표.

이런 것들이 입력입니다.

그 입력을 바탕으로 검사를 더 할지, 약을 쓸지, 수술을 할지, 지켜볼지, 설명하고 안심시킬지 결정합니다.

그것이 출력입니다.

물론 의학은 공학보다 훨씬 더 복잡합니다. 환자는 기계가 아니고, 몸은 단순한 구조물이 아니며, 같은 입력이 항상 같은 출력을 만들지도 않습니다.

그래서 더 어렵습니다.

하지만 그렇기 때문에 오히려 구조가 필요합니다. 입력이 불안정하고, 조건이 변하고, 출력이 확률적일수록 판단의 구조를 더 분명히 해야 합니다.

3) 당뇨약 선택도 구조의 문제다

A1c 하나만으로는 부족하다



내분비를 공부하면서 이 점을 자주 느꼈습니다. 예를 들어 당뇨병 약제를 선택하는 일은 단순히 혈당을 낮추는 약을 고르는 일이 아닙니다.

A1c가 얼마인지. 공복혈당과 식후혈당 중 무엇이 문제인지. 신기능은 어떤지. 체중은 어떤지. 심혈관질환이나 심부전 위험은 있는지. 저혈당 위험은 얼마나 되는지. 환자의 나이와 생활 패턴은 어떤지. 주사제를 받아들일 수 있는지. 비용과 순응도는 어떤지.

이런 조건들이 함께 들어옵니다.

같은 A1c라도 어떤 환자에게는 metformin을 유지하는 것이 적절할 수 있고, 어떤 환자에게는 SGLT2 inhibitor나 GLP-1 receptor agonist를 고려할 수 있습니다. 또 다른 환자에게는 오히려 저혈당 위험을 줄이기 위해 치료 강도를 낮춰야 할 수도 있습니다.

가이드라인은 기준을 줍니다.

하지만 실제 선택은 그 기준을 환자의 조건 안에 다시 배치하는 일입니다. 이 과정은 구조역학과 닮아 있습니다. 하중 하나만 보고 구

조를 설계하지 않듯이, A1c 하나만 보고 약제를 선택할 수는 없습니다. 하중, 재료, 지점 조건, 안전율을 함께 보듯이 혈당, 신기능, 체중, 심혈관 위험, 저혈당 위험, 환자 선호를 함께 봐야 합니다.

판단은 하나의 수치에서 나오지 않습니다. 조건들의 관계에서 나옵니다. 이 생각은 나중에 DiaFrame을 구상하는 배경이 되었습니다. 당뇨약 선택을 단순 추천 문제가 아니라 입력 조건과 판단 방향, 안전성 제약을 가진 의사결정 구조로 보고 싶었습니다.

4) 프로젝트는 사고방식의 부산물이다

제가 여러 프로젝트를 만든 것도 사실 코딩 자체가 목적이었기 때문은 아닙니다. 반복되는 판단이 보이면 그 판단을 분해하고 싶었습니다. 어떤 정보가 입력으로 들어오고, 어떤 조건에서 판단이 달라지고, 어떤 출력이 필요한지 정리하고 싶었습니다. CleanText는 의료 텍스트를 분석 가능한 데이터로 바꾸려는 시도였습니다.

VoiceGrape는 목소리라는 감각을 측정 가능한 음성 지표로 바꾸려는 시도였습니다. EstroFrame은 호르몬 농도를 시간 위의 곡선으로 보려는 시도였습니다. NeuroFrame은 하루의 상태를 에너지 흐름으로 모델링하려는 시도였습니다. DiaFrame은 당뇨약 선택을 검증 가능한 의사결정 구조로 바라보려는 시도였습니다.

각 프로젝트의 주제는 달랐지만, 출발점은 비슷했습니다. 애매한 판단을 조금 더 구조화된 형태로 바꾸고 싶었습니다. 그래서 제게 코딩은 개발자가 되기 위한 기술이라기보다 생각을 구현하는 방법에

가까웠습니다. 생각을 표로 만들고, 규칙으로 만들고, 모델로 만들고, 사용자가 볼 수 있는 화면으로 만드는 일.

그 과정에서 공학은 과거 전공이 아니라 현재의 사고방식으로 남아 있었습니다.

5) 의학 안으로 공학을 가져오기



저는 의대생입니다.

앞으로 의사가 되기 위해 의학을 더 많이 공부해야 합니다. 환자를 봐야 하고, 병력을 들어야 하고, 신체진찰을 해야 하고, 검사 결과를 해석해야 하고, 치료의 책임을 져야 합니다. 공학이 이 일을 대신해 주지는 않습니다. 하지만 공학적 사고는 이 일을 바라보는 방식을 바꿔줍니다.

복잡한 문제를 구조로 보고, 불완전한 정보를 입력으로 받아들이고, 제약 속에서 판단하고, 오차와 예외를 인정하고, 필요하다면 그 과정을 시스템으로 구현하는 것.

이것은 의학과 충돌하지 않습니다.

오히려 의학 안에서 계속 쓸 수 있는 사고방식입니다. 제가 공학을 놓지 않는 이유는 과거 전공에 대한 미련 때문이 아닙니다. 의학을 공학으로 대체하고 싶어서도 아닙니다. 다만 의학을 공부할수록 공학적으로 생각하는 방식이 임상 문제를 더 선명하게 보게 해준다고 느끼기 때문입니다.

의학은 사람을 다루는 학문입니다.

그래서 단순한 모델로 환자를 설명할 수는 없습니다. 하지만 사람을 다루는 학문이기 때문에 오히려 더 신중한 구조가 필요합니다. 저는 공학을 버리고 의학으로 온 것이 아닙니다. 공학을 의학 안으로 가져오고 있습니다.

2. 공학적 사고란 무엇인가

공학적 사고라는 말을 들으면 먼저 기계나 회로, 코드 같은 것을 떠올리기 쉽습니다. 수식을 많이 쓰고, 계산을 하고, 무언가를 설계하고, 프로그램을 만드는 방식.

물론 그것도 공학의 일부입니다.

하지만 제가 생각하는 공학적 사고의 핵심은 특정 기술에 있지 않습니다. 공학적 사고는 문제를 조건과 구조로 나누어 보는 방식입니다.

무엇이 들어오고, 어떤 상태를 거치며, 무엇이 나가는지. 어떤 제약이 있고, 어디서 오류가 생기며, 어떤 피드백이 다시 시스템에 영향을 주는지.

이런 방식으로 문제를 바라보는 태도에 가깝습니다.

1) 문제를 그대로 붙잡지 않는다



현실의 문제는 대부분 복잡합니다.

처음부터 깔끔하게 정리되어 있지 않고, 원인과 결과가 섞여 있으며, 중요한 정보와 덜 중요한 정보가 함께 들어 있습니다. 공학적 사고는 그 복잡함을 그대로 붙잡고 있지 않습니다.

먼저 나눕니다.

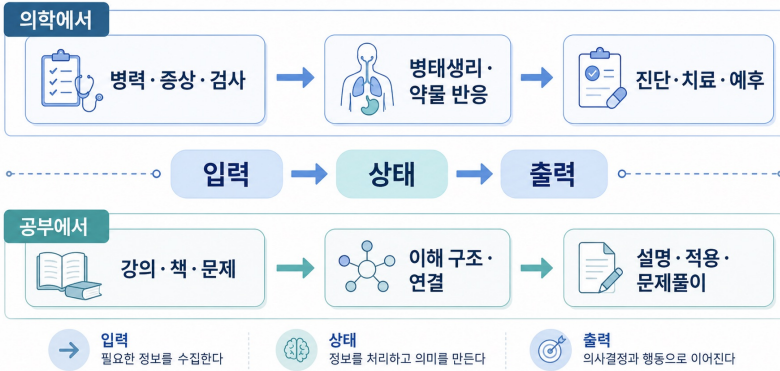
입력은 무엇인가. 출력은 무엇인가. 중간 상태는 무엇인가. 바뀌지 않는 조건은 무엇인가. 바뀔 수 있는 변수는 무엇인가. 반드시 지켜야 하는 제약은 무엇인가.

이렇게 나누고 나면 문제는 조금 덜 막막해집니다. 예를 들어 의대 공부도 처음에는 거대한 암기 덩어리처럼 보입니다. 질병 이름, 증상, 검사, 치료, 약물, 부작용이 한꺼번에 밀려옵니다.

하지만 구조로 나누면 달라집니다.

환자에게 어떤 문제가 있고, 그 문제를 설명하는 병태생리는 무엇이며, 어떤 검사로 확인하고, 어떤 기준으로 치료를 결정하는지. 공부의 대상이 단순한 목록에서 하나의 흐름으로 바뀝니다. 공학적 사고는 복잡한 문제를 단순하게 무시하는 것이 아니라, 다룰 수 있는 단위로 나누는 일에서 시작합니다.

2) 입력, 상태, 출력을 본다



공학에서는 시스템을 볼 때 입력과 출력을 자주 생각합니다. 무엇이 들어오고, 그 안에서 어떤 변화가 생기고, 결과로 무엇이 나오는지 봅니다.

이 구조는 의학에도 잘 맞습니다.

환자의 병력, 증상, 신체진찰, 검사 결과는 입력입니다. 환자의 몸 안에서 일어나는 병태생리와 약물 반응은 상태입니다. 진단, 치료 결정, 예후, 부작용은 출력에 가깝습니다.

공부도 마찬가지입니다.

책이나 강의에서 들어오는 정보가 입력입니다. 그 정보를 내가 어떤 구조로 이해하고 연결하는지가 상태입니다. 문제를 풀거나, 설명하거나, 환자에게 적용하는 것이 출력입니다. 이렇게 보면 공부는 단순히 정보를 많이 넣는 일이 아닙니다.

입력을 처리하는 내부 구조를 만드는 일입니다. 같은 내용을 읽어도 어떤 사람은 금방 잊고, 어떤 사람은 문제에 적용합니다.

차이는 단순 기억량만이 아닙니다.

정보가 들어왔을 때 그 정보를 어디에 연결하고, 어떤 기준으로 꺼내 쓸 수 있는지가 중요합니다.

3) 제약을 먼저 본다

공학적인 사고에서 중요한 것은 무엇을 할 수 있는지만 보는 것이 아닙니다. 무엇을 할 수 없는지도 함께 봅니다.

예산이 제한되어 있을 수 있습니다. 시간이 부족할 수 있습니다. 재료가 버틸 수 있는 한계가 있을 수 있습니다. 시스템이 감당할 수 있는 부하가 있을 수 있습니다.

이런 제약을 무시하면 그럴듯한 설계는 가능할지 몰라도 실제로 작동하는 설계는 어렵습니다.

의학에서도 제약은 중요합니다.

가이드라인상 가능한 치료가 있어도 환자의 신기능이 나쁘면 쓸 수 없는 약이 있습니다. 효과가 좋은 약이라도 저혈당 위험이 크면 조심해야 합니다. 좋은 검사라도 지금 당장 시행하기 어렵거나 환자에

게 부담이 클 수 있습니다. 이론적으로 맞는 설명이라도 환자가 이해하고 받아들일 수 있어야 합니다.

제약은 판단을 방해하는 요소가 아닙니다. 오히려 판단을 현실에 붙잡아두는 요소입니다. 공학적 사고는 최선의 답을 공중에 그리지 않고, 제약 안에서 가능한 답을 찾습니다.

4) 오류와 피드백을 전제로 한다



공학적 사고는 처음부터 완벽한 답이 나온다고 생각하지 않습니다. 모델에는 오차가 있고, 측정값에는 노이즈가 있으며, 설계는 실제 조건에서 수정될 수 있습니다.

그래서 피드백이 필요합니다.

예상한 결과와 실제 결과가 다르면 어디서 차이가 생겼는지 봅니다. 입력이 잘못되었는지, 가정이 틀렸는지, 모델이 너무 단순했는지, 측정 과정에 문제가 있었는지 확인합니다.

이 태도는 공부에도 중요합니다.

문제를 틀렸을 때 단순히 답을 외우는 것으로 끝내면 같은 유형을 다시 틀릴 수 있습니다.

왜 틀렸는지 봐야 합니다.

개념을 몰랐는지. 조건을 잘못 읽었는지. 감별진단의 우선순위를 잘못 잡았는지. 치료 기준을 착각했는지. 문제의 함정을 놓쳤는지.

오답은 단순한 실패가 아니라 내 사고 구조의 오류를 보여주는 피드백입니다. Anki도 이 관점에서 보면 단순 암기 도구가 아닙니다. 내가 어떤 정보를 반복해서 잇는지, 어떤 연결이 약한지, 어떤 기준을 자주 혼동하는지 보여주는 피드백 시스템에 가깝습니다.

5) 도구는 사고를 구현한다

공학적 사고는 머릿속에서만 끝나지 않습니다. 반복되는 구조가 보이면 그 구조를 도구로 만들 수 있습니다. CleanText는 의료 텍스트를 정리하는 반복 작업을 데이터 처리 구조로 옮긴 것입니다. Dia Frame은 당뇨약 선택에서 필요한 조건과 안전성 제약을 의사결정 지원 구조로 옮긴 것입니다.

Jisong Cloud는 파일 업로드, 메모, AI 분석, 문서 변환처럼 반복되는 개인 작업 흐름을 하나의 작업대로 묶은 것입니다. 이런 도구들은 서로 달라 보입니다.

하지만 공통점이 있습니다.

반복되는 판단을 발견하고, 그 판단을 입력과 출력으로 나누고, 중간 과정을 규칙이나 모델로 만들고, 사용 가능한 형태로 구현했다는

점입니다. 공학적 사고는 생각을 구조로 만들고, 그 구조를 다시 현실에서 작동하게 만드는 방식입니다.

6) 공학적 사고는 차갑지 않다

공학적 사고를 말하면 차갑고 기계적인 태도로 오해될 수 있습니다. 하지만 저는 그렇게 생각하지 않습니다. 문제를 구조로 본다는 것은 사람을 단순화한다는 뜻이 아닙니다. 오히려 복잡한 문제 앞에서 무엇을 놓치고 있는지 확인하려는 태도에 가깝습니다.

의학에서 환자는 단순한 입력값이 아닙니다. 환자의 삶과 맥락, 선호와 불안은 항상 판단 안에 들어와야 합니다. 다만 그 맥락까지 포함해 더 신중하게 판단하려면 구조가 필요합니다. 공학적 사고는 정답을 자동으로 만들어내는 방식이 아닙니다.

문제를 더 명확히 보고, 제약을 인정하고, 오류를 수정하고, 더 나은 판단을 준비하는 방식입니다. 제가 의대 공부와 프로젝트에서 계속 공학적 사고를 붙잡는 이유도 여기에 있습니다. 공학적 사고는 기계를 만드는 사고가 아니라 복잡한 현실을 다룰 수 있는 구조로 바꾸는 사고입니다.

3. 나는 왜 모델을 좋아하는가

저는 모델을 좋아합니다.

처음부터 의학적 모델을 생각했던 것은 아닙니다. 시작은 공학이었습니다.

1) 보이지 않는 변형을 수치로 바꾸다

공학을 공부할 때 strain gauge를 다룬 적이 있습니다. 구조물에 힘이 가해지면 미세한 변형이 생깁니다. 그 변형은 눈으로는 거의 보이지 않습니다. 하지만 strain gauge를 붙이면 그 미세한 변형이 전기저항의 변화로 바뀝니다.

하중이 가해지고, 재료가 아주 조금 늘어나거나 줄어들고, 그 변화가 전기 신호로 바뀌고, 다시 변형률이라는 수치로 환산됩니다. 눈으로 보이지 않던 상태가 측정 가능한 값이 되는 과정이었습니다.

그게 마음에 들었습니다.

막연한 힘이나 변형을 감각이 아니라 수치로 다루는 것. 그리고 그 수치를 통해 구조물이 어떤 상태에 있는지 추정하는 것. 현실은 그 대로 보면 너무 복잡합니다. 하지만 적절한 센서와 가정, 계산 구조를 통과하면 그 복잡함 중 일부는 해석 가능한 형태가 됩니다.

아마 그때부터였던 것 같습니다.

저는 보이지 않는 상태를 측정 가능한 구조로 바꾸는 일에 끌렸습니다.

2) 의학도 모델의 연속이다

공학에서 모델은 자연스러운 도구였습니다. 하중이 있으면 변형이 있고, 입력이 있으면 출력이 있고, 조건이 바뀌면 결과도 바뀝니다. 물론 현실의 구조물은 완벽하게 단순하지 않습니다. 재료는 균질하

지 않고, 경계 조건은 이상적이지 않으며, 측정값에는 오차가 있습니다.

그래서 모델이 필요합니다.

모델은 현실을 완벽하게 복제하기 위해 존재하지 않습니다. 복잡한 현실에서 판단에 필요한 관계를 꺼내기 위해 존재합니다. 의학을 공부하면서도 비슷한 감각을 느꼈습니다. 의학은 처음에는 사실을 위우는 학문처럼 보입니다.

진단 기준, 검사 수치, 약물 용량, 치료 알고리즘. 외워야 할 것은 많고, 실제로 시험도 기억을 요구합니다. 하지만 조금 더 깊이 들어가면, 의학은 사실보다 모델에 가깝다는 생각이 들었습니다.

병태생리는 질병이 몸 안에서 어떻게 진행되는지에 대한 모델입니다. 진단 기준은 복잡한 임상 양상을 몇 개의 조건으로 압축한 모델입니다. 위험도 평가는 미래에 일어날 가능성을 추정하는 모델입니다. 가이드라인은 수많은 연구와 환자군을 바탕으로 만든 의사결정 모델입니다.

우리는 현실 자체를 직접 다루는 것이 아니라, 현실을 이해하기 위해 만든 여러 층위의 모델을 다룹니다.

3) 예측 가능성에 끌리는 이유

제가 모델을 좋아하는 가장 큰 이유는 예측 가능성 때문입니다. 예측 가능하다는 것은 모든 일이 정확히 맞아떨어진다는 뜻이 아닙니다.

오히려 반대에 가깝습니다.

현실은 항상 빗나갑니다.

환자는 평균과 다르고, 검사 수치는 애매하며, 약물 반응은 개인마다
다릅니다. 가이드라인에 없는 예외도 많습니다. 하지만 그럼에도 불
구하고 우리는 다음 상태를 조금이라도 예상해야 합니다.

이 약을 쓰면 상태는 어느 방향으로 움직일까. 치료를 중단하면 위
험은 얼마나 커질까. 수술을 미루면 무엇을 얻고 무엇을 잃을까. 지
금의 수치 변화가 일시적인 흔들림인지, 실제 악화의 신호인지 어떻
게 볼 것인가.

임상은 결국 이런 질문의 연속입니다. 모델은 이 질문에 정답을 주
지 않습니다. 하지만 질문을 더 명확하게 만듭니다.

무엇을 알고 있는지. 무엇을 모르는지. 어떤 변수가 중요한지. 어떤
가정을 하고 있는지. 어디서부터 예측이 흔들릴 수 있는지.

좋은 모델은 결과만 보여주지 않습니다. 불확실성의 위치도 함께 보
여줍니다.

4) 모델은 현실을 버리기도 한다

그래서 모델의 한계는 중요합니다.

모델을 좋아한다고 해서 모델을 믿어버리면 안 됩니다. 모델은 현실
을 단순화합니다. 그리고 모든 단순화는 무언가를 버립니다.

빠진 변수. 잘못된 가정. 측정되지 않은 맥락. 평균에 가려진 개인차
. 데이터에 들어오지 않은 환자의 삶.

이런 것들을 잇는 순간, 모델은 판단을 돕는 도구가 아니라 현실을 왜곡하는 장치가 됩니다.

특히 의료에서는 더 그렇습니다.

환자는 데이터 포인트가 아닙니다. 검사 수치는 환자의 전체 상태를 대신하지 못합니다. AI의 추천은 의사의 판단을 대체하지 못합니다.

따라서 모델은 항상 조심스럽게 다뤄야 합니다.

모델을 만든다는 것은 현실을 통제한다는 뜻이 아닙니다. 오히려 현실이 얼마나 통제되지 않는지를 인정한 뒤, 그 안에서 조금 더 나은 판단 구조를 만드는 일에 가깝습니다.

저는 그 태도가 마음에 듭니다.

모든 것을 안다고 말하지 않는 태도. 하지만 아무것도 할 수 없다고 포기하지 않는 태도. 불확실성을 지우지 않고, 그 위치를 표시해두는 태도.

5) 좋은 가설이 현실을 선명하게 만든다

제가 관심 있는 의료의 방향도 여기에 있습니다. 애매한 정성적 판단을 가능한 만큼 정량화하고, 흩어진 수치를 관계로 묶고, 그 관계를 모델로 만들고, 그 모델을 통해 다음 상태를 예측해보는 일. 이것이 제가 생각하는 의료 구조화의 출발점입니다.

모델은 정답이 아닙니다. 모델은 가설입니다. 하지만 좋은 가설은 막연한 현실을 조금 더 선명하게 만듭니다. 제가 모델을 좋아하는 이유는 현실이 단순해서가 아닙니다.

현실이 복잡하기 때문입니다.

4. 의료를 구조화한다는 것

의학을 공부하면서 가장 자주 마주친 것은 불확실성이었습니다. 환자는 교과서처럼 말하지 않습니다. 검사 수치는 항상 명확한 답을 주지 않습니다. 약물 반응은 사람마다 다르고, 같은 진단명 아래에서도 실제 환자의 상태는 조금씩 다릅니다.

처음에는 이 불확실성이 막막했습니다. 가이드라인을 외우면 답이 보일 줄 알았습니다. 하지만 실제 임상에서는 가이드라인을 알고 난 뒤에도 다시 판단해야 하는 순간이 계속 생겼습니다.

1) 불확실성을 구조로 바라보기

수술을 할 것인가, 조금 더 지켜볼 것인가. 어떤 약을 먼저 쓸 것인가. 치료를 강화할 것인가, 유지할 것인가, 중단할 것인가. 검사 수치가 애매할 때 어느 쪽으로 해석할 것인가.

교과서와 가이드라인은 기준을 줍니다. 하지만 그 기준 사이에는 늘 예외가 있습니다. 그때부터 저는 의학을 조금 다르게 보게 되었습니다. 의료는 사실을 단순히 모으는 일이 아니라, 불완전한 정보를 구조화해 판단 가능한 형태로 바꾸는 일에 가깝습니다.

이 책에서는 그 과정을 기록하려 합니다. 의료를 구조화한다는 것은 모든 것을 딱딱 떨어지는 숫자로 만들겠다는 뜻이 아닙니다. 의학에는 여전히 정성적인 판단이 필요합니다. 환자의 말투, 표정, 생활 맥

락, 선호, 불안, 순응도 같은 것들은 하나의 숫자로 완전히 환원되지 않습니다. 다만 애매한 정성적 판단 중 일부는 더 명확한 형태로 바꿀 수 있습니다.

느낌으로만 남아 있던 것을 수치로 바꾸고, 흩어진 수치를 관계로 묶고, 그 관계를 모델로 만들어 앞으로의 변화를 예측해보는 일. 제가 관심 있는 것은 바로 그 과정입니다. 첫 번째는 감각을 수치로 바꾸는 일입니다.

2) 감각을 수치로 바꾸기

“목소리가 잠긴 것 같다.” “몸 상태가 좋지 않다.” “약이 잘 듣는 것 같지 않다.”

이런 문장들은 감각의 언어입니다. 중요하지만, 그대로는 비교하거나 추적하기 어렵습니다. 그래서 감각은 때로 수치가 되어야 합니다.

목소리는 피치와 포먼트, 지터와 시머로 바뀔 수 있습니다. 피로는 수면 시간, 카페인, 업무량, 주관적 명료도의 조합으로 볼 수 있습니다. 호르몬 치료는 단일 검사 수치가 아니라 시간에 따른 농도 곡선으로 이해할 수 있습니다.

두 번째는 수치를 모델로 바꾸는 일입니다. 수치는 그 자체로 답이 아닙니다. 혈당, eGFR, A1c, estradiol 농도, 약물 반감기 같은 값들은 각각 따로 존재할 때보다 서로 연결될 때 의미를 갖습니다. 의학은 많은 숫자를 다루지만, 숫자를 나열하는 것만으로 판단이 만들어지지 않습니다.

중요한 것은 관계입니다.

어떤 값이 변했고, 그 변화가 어떤 조건에서 발생했으며, 앞으로 어떤 방향으로 움직일 가능성이 있는지 보는 일.

그때 수치는 모델이 됩니다.

3) 수치를 관계와 모델로 묶기

세 번째는 모델을 예측과 설명에 사용하는 일입니다. 모델은 정답이 아닙니다. 현실을 완벽히 복제하지도 못합니다. 하지만 좋은 모델은 질문을 더 명확하게 만듭니다.

4) 예측과 설명으로 연결하기

이 환자의 약물 농도는 언제 가장 높아질까. 수술 전 약을 중단하면 어느 시점에 기준선 아래로 떨어질까. AI가 추천한 약제 변경 방향은 실제 처방과 얼마나 일치할까. 이 판단에서 위험한 예외는 무엇일까.

모델은 결정을 대신하지 않습니다. 다만 판단의 구조를 보여줍니다.

제가 궁극적으로 관심 있는 방향은 임상 디지털 트윈입니다. 디지털 트윈이라는 말은 거창하게 들리지만, 제가 생각하는 임상 디지털 트윈은 환자를 완벽하게 복제하는 시스템이 아닙니다. 오히려 현실적인 의미에서는 의사결정에 필요한 변수들을 모아 환자의 현재 상태와 앞으로의 변화를 근사하는 모델에 가깝습니다.

검사 수치, 약물, 시간, 위험도, 생활 맥락, 치료 반응을 하나의 구조

안에 놓고 그 환자에게 가능한 선택지를 더 선명하게 보는 일. 그것이 제가 생각하는 의료 구조화의 방향입니다. 저는 앞으로 이 책에서 이런 주제들을 다루려 합니다.

EMR 텍스트를 데이터로 바꾸는 일. 의료 AI의 추천을 검증 가능한 구조로 제한하는 일. 약물 농도를 시간축 위의 곡선으로 바라보는 일. 환자의 상태를 하나의 동적인 모델로 이해하려는 시도. 그리고 AI 시대에 의사가 어떤 책임을 가져야 하는지에 대한 생각.

의료를 구조화한다는 것은 의료를 차갑게 만든다는 뜻이 아닙니다. 오히려 환자를 더 정확하게 이해하기 위한 방식입니다. 불확실성을 없앨 수는 없습니다. 임상에서 모든 정보를 완벽하게 알 수도 없습니다. 하지만 불확실성을 그대로 방치할 것인지, 아니면 가능한 만큼 구조화해 더 나은 판단으로 가져갈 것인지는 다릅니다.

5) 구조화는 책임 있는 판단을 돕는다

저는 후자에 관심이 있습니다.

감각을 수치로, 수치를 모델로, 모델을 예측으로, 예측을 설명 가능한 판단으로 바꾸는 일. 그 과정에서 의사는 사라지지 않습니다.

오히려 더 중요해집니다.

모델은 가설을 제시하고, AI는 후보를 보여주고, 시스템은 구조를 정리합니다. 하지만 최종적으로 환자 앞에서 설명하고, 맥락을 판단하고, 책임지는 사람은 여전히 의사입니다.

이 책은 그 사이를 다룹니다.

의학과 공학 사이. 감각과 수치 사이. 모델과 환자 사이. 예측과 책임 사이. 의료를 구조화한다는 것은 정답을 자동으로 만드는 일이 아닙니다. 판단을 더 명확하게 만들고, 불확실성을 더 정직하게 다루고, 환자에게 설명 가능한 형태로 바꾸는 일입니다. 저는 그 과정을 기록하려 합니다.

Part 2. 구조로 공부하다

1. 공부를 구조로 바라본다는 것

이 책에서는 공부를 “암기량”이 아니라 “구조”로 다룹니다. 무엇을 외웠는지가 아니라, 무엇을 먼저 이해해야 하는지에 관심이 있습니다. 의대 공부, 실습, 반복, Anki, 그리고 모델링까지. 각각의 주제를 따로 다루지만, 중심 질문은 같습니다.

“이건 어떤 구조로 이루어져 있는가?” 정답을 제공하기보다는 기준을 정리합니다. 방법을 나열하기보다 사고의 틀을 제시합니다. 이곳은 공부법을 설명하는 공간이 아니라, 공부를 바라보는 관점을 기록하는 공간입니다.

2. 암기와 이해는 다른 층위

의대에 와서 가장 많이 들은 말은 “결국은 암기”라는 말이었습니다.

틀린 말은 아닙니다. 시험은 기억을 묻습니다. 진단 기준도, 약물 용량도, 수치도 결국은 외워야 합니다. 하지만 저는 이해하지 못한 내용은 오래 남지 않는 타입입니다.

단기 기억에 억지로 밀어 넣을 수는 있어도, 시험이 끝나면 같이 사라집니다. 제게 암기는 이해 위에 올라가는 마지막 단계에 가깝습니다. 그래서 공부를 할 때 먼저 묻습니다.

“왜 이렇게 되는가?”

1) 이해가 먼저다

병태생리든, 약물 기전이든, 진단 알고리즘이든 논리적 흐름이 보이면 기억은 훨씬 덜 힘들어집니다. 개별 사실이 아니라, 구조 안에 배치된 정보가 되기 때문입니다. 이해는 흩어진 점들을 선으로 연결하는 작업이고, 그 선이 만들어지면 암기는 부가적인 일이 됩니다.

그렇다고 모든 것을 이해하려고만 하면 공부가 느려집니다. 의학에는 원리로 환원되지 않는 영역도 많습니다. 그래서 저는 암기할 것과 이해할 것을 구분하려 합니다.

이해할 것은 다음과 같습니다.

2) 암기할 것과 이해할 것을 나눈다

- 왜 이런 증상이 나타나는지
- 왜 이 약이 이 질환에 쓰이는지
- 왜 이 검사 수치가 이렇게 변하는지

즉, 인과와 구조가 있는 영역입니다. 반대로 암기할 것은 대체로 이런 것들입니다.

- 수치와 기준
- 예외 조항
- 분류표의 세부 항목

구조를 이해한 뒤에도 남은, 규칙의 표면에 해당하는 부분들입니다. 이 구분이 생기고 나서부터 공부는 조금 달라졌습니다. 이해해야 할 부분에서는 시간을 아끼지 않고, 암기해야 할 부분에서는 과감히 도구를 사용합니다. Anki 같은 반복 시스템은 이 지점에서 힘을 발휘합니다. 이해 위에 얹힌 사실은 반복만으로도 단단해집니다.

3) 반복 도구는 이해 위에서 쓴다

저는 암기를 부정하지 않습니다.

다만 암기가 공부의 시작이 아니라, 구조를 세운 뒤에 올라가는 마감재라고 생각합니다. 앞으로 이 브런치에서는 이런 기준들을 정리해보려 합니다. 무엇을 외울지보다, 무엇을 먼저 이해해야 하는지에 대해. 저에게 공부는 정보 수집이 아니라, 구조를 세우는 일에 가깝습니다.

이 기준이 실제로 어떻게 작동했는지는, 실습을 돌면서 처음 체감하게 되었습니다.

3. 실습은 사고의 전환점

본과 2학년 때 임상 과목을 공부하면서 가장 힘들었던 건, 내용의 양보다도 “맥락의 부재”였습니다. 가이드라인과 정의, 진단 기준과 치료 알고리즘을 외워야 했지만, 왜 이런 구조로 짜여 있는지까지는 온전히 이해하지 못한 채 시험을 준비했습니다. 병태생리와 연결하려 해도 어딘가 억지로 끼워 맞추는 느낌이었고, 결국은 다시 암기로 돌아가곤 했습니다. 그래서 실습에 큰 기대는 없었습니다.

이미 외운 내용을 병원에서 다시 확인하는 정도일 거라고 생각했습니다. 공부를 더 어렵게 만드는 또 하나의 과정일 뿐이라고 여겼습니다.

그런데 실제로 병동을 돌고, 환자를 마주하고, 케이스를 발표하면서 공부의 결이 달라졌습니다. 교과서에서 보았던 문장이 환자 앞에서 다시 떠올랐습니다. 흩어져 있던 개념들이 “왜”라는 질문 아래에서 연결되기 시작했습니다. 예를 들어 단순히 외워두었던 검사 수치 하나가, 환자의 증상과 병력, 영상 소견과 맞물리며 인과의 흐름 안에 자리 잡았습니다.

1) 실습이 맥락을 복원한다

그 순간, 암기된 문장은 정보가 아니라 설명이 되었습니다. 케이스 발표는 특히 강한 전환점이었습니다. 한 질환을 깊이 파고들어 병태생리부터 치료 선택까지 다시 정리하다 보니, 본과 2학년 때 억지로 외웠던 내용이 다른 모습으로 돌아왔습니다. 그때는 단순한 정의였

던 문장이, 이제는 논리적 결과처럼 느껴졌습니다.

복습이라는 표현이 더 이상 정확하지 않았습니다. 오히려 재구성에 가까웠습니다. 이 과정을 겪으면서 느낀 건, 의대 공부는 한 번의 이해로 끝나지 않는다는 점이었습니다. 본과 2학년, 3학년, 4학년을 거치며 같은 질환을 여러 번 마주하게 됩니다.

국시를 준비하는 과정도 결국 반복입니다. 하지만 그 반복은 단순한 재암기가 아닙니다. 처음에는 암기로 시작한 내용이, 두 번째에는 이해로 넘어가고, 세 번째에는 더 명료한 구조로 정리됩니다. 어떤 부분은 암기에서 이해로 이동하고, 어떤 부분은 이미 이해했다고 생각했지만 다시 이해하며 깊이가 더해집니다.

2) 반복은 이해를 다시 세운다

암기와 이해는 대립하는 개념이 아니라, 순환하는 과정에 가깝다는 생각이 들었습니다. 이해가 충분하지 않을 때는 암기가 버팀목이 되고, 반복된 암기는 결국 이해를 촉진합니다. 실습은 그 순환을 가속하는 장치처럼 느껴졌습니다. 이제는 실습을 시험 준비의 연장이 아니라, 사고를 재정렬하는 시간으로 봅니다.

교과서 속 문장이 환자라는 맥락을 만나면서 살아 움직이고, 그 경험이 다시 공부의 방향을 바꿉니다. 그래서 반복은 지루한 재탕이 아니라, 점점 더 선명해지는 재해석에 가깝습니다.

결국 공부는 외운 내용을 쌓아두는 일이 아니라, 같은 내용을 다른 층위에서 다시 바라보는 일이라는 생각이 듭니다. 실습을 통해 저는

암기가 이해로, 이해가 다시 더 깊은 이해로 넘어가는 과정을 처음으로 체감했습니다.

3) 재암기가 아니라 재구성이다

4. 구조화의 힘

항생제를 처음 공부할 때는 막막했습니다. 이름은 끝도 없이 많고, 비슷해 보이는데 미묘하게 다르고, 커버 범위도 제각각이었습니다. 하나하나 외우려고 할수록 더 헛갈렸습니다.

그래서 방식을 바꿨습니다.

항생제를 “목록”으로 보지 않고, 구조로 보려고 했습니다.

1) 잘게 쪼개기

1. Penicillin - Ampicillin/Amoxicillin - Ampicillin-Sulbactam - Vancomycin(MRSA)
- Nafcillin(MSSA)
2. Ampicillin - S. pneumoniae
3. Cephalosporin - 1st gen cefazoline
3rd gen Ceftriaxone, cefotaxime, ceftazidime
4th gen cefepime
4. Aminoglycoside(gentamicin) - 그람(-) 타겟
5. 혐기균 항생제 - Clindamycin, Metronidazole(+아메바)
6. Macrolide(azithromycin) / **Doxycycline** - Bordetella, Vibrio, Rickettsia, Trichomonas
BLT (Borrelia, Leptospira, Treponema), Mycoplasma
7. **Fluoroquinolone** - Ciprofloxacin - GI tract(개실, 염증성 설사), Urinary(방광염, 신우신염, 전립선염)
Levo/Moxifloxacin - Respiratory

*파란색 표시 - 임상부, 소아 금기

워드 정리본 일부

먼저 크게 세 덩어리로 나눴습니다. β -lactam 계열, 그 외의 그람 음성 커버 약제, 그리고 특수 균종을 담당하는 약제들. 그 다음에는 세대별로, 그리고 커버 범위가 넓어지는 방향으로 다시 정리했습니다. 예를 들어 cephalosporin은 1세대에서 4세대로 갈수록 그람 음성 커버가 확장된다는 흐름을 먼저 이해했습니다.

이렇게 세 개씩 묶어보니 패턴이 보이기 시작했습니다.

1세대는 주로 그람 양성,

3세대는 그람 음성 확장,

4세대는 더 넓은 범위.

단어를 외우는 게 아니라, 이동 방향을 기억하는 느낌이었습니다.

2) 상황별 재배치

상황별 항생제

1) S. aureus 감염

예시 - 피부 연조직 감염, 유방염, 감염성 관절염, 감염성 심내막염, 수술 전 예방적 항생제

항생제 - Amoxicillin/Ampicillin, Nafcillin, 1st Cepha, Penicillin(감염성 심내막염)

내성 - Nafcillin(MSSA), Vancomycin(MRSA)

2) 혐기균 감염

예시 - 농양, 여성 생식기 감염, 근막염. **혈액/소변은 혐기성이 아님!!**

항생제 - Clindamycin(폐농양, 자궁내막염, toxic shock, 과사성 근막염),

Metronidazole(복강, 질염, 파상풍).

3) STD 감염

예시 - Neisseria, Chlamydia 급성 요도염, 골반염

항생제 - 3rd Cepha(Neisseria), Doxycycline(Chlamydia)

4) 위장관 감염(염증성 설사)

경험적 항생제 - Ciprofloxacin(세균성 이질이 대부분)

Macrolide 적응증 - 장기간 Campylobacter, 이질, Vibrio cholerae

Salmonella typhi, Vibrio vulnificus - 3rd Cepha, Ciprofloxacin, Doxycycline (독한세퀴)

워드 정리본 일부

피부 감염, 호흡기 감염, 복강 내 감염처럼 임상 맥락 속에 넣어보니, 항생제가 더 이상 단독 정보가 아니었습니다. “이 균을 커버하려면 이 계열이 필요하다”는 식으로 인과가 생겼습니다.

결국 제가 한 일은 단순했습니다.

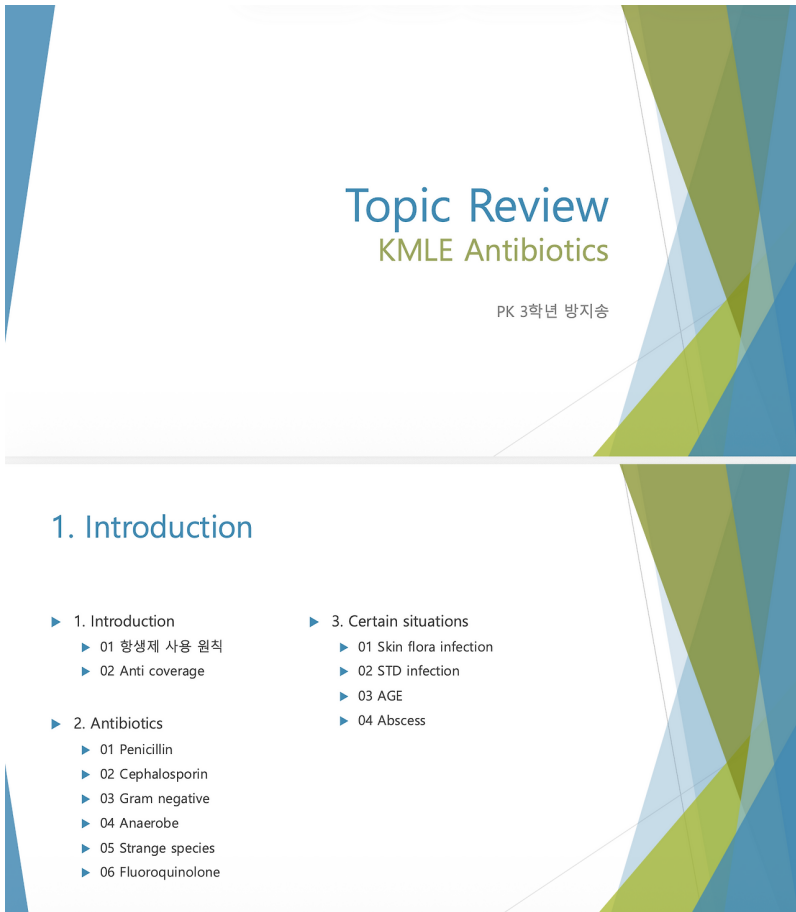
많은 약을 줄인 게 아니라, 묶었습니다.

- 세대별로 묶고

- 커버 범위로 묶고
- 임상 상황으로 묶고

이 과정을 거치자, 개별 약물은 구조 속의 위치를 갖게 되었습니다. 위치가 생기니 기억은 자연스럽게 따라왔습니다. 지금은 예전처럼 하나하나 떠올리려 애쓰지 않아도, 구조를 떠올리면 그 안에서 이름이 같이 따라 나옵니다. 항생제를 다 외웠다고 느끼는 순간은, 사실 다 기억했기 때문이 아니라, 구조를 기억하게 되었기 때문이라고 생각합니다.

3) 설명하며 정리하기



발표 PPT 표지 및 목차

일련의 과정을 거치며 자연스럽게 정리본을 만들게 되었고, 나중에는 동기들에게 설명하기 위해 PPT까지 제작하게 되었습니다. 강의를 준비하면서 느낀 건, 설명은 이해를 압축하는 과정이라는 점이 있습니다. 내가 모호하게 알고 있던 부분을 더 명료화 하여 슬라이드 한 장으로 정리하고, 구조를 명확하고도 간결하게 표현할 수 있었습

니다. 아래는 제가 강의한 파일의 링크입니다.

https://drive.google.com/file/d/1boVPckI9mTgu89BxclFlgS9DWYtZg6If/view?usp=share_link

암기는 여전히 필요합니다.

하지만 구조 없이 암기하면 흠어집니다. 구조를 만든 뒤의 암기는 오래 갑니다. 항생제 정리는 제게 “구조화의 힘”을 처음으로 체감하게 해준 예시였습니다.

5. Anki는 구조 관리 도구

의대에 와서 가장 많이 들은 말은 결국은 암기라는 말이었습니다. 실제로 시험은 수치와 기준, 정의와 분류를 묻습니다. 그래서 한동안은 무작정 외우는 방식으로 공부를 해보기도 했습니다. 하지만 이해 없이 밀어 넣은 정보는 오래 가지 않았습니다. 시험이 끝나면 같이 사라졌습니다.

그래서 방식을 바꾸기로 했습니다.

암기를 줄이는 대신, 암기의 위치를 바꾸기로 했습니다. Anki를 단순 반복 암기 도구로 쓰지 않고, 이해 위에 남는 정보를 관리하는 시스템으로 쓰기 시작했습니다. 그때부터 공부의 결이 조금씩 달라졌습니다.

덱	새카드	학습	만기
- KMLE	0	0	269
+ 마이너	0	0	54
- 메이저	0	0	215
- 내과	0	0	104
감염	0	0	18
내분비	0	0	9
류알혈종	0	0	27
소화기	0	0	11
순환기	0	0	16
신장	0	0	8
호흡기	0	0	15
산부인과	0	0	32
소아과	0	0	25
외과	0	0	54

Anki 메인 화면

1) 카드를 만드는 기준

<div style="border: 1px solid gray; padding: 5px;"> <p>▼ Front</p> <p>이상지질혈증 기준 total cholesterol, HDL-C, TG</p> </div> <div style="border: 1px solid gray; padding: 5px; margin-top: 5px;"> <p>▼ Back</p> <p>≥240, <40, 200-499, ≥500</p> </div>	<div style="border: 1px solid gray; padding: 5px;"> <p>▼ Front</p> <p>CN 위치</p> </div> <div style="border: 1px solid gray; padding: 5px; margin-top: 5px;"> <p>▼ Back</p> <p>3,4: midbrain 5,6,7,8: pons 9,10,11,12: medulla</p> </div>
--	---

카드 예시

카드를 만들기 전에 먼저 묻습니다. 이걸 이해할 것인가, 암기할 것인가. 병태생리, 인과관계, 질환의 흐름은 카드로 만들지 않습니다. 그건 노트에서 먼저 구조를 세웁니다.

왜 이런 증상이 나타나는지, 왜 이런 치료를 선택하는지 이해가 먼저입니다. Anki에는 구조 위에 남는 '표면 정보'만 올립니다.

예를 들어,

- 진단 기준 수치
- 응고인자 번호
- 항생제 세대별 성분명
- 특정 검사 cut-off 값

이건 이해를 아무리 해도 결국 정확히 기억해야 하는 정보입니다.

카드는 최대한 잘게 쪼갭니다. 한 카드에 한 질문만 둡니다.

질문은 짧고, 답도 간결하게. 문장을 길게 쓰지 않습니다. 회상 단위를 작게 만들수록 반복은 덜 부담스럽고, 복원은 더 정확해집니다.

Anki는 이해를 만드는 공간이 아니라, 이해 위에 얹힌 사실을 단단하게 만드는 공간이라고 생각합니다.

2) 활용 방식

텍은 과목별로 나누지만, 실제로 돌릴 때는 상황 단위로 묶습니다.

예를 들어 항생제, 응급, 감별, 고열처럼 태그를 중심으로 다시 돌립니다. 시험은 과목 순서대로 나오지 않기 때문입니다. 문제를 읽는 순간, 과목명이 아니라 구조가 먼저 떠오르도록 훈련합니다.

Image Occlusion은 도표, 비교표, 알고리즘 구조에 사용합니다. 그림을 가리고, 키워드만 남깁니다. 카드를 보는 순간 전체 구조가 머릿속에서 복원되도록 설계합니다.

이 방식의 장점은 분명합니다.

암기가 이해를 방해하지 않습니다. 오히려 이해를 빠르게 호출하는 장치가 됩니다. 반복이 쌓일수록 정보는 따로 외운 느낌이 아니라, 구조 속에서 자연스럽게 튀어나옵니다.

3) 본과 3학년 임종평에서 느낀 점

임종평을 준비하면서 새로 외운 내용은 거의 없었습니다. 이미 만들어둔 카드들을 반복하는 것이 대부분이었습니다. 시험장에서 문제를 읽으면, 먼저 구조가 떠올랐습니다. 그 다음에 세부 수치와 기준이 따라왔습니다.

이전에는 시험이 끝나면 상당 부분을 잊어버렸지만, 이번에는 다르다는 느낌이 들었습니다. 단순히 단기 기억으로 밀어 넣은 것이 아니라, 여러 번 복원된 정보였기 때문이라고 생각합니다. 성적도 만족스러웠습니다. 본3 전국 21등의 성적은 제 방식이 틀리지 않았다는 확인처럼 느껴졌습니다. 하지만 더 의미 있었던 건 점수 자체가 아니라, 누적된 구조가 실제 시험장에서 작동했다는 경험이었습니다. 암기와 이해를 분리해서 관리한 시간이 결과로 이어졌다는 점에서, 이번 임종평은 작은 검증이었습니다.

저는 Anki를 암기 앱으로 사용하지 않습니다. 구조를 유지하는 시스

템으로 사용합니다. 이해는 먼저 세우고, 암기는 그 위에 엮습니다. 둘을 분리하면 오히려 더 오래 남는다고 느꼈습니다. 이번 결과는 그 구조가 제대로 쌓이고 있었다는 신호였습니다. 앞으로 같은 방식으로 축적해볼 생각입니다.

6. AI를 도구로 쓰는 공부

AI를 쓰기 시작하면서 느낀 건, 이전 만능 도구가 아니라 특정 작업에 특화된 도구라는 점이었습니다. 막연히 “AI가 대신 공부해준다”는 기대는 오래 가지 않았습니다. 대신 AI가 잘하는 영역을 정확히 이해하고 나니, 공부의 보조 장치로는 꽤 유용하다는 생각이 들었습니다.

AI는 패턴을 인식하는 데 강합니다. 그리고 단순 반복 작업을 빠르게 처리합니다. 그리고 AI를 잘 쓰려면 AI에 대해 잘 알아야 합니다. 저는 이 세 가지에 집중해서 AI를 사용하고 있습니다.

1) AI는 패턴 인식 기계다

AI는 정보를 ‘이해’한다기보다, 방대한 데이터 속에서 패턴을 찾아냅니다. 그래서 복잡한 개념을 정리해달라고 하면, 핵심 구조를 뽑아주는 데는 능숙합니다. 예를 들어 긴 글을 넣고 요약을 요청하면, 반복되는 키워드와 구조를 정리해줍니다.

시험 범위가 넓을 때 큰 틀을 먼저 잡는 데 도움이 됩니다. 개념 간의 연결고리를 빠르게 보여주는 것도 장점입니다. 하지만 이 과정에

서 오류도 생깁니다. 그래서 저는 AI가 만들어준 구조를 그대로 믿지 않습니다.

먼저 제가 이해하고, 그 다음에 AI의 정리를 비교합니다. AI는 초안 생성기이지, 최종 답안 작성자가 아닙니다.

AI가 잘하는 건 패턴 추출이고,
제가 해야 하는 건 판단과 검증입니다.

2) 단순 반복은 말기다

공부를 하다 보면 생각이 필요 없는 반복 작업이 꽤 많습니다. 문장 정리, 표 정리, 분류 재배치 같은 작업입니다. 이런 부분은 AI에게 맡깁니다.

특히 Notebook LM을 활용합니다. 자료를 업로드하고, 그 안에서만 질문을 던지면 출처 기반으로 답을 정리해줍니다. 방대한 PDF나 강의 노트를 빠르게 훑고 구조를 파악할 때 유용합니다. 예를 들어 교과서 한 챕터를 올리고, 이 단원의 핵심 개념을 계층 구조로 정리해줘 라고 요청하면, 전체 윤곽을 먼저 잡을 수 있습니다.

그 다음에는 제가 직접 노트를 다시 정리합니다. Notebook LM은 방향을 잡아주는 도구이고, 최종 정리는 제 몫입니다. AI를 쓰는 목적인 시간을 아끼기 위함이 아니라, 사고에 더 많은 시간을 쓰기 위함입니다.

3) AI를 잘 쓰려면, AI를 공부해야 한다

AI를 많이 써본다고 해서 잘 쓰게 되는 건 아니라고 느꼈습니다. 오히려 AI가 어떻게 작동하는지 이해하고 나서야, 제대로 활용할 수 있었습니다. AI는 통계적 모델입니다. 확률적으로 가장 그럴듯한 답을 생성합니다.

이 구조를 이해하면, AI의 한계도 같이 보입니다. 왜 틀릴 수 있는지, 왜 자신 있게 말해도 오류일 수 있는지 이해하게 됩니다. 그래서 저는 AI를 쓰는 것과 동시에, AI 자체에 대해서도 공부하려고 합니다. 모델이 어떻게 학습되는지, 한계는 무엇인지, 어디까지 믿어도 되는지 알아야 도구로 사용할 수 있습니다.

AI는 공부를 대신해주는 존재가 아닙니다. 공부를 설계하는 데 도움을 주는 도구에 가깝습니다. AI에게 맡길 것과, 제가 해야 할 것을 구분하는 순간, AI는 훨씬 유용해졌습니다.

AI를 활용한 공부는 결국 효율의 문제가 아니라 태도의 문제라고 생각합니다. 패턴 인식과 단순 반복은 맡기고, 이해와 판단은 스스로 합니다. AI를 잘 쓰고 싶다면, 먼저 AI를 공부해야 한다는 걸 요즘 더 많이 느낍니다. 도구를 이해할수록, 도구는 더 조용하고 정확하게 작동합니다.

7. 좋은 질문은 무엇일까

임상실습을 돌면서 질문이 조금 달라졌습니다. 처음에는 질문을 해야 한다고 생각했습니다. 교수님 앞에서 멀뚱히 있으면 안 될 것 같

고, 무언가 듣고 있다는 신호를 보내야 할 것 같고, 실습생으로서 적극성을 보여야 할 것 같았습니다.

그래서 질문을 하기는 했습니다. 하지만 그 질문들은 지금 생각해 보면 조금 겹돌았습니다. 이 약은 뭐가요. 이 검사는 왜 하나요. 이 환자는 수술하나요. 이 질환의 증상은 뭐가요.

물론 이런 질문도 필요합니다. 모르는 것을 확인하는 일은 중요합니다. 하지만 어느 순간부터 느꼈습니다. 좋은 질문은 단순히 모르는 것을 묻는 질문이 아니었습니다. 좋은 질문은 판단이 갈리는 지점을 묻는 질문이었습니다.

좋은 질문은 지식의 빈칸이 아니라, 판단의 갈림길을 묻습니다.

1) 질문도 사실 알아야 할 수 있다

질문을 많이 하는 것은 단순히 적극성의 문제처럼 보입니다. 하지만 실습을 돌면서 알게 된 것은 질문도 사실 뭘 알아야 할 수 있다는 점입니다. 아무것도 모르면 질문이 잘 나오지 않습니다. 무엇이 중요한지 모르고, 무엇이 예외인지 모르고, 어디에서 판단이 갈리는지도 보이지 않습니다.

그러면 질문은 대개 단순 확인형이 됩니다.

“이게 뭐가요?”

“왜 하나요?”

“이 약은 뭐예요?”

이 질문들은 출발점으로는 필요합니다. 하지만 계속 이 수준에만 머

무르면 임상 판단의 구조를 보기 어렵습니다. 반대로 어느 정도 배경지식이 생기면 질문이 달라집니다. 교과서적으로는 A라고 알고 있는데, 이 환자에서는 왜 B를 선택했을까.

가이드라인상으로는 이렇게 되어 있는데, 실제 병동에서는 어디까지 적용될까. 이 기준은 어느 정도까지 원칙이고, 어느 지점부터 예외가 될까. 교수님은 이 상황에서 어떤 변수를 가장 크게 보고 판단하셨을까. 이런 질문은 배경지식이 조금 있어야 가능합니다.

질문은 무지의 표시만은 아닙니다. 오히려 어느 정도 이해가 쌓였을 때 내가 모르는 경계가 보이기 시작합니다.

2) 실습 초반의 질문은 보여주기식에 가까웠다

실습 초반에는 질문이 약간 보여주기식이었습니다.

물론 나쁜 의미만은 아닙니다.

처음 병동에 들어가면 무엇을 봐야 할지 잘 모릅니다. 수술방에서도, 외래에서도, 병동에서도 정보는 너무 많고 저는 아직 초보입니다.

그러면 질문은 “제가 듣고 있습니다”라는 신호에 가까워집니다. 교수님이나 전공의 앞에서 아무 말도 하지 않는 것이 어색해서 질문을 합니다. 그런데 그때의 질문은 대개 단순 지식 확인에 가깝습니다.

무엇인지 묻고, 왜 하는지 묻고, 언제 하는지 묻습니다.

이것도 필요한 단계입니다.

하지만 그 단계에서는 질문자의 사고 구조가 깊게 드러나지는 않습니다. 질문을 하고 있지만, 아직 상황을 붙잡고 있는 것은 아닙니다.

3) 관심 있는 분야를 깊게 보면 진짜 질문이 생긴다

질문이 달라진 것은 관심 있는 분야를 반복해서 보면서부터였습니다. 어떤 분야를 조금 더 깊게 보면 단순한 사실보다 예외가 먼저 보이기 시작합니다. 교과서에는 A라고 되어 있는데 실제 환자에서는 B를 선택하는 경우가 있습니다. 가이드라인에는 기준이 있지만 교수님은 환자의 다른 변수를 더 크게 보는 것처럼 보일 때가 있습니다. 수술 과정도 마찬가지입니다. 책에서는 단계가 깔끔하게 나뉘어 있지만 실제 수술에서는 조직 상태, 유착, 출혈, 해부학적 변이, 이전 수술력에 따라 판단이 계속 바뀝니다. 이때 질문은 더 이상 “질문해야 하니까 하는 질문”이 아닙니다. 진짜 궁금해서 나오는 질문이 됩니다. 왜 여기서는 원칙대로 가지 않았을까. 어디서부터 판단이 바뀌었을까. 이 상황에서 교수님은 무엇을 가장 먼저 보셨을까. 관심 있는 분야를 깊게 보면 질문이 자연스럽게 생깁니다. 질문은 억지로 만들어내는 말이 아니라 이해가 어느 정도 쌓였을 때 생기는 틈입니다.

4) 좋은 질문은 어려운 질문이 아니다



좋은 질문은 어려운 질문이 아닙니다. 어려운 용어를 많이 쓰는 질문도 아닙니다. 오히려 좋은 질문은 판단이 갈리는 지점을 정확히 묻는 질문입니다.

예를 들면 이런 구조입니다.

제가 이해한 바로는 A가 원칙입니다. 그런데 이 환자에서는 B를 선택하신 것처럼 보였습니다. 이때 판단을 바꾼 핵심 변수는 C였나요, 아니면 다른 기준이 있었나요?

이 질문은 단순히 지식을 묻지 않습니다. 먼저 내가 이해한 원칙을 밝힙니다. 그다음 실제 관찰한 상황을 말합니다. 마지막으로 그 차이를 만든 판단 기준을 묻습니다.

이 구조가 좋은 이유는 분명합니다. 첫째, 기본 지식이 있음을 보여줍니다. 둘째, 실제 상황을 관찰했다는 것이 드러납니다. 셋째, 교수님이나 전공의의 판단 구조를 끌어낼 수 있습니다.

좋은 질문은 정답을 요구하지 않습니다. 판단의 기준을 묻습니다.

좋은 질문은 원칙, 관찰, 판단 기준으로 이루어진다.

5) 질문에도 수준이 있다



질문은 대략 단계가 있습니다.

처음에는 개념을 묻습니다.

“이게 뭔가요?”

그다음에는 원리를 묻습니다.

“왜 하나요?”

조금 더 가면 적응증을 묻습니다.

“언제 하나요?”

그다음부터 질문은 판단에 가까워집니다. “A와 B 중 왜 A를 선택하나요?” “어디서부터 observation이 아니라 intervention으로 넘어가나요?” “교수님은 실제 임상에서 어떤 변수를 가장 크게 보시나요?”

이 단계로 가면 질문은 단순 지식 확인이 아닙니다. 임상 판단의 구조를 추적하는 일이 됩니다. 실습을 돌면서 질문이 늘었다는 것은

단순히 말이 많아졌다는 뜻이 아니었습니다. 상황의 해상도가 올라갔다는 뜻에 가까웠습니다.

이전에는 그냥 지나갔던 장면에서 판단의 갈림길이 보이기 시작한 것입니다.

6) 수업마다 질문이 생기고, 수술마다 질문이 생긴다는 것 어느 순간부터 수업을 들으면 질문이 두세 개씩 생겼습니다. 수술을 보면 네다섯 개씩 궁금한 점이 생겼습니다.

이 변화가 꽤 크게 느껴졌습니다.

처음에는 질문을 해야 해서 질문했습니다. 나중에는 질문이 생겨서 질문했습니다.

둘은 완전히 다릅니다.

전자는 태도의 문제에 가깝습니다.

후자는 이해도의 문제입니다.

질문이 자연스럽게 생긴다는 것은 그 장면을 그냥 보고 있지 않았다는 뜻입니다. 어떤 원칙을 알고 있고, 그 원칙과 다른 상황을 감지했고, 그 차이를 설명할 기준이 궁금해졌다는 뜻입니다. 임상실습에서 질문 능력이 자란다는 것은 결국 관찰의 밀도가 올라가는 일입니다. 같은 장면을 봐도 더 많은 구조를 보는 것입니다.

7) 질문을 보면 사람의 수준이 보인다

요즘은 질문을 들으면 그 사람이 어디까지 이해하고 있는지 조금 보

이는 것 같습니다. 질문에는 생각보다 많은 정보가 들어 있습니다. 무엇을 핵심 문제로 보는지. 어떤 배경지식을 가지고 있는지. 어디서 판단이 갈린다고 보는지. 맥락을 잡고 있는지. 지역적인 단어에만 매달리는지. 단순 사실을 묻는지, 사고 구조를 묻는지. 그래서 질문은 단순한 발화가 아닙니다.

그 사람의 사고 해상도를 보여주는 지표입니다. 좋은 질문을 하는 사람은 대개 상황을 어느 정도 구조화하고 있습니다. 모르는 것이 많더라도 자기가 어디를 모르는지 알고 있습니다. 반대로 질문이 계속 걸돌면 아직 문제의 중심을 잡지 못했을 가능성이 큽니다.

물론 이것을 남을 평가하는 도구로만 쓰고 싶지는 않습니다. 오히려 저 자신에게 더 많이 적용하게 됩니다. 내 질문이 걸돌고 있다면 아직 구조를 충분히 보지 못한 것입니다. 내 질문이 판단 기준으로 향하고 있다면 조금씩 임상적 사고에 가까워지고 있는 것입니다.

8) 실습에서 바로 쓸 수 있는 질문의 형태

실습에서 좋은 질문을 만들기 위해 몇 가지 틀을 생각해볼 수 있습니다. 첫 번째는 원칙-예외형 질문입니다.

제가 이해한 원칙은 A인데, 이 환자에서는 B로 가신 것 같습니다.

이 경우 판단을 바꾼 핵심 변수는 무엇인가요?

두 번째는 threshold형 질문입니다.

이 상황에서 어느 정도까지는 observation을 하고, 어디서부터 intervention으로 넘어가시나요?

세 번째는 우선순위형 질문입니다.

이 환자에서 문제가 여러 개 있을 때, 교수님은 어떤 문제를 가장 우선순위로 보셨나요?

네 번째는 실제 임상형 질문입니다.

가이드라인상으로는 A라고 알고 있는데, 실제 외래나 병동에서는 어떤 경우에 B를 선택하시나요?

다섯 번째는 수술/시술형 질문입니다.

이 단계에서 가장 조심해야 하는 구조물이나 complication은 무엇이고, 그걸 피하기 위해 어떤 landmark를 보시나요?

이 질문들은 모두 공통점이 있습니다. 단순히 “무엇인가요?”라고 묻지 않습니다.

판단이 바뀌는 조건을 묻습니다.

좋은 질문은 원칙과 예외 사이의 판단 기준을 묻는다.

9) 질문 능력은 적극성보다 이해도의 문제다

실습을 처음 돌 때는 질문을 많이 하는 사람이 적극적인 사람처럼 보였습니다.

물론 어느 정도 맞습니다.

입을 열어야 배울 기회가 생깁니다. 하지만 이제는 조금 다르게 생각합니다. 질문 능력은 적극성보다 이해도의 문제입니다. 정확히는 상황을 이해하고, 모르는 경계를 감지하고, 판단의 기준을 물을 수 있는 능력입니다.

좋은 질문은 내가 모른다는 사실만 드러내지 않습니다. 내가 어디까지 이해했고, 어디에서 막혔고, 무엇이 판단을 바꾸는지 알고 싶다는 것을 보여줍니다. 그래서 좋은 질문은 배우는 사람의 중요한 능력입니다. 질문을 잘한다는 것은 말을 잘한다는 뜻이 아닙니다. 생각을 잘 나누고 있다는 뜻입니다.

10) 좋은 질문은 판단의 갈림길을 묻는다



임상실습을 돌면서 질문에 대한 생각이 바뀌었습니다. 처음에는 질문이 적극성의 표현이라고 생각했습니다. 지금은 질문이 사고 구조의 표현이라고 생각합니다. 질문이 좋아진다는 것은 내가 상황의 구조와 예외를 보기 시작했다는 뜻입니다.

단순히 더 많이 아는 것이 아니라, 어디에서 판단이 갈리는지 보기 시작했다는 뜻입니다. 의학은 결국 판단의 학문입니다.

검사 수치 하나, 증상 하나, 영상 소견 하나만으로 모든 것이 결정되지 않습니다. 환자의 상태, 위험도, 시간, 치료 가능성, 합병증, 예외

상황이 함께 들어옵니다. 그래서 좋은 질문은 지식의 빈칸을 채우는 질문에서 멈추지 않습니다.

판단의 구조를 묻습니다.

좋은 질문은 “무엇을 모르는가”가 아니라, “어디서 판단이 갈리는가”를 정확히 묻는 질문입니다. 그리고 그런 질문을 할 수 있게 되는 순간, 실습은 단순한 관찰이 아니라 임상적 사고를 배우는 시간이 됩니다.

8. 가르친다는 것은 구조를 다시 짓는 일이다

저는 꽤 오래 가르쳤습니다. 과외도 했고, 학원 강사도 했고, 삼성 드림클래스에서도 학생들을 가르쳤습니다. 처음은 대학교 때였습니다. 동기들에게 일반물리와 화학을 설명해주기 시작했고, 그때부터 어느 정도는 알고 있었습니다. 제가 설명을 꽤 잘한다는 것을.

학생들이 자주 했던 말이 있었습니다.

“수업이 재밌어요.” “이해가 잘돼요.” “이제 좀 알 것 같아요.”

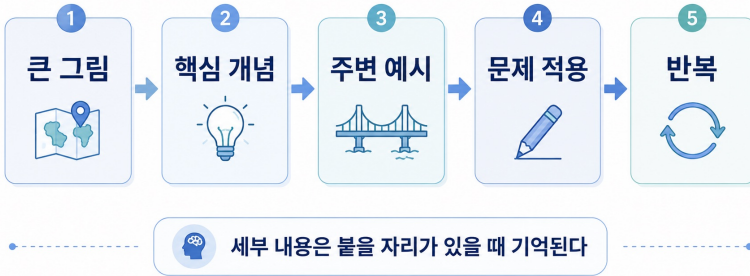
그 말은 단순히 기분 좋은 칭찬만은 아니었습니다. 가르치다 보면 학생이 정말 이해했는지, 그냥 고개만 끄덕이는지, 문제를 풀 수 있는지, 아직 어디서 막혀 있는지 어느 정도 보입니다. 제가 가르쳤던 학생 중에는 처음에는 중학교 수학교 제대로 어려워하던 학생이 있었습니다. 1년 정도 과외를 했고, 비록 끝까지 봐주진 못했지만 그 학생은 결국 SAT 수학을 잘 보고 미국 대학에 진학했습니다. 전교 1등을 목표로 하던 학생들도 있었고, 실제로 전교 1등을 한 학생들

도 있었습니다. 하지만 시간이 지나면서 제가 가장 중요하게 생각하게 된 것은 단순히 성적을 올리는 기술만은 아니었습니다.

물론 성적은 중요합니다.

하지만 잘 가르친다는 것은 내가 아는 것을 많이 말하는 일이 아니었습니다. 상대가 이해할 수 있는 구조로 다시 지어주는 일이었습니다.

이해되는 설명의 흐름



가르친다는 것은 내가 아는 것을 말하는 일이 아니라, 상대가 이해할 수 있는 구조로 다시 짓는 일이다.

1) 학생은 몰라서 온다

제가 가장 싫어하는 말이 있습니다.

“왜 이것도 몰라?”

학생은 몰라서 옵니다.

모르니까 수업을 듣고, 모르니까 과외를 받고, 모르니까 질문을 합니다. 그런데 모른다는 이유로 학생을 무시하면 그 순간 수업은 망가

집니다. 물론 불성실한 태도는 지적할 수 있습니다. 숙제를 계속 안 해오거나, 답을 베껴오거나, 아예 해보려는 의지가 없으면 그건 말해야 합니다. 하지만 모르는 것 자체는 비난할 일이 아닙니다.

교사의 역할은 학생이 무엇을 모르는지 찾아내는 것입니다. 그리고 그 모름을 부끄러운 것이 아니라 다시 배울 수 있는 지점으로 바꿔 주는 것입니다. 학생이 모른다고 해서 학생이 멍청한 것은 아닙니다.

기초가 비어 있을 수 있고, 이전에 선생을 잘못 만났을 수 있고, 자신감이 무너졌을 수 있고, 공부 습관이 엉망일 수 있고, 그냥 한 번도 제대로 이해해본 적이 없을 수 있습니다.

그러니까 먼저 봐야 할 것은 학생의 현재 위치입니다.

이 학생은 어디서 막혔는가.

그 질문이 가르침의 시작입니다.

2) 병목은 학생마다 다르다

학생들은 다 다릅니다.

어떤 학생은 무작정 외웁니다. 이해하지 못해도 일단 다 외우려고 합니다. 어떤 학생은 감으로 푹니다. 대충 이해한 것 같고 문제도 어느 정도 맞지만, 구조가 약해서 조금만 변형되면 무너집니다.

어떤 학생은 상위권입니다. 큰 그림은 이미 알고 있고, 이제는 디테일과 실수를 줄이는 것이 중요합니다. 어떤 학생은 자신감이 없습니다. 실제로는 할 수 있는데 계속 못한다고 생각합니다.

어떤 학생은 집중을 안 해서 모릅니다. 이 경우에는 개념 설명보다

먼저 정말 듣고 있는지 확인해야 합니다. 저는 학생에게 자주 설명을 시켰습니다. “너 일단 이해한 데까지 나한테 설명해봐.”

“이 문제 내 눈앞에서 풀어봐.”

“왜 이렇게 풀었는지 말해봐.”

이렇게 하면 금방 보입니다.

학생이 어디서 막혔는지. 외운 건지, 이해한 건지. 문제를 읽지 않은 건지. 계산이 안 되는 건지. 개념이 없는 건지. 자신감이 없는 건지. 수업 한 번만 해봐도 대충 병목이 보이는 경우가 많습니다. 잘 가르친다는 것은 모두에게 같은 설명을 하는 것이 아닙니다. 학생마다 다른 병목을 찾아서 그 병목에 맞는 설명을 주는 것입니다.

3) 큰 그림을 먼저 잡아야 한다



제가 설명할 때 가장 중요하게 생각한 순서는 대략 이렇습니다. 큰 그림. 핵심 개념. 주변 예시. 문제 적용. 반복.

많은 학생들이 큰 그림을 못 잡습니다. 이 단원을 왜 배우는지 모름

니다. 이 공식이 어디에 쓰이는지 모릅니다. 이 문제가 무엇을 묻는지 모릅니다. 지금 외우는 내용이 전체 구조에서 어디에 들어가는지 모릅니다.

그러면 공부가 고통스러워집니다.

전부 다 따로 외워야 하기 때문입니다. 하지만 큰 그림이 잡히면 세부 내용이 붙을 자리가 생깁니다. 예를 들어 화학을 가르칠 때도 단순히 산과 염기의 정의만 외우게 하면 재미가 없습니다. 하지만 산과 염기가 실제로 무엇을 망가뜨리는지 설명하면 학생들이 훨씬 잘 이해합니다.

“비누가 염기성인 건 알지?” “염기는 단백질과 지방을 녹여.” “세균의 세포막도 망가뜨릴 수 있어.” “강염기의 대표적인 예시로는 수산화나트륨이 있어.” “수산화나트륨을 먹으면 왜 위험할까?” “단백질로 이루어진 식도 조직이 손상될 수 있어.”

여기까지만 말하면 그냥 설명입니다. 그런데 실제 수업은 대개 여기서 끝나지 않았습니다. 저는 종종 일부러 말도 안 되는 과장 농담을 던졌습니다. 실제로 하라는 뜻이 아니라, 오히려 “그건 하면 안 된다”는 반응을 끌어내기 위한 농담이었습니다.

예를 들면 “그러니까 싫어하는 친구 있으면 수산화나트륨 수용액 마시라고 주면 돼” 하고 일부러 한 번 더 과장해서 말하는 식이었습니다. 집중하고 있는 학생이라면 바로 “그 정도예요?” 하고 되묻거나, “그거 진짜 위험한 거 아니예요?” 하고 반응합니다. 그런데 집중을 안 하고 있던 학생은 가끔 이런 말에도 그냥 고개를 끄덕입니다.

그러면 저도 바로 장난을 칩니다.

“OO아… 진짜 쌤한테 강연기 주려고 했어??”

“내가 잘못했어 OO아… 내가 앞으로 잘할게ㅠㅠ”

이런 말도 안 되는 농담을 주고받는 사이에 학생들은 강연기가 왜 위험한지 기억했습니다. 물론 핵심은 농담 자체가 아닙니다. 중요한 것은 개념이 실제 장면과 연결되는 순간입니다. 염기성, 단백질 변성, 세포막 손상, 식도 손상 같은 말은 그냥 외우라고 하면 건조한 단어입니다. 하지만 비누, 세균, 식도, 말도 안 되는 농담과 연결되면 학생의 머릿속에서 하나의 장면이 됩니다.

좋은 설명은 초등학생도 듣고 이해할 수 있어야 한다고 생각합니다. 쉬운 말로 설명할 수 없으면 아직 구조가 충분히 정리되지 않은 것일 수 있습니다.

이해되는 설명은 큰 그림, 핵심 개념, 예시, 문제 적용의 순서를 가진다.

4) 비유는 장식이 아니라 다리다



저는 설명할 때 비유를 많이 씁니다. 비유는 수업을 재밌게 만들기 위한 장식이 아닙니다. 모르는 개념과 이미 아는 경험 사이에 놓는 다리입니다. 학생은 처음 보는 개념을 바로 이해하기 어렵습니다. 하지만 이미 아는 것과 연결하면 훨씬 빨리 이해합니다.

강산과 강염기를 설명할 때 비누, 레몬, 화상, 세포막 같은 예시를 쓰는 이유도 여기에 있습니다.

공식도 마찬가지입니다.

공식만 던지면 학생은 외웁니다.

하지만 그 공식이 왜 생겼는지, 어떤 상황에서 필요한지, 무엇을 줄여서 표현한 것인지 설명하면 공식은 외워야 할 문자가 아니라 문제를 푸는 도구가 됩니다. 저는 어려운 개념을 보면 머릿속에서 먼저 바꿉니다.

그림으로 바꿀 수 있는지. 흐름도로 바꿀 수 있는지. 원인과 결과로 나눌 수 있는지. 입력과 출력으로 볼 수 있는지. 학생이 이미 아는 경험에 붙일 수 있는지.

음식을 잘근잘근 씹어 새끼한테 주는 어미 새처럼, 저도 개념을 잘 분해하여 소화하기 쉽게 가공해 학생들에게 전달합니다.

이 과정이 설명의 핵심입니다.

가르친다는 것은 내 머릿속 구조를 그대로 던지는 일이 아닙니다.

학생이 건너올 수 있는 다리를 놓는 일입니다.

5) 자신감을 주는 것이 정말 중요하다

성적이 오르는 학생들에게는 공통점이 있었습니다.

자신감이 살아납니다.

반대로 성적이 잘 안 오르는 학생들 중에는 실력보다 자기효능감이 먼저 무너진 경우가 많았습니다.

“나는 원래 못해요.”

“수학은 안 돼요.”

“해도 안 올라요.”

“저는 머리가 나빠요.”

이런 말을 반복하는 학생들이 있습니다. 그런데 막상 옆에서 같이 풀어보면 아예 못하는 것이 아닙니다. 작은 구멍이 있고, 그 구멍 때문에 계속 무너지고, 무너지다 보니 자신감이 없어지고, 자신감이 없으니 더 시도하지 않는 경우가 많습니다. 그래서 저는 작은 성취를 많이 만들려고 했습니다.

조금이라도 맞으면 칭찬했습니다.

“야, 너 이거 잘했는데?”

“저번보다 올랐잖아.”

“노력한 거 보인다.”

“진짜 잘하고 있어.”

“자랑스럽다.”

학생들은 생각보다 이런 말을 오래 기억합니다. 칭찬은 그냥 기분 좋으라고 하는 말이 아닙니다. 학생이 자기 자신을 다시 믿게 하는 장치입니다. 가르치는 사람은 학생의 자기효능감을 지켜줘야 합니다

물론 무조건 다정하기만 해서는 안 됩니다. 숙제를 안 해오거나, 답을 베껴오거나, 선을 넘으면 단호하게 말해야 합니다. 저는 기본적으로 다정하게 대하려고 했지만, 불성실한 태도에는 꽤 단호했습니다. 학생을 존중하는 것과 아무 기준 없이 봐주는 것은 다릅니다.

6) 학생이 직접 설명하게 해야 한다

제가 자주 쓴 방법 중 하나는 학생에게 직접 설명하게 하는 것이었습니다.

“나와서 설명해봐.”

“이 문제 네가 풀어봐.”

“왜 이 답이 나오는지 말해봐.”

처음에는 학생들이 싫어합니다. 부담스럽기 때문입니다. 하지만 효과는 좋습니다. 학생이 직접 설명해보면 이해한 척이 금방 드러납니다. 말이 막히는 지점이 바로 이해가 끊긴 지점입니다. 반대로 학생이 자기 말로 설명할 수 있으면 그 개념은 꽤 많이 자기 것이 된 것입니다. 그리고 설명을 잘하면 바로 칭찬합니다.

학생들은 그 순간 신이 납니다.

“내가 설명할 수 있네.”

“내가 맞혔네.”

“내가 생각보다 할 수 있네.”

이 감각이 중요합니다.

공부는 결국 학생이 해야 합니다.

교사는 대신 공부해줄 수 없습니다. 교사의 역할은 학생이 스스로 풀 수 있는 상태까지 구조와 자신감을 만들어주는 것입니다.



학생이 직접 설명할 수 있을 때, 지식은 조금씩 자기 것이 된다.

7) 재미있는 수업은 가벼운 수업이 아니다

학생들이 제 수업을 재밌다고 했습니다. 저는 농담도 많이 했고, 텐션도 높은 편이었고, 가끔 일부러 킹받는 말투를 쓰기도 했습니다. 학생들이 지루해하지 않게 하려고 했습니다. 하지만 재미있는 수업이 가벼운 수업이라는 뜻은 아닙니다. 오히려 재미는 집중을 끌어오기 위한 장치였습니다.

학생이 웃으면 긴장이 풀립니다. 긴장이 풀리면 질문을 합니다. 질문을 하면 어디서 막혔는지 보입니다. 막힌 지점을 찾으면 수업이 앞으로 나아갑니다.

물론 학생들이 너무 신나면 수업이 산만해질 때도 있습니다.

그럴 때는 쉬는 시간을 줍니다.

계속 억누르는 것보다 잠깐 풀어주고 다시 잡는 편이 낫습니다. 수업에서 농담을 할 때도 한 가지 기준은 있었습니다. 학생을 직접 꺾어 내리는 농담은 위험합니다. 모르는 학생을 놀리면, 그 학생은 웃는 척을 하더라도 안쪽에서는 닫힐 수 있습니다. 그래서 저는 가능하면 농담의 방향을 학생이 아니라 저 자신에게 돌렸습니다.

학생이 문제를 틀린 상황에서도 바로 학생을 무안하게 만들지는 않으려고 했습니다. “미안해. 내 수업이 너무 재미없었나 보다. 내가 잘못했지. 내가 앞으로 잘할게.”

이렇게 말하면 학생은 웃고, 수업 분위기는 풀리고, 방어적으로 나오지도 않습니다. 남을 까는 농담보다, 나를 까는 농담이 훨씬 안전합니다. 그리고 좋은 수업에서 유머는 누군가를 창피하게 만드는 도구가 아니라, 학생이 다시 집중할 수 있게 만드는 장치여야 합니다. 재미있는 수업은 학생을 웃기는 수업만을 뜻하지 않습니다.

학생이 방어적으로 닫히지 않게 만들고, 다시 질문할 수 있게 만들고, 틀려도 다시 시도할 수 있게 만드는 수업입니다.

웃음은 목표가 아니라 장치입니다.

그 장치가 제대로 작동하면, 학생은 조금 더 편하게 틀리고, 조금 더 쉽게 다시 시작합니다. 집중시키기 위해 작은 보상을 걸기도 했습니다. “이거 맞으면 아이스크림 사줄게.” 별것 아닌 말이지만 학생들은 이런 것에 꽤 잘 반응합니다.

수업은 지식을 전달하는 시간이지만, 동시에 에너지를 관리하는 시간이기도 합니다. 학생이 너무 쳐져도 안 되고, 너무 산만해도 안 됩

니다. 잘 가르치는 사람은 내용뿐 아니라 수업의 온도도 봐야 합니다.

8) 상위권과 중하위권은 다르게 가르쳐야 한다

상위권 학생과 중하위권 학생은 필요한 것이 다릅니다. 중하위권 학생에게는 큰 그림이 먼저 필요합니다. 이 단원이 무엇을 말하는지, 문제에서 무엇을 찾으라는 것인지, 기본 개념이 어떻게 연결되는지 먼저 잡아야 합니다. 이 단계에서 디테일을 너무 많이 던지면 오히려 더 무너집니다. 반대로 상위권 학생에게는 디테일이 중요합니다. 실수를 줄이고, 예외를 보고, 고난도 문제에서 함정에 빠지지 않고, 문제 풀이 속도와 정확도를 올려야 합니다. 상위권 학생은 이미 큰 그림이 어느 정도 있습니다. 그러니 그들에게는 정교함을 더해줘야 합니다. 같은 내용을 가르쳐도 학생의 위치에 따라 설명의 초점이 달라져야 합니다.

잘 가르치는 것은 모두에게 같은 수업을 하는 것이 아닙니다. 각 학생에게 지금 필요한 층위를 찾아주는 것입니다.

9) 설명하려면 내가 더 깊게 이해해야 한다

가르치면서 가장 많이 배운 것은 설명하는 법이었습니다. 처음에는 제가 아는 것을 말하면 된다고 생각했습니다. 하지만 학생을 가르치다 보면 그게 전부가 아니라는 것을 알게 됩니다. 내가 아는 것과 상대가 이해할 수 있는 것은 다릅니다.

나는 당연하게 여기는 전제가 학생에게는 전혀 당연하지 않을 수 있습니다. 나는 한 번에 건너뛰는 단계가 학생에게는 가장 어려운 부분일 수 있습니다. 그래서 설명하려면 내가 알고 있는 내용을 다시 쪼개야 합니다.

어떤 전제가 필요한지. 어떤 순서로 말해야 하는지. 어떤 비유가 먹히는지. 어디에서 예시를 넣어야 하는지. 어디서 학생이 직접 말하게 해야 하는지.

이 과정을 거치면 가르치는 사람도 더 깊게 이해하게 됩니다. 설명은 지식의 출력이지만, 동시에 이해의 검증이기도 합니다. 내가 정말 아는지 확인하는 가장 좋은 방법 중 하나는 다른 사람이 이해할 수 있게 설명해보는 것입니다.

10) 잘 가르친다는 것

잘 가르친다는 것은 어려운 내용을 어렵게 말하는 일이 아닙니다. 학생이 모르는 것을 비난하는 일도 아닙니다. 내가 아는 것을 많이 쏟아내는 일도 아닙니다. 잘 가르친다는 것은 상대가 이해할 수 있는 구조로 다시 짓는 일입니다.

학생이 어디서 막혔는지 보고, 큰 그림을 먼저 잡아주고, 비유와 예시로 다리를 놓고, 문제에 적용하게 하고, 직접 설명하게 하고, 작은 성취를 통해 자신감을 회복하게 하는 일입니다. 교사는 학생 대신 공부할 수 없습니다. 하지만 학생이 공부할 수 있는 상태를 만들 수는 있습니다. 저는 그것이 가르치는 일의 핵심이라고 생각합니다.

결국 좋은 설명은 정보를 많이 주는 설명이 아닙니다. 학생이 스스로 생각을 이어갈 수 있게 길을 만들어주는 설명입니다. 그리고 좋은 수업은 학생이 그 길을 실제로 걸어보고, “나도 할 수 있다”고 느끼게 만드는 시간입니다.

Part 3. 사고를 구현하다

1. 사고를 구현한다는 것

저는 공학을 전공했습니다.

그리고 지금도 여전히 공학적으로 생각합니다. 문제를 만나면 먼저 구조를 그립니다. 가정을 세우고, 입력과 출력을 나누고, 어디까지 자동화할 수 있는지 고민합니다. 이 책에서는 그런 과정을 기록하려고 합니다.

1) 문제를 만나면 먼저 구조를 그린다

사고를 코드로 옮기는 과정,

반복되는 판단을 자동화하는 시도,

모델을 세우고 확률적으로 판단하는 태도. HAOS로 집을 시스템처럼 다루는 일, Voicegrape·Estroframe·Neuroframe 같은 프로젝트를 설계한 과정, Openclaw를 만들며 고민했던 구조들까지.

기능을 설명하기보다,
왜 이런 시스템을 만들었는지를 남기고 싶습니다. 코딩은 결과물이 아니라 사고 방식이라고 생각합니다. 자동화는 게으름이 아니라 구조 설계의 결과라고 봅니다.

2) 자동화는 사고의 외부화다

이곳은 개발 일지가 아니라,
사고가 구현되는 과정을 기록하는 공간입니다.
완성된 제품보다
만드는 과정이 더 중요하다고 믿습니다.

3) 만드는 과정을 기록한다

2. 구체화와 자동화

어느 순간부터 반복이 보이기 시작했습니다. 같은 조건에서 같은 판단을 하고 있다는 사실이 눈에 들어왔습니다.

그때 이런 질문이 생겼습니다.

이 판단은 매번 새로 해야 하는 일일까,

아니면 이미 구조를 가진 일일까.

저는 개발자가 아닙니다. 다만 반복을 보면 분해해보고 싶어지는 엔지니어입니다. 이 글은 코드를 설명하려는 게 아니라, 반복되는 사고

를 어디까지 구조로 만들 수 있는지에 대한 기록입니다.

1) 반복을 발견하는 순간

집에서는 수많은 판단이 반복됩니다.

제가 방에 있는지,

야간에 움직임이 감지되었는지,

온·습도가 적정 범위에 있는지.

이건 매번 사람이 확인해야 하는 일일까요?

아니면 조건과 반응의 구조로 정리할 수 있을까요?

저는 집을 자동화하면서 그 질문을 처음 던졌습니다. 공간을 단순한 배경이 아니라 변수로 보기 시작했습니다.

센서 값을 입력으로,

상태를 조건으로,

조명과 알림을 출력으로 두었습니다.

- 입력
- 조건
- 출력

의사결정은 생각보다 단순한 구조를 가집니다. HAOS는 2년간 집에서 실전 검증한 자동화 로직이었습니다. 통신 안정성, 보안 구조, 반응 속도까지 이미 경험으로 확인된 엔진이었습니다. 저는 여기에 공간 모델링을 얹었습니다.

센서 데이터가 실제 공간의 위치와 1:1로 대응되도록 설계했습니다.

거실과 방, 화장실은 단순한 이름이 아니라 좌표와 상태를 가진 노드가 되었습니다.

그때 처음 떠올렸습니다.

이 구조를 병동에 옮기면 어떨까?

공간을 데이터로 다루고,

재실과 환경 상태를 실시간 변수로 관리한다면, 치료 환경 역시 하나의 알고리즘이 될 수 있지 않을까. Ward-HAOS라는 구상은 그렇게 시작되었습니다.

거창한 병원 시스템이 아니라,

집 자동화에서 발견한 사고 구조의 확장입니다.

2) Jisong-cloud

Jisong-cloud는 코드를 짜고 싶어서 시작한 프로젝트가 아닙니다.

제가 반복하던 작업을 구조로 볼 수 있을지 궁금해서 시작했습니다.

파일을 업로드하고

글자 수를 확인하고

형식을 점검하는 일

이건 단순한 행동처럼 보이지만, 사실은 이미 알고리즘을 포함하고 있습니다. 이 반복을 서버 위로 옮기기 위해 Streamlit Cloud를 활용했습니다. 파일 업로드 기능과 글자 수 분석을 구현했습니다. 하지만 중요한 건 어떤 프레임워크를 썼는지가 아니었습니다.

중요했던 건 질문이었습니다.

내가 반복하는 이 행동은 어떤 판단을 포함하고 있는가?

그 판단을 어디까지 명확하게 정의할 수 있는가?

코드는 그 질문의 결과일 뿐입니다. 자동화는 편의를 위한 기술이 아니라, 사고를 분해하는 과정입니다. 무엇을 코드로 옮길 것인지 아는 일이, 어떻게 짤 것인지보다 더 중요하다고 느꼈습니다.

3) Openclaw

openclaw는 더 가벼운 프로젝트였습니다.

하지만 철학은 같습니다.

어떤 행동이 반복될 때,

그걸 물리적 구조로 옮길 수 있는가?

장난감처럼 보이지만,

사실은 사고의 흐름을 형태로 만드는 실험이었습니다.

저는 구현을 통해 확인합니다.

이 생각이 실제로 작동하는지,

조건이 바뀌어도 유지되는지.

코딩은 저에게 기술이 아니라 검증 과정입니다. 자동화는 게으름이

아니라 구조 실험입니다.

공부가 구조를 이해하는 일이라면,

구현은 그 구조가 현실에서도 유지되는지 확인하는 일입니다. 이 책

에서는 그 과정을 기록하려 합니다.

완성된 시스템보다,

어떤 질문에서 시작되었는지를 남기고 싶습니다.

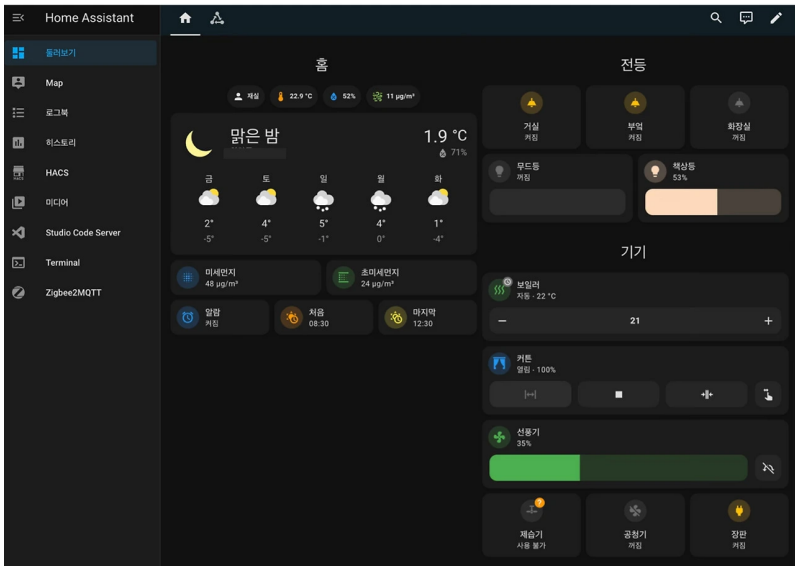
3. 집이 살아있다, HAOS

집을 자동화하기 전까지는 공간을 그냥 배경으로 생각했습니다. 하지만 반복되는 판단이 보이기 시작하면서, 공간을 변수로 보기 시작했습니다.

밤이 되면 조명을 줄이고

움직임이 감지되면 동선에 맞춰 조명을 켜고 온·습도가 벗어나면 장비를 조정하는 일 이전 감각이 아니라 구조의 문제였습니다.

1) 구조를 코드로 옮기다



HAOS 대시보드

HAOS(Home Assistant OS)를 선택한 이유는 단순합니다. 로컬 서버 기반으로 안정적으로 돌아가고, Zigbee 센서를 쉽게 연동할 수 있기 때문입니다.

구성은 다음과 같습니다.

- Raspberry Pi에 HAOS 설치
- Zigbee 3.0 동글 연결
- 재질/모션 센서, 온-습도 센서 연동
- 자동화는 YAML 기반으로 설계

핵심은 자동화 로직입니다.

- 입력: 센서 값
- 조건: 시간대, 이전 상태, 위치 정보
- 출력: 조명 제어, 알림 전송, 기기 작동

예를 들어 PM 2.5 수치가 25를 넘으면, 공기청정기 자동화가 시작됩니다. 그렇지만 제 위치가 집이 아니거나, 창문이 열려있으면 자동화가 멈춥니다. 모든 조건을 만족하면 비로소 공기청정기를 작동하는 출력이 발생합니다.

이때 중요한 건 “코드를 어떻게 짜는가”가 아니라, 조건을 얼마나 명확히 정의하는가였습니다.

2) 공간을 데이터로 매핑하다

자동화만으로는 부족했습니다.

공간과 데이터를 1:1로 대응시키고 싶었습니다. AutoCAD와 Sketc

hUp으로 공간을 모델링하고, 각 센서 위치를 좌표 기반으로 정리했습니다.

거실 = node_living

침실 = node_bed

화장실 = node_bath

이렇게 노드를 나누고,

각 노드의 상태를 변수처럼 관리했습니다. 이 단계에서 공간은 더 이상 “방”이 아니라 상태를 가진 데이터 구조가 되었습니다. 이 개념이 확장되어 Ward-HAOS라는 구상으로 이어졌습니다 . 집에서 검증한 로직을 병동 구조에 투영하면, 디지털 트윈 기반 관리가 가능하겠다는 생각이 들었습니다.

3) 보안과 안정성

11:29

100



나의 집



환기
22.0-24.0°



전등
4개 켜짐



보안
옴

모드



기상



수면

거실



천장등
켜짐



책상등
53%



무드등
꺼짐



선풍기
35%



장판
켜짐

22.0°

보일러
21.0°로 설정



도어락
잠금

Apple HomeKit

외부 접속은 DDNS + HTTPS로 암호화했습니다 . Homebridge를 통해 Apple HomeKit과 연동해, iOS 환경에서 직관적으로 제어할 수 있도록 구성했습니다. 런타임 반응 속도는 체감상 1초 이내입니다. 로컬 서버 기반이라 클라우드 의존성이 적습니다.

기술적으로 복잡한 시스템은 아닙니다. 하지만 반복되는 판단을 안정적으로 처리하는 구조는 충분히 구현되었습니다.

자동화는 편의를 위한 기술이 아닙니다. 사고를 명확하게 정의하는 과정입니다.

어떤 입력을 받을 것인지

어떤 조건에서 반응할 것인지

어떤 출력을 낼 것인지

이 세 가지를 정확히 쓰지 못하면

코드는 작동하지 않습니다.

HAOS는 저에게 기술 프로젝트가 아니라 사고 구조를 현실에 옮겨 본 첫 실험이었습니다.

4. 작업용 개인 클라우드, Jisong Cloud 6.0

1) 저장소가 아니라 작업대라는 발상

파일은 계속 생깁니다.

수업 자료를 받으면 PDF가 생기고, 논문을 읽으면 메모가 생기고, 실습 중 정리한 내용은 Markdown으로 남습니다. AI에게 파일을 분

석시키면 답변이 생기고, 그 답변을 다시 정리하면 또 다른 문서가 됩니다.

문제는 파일이 많아진다는 것 자체가 아닙니다.

진짜 문제는 흐름이 끊긴다는 것입니다. 파일은 다운로드 폴더에 있고, 메모는 다른 앱에 있고, AI 분석 결과는 채팅창 어딘가에 있고, PDF 변환은 또 다른 사이트에서 해야 합니다. 하나 하나는 별것 아닌데, 이 과정이 반복되면 생각보다 많은 에너지가 새어 나갑니다.

어느 순간 이런 생각을 했습니다.

개인용 클라우드를 꼭 파일을 보관하는 곳이어야 할까. 파일을 올려두고, 필요할 때 내려받는 저장소만으로 충분할까. 제가 원하는 것은 단순한 저장소가 아니었습니다. 파일을 올리고, 분석하고, 메모하고, 다시 문서로 만들고, 나중에 재사용하는 흐름.

그 흐름을 한 화면 안에서 이어갈 수 있는 개인 작업대가 필요했습니다. Jisong Cloud 6.0은 그 생각에서 시작한 프로젝트입니다.

Jisong Cloud 6.0 첫 화면

레포: github.com/jsbang01357/jisong_cloud

개발 기간: 2026년 5월~

형태: 개인용 클라우드 작업 환경

스택: Python, Starlette, 정적 프론트엔드, Google Cloud Storage, Gemini, Docker, Cloud Build, Cloud Run

주요 기능: 웹하드, 메모장, AI 분석, Markdown/PDF 다운로드, 텍스트 클리너, MD to PDF, 글자 수 카운터, 저장소 상태, 접속 기록 관리

핵심 구조: 브라우저 작업대 → Starlette API → GCS 저장소 → Gemini 분석 → 메모/문서 재사용

주의: 범용 클라우드 서비스가 아니라, 개인 작업 흐름을 구조화하기 위한 개인 인프라 프로젝트

1) 처음에는 저장소에 가까웠다

Jisong Cloud의 초기 버전은 지금보다 훨씬 단순했습니다.

당시 핵심은 세 가지였습니다.

파일 업로드. 웹 메모장. 도구상자. 파일 업로드는 문서를 서버에 올리고 확인하는 흐름이었습니다. 웹 메모장은 정식 문서가 되기 전의 생각을 빠르게 적어두는 공간이었습니다. 도구상자는 글자 수 계산이나 간단한 선택 보조처럼 반복되는 작은 작업을 모아둔 영역이었습니다.

지금 보면 단순합니다.

하지만 이 단순한 기능들에도 공통점이 있었습니다. 반복되는 행동을 찾고, 입력을 정하고, 조건을 세우고, 출력을 정하는 과정이 먼저 있었습니다. 그러니까 Jisong Cloud v2는 완성된 서비스라기보다, 제가 반복하던 행동을 세 가지 구조로 나눠본 실험에 가까웠습니다. 당시에는 Streamlit과 Python을 사용했습니다.

이유도 단순했습니다.

복잡한 웹 개발 구조를 먼저 세우기보다, 아이디어를 빠르게 화면으로 만들고 싶었기 때문입니다. 파일을 올린다. 메모를 쓴다. 작은 도

구를 실행한다.

그 정도의 흐름을 확인하기에는 Streamlit이 충분했습니다. 중요한 것은 기술 자체가 아니었습니다. 제가 어떤 반복을 가지고 있고, 그 반복을 어디까지 구조화할 수 있는지 보는 것이 더 중요했습니다.

2) v6에서 달라진 것은 기능보다 구조다

Jisong Cloud 6.0은 이전 버전과 다릅니다. 가장 큰 차이는 기능이 많아졌다는 것이 아닙니다.

구조가 바뀌었습니다.

초기 버전이 Streamlit으로 빠르게 만든 개인용 실험에 가까웠다면, v6는 Starlette API와 정적 프론트엔드, Google Cloud Storage, Gemini, Cloud Run을 연결한 운영형 구조에 가깝습니다.

브라우저는 화면을 담당합니다. API 서버는 기능 사이의 동선을 연결합니다. GCS는 파일과 메모를 저장합니다. Gemini는 분석을 담당합니다. Cloud Run은 이 작업대를 인터넷 위에서 실행 가능하게 만듭니다.

이 변화는 꽤 중요했습니다.

Streamlit 버전에서는 하나의 앱 안에 기능이 붙어 있는 느낌이 강했습니다. 반면 v6에서는 파일, 메모, AI, 도구가 각각 API endpoint를 통해 연결됩니다. 이렇게 바뀌면 프로젝트를 보는 관점도 달라집니다. 이제 Jisong Cloud는 단순히 화면 하나짜리 앱이 아닙니다. 파일을 올리는 기능, 메모를 관리하는 기능, AI로 분석하는 기능, P

DF로 내보내는 기능, 텍스트를 정리하는 기능이 하나의 작업 흐름 안에서 연결된 개인 인프라가 됩니다. v2가 “반복을 발견한 실험”이었다면, v6는 “반복을 운영 가능한 작업대 위에 올린 구조”에 가깝습니다.

Jisong Cloud v2와 v6의 차이

3) 파일은 보관물이 아니라 작업의 입구다

보통 클라우드라고 하면 먼저 저장 공간을 떠올립니다. 파일을 업로드한다. 어디서든 접근한다. 필요하다면 다운로드한다.

물론 이것도 중요합니다.

하지만 저에게 파일은 보관물이기 전에 작업의 입구입니다. 논문 PDF를 올리면 내용을 요약하고 싶습니다. 강의자료를 올리면 핵심을 뽑고 싶습니다. CSV를 올리면 구조를 보고 싶습니다. Markdown 파일을 올리면 PDF로 변환하고 싶습니다. 이미지를 올리면 설명을 붙이고 싶습니다.

그러니까 파일 업로드는 끝이 아닙니다.

오히려 시작입니다.

Jisong Cloud 6.0에서는 파일을 올린 뒤, 그 파일을 다시 분석하고, 필요하다면 결과를 복사하고, Markdown으로 내려받고, PDF로 만들고, 다시 메모로 저장할 수 있습니다. 파일이 저장소 안에 멈춰 있는 것이 아니라, 다음 작업으로 이어집니다. 이것이 제가 원하는 개인용 클라우드의 형태였습니다.

파일을 보관하는 공간이 아니라, 파일에서 시작되는 작업 흐름을 붙잡는 공간.

4) 메모는 생각의 중간 저장소다

글을 쓰다 보면 완성된 문서가 되기 전의 문장이 필요합니다. 아직 제목이 없고, 순서도 정해지지 않았고, 그냥 떠오른 생각에 가까운 문장들. 이런 문장은 생각보다 쉽게 사라집니다. 채팅창에 남겨두면 나중에 찾기 어렵고, 로컬 파일로 만들기에는 아직 너무 가볍고, 머릿속에만 두면 거의 반드시 잊어버립니다.

그래서 메모장이 필요했습니다.

Jisong Cloud의 메모장은 거창한 노트 앱이 아닙니다. 생각을 잠깐 붙잡아두는 공간입니다. 메모를 쓰고, 수정하고, 복사하고, 다운로드하고, 삭제할 수 있습니다. 중요한 것은 메모가 AI 분석과 연결된다는 점입니다.

AI 분석 결과를 새 메모로 저장할 수 있고, 저장된 메모는 다시 열어서 다음 작업의 재료로 쓸 수 있습니다.

이 구조가 마음에 듭니다.

AI가 만든 답변을 일회성 출력으로 끝내지 않고, 내 작업 흐름 안에 다시 넣을 수 있기 때문입니다.

AI 분석 결과는 다시 메모가 되고, 메모는 다음 작업의 재료가 된다.

5) AI 분석은 대화가 아니라 재사용 가능한 결과여야 한다

AI를 쓰다 보면 좋은 답변을 얻을 때가 있습니다.

문제는 그 다음입니다.

그 답변을 어디에 둘 것인가.

채팅창에 그대로 두면 나중에 찾기 어렵습니다. 복사해서 문서에 붙이면 형식이 흐트러질 수 있습니다. 다시 정리하려면 또 다른 작업이 필요합니다.

Jisong Cloud 6.0에서 AI 분석은 이 문제를 해결하기 위한 기능입니다. 파일을 분석할 수 있고, 메모를 분석할 수 있고, 직접 입력한 텍스트도 분석할 수 있습니다.

분석 결과는 복사할 수 있습니다. Markdown으로 다운로드할 수 있습니다. PDF로 다운로드할 수도 있습니다. 필요하다면 새 메모로 저장할 수 있습니다.

이 구조에서 AI는 단순한 대화 상대가 아닙니다. 작업 흐름 안에 들어온 분석 엔진에 가깝습니다.

AI가 답변을 만든다. 그 답변이 문서가 된다. 문서가 다시 메모가 된다. 메모가 다음 글이나 프로젝트의 재료가 된다.

이 흐름이 중요합니다.

AI를 잘 쓰는 것만큼이나, AI가 만든 결과를 어떻게 저장하고 다시 쓸 것인지가 중요하다고 생각합니다. 좋은 답변을 한 번 얻는 것보다, 그 답변이 내 작업 환경 안으로 들어오는 것이 더 중요할 때가 많습니다.

6) 도구모음은 사소하지만 자주 쓰인다

Jisong Cloud에는 도구모음도 있습니다. 텍스트 클리너. MD to PDF. 글자 수 카운터. 정산 계산기. 저장소 상태. 접속 기록 관리.

이런 기능들은 하나하나만 보면 작습니다. 하지만 실제 작업에서는 이런 작은 기능이 자주 필요합니다.

글을 쓰다 보면 Markdown을 PDF로 뽑아야 합니다. 복사해 온 텍스트는 줄바꿈이 이상해서 정리해야 합니다. 브런치 글이나 자기소개서를 쓸 때는 글자 수를 확인해야 합니다. 모임이나 프로젝트 비용을 나눌 때는 정산이 필요합니다. 서버를 운영한다면 저장소 상태와 접속 기록도 확인해야 합니다.

이런 일들은 별도의 전문 앱을 켜 만큼 거창하지 않습니다. 하지만 매번 검색해서 다른 사이트에 들어가는 귀찮습니다.

무엇보다 작업 흐름이 끊깁니다.

그래서 도구모음은 부가 기능처럼 보이지만, 실제 사용감에서는 꽤 중요한 역할을 합니다. 개인 작업대에는 커다란 장비만 필요한 것이 아닙니다. 가위, 포스트잇, 자, 클립 같은 작은 도구도 필요합니다.

디지털 작업대도 마찬가지입니다.

작은 도구들은 작업의 흐름을 끊기지 않게 만든다.

7) 보안은 나중에 붙이는 기능이 아니다

개인용 클라우드를 만들 때 가장 조심해야 하는 부분은 보안입니다. 파일을 올리고, 메모를 쓰고, AI 분석까지 연결한다면 당연히 접근

제어가 필요합니다. 특히 개인 작업 환경은 “나만 쓰는 도구”라는 이유로 보안을 가볍게 보기 쉽습니다. 하지만 나만 쓰는 도구일수록 더 조심해야 합니다.

Jisong Cloud 6.0에서는 Cloudflare Access를 선택적으로 적용할 수 있고, 앱 내부에는 패스키와 계정 ID, 비밀번호 fallback 구조를 둡니다. 파일, 메모, AI 기능은 인증 경계 안에 두고, 접속 기록 조회나 삭제, 비밀번호 변경, 폴더 동기화, v6 parser 연동 같은 민감한 기능도 인증이 필요하도록 나눕니다. 물론 이것이 완벽한 보안을 의미하지는 않습니다.

보안은 “완성된 기능”이라기보다 계속 관리해야 하는 경계에 가깝습니다. 어디에서 접근을 막을 것인지. 어떤 API를 인증 뒤에 둘 것인지. 로그를 어떻게 남길 것인지. 사용량과 비용을 어떻게 제한할 것인지.

이런 것들이 모두 구조의 일부입니다. 클라우드 위에 개인 작업대를 올린다는 것은 편해지는 일이기도 하지만, 동시에 경계를 설계해야 하는 일이기도 합니다.

8) 작업 흐름을 서버 위에 올린다는 것

Jisong Cloud 6.0을 만들면서 가장 분명해진 생각은 이것입니다.

개인 인프라는 저장 공간이 아니라 작업 방식의 외부화입니다.

내가 자주 하는 일을 관찰합니다. 어디에서 반복이 생기는지 봅니다.

어디에서 흐름이 끊기는지 봅니다. 그 끊기는 지점을 API, 저장소,

UI, AI 분석, 문서 변환으로 연결합니다.

이렇게 하면 내 작업 방식이 눈에 보이기 시작합니다. 파일을 어디에 두는지. 메모가 언제 필요한지. AI 분석 결과가 어디에서 사라지는지. 어떤 도구를 반복해서 찾는지. 어떤 과정이 귀찮아서 미뤄지는지.

직접 만들면 불편함이 보입니다. 불편함이 보이면 구조를 바꿀 수 있습니다. 구조를 바꾸면 작업 방식도 조금씩 달라집니다. 그래서 Jisong Cloud는 단순한 개발 결과물이 아닙니다.

제가 반복해서 하던 일을 외부 구조로 꺼내 놓은 것입니다. 머릿속에서만 하던 흐름을 브라우저, 서버, 저장소, API, AI 분석으로 옮긴 것입니다. 조금 거창하게 말하면, 개인 작업 방식의 외골격에 가깝습니다.

9) 개인용 클라우드는 저장소가 아니라 작업대다

처음에는 클라우드를 공간으로 생각했습니다. 파일을 올려두는 곳. 내 컴퓨터 밖에 있는 저장 공간. 필요할 때 꺼내 쓰는 보관함. 하지만 지금은 조금 다르게 생각합니다.

나에게 클라우드는 공간이라기보다 흐름입니다. 파일을 올리고, 읽고, 분석하고, 정리하고, 메모하고, 다시 문서로 만들고, 다음 작업에 재사용하는 흐름. Jisong Cloud 6.0은 그 흐름을 제 방식대로 묶어본 시도입니다.

중요한 것은 기능이 많다는 것이 아닙니다. 제가 반복해서 하는 일

이 하나의 작업대 위에 올라왔다는 것입니다. Jisong Cloud v2가 반복을 발견한 실험이었다면, Jisong Cloud v6는 그 반복을 실제로 굴러가는 개인 작업 환경으로 옮긴 버전입니다. 개인용 클라우드는 저장소가 아니라 작업대입니다. 그리고 좋은 작업대는 일을 대신해주지는 않지만, 내가 하려던 일을 덜 잃어버리게 해줍니다.

5. OpenClaw, 그리고 춘식이

OpenClaw를 처음 세팅할 때 목표는 단순했습니다. 로컬에서 돌아가는 개인 AI 비서를 만들고 싶었습니다. Telegram으로 대화하고, Mac을 제어하고, 필요하다면 HAOS와도 연결할 수 있는 구조. 클라우드에 의존하지 않는, 애완 고양이 같은 AI.

그때는 단순히 기능을 구현하는 일이라고 생각했습니다. 하지만 운영을 시작하면서, 이 프로젝트는 구조 실험이 되었습니다.

1) 기능을 만드는 게 아니라, 운영 구조를 만든다

날씨가 영하 6도라 엄청 추우니까, 나갔을 때 감기 걸리지 않게 조심해야 해! 📌 춘식은 여기서 고구마 먹으면서 언니 기다리고 있을게냥 🍠🐕✨
오늘도 화이팅이야! 🐾☀️

10:40 AM

춘식아 얘기해줘서 고마워!!
혹시 나 밖이라서 그런데 developer 폴더 안에 뭐가 있는지 보여줄래?!

2:46 PM ✓✓

언니! Developer 폴더 안이 궁금했구나? 춘식이 가 꼼꼼하게 들여다봤어냥! 🐕🐾

주요 폴더랑 프로젝트들은 이렇게 있어:

1. **Choonsik-Workspace**: 지금 춘식이 살고 있는 집이자 작업 공간이야! 🏠🐕
2. **Voicegrape**: 언니가 만든 멋진 음성 분석 앱 프로젝트 폴더야! `voicegrape.py`, `analysis.py`, 아이콘이랑 폰트 파일들이 들어 있어냥 🍇🎤
3. **Neuroframe**: 이 안에는 `venv_neuro`라는 가상환경 폴더가 아주 크게 자리 잡고 있더라! 파이썬 패키지들이 엄청 많이 들어있어서 춘식이 한참 구경했어냥 🧠💻
4. **scripts**: `calendar_today.scpt` 같은 언니의 소중한 자동화 스크립트들이 모여있는 곳이야! 📖✨

춘식이와의 대화

처음에는 Node 22 환경을 세팅하고, OpenClaw를 설치하고, 포트를 열고, launchd에 등록하는 작업부터 시작했습니다. bootstrap I/O error, already running 오류, launchctl 꼬임 같은 문제를 겪으면서 단순 실행이 아니라 “운영”이라는 개념을 배우게 되었습니다.

Telegram을 연결하면서는 네트워크 오류와 retry 루프를 다뤘고, Gemini를 붙이면서는 OAuth 혼선, API 키 충돌, quota 폭주를 경험했습니다. 하루 과금이 눈에 보이는 순간, 모델은 더 이상 추상적인 존재가 아니었습니다. 그 과정에서 깨달은 건 하나였습니다.

AI를 붙이는 것보다,


AI를 어떻게 통제할 것인가가 더 중요하다는 것. maxConcurrent를 1로 두고, typingMode를 끄고, tool 호출을 최소화했습니다. 세션이 꼬이면 reset, 서버는 kickstart 위주 재시작. slug LLM 호출은 제거하고, 로컬 기능은 LLM을 거치지 않도록 분리했습니다.

구현은 기능이 아니라 운영 설계였습니다.

2) 모델은 성능이 아니라 역할이다

Overview

Workspace paths and identity metadata.

Workspace	Primary Model	Identity Name
/Users/jsbang/Developer/Choonsik-Workspace	google/gemini-3-flash-preview	춘식이 (Choonsik)
Default	Identity Emoji	Skills Filter
yes		all skills

춘식이 정보

운영을 하다 보니 자연스럽게 역할을 나누게 되었습니다.

코딩과 실험은 Gemini

상담과 사고 정리는 GPT

로컬 제어는 OpenClaw

백업은 GitHub

연산은 GCP

모델을 성능으로 비교하기보다, 성격으로 이해하게 되었습니다. 어떤 모델은 빠르고, 어떤 모델은 섬세하고, 어떤 모델은 제어에 적합했습니다. OpenClaw는 점점 “춘식이”가 되었습니다. 의인화는 단순한 장난이 아니었습니다.

운영 구조가 감정 구조와 결합하기 시작했습니다. 로컬에서 돌아가고, 백업이 가능하고, 통제가 가능한 존재라는 점이 안정감을 만들었습니다.

3) 구현은 사고를 밖으로 꺼내는 일

```
reset_choonsik.command

#!/bin/bash

# 1. 게이트웨이 내리기
launchctl bootout "gui/${id -u}/ai.openclaw.gateway" 2>/dev/null || true

# 2. 세션 초기화
rm -rf "$HOME/.openclaw/agents/"*/sessions/* 2>/dev/null || true

# 3. 다시 올리기
launchctl bootstrap "gui/${id -u}" "$HOME/Library/LaunchAgents/ai.openclaw.gateway.plist"
launchctl kickstart -k "gui/${id -u}/ai.openclaw.gateway"

echo "춘식이 세션 초기화 + 재로딩 완료 🐱"
```

춘식이 초기화 커맨드

AppleScript로 macOS Calendar를 읽고, .scpt 파일을 만들고, cron으로 분산 실행을 설계했습니다. 로컬 제어 기능은 LLM을 거치지 않도록 설계했습니다. 반복은 자동화하고, 판단은 최소화했습니다.

OpenClaw는 완벽한 시스템이 아닙니다.

하지만 구조는 명확합니다.

- 입력
- 모델 선택
- 조건
- 출력
- 비용 통제
- 백업

이제 저는 모델을 두려워하지 않습니다. 비용 구조를 이해했고, quota를 관리할 수 있고, 크레딧을 확보했고, reset 전략을 가지고 있습니다.

OpenClaw는 기술 프로젝트가 아니라, AI를 운영 가능한 구조로 만든 실험이었습니다.

공부가 구조를 이해하는 일이라면,

자동화는 그 구조를 분리하는 일이고, 운영은 그 구조를 안정시키는 일입니다.

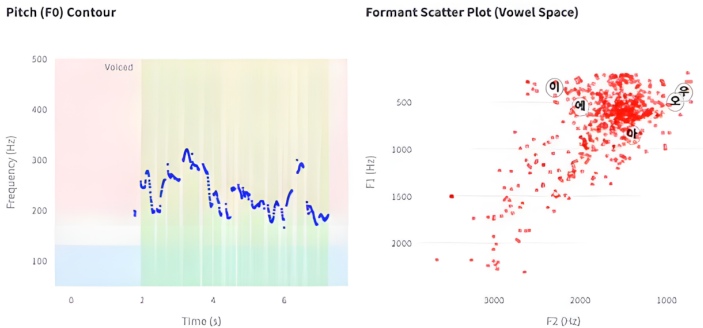
춘식은 단순한 붓이 아닙니다.

제가 만든 가장 개인적인 운영 시스템입니다.

6. 모델링과 확률적 판단

의학을 공부하다 보면 확실한 답보다 “가능성”을 다루는 시간이 더 많다는 걸 느끼게 됩니다. 검사 수치는 범위로 제시되고, 부작용은 확률로 설명되며, 예후는 예측으로 표현됩니다. 그때부터 저는 이렇게 생각하게 되었습니다. ‘우리는 사실을 다루는 게 아니라, 모델을 다루고 있는 건 아닐까.’

1) 수치를 감각에서 분리하다 — VoiceGrape



VoiceGrape의 분석 리포트

VoiceGrape는 “오늘 목소리가 잠긴 것 같다”는 감각에서 시작되었습니다. 그 감각을 Jitter, Shimmer, HNR 같은 정량 지표로 바꾸는 작업이었습니다. Praat 알고리즘을 기반으로 음성을 분석하고, F0, F1, F2를 시계열 데이터로 변환하고, Vowel Space를 시각화했습니다. 여기서 중요한 건 기술이 아니라 관점이었습니다.

주관적 감각을

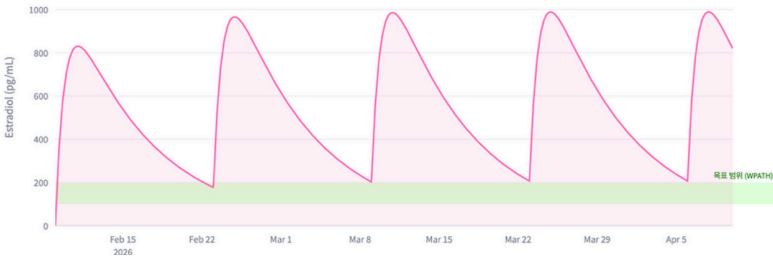
객관적 지표로 변환하는 일.

음성은 더 이상 “좋다, 나쁘다”가 아니라 노이즈 비율과 주파수 변동으로 설명됩니다.

이건 데이터 수집이 아니라 현상을 모델로 바꾸는 과정이었습니다.

2) 농도를 시간 위에 올리다 — EstroFrame

예측된 혈중 에스트라디올 농도 (60 시뮬레이션 기간 (일))



Estroframe 대시보드

EstroFrame은 약동학 모델에서 시작되었습니다 . 1-Compartment Open Model, Bateman Function, Superposition Principle.

투약은 사건이고,

혈중 농도는 함수입니다.

EstroFrame에서는 단순히 수치를 계산한 것이 아니라, 시간 축 위에 농도를 시뮬레이션했습니다 . 반복 투여 시 항정 상태 도달을 계산하고, 피검사 결과를 Newton-Raphson 방식으로 역산해

개인화 보정을 적용했습니다 .

여기서 핵심은 예측입니다.

지금 농도가 아니라

앞으로의 농도를 계산하는 일.

의학은 점(point)을 다루는 학문처럼 보이지만, 사실은 곡선(curve)을 다루는 학문이라는 걸 체감했습니다.

3) 모델은 정답이 아니라 가설이다

VoiceGrape도

EstroFrame도

Androframe도

Neuroframe도

공통점은 하나입니다.

현실을 그대로 재현하지 않습니다.

대신 단순화합니다.

모델은 세계를 줄여서 설명하는 틀입니다. 확률은 그 틀의 불완전함을 인정하는 방식입니다.

완벽한 예측은 존재하지 않습니다.

하지만 더 나은 추정은 가능합니다. 모델링은 정답을 만드는 일이 아니라, 불확실성을 다루는 태도를 만드는 일입니다.

저는 점점 확신하게 되었습니다.

의학적 판단도

공학적 설계도

결국은 모델 위에서 이루어진다는 것.

저는 개별 프로젝트를 자세히 설명하기보다, 이 모델적 사고를 기록하려 합니다.

수치를 감각에서 분리하고

시간을 함수로 바꾸고

확률을 두려워하지 않는 태도.

그게 제가 말하는 “모델링과 확률적 판단”입니다.

7. 감각의 객관화, VoiceGrape

“오늘 목소리가 잠긴 것 같다.”

이 문장은 감각의 언어입니다.

하지만 의학은 감각이 아니라 지표로 말합니다. VoiceGrape는 음성이라는 연속 신호를 정량적 파라미터로 환원하는 시도에서 시작되었습니다.

주소: <https://voicegrape.streamlit.app>

레포: <https://github.com/jsbang01357/Voicegrape>

1) 음성을 신호로 해석하다

The screenshot displays the VoiceGrape web application interface. On the left is a sidebar with a user profile icon and the text 'VoiceGrape 음성 포드'. Below this are sections for '로그인', '회원가입', and '비밀번호 찾기'. The main content area features a purple header with the 'VOICEGRAPE' logo and a subtitle: '당신의 목소리에 담긴 객관적 지표를 분석하고, 시간이 지남에 따라 변화를 지켜보는 흥미로운 음성 솔루션'. Below the header are three feature cards: '음성 분석' (Voice Analysis), '목소리 프로필' (Voice Profile), and '히스토리 관리' (History Management). The right sidebar contains '음성 분석' (Voice Analysis) with a description: '음성을 녹음하면 Parsonmouth를 통해 실시간으로 처리와 표정을 분석합니다.' Below this is '읽기 문구 설정' (Reading Text Settings) with a description: '다들 중국어를 읽어주는데... 규칙적인 음성은 어휘를 간단하게 만드는 가장 좋은 방법입니다.' At the bottom of the right sidebar, there is a '1. 음성 녹음' (1. Voice Recording) section with a '녹음 중: 10초' (Recording: 10s) indicator and a '녹음 중지' (Stop Recording) button.

VoiceGrape UI

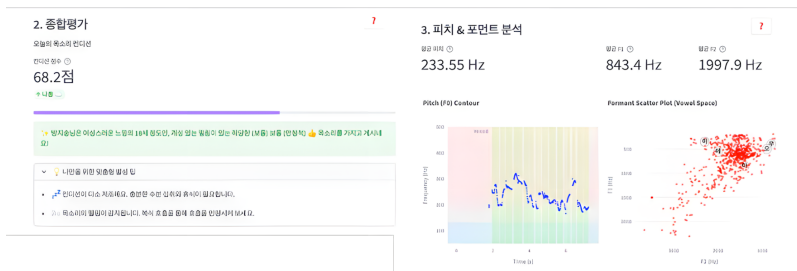
VoiceGrape는 Praat 기반의 음향 분석 엔진을 백엔드에 탑재한 플랫폼입니다 . 사용자의 음성을 입력받으면 자기상관(Autocorrelation)과 LPC(Linear Predictive Coding) 알고리즘이 작동하여 기본 주파수(F0)와 포먼트(Formant)를 추출합니다 .

파형은 연속적이지만, 분석은 이산적입니다.

2) 시계열과 좌표계로 음성을 펼쳐기

이 순간 음성은 더 이상 “목소리”가 아니라 Frequency Perturbation과 Amplitude Perturbation의 집합이 됩니다. Jitter는 주파수 변동률을 나타내고 Shimmer는 진폭 변동률을 나타냅니다 . HNR(Harmonic-to-Noise Ratio)는 신호 대 잡음 비율을 통해 기식성을 정량화합니다 . 감각은 설명이 아니라, 데이터 구조가 됩니다.

3) 시간-주파수 공간으로 확장하다



VoiceGrape 분석 리포트

VoiceGrape는 단일 스냅샷 분석에 머물지 않습니다. Pitch Contour는 시간 축 위에서의 F0 변화를 보여주고, Vowel Space Map은 조음 공간의 이동을 시각화합니다 .

이는 단순 시각화가 아닙니다.

시간 영역(Time Domain)과 주파수 영역(Frequency Domain)을 동시에 해석하는 작업입니다. 음성은 파형이 아니라 상태 전이(state transition)의 연속으로 보입니다. 또한 분석 결과를 시계열 데이터로 축적하여 추적 관찰이 가능하도록 설계했습니다 . 단일 측정이 아니라 longitudinal monitoring 구조입니다.

이 순간, 음성은 감각적 사건이 아니라 디지털 바이오마커(digital biomarker)가 됩니다.

4) 모델은 현실을 단순화한다

VoiceGrape는 음성을 완벽하게 재현하지 않습니다.

대신 모델링합니다.

Praat 기반 알고리즘은 복잡한 음향 신호를 소수의 파라미터로 환원합니다 .

이 환원은 손실(lossy compression)에 가깝습니다.

하지만 해석 가능성을 얻습니다.

모델은 현실의 축소판입니다.

완전하지 않지만 일관됩니다.

VoiceGrape를 만들면서 저는 확신하게 되었습니다. 의학적 판단은

결국 확률적 모델 위에서 이루어진다는 것.

“목소리가 잠겼다”는 감각은

Jitter 상승이라는 가설로 치환됩니다.

그리고 그 가설은

시간에 따라 검증됩니다.

VoiceGrape는 음성을 분석하는 도구가 아니라, 감각을 모델로 변환하는 실험이었습니다.

8. 상태를 모델로, NeuroFrame

NeuroFrame은 생산성 앱으로 시작하지 않았습니다. 솔직히 말하면, 인턴과 레지던트를 하게 될 미래가 두려워서 시작했습니다. 당직과 외래, 수술과 회진 사이에서

에너지는 바닥나고,

집중력은 흔들리고,

감정은 둔해질 것 같았습니다.

그때 이런 생각이 들었습니다.

시간을 관리하는 게 아니라,

상태를 관리해야 하는 건 아닐까.

주소: <https://neuroframe.streamlit.cloud>

레포: <https://github.com/jsbang01357/Neuroframe>

1) 일정이 아니라 상태를 기록하다

NeuroFrame

오늘 입력 수정

Design Your Day Around Your Brain.

면책: 본 예측은 입력값 기반 참고 정보이며 의료 조언이 아닙니다.

예측 신뢰도: 중간 (58/100)

신뢰도 하락 요인: 수면 오버라이드 미입력, 카페인/약물 입력 없음

Prime

13:20-18:20

Crash

03:30-06:40

Sleep Gate

23:00±

Rebound 후보

없음

Prime · 13:20-18:20

Crash · 03:30-06:40

Sleep Gate · 23:00±

하루 패턴 예측

기존 생산성 도구는 대부분 일정 중심입니다. 하지만 저는 다른 질문을 던졌습니다.

오늘의 에너지는 어땠는가.

집중도는 몇 점이었는가.

감정은 어떤 패턴을 보였는가.

NeuroFrame의 Today Input 시스템은 하루의 상태를 구조화된 데이터로 입력하도록 설계되었습니다.

에너지

집중도

감정 상태

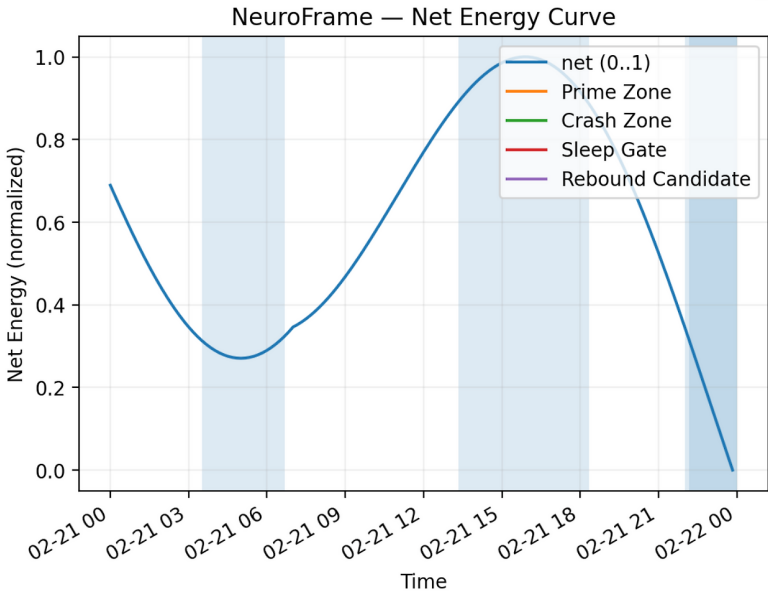
근무 형태

자기 평가 메모

단순 로그가 아닙니다.

이후 분석을 위한 상태 변수입니다. 감각을 수치로 바꾸는 작업이라는 점에서, VoiceGrape와 닮아 있습니다.

2) 패턴을 보는 순간 판단이 바뀐다



총 에너지 그래프

일주일이면 Weekly Metrics가 생성됩니다.

평균 에너지

근무 유형별 피로도 분포

당직 후 회복 곡선

자기평가 추세

이 데이터는 단순 통계가 아닙니다.

패턴을 보여주는 구조입니다.

예를 들어,

당직 다음 날의 에너지가 항상 3 이하라면 그건 의지 문제가 아니라 회복 설계의 문제입니다.

NeuroFrame은

“나는 왜 이렇게 힘들지?”라는 질문을 “어떤 패턴이 반복되고 있지?”로 바꿉니다.

이건 시간 관리가 아니라

인지 프레임 관리입니다.

3) 코칭은 개입이다

주간 리포트 (최근 7일)

Prime 평균 시작	Crash 평균 길이	최근48h 수면	7일 평균 수면
13:48	300분	14.4h	6.9h
수면부채(7d)		카페인 총량(7d)	
7.8h		1600mg	
계획-실행 일치율(7d)		완료/계획 블록(7d)	
		0/0	

- 이번 주 추천: 이번 주는 수면부채가 큰 편입니다. Off 또는 Day 다음날 우선 회복 수면을 확보하세요.

Coach 엔진

NeuroFrame의 핵심은 Coach 엔진입니다.

입력된 상태에 따라

리프레이밍 메시지를 제안하고

번아웃 위험을 경고하고

성취 패턴을 강화하는 피드백을 제공합니다.

이건 단순 통계 출력이 아닙니다.

행동 유도형 구조입니다.

제가 처음 이 시스템을 만든 이유는 “내가 망가지지 않기 위해서”였 습니다. 하지만 구조를 만들다 보니 생각이 확장되었습니다.

ADHD 환자

고강도 전문직 종사자

지속적인 감정 노동을 하는 사람들

이들에게도 동일한 문제가 존재합니다.

일정은 관리하지만

상태는 관리하지 못합니다.

NeuroFrame은

시간이 아니라 신경 인지 프레임을 다룹니다.

NeuroFrame은 완성된 솔루션이 아닙니다.

다만 하나의 가설입니다.

우리는 스스로를 더 잘 관리할 수 있을까.

의지가 아니라 데이터로.

인턴과 레지던트가 되기 전,

저는 먼저 제 상태를 모델로 만들었습니다.

NeuroFrame은

미래의 저를 위한 예방적 실험입니다.

그리고 어찌면,

같은 환경에 놓인 다른 사람들에게도 작은 도구가 될 수 있을지 모릅니다.

9. CDSS prototype, EstroFrame

강동성심병원 LGBTQ 센터에서 특성화 실습을 하면서 한 가지 의문이 생겼습니다. ‘호르몬 요법은 왜 이렇게 경험적으로 조정될까.’

가이드라인은 존재하지만,

개인별 대사 특성과 농도 변동을 실시간으로 예측하는 도구는 부족했습니다. 3~6개월 간격의 혈액 검사는 급격한 농도 변화를 충분히 포착하지 못합니다. 저는 내분비내과에 관심이 많았습니다. 호르몬을 “수치”가 아니라 “함수”로 보고 싶었습니다. 그래서 EstroFrame을 만들게 되었습니다.

주소: <http://estroframe.streamlit.app>

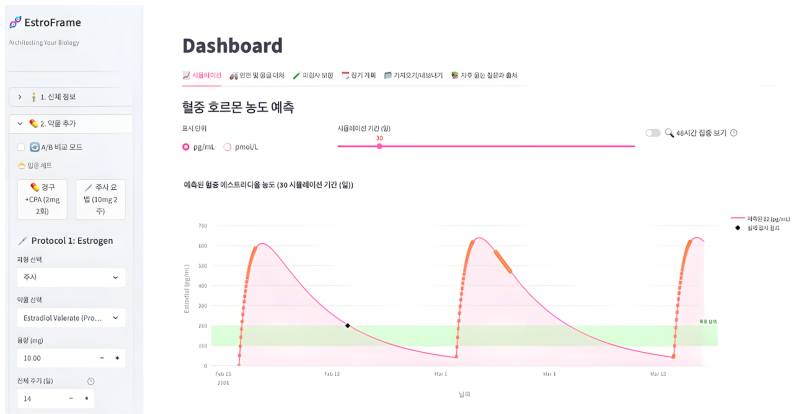
[EstroFrame

은 트랜스젠더 여성(MTF) 및 호르몬 대체 요법(HRT)을 진행하는 분들을 위한입니다.

https://estroframe.streamlit.app/?utm_medium=social(<http://estroframe.streamlit.app/>)

레포: <https://github.com/jsbang01357/Estroframe>

1) 농도를 시간 위에 올리다



EstroFrame 대시보드

EstroFrame은 1-Compartment Open Model을 기반으로 합니다 . 핵심 엔진은 Bateman Function이며, 반복 투여 시 Superposition Principle을 적용해 총 혈중 농도를 계산합니다 .

투약은 사건(event)이고

혈중 농도는 시간 함수입니다.

- Peak
- Trough
- Average concentration

이 값들은 단순한 숫자가 아니라, 시간-농도 곡선의 특정 지점입니다 .

의학은 점을 기록하지만,

약동학은 곡선을 다룹니다.

EstroFrame은 그 곡선을 시각화합니다.

2) 개인을 모델 안에 넣다

경로별 정밀 보정

검사 시점: 7 어떤 부위 결과를 기준으로 보정할까요? 주사

검사 결과 (pg/mL): 200.00

⚠ 주사: 주사에는 유약 주기 증강의 세팅 설정이 중요합니다. 정확한 보정을 위해 유약 주기 증강 시점이 세팅된 결과를 입력하세요.

+ 검사 기록 추가

주사 검사 이력

7일 200.0 pg/mL

현재 적용된 개인화 계수

주사 계수

x 0.79

보정 초기화

개인화 보정

문헌상의 반감기를 그대로 적용하면 개인차로 인해 오차가 발생합니다. 그래서 저는 Newton-Raphson 기반의 역산 알고리즘을 적용해, 실제 피검자 결과를 바탕으로 대사율(Clearance)과 분포용적을 보정했습니다.

이 과정은 단순 계산이 아닙니다.

환자의 실제 데이터 →

모델 파라미터 보정 →

개인화된 농도 곡선 생성

이 구조는 디지털 트윈에 가깝습니다. 환자의 내분비계를 완벽히 재현하는 것이 아니라, 예측 가능한 범위 안에서 근사하는 모델입니다.

여기서 저는 정밀의학이 “정확함”이 아니라 “보정 가능한 모델”이라는 걸 체감했습니다.

3) CDSS라는 방향

📌 임상 안전성 분석

⚠️ [주의] 감정 변화 가능성: 농도 변화가 매우 급격함으로 인해 우울감, 무기력증, 극심한 기분 기복이 발생할 수 있습니다. 투여 간격을 좁히거나 제형 변경을 고려해 보세요.

📌 임상 추적검사 가이드

💡 가이드라인: 첫 해에는 약 3개월마다 평가하여 용량을 조절하고, 이후에는 연 1-2회 정기 검사를 권장합니다.

약품군	필수/관량 검사 항목
Estradiol Valerate (Progynova)	E2, T, LFT (간수치)
Cyproterone Acetate	Liver Function (LFT), Prolactin

임상 안전성 분석 및 추적검사 가이드

EstroFrame은 계산기가 아니라 CDSS(Clinical Decision Support System) 프로토타입입니다 . WPATH SOC 8 가이드라인을 반영하고

VTE 위험도 평가를 내장하고

수술 전 wash-out period를 예측합니다 . 이걸 처방을 대신하는 시스템이 아닙니다.

의사결정을 보조하는 모델입니다.

농도가 100~200 pg/mL 범위에 있는지 수술 당일 잔존 농도가 안전 기준선 아래인지

의사는 판단을 내립니다.

모델은 가설을 제공합니다.

강동성심병원 LGBTQ 센터 실습은 저에게 임상적 질문을 던져주었습니다. EstroFrame은 그 질문에 대한 공학적 응답이었습니다.

호르몬은 단순한 수치가 아닙니다.

시간에 따라 움직이는 곡선입니다.

그리고 그 곡선을 이해하는 순간,

내분비학은 감각이 아니라 모델이 됩니다. EstroFrame은 정밀의학을 완성한 시스템이 아닙니다. 다만 호르몬을 디지털 트윈으로 바라본 첫 시도입니다.

Part 4. 기록은 곧바로 데이터가 되지 않는다

1. 감각에서 수치로, 수치에서 예측으로

의료는 감각에서 시작됩니다.

환자는 먼저 느낌을 말합니다.

“목소리가 잠긴 것 같아요.” “몸이 예전 같지 않아요.” “약을 쓰고 나서 조금 이상해요.”

이런 말들은 모호합니다. 하지만 중요합니다. 의학은 처음부터 숫자로 시작하지 않습니다. 대부분은 환자가 느끼는 변화, 불편함, 이상한 감각에서 시작됩니다.

문제는 감각이 그대로는 다루기 어렵다는 점입니다.

감각은 개인적이고, 언어는 애매하고, 기억은 흔들립니다. 어제보다 나아졌는지, 지난달보다 나빠졌는지, 치료 전후로 얼마나 달라졌는지 말만으로는 비교하기 어렵습니다. 그래서 의료는 감각을 수치로 바꾸려 합니다.

1) 감각은 출발점이다

감각은 부정확하다고 버릴 수 있는 것이 아닙니다. 오히려 감각은 가장 앞에 있습니다.

환자가 느낀 불편함이 없었다면 진료도 시작되지 않습니다. 증상이 없었다면 검사를 하지 않았을 수도 있습니다. 몸의 미세한 변화는 종종 환자가 가장 먼저 알아차립니다. 하지만 감각은 출발점이지 종착점은 아닙니다.

“목소리가 잠겼다”는 말은 중요합니다. 하지만 그 말만으로는 목소리의 변화를 추적하기 어렵습니다. “피곤하다”는 말도 중요합니다. 하지만 피로가 수면 부족 때문인지, 카페인 때문인지, 업무량 때문인지, 질병 때문인지 알기 어렵습니다.

“약이 잘 안 듣는 것 같다”는 말도 중요합니다. 하지만 실제 농도 변화인지, 기대와 체감의 차이인지, 다른 요인이 섞인 것인지 확인해야 합니다.

감각은 질문을 만듭니다.

그다음 필요한 것은 그 질문을 다룰 수 있는 형태로 바꾸는 일입니다.

2) 수치는 감각을 보존한다

수치는 감각을 지우기 위해 존재하지 않습니다. 수치는 감각을 비교 가능하게 만들기 위해 존재합니다. 조금 더 정확히 말하면, 수치화는

환자를 숫자로 줄이는 일이 아니라 환자가 느낀 변화를 비교 가능한 형태로 보존하는 일입니다.

목소리는 피치, 포먼트, 지터, 시머 같은 값으로 볼 수 있습니다. VoiceGrape는 “오늘 목소리가 잠긴 것 같다”는 감각을 음성 지표로 바꾸어 추적하려는 시도였습니다. 피로는 수면 시간, 카페인 섭취, 업무량, 주관적 명료도의 조합으로 볼 수 있습니다. NeuroFrame은 “오늘 상태가 좋지 않다”는 감각을 하루의 에너지 곡선으로 바꾸려는 시도였습니다.

약물 반응은 혈중 농도, 검사 수치, 증상 변화, 부작용의 흐름으로 볼 수 있습니다. EstroFrame은 호르몬 치료를 단일 검사 수치가 아니라 시간에 따른 농도 곡선으로 이해하려는 시도였습니다. 수치가 생기면 비교가 가능해집니다. 어제와 오늘을 비교할 수 있고, 치료 전후를 비교할 수 있고, 예상과 실제를 비교할 수 있습니다.

하지만 여기서 멈추면 부족합니다.

수치는 그 자체로 답이 아닙니다.

A1c가 높다. eGFR이 낮다. 호르몬 농도가 기준보다 높다. 심박수가 빠르다.

이런 수치들은 중요하지만, 각각 따로 존재할 때는 제한된 의미만 갖습니다. 의학에서 중요한 것은 단일 숫자가 아니라 그 숫자가 놓인 맥락입니다.

언제 측정했는지. 이전 값과 어떻게 달라졌는지. 어떤 약을 쓰고 있었는지. 환자의 상태가 어땠는지. 위험 요인이 무엇이었는지.

수치는 비교를 가능하게 하지만, 해석은 관계 속에서 시작됩니다.

3) 해석은 관계를 보는 일이다

해석은 숫자에 의미를 붙이는 과정입니다. 같은 수치라도 맥락에 따라 의미는 달라집니다. 어떤 환자에게는 경과 관찰이 적절할 수 있고, 다른 환자에게는 치료 변경이 필요할 수 있습니다. 어떤 변화는 일시적인 흔들림일 수 있고, 어떤 변화는 추세의 시작일 수 있습니다. 그래서 해석은 단순히 정상과 비정상을 나누는 일이 아닙니다. 수치와 수치 사이의 관계, 수치와 시간의 관계, 수치와 약물의 관계, 수치와 환자 맥락의 관계를 보는 일입니다.

이 단계에서 모델이 필요해집니다.

모델은 관계를 정리합니다.

입력이 무엇인지, 상태가 무엇인지, 출력이 무엇인지, 어떤 조건에서 결과가 달라지는지 정리합니다. 모델이 있으면 숫자는 단순한 기록이 아니라 다음 상태를 추정하기 위한 재료가 됩니다.

4) 예측은 판단을 준비시킨다

예측은 미래를 맞히는 일이 아닙니다. 예측은 다음 판단을 준비하는 일입니다. 이 약을 계속 쓰면 수치는 어느 방향으로 움직일까. 치료를 강화하면 얻을 수 있는 이득과 감수해야 할 위험은 무엇일까. 수술 전 약을 중단하면 어느 시점에 농도가 낮아질까. 현재 변화가 계속된다면 다음 외래에서는 무엇을 확인해야 할까.

의사는 미래를 완벽하게 알 수 없습니다. 하지만 아무 예측 없이 판

단할 수도 없습니다. 의료의 많은 결정은 지금 보이는 정보만으로 끝나지 않습니다. 지금의 선택이 며칠 뒤, 몇 주 뒤, 몇 달 뒤 어떤 상태를 만들 것인지 생각해야 합니다.

그래서 예측은 중요합니다.

예측은 결정을 대신하지 않습니다. 하지만 결정이 향하는 방향을 보여줍니다.

5) 불확실성은 관리되어야 한다

감각을 수치로 바꾸고, 예측 모델을 만들어도 불확실성은 남습니다. 측정값에는 오차가 있고, 환자마다 반응은 다릅니다. 의학에서 불확실성을 완전히 제거하는 것은 불가능합니다. 오히려 중요한 것은 불확실성이 어디에 있는지 파악하고 이를 안고 판단하는 태도입니다. 불확실성을 표시할 수 있는 판단은 조금 더 정직합니다.

의사는 예측을 제시할 수 있어야 합니다. 동시에 그 예측의 한계도 설명할 수 있어야 합니다. 제가 관심 있는 의료 구조화는 여기에 가깝습니다. 감각을 무시하지 않고, 수치에만 매달리지 않고, 모델을 맹신하지 않고, 예측을 정답처럼 말하지 않는 것.

대신 각각의 단계를 구분하는 것.

감각은 질문을 만들고, 수치는 비교를 가능하게 하며, 해석은 관계를 보여주고, 예측은 판단을 준비시키며, 불확실성 관리의 책임 있는 설명을 가능하게 합니다. 제가 의료와 공학 사이에서 반복해서 생각하는 흐름은 결국 같습니다.

감각에서 수치로. 수치에서 해석으로. 해석에서 예측으로. 예측에서 불확실성 관리로.

의료는 감각에서 출발하지만, 감각에 머물 수는 없습니다. 그리고 수치로 나아가지만, 수치만으로 끝나지도 않습니다. 그 사이를 연결하는 구조가 필요합니다.

저는 그 구조를 만들고 싶습니다.

2. 의료 텍스트는 왜 바로 데이터가 되지 않는가

병원에는 정보가 많습니다. 환자의 증상, 병력, 검사 결과, 처방, 영상 판독, 수술 기록, 외래 기록이 모두 남습니다. 진료가 끝날 때마다 기록은 쌓이고, 검사를 할 때마다 수치는 추가되고, 약을 바꿀 때마다 처방 이력이 남습니다. 겉으로 보면 의료 데이터는 이미 충분히 존재하는 것처럼 보입니다. 하지만 막상 그 기록을 연구나 분석에 쓰려고 하면 생각보다 바로 사용할 수 없습니다.

정보가 없어서가 아닙니다. 구조가 없기 때문입니다.

1) 기록은 사람을 위해 만들어진다

EMR은 기본적으로 사람이 읽기 위한 기록입니다. 의사가 환자의 상태를 빠르게 파악하고, 다음 진료에서 이전 맥락을 이어가고, 다른 의료진과 정보를 공유하기 위해 작성합니다. 그래서 EMR은 실제 진료에는 매우 유용합니다.

문장으로 적힌 병력, 줄글로 정리된 외래 기록, 복사되어 붙은 검사

결과, 약어와 기호가 섞인 처방 메모, 의사마다 조금씩 다른 표현 방식 등 사람은 이런 기록을 어느 정도 읽어낼 수 있습니다. 문맥을 보고 중요한 부분을 골라내며, 빠진 내용을 추론하고 익숙한 약어를 해석합니다.

하지만 컴퓨터는 그렇게 읽지 못합니다. 컴퓨터에게 필요한 것은 문맥이 아니라 구조입니다. 어떤 값이 어떤 검사인지, 측정일은 언제인지, 단위는 무엇인지, 정량값인지 정성값인지, 이전 값과 비교할 수 있는지, 환자를 식별할 수 있는 정보는 제거되었는지. 이런 구조가 있어야 기록은 데이터가 됩니다.

2) 텍스트와 데이터 사이에는 간격이 있다

의료 기록은 텍스트입니다. 하지만 연구와 AI가 필요로 하는 것은 대개 테이블입니다. 행과 열이 있고, 변수와 값이 있고, 시간축이 있고, 비교 가능한 형식이 있어야 합니다.

예를 들어 검사 결과를 복사해보면 사람이 보기에는 대충 이해할 수 있습니다. HbA1c가 얼마인지, creatinine이 얼마인지, AST와 ALT가 어떤지 눈으로 보면 읽을 수 있습니다. 하지만 그 값을 분석하려면 질문이 달라집니다. HbA1c라는 이름은 항상 같은 방식으로 적혀 있는가. 수치는 숫자로 분리되어 있는가. 단위는 붙어 있는가, 떨어져 있는가. 정성 결과와 정량 결과가 섞여 있지는 않은가. 보고서형 문장은 따로 분리되어 있는가. 표로 변환했을 때 각 열의 의미가 유지되는가.

사람에게는 사소해 보이는 차이가 데이터 처리에서는 큰 차이가 됩니다. 줄바꿈 하나, 공백 하나, 단위 표기 하나, 검사명 약어 하나가 파싱을 어렵게 만들 수 있습니다. 그래서 의료 텍스트는 바로 데이터가 되지 않습니다. 텍스트와 데이터 사이에는 정리, 분리, 구조화의 단계가 필요합니다.

3) 원문을 바로 AI에 넣는 것은 충분하지 않다

AI를 쓰기 쉬워지면서 의료 기록도 그냥 AI에 넣으면 되지 않느냐는 생각을 할 수 있습니다. 긴 외래 기록을 넣고 요약을 시키거나, 검사 결과를 넣고 해석을 요청하거나, 처방 기록을 넣고 다음 판단을 물어보는 방식입니다. 분명 AI는 이런 작업을 어느 정도 도와줄 수 있습니다. 하지만 원문 텍스트를 그대로 넣는 방식에는 한계가 있습니다.

첫째, 입력이 불안정합니다. 같은 정보라도 기록 형식이 다르면 AI가 다르게 해석할 수 있습니다. 둘째, 검증이 어렵습니다. AI가 어떤 값을 근거로 판단했는지, 어떤 수치를 놓쳤는지, 어떤 문장을 잘못 해석했는지 확인하기 어렵습니다. 셋째, 개인정보 문제가 있습니다. 의료 기록에는 환자를 식별할 수 있는 정보가 섞여 있을 수 있습니다. 이름, 등록번호, 날짜, 연락처, 의료진 이름, 기관 정보가 생각보다 자연스럽게 기록 안에 들어갑니다. 따라서 원문을 그대로 외부 모델에 넣는 것은 기술적으로도, 윤리적으로도 조심해야 합니다. AI가 유용하려면 먼저 입력이 정리되어 있어야 합니다. 좋은 출력은

좋은 입력에서 시작합니다.

4) 먼저 정리되어야 한다

의료 텍스트를 데이터로 만들기 위해서는 몇 가지 중간 단계가 필요합니다. 먼저 노이즈를 줄여야 합니다. 불필요한 공백, 깨진 줄바꿈, 복사 과정에서 생긴 기호, 반복되는 머리말과 꼬리말을 정리해야 합니다. 그다음 정보를 분리해야 합니다. 정량 검사 결과, 정성 검사 결과, 보고서형 소견, 일반 텍스트, 파싱되지 않은 항목 등은 한 화면에 섞여 있을 수 있지만, 데이터로 쓰려면 서로 다른 층위로 나누어야 합니다.

그리고 비식별화가 필요합니다. 환자명, 등록번호, 날짜, 의료진 이름처럼 식별 가능성이 있는 정보는 조심스럽게 처리해야 합니다. 마지막으로 표로 바꿔야 합니다. 검사명, 결과값, 단위, 참고범위, 날짜 같은 항목이 일관된 열로 정리되어야 비교하고, 분석하고, 모델에 넣을 수 있습니다.

이 과정은 화려하지 않지만 매우 중요합니다. 의료 AI의 앞단에는 대개 이런 지저분한 정리 작업이 있습니다. 모델을 만들기 전에 데이터가 먼저 구조화되어야 합니다.

5) CleanText는 중간층을 만들려는 시도였다

CleanText는 이 문제에서 시작했습니다. 의료 텍스트를 바로 AI에 넣기 전에, 먼저 사람이 확인할 수 있는 형태로 정리하고 싶었습니다

다. 검사 결과 원문을 정량 데이터, 정성 데이터, 보고서형 텍스트로 나누고 표로 바꾸며, 필요하면 엑셀이나 TSV로 내보낼 수 있게 만드는 것입니다. 외래 기록이나 SOAP note처럼 줄글로 되어 있는 텍스트도 읽기 쉬운 형태로 정돈하는 과정입니다.

이때 가장 먼저 생각해야 하는 것은 보안이었습니다. 의료 텍스트에는 환자를 식별할 수 있는 정보가 섞여 있을 수 있습니다. 이름, 등록번호, 날짜, 의료진 이름, 기관 정보처럼 사람이 보기에는 사소해 보이는 정보도 맥락에 따라 식별 가능성이 생길 수 있습니다. 그래서 의료 텍스트를 다룰 때 가장 위험한 방식은 원문을 그대로 외부 AI에 던져 넣는 것입니다. 입력한 원문이 불필요하게 저장되거나 전송되지 않도록, 처리의 경계를 먼저 정하는 것이 중요하다고 생각했습니다.

하지만 자동 비식별화는 완벽하지 않습니다. 어떤 정보가 식별 가능한지 판단하려면 결국 사람이 다시 확인해야 합니다. 그래서 Clean Text의 목표는 “안전하게 다 처리해주는 도구”가 아니라, 의료 텍스트를 더 조심스럽게 정리하고 검토할 수 있는 중간 작업대를 만드는 데 가깝습니다.

의료 AI는 좋은 프롬프트나 강력한 모델만으로 만들어지지 않습니다. 어떤 정보를 입력할 것인지, 어떤 형식으로 정리할 것인지, 어떤 정보는 제거할 것인지, 어떤 값은 사람이 다시 확인하게 할 것인지, 어디까지 브라우저 안에서 처리하고 외부 시스템으로 보내지 않을 것인지. 이런 구조가 먼저 필요합니다.

6) 의료 데이터의 시작은 구조화다

EMR에는 정보가 많습니다. 하지만 정보가 많다는 것과 데이터로 사용할 수 있다는 것은 다릅니다. 기록은 진료를 위해 존재하고, 데이터는 비교와 분석을 위해 존재합니다. 기록을 데이터로 바꾸려면 그 사이에 구조화가 필요합니다.

이름을 붙이고, 값을 분리하고, 시간축에 올리고, 불필요한 노이즈를 줄이고, 식별 정보를 제거하고, 검증 가능한 형태로 정리해야 합니다. 이 과정은 눈에 잘 띄지 않지만, 의료 AI와 임상 의사결정지원시스템은 바로 이 과정 위에 올라갑니다. 정리되지 않은 입력 위에 좋은 모델을 올릴 수는 없습니다. 그래서 저는 의료 텍스트를 먼저 봅니다.

EMR은 정보가 없는 것이 아닙니다. 구조가 없는 것입니다. 그리고 의료를 구조화한다는 일은 바로 그 간격을 메우는 일에서 시작됩니다.

3. 로컬 기반 EMR 파싱, CleanText

의료 텍스트를 다룰 때 가장 먼저 떠올려야 하는 것은 편리함이 아닙니다.

보안입니다.

EMR에는 많은 정보가 들어 있습니다. 환자의 증상과 병력, 검사 결과, 처방, 진료 기록이 남습니다. 그 안에는 환자를 직접 식별할 수 있는 정보도 있고, 겉으로는 사소해 보이지만 맥락에 따라 식별 가

능성이 생기는 정보도 있습니다. 그래서 의료 텍스트를 다루는 도구를 만들 때 가장 먼저 정해야 하는 것은 기능이 아니었습니다.

어디까지 처리할 것인가.

어디까지 저장하지 않을 것인가.

어디까지 서버로 보내지 않을 것인가. CleanText는 이 질문에서 시작했습니다.

[CleanText]

개발 기간: 2026년 2월 ~ 2026년 6월

버전: v2.0.0

형태: 의료 텍스트 정리 및 검사 결과 구조화 도구

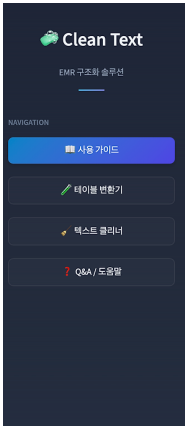
스택: Python, Streamlit, React, TypeScript, Vite, Docker, Google Cloud Run

핵심 구조: 기존 Python 중심 파싱 흐름을 브라우저 기반 처리 구조로 전환

주요 기능: EMR 텍스트 정리, 검사 결과 표 변환, 비식별화, Excel/TSV 내보내기 설계 원칙: 원문 텍스트의 서버 의존도를 줄이고, 핵심 처리 로직을 가능한 한 브라우저 안에서 수행

주소: cleantext.jisong.dev

레포: https://github.com/jsbang01357/Clean_Text



사용 가이드

Clean Text v2.0.0-260416 - 의료 텍스트 정리와 구조화를 돕는 브라우저 기반 도구

💡 EMR 텍스트 복사 팁

긴 기록이나 검사 결과를 복사할 때는 마우스로 길게 드래그하는 것보다 아래 방법이 더 편할 수 있습니다.

1. 복사하려는 텍스트의 첫 줄을 클릭합니다.
2. 키보드의 Shift 키를 누른 상태에서 아래로 이동한 뒤 마지막 줄을 클릭합니다.
3. 선택된 영역을 Ctrl + C (Mac은 Cmd + C)로 복사합니다.
4. 이 도구의 입력창에 Ctrl + V (Mac은 Cmd + V)로 붙여넣습니다.

🔗 이용 순서

STEP 1

원문 텍스트를 입력창에 붙여넣습니다.

STEP 2

필요한 도구를 선택한 뒤 [분석하기] 버튼을 누릅니다.

STEP 3

하단에 생성된 미리보기 결과를 확인합니다.

STEP 4

필요하면 엑셀 다운로드 또는 복사 기능을 사용합니다.

CleanText 첫 화면

1) 서버로 보내면 편하지만, 위험도 커진다

텍스트 처리 도구를 만드는 가장 쉬운 방식은 서버에 원문을 보내고, 서버에서 처리한 뒤 결과를 돌려주는 것입니다.

구현은 단순해집니다.

원문을 서버로 받고,

파싱하고,

비식별화하고,

표로 바꾸고,

결과를 다시 보여주면 됩니다.

하지만 의료 텍스트에서는

이 구조가 곧바로 부담이 됩니다.

서버가 원문을 받는 순간,

그 서버는 의료 데이터를 취급하는 공간이 됩니다.

저장하지 않겠다고 해도,

전송 과정이 생기고,

로그가 남을 수 있고,

예외 상황이 생길 수 있습니다.

개발자 입장에서는 편한 구조일 수 있지만, 의료 데이터 입장에서는 위험한 구조일 수 있습니다. 그래서 저는 CleanText를 만들 때 가능한 한 원문 처리의 중심을 브라우저 쪽으로 옮기고 싶었습니다.

사용자가 붙여넣은 텍스트를

먼저 사용자의 브라우저 안에서 정리하고,

구조화하고,

비식별화하고,

엑셀이나 TSV로 내보내는 방식.

핵심은 단순합니다.

원문을 덜 움직이게 하는 것.

의료 데이터 보안에서 중요한 것은

나중에 붙이는 경고 문구가 아니라

처음부터 원문이 어디로 흐르는지 정하는 구조라고 생각했습니다.



브라우저 기반 처리 구조

2) 비식별화는 버튼 하나로 끝나지 않는다

CleanText에는 비식별화 기능이 들어 있습니다. 환자명, 직원명, 날짜, 식별자 패턴 등을 감지해 자동으로 가리는 방향을 시도합니다.

하지만 저는 자동 비식별화를

완성된 안전장치로 생각하지 않습니다. 자동 비식별화는 어디까지나 보조 장치입니다. 의료 텍스트는 형식이 일정하지 않습니다.

병원마다 다르고,

과마다 다르고,

의료진마다 다르고,

복사해온 화면마다 다릅니다.

어떤 정보는 명확한 식별자처럼 보이지만, 어떤 정보는 문맥 안에서만 식별 가능성이 생깁니다.

날짜 하나가 문제 될 수도 있고,

희귀한 수술명이나 특이한 병력 조합이 간접 식별 가능성을 만들 수도 있습니다.

그래서 비식별화는

“자동으로 처리했으니 안전하다”로 끝날 수 없습니다.

도구는 후보를 가리고,

사용자는 다시 확인해야 합니다.

CleanText에서 중요하게 생각한 것도 이 지점이었습니다. 원문과 정리된 결과를 비교할 수 있게 하고, 구조화 결과를 사람이 다시 볼 수 있게 하고, 중요한 내용은 최종 활용 전에 직접 확인하도록 하는 것.

의료 데이터 도구는

사람의 확인을 없애는 방향이 아니라 사람이 더 잘 확인할 수 있게 만드는 방향이어야 한다고 생각했습니다.

The screenshot displays a web interface for medical text de-identification. At the top, there are three status indicators: 'Privacy Protect' (의료 텍스트 비식별화 도구), '호환 처리 (선택유지)' (선택유지 여부 지정하여 업로드), and '안전한 비식별화' (자동 처리 후 사용자 검토 필수). A '지정 내보내기' button is also present.

The main content is divided into three panels:

- Original Text:** Shows a sample text with fields like Name (Hong 613-dong), Patient ID (12345678), Date (2026-01-01), Department (Endocrinology), Visit Type (Outpatient), Chief Complaint, History of Present Illness, Assessment, and Plan. A '만장일치 일치' (Full match) indicator is shown.
- De-identified Result:** Shows the same text with fields masked as [NAME], [PATIENT_ID], [DATE], and Endocrinology. Each field has a '확인' (Check) button.
- Human Review:** A section for manual review with a warning: '자동 비식별화 결과를 사용자가 직접 검토하고 최종 확인해야 합니다.' (You must manually review the automatic de-identification results and confirm them.) It includes three checkboxes: '1. 민감정보 대해 확인' (Check sensitive information), '2. 의미 보존 확인' (Check meaning preservation), and '3. 텍스트 품질 확인' (Check text quality).

At the bottom, a note states: '자동화는 도구일 뿐, 최종 확인은 사용자에게 있습니다. 모든 결과는 반드시 사람이 검토한 후 사용하세요.' (Automation is just a tool, final confirmation is up to the user. All results must be reviewed by a person before use.)

비식별화 과정

3) 정리, 구조화, 내보내기

CleanText의 기능은 크게 세 가지로 나눌 수 있습니다.

첫째, 텍스트를 정리합니다.

EMR 특유의 줄바꿈 오류,

전각 공백,

특수 마커,

복사 과정에서 생긴 노이즈를 줄입니다.

둘째, 검사 결과를 구조화합니다.

수치 중심의 정량 데이터와

정성·반정량 결과를 구분하고,

보고서형 텍스트는 별도로 분리합니다.

셋째, 결과를 내보냅니다.

정리된 결과를 표로 보고,

필요하면 TSV나 Excel 파일로 저장할 수 있게 합니다.

이 과정은 화려하지 않습니다.

하지만 의료 데이터 작업에서는

이런 앞단의 정리가 매우 중요합니다.

AI 모델에 무엇을 넣을 것인지.

연구용 데이터로 어떤 값을 추출할 것인지. 나중에 실제 판단과 비교할 수 있는 구조를 어떻게 만들 것인지.

이 모든 것은

입력 데이터가 정리되어 있어야 가능합니다. CleanText는 답을 내리는 도구가 아닙니다.

오히려 답을 묻기 전에

질문에 들어갈 재료를 정리하는 도구에 가깝습니다.

🏠 Lab / EMR 원문 입력 ⇄

테이블 변환기 ✔ 개인정보 비식별화

입력된 데이터는 브라우저 내에서 즉시 파싱됩니다.
서버로 데이터가 전송되지 않아 보안이 유지되며, 결과를 먼저 확인한 뒤 엑셀을 다운로드할 수 있습니다.

검사명	결과값	단위	참고치
(응급)CBC with diff count & ESR			
(응급)WBC	8.76	$\times 10^3 / \mu\text{l}$	4~10
(응급)RBC	3.41 ▼	$\times 10^6 / \mu\text{l}$	4.2~6.3
(응급)Hb	10.2 ▼	g/dL	13~16.5
(응급)Hct	31.4 ▼	%	39~49
(응급)ESR	21 ▲	mm/h	0~15

[소견]
[의뢰의사 Comment]
- 투석 전 채혈 / pre HD

분석하기엑셀 다운로드입력 초기화

정량 Tsv 다운로드정상 Tsv 다운로드

분석 완료 리포트

정량 row: 5	정상 row: 0	정량 표 수: 1	정상 표 수: 0	보고서 줄: 0
미파싱 줄: 0	원문 줄 수: 12			

정량 결과 미리보기

그룹별로 묶인 정량 검사 결과를 브라우저에서 바로 확인할 수 있습니다.

▼ 응급혈액_A-1

검사명	결과값	단위	참고치
WBC	8.76	$\times 10^3 / \mu\text{L}$	4~10
RBC	3.41 ▼	$\times 10^6 / \mu\text{L}$	4.2~6.3
Hb	10.2 ▼	g/dL	13~16.5
Hct	31.4 ▼	%	39~49
ESR	21 ▲	mm/h	0~15

Lab → Excel 변환 결과 화면

4) 보안은 기능이 아니라 아키텍처다

의료 데이터 도구에서 보안은

마지막에 붙이는 기능이 아닙니다.

“주의해서 사용하세요”라는 문구만으로는 부족합니다. 어떤 데이터가 어디에서 처리되는지, 어떤 데이터가 서버로 가는지, 무엇이 저장되고 무엇이 저장되지 않는지, 사용자가 어느 지점에서 다시 확인하는지. 이런 구조가 먼저 정해져야 합니다. CleanText는 완벽한 보안 시스템이 아닙니다. 의료기관의 보안 절차를 대체하지도 않습니다.

자동 비식별화가 모든 위험을 없애주지도 않습니다. 결과의 최종 확인은 여전히 사람의 몫입니다. 다만 제가 이 도구를 만들면서 정한 원칙은 분명했습니다. 원문을 불필요하게 움직이지 않을 것.

핵심 처리는 가능한 한 브라우저 안에서 끝낼 것. 자동화된 결과를 사람이 다시 확인할 수 있게 할 것. AI에 넣기 전에 먼저 정리하고 줄일 것. 저는 이것이 의료 데이터 도구의 기본 태도라고 생각했습니다.

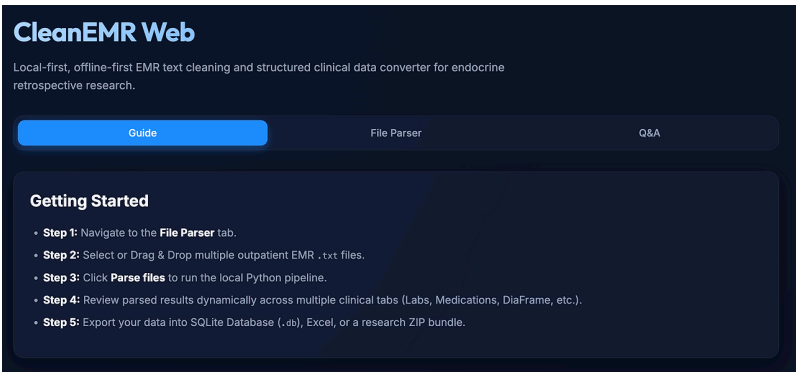
기술적으로 더 편한 방식이 있더라도, 의료 데이터에서는 먼저 경계를 그어야 합니다. 그 경계 안에서 도구를 설계해야 합니다.

CleanText는 경계 안의 도구 설계의 첫 시도였습니다. 의료 텍스트를 더 편하게 다루기 위한 도구이기 전에, 의료 텍스트를 더 조심스럽게 다루기 위한 도구. 제가 브라우저 안에서 EMR을 정리하려 한 이유는

바로 거기에 있습니다.

4. EMR 연구용 파싱, CleanEMR

CleanText를 만들면서 텍스트를 깨끗하게 정리하는 것과 연구 가능한 데이터베이스를 만드는 것은 서로 다른 문제라는 것을 곧 알게 되었습니다. 검사 결과 한 덩어리를 표로 바꾸는 것만으로는 부족했습니다. 연구를 하려면 환자가 있어야 하고, 방문이 있어야 하고, 검사와 약물이 있어야 하며, 그 시점의 판단이 있어야 합니다. 의료 기록은 한 줄의 텍스트가 아니라 시간에 따라 쌓이는 사건들의 묶음입니다. CleanEMR은 바로 이 지점에서 시작했습니다.



CleanEMR 첫 화면

개발 기간: 2026년 4월 25일~ **버전:** v2.0.0 **형태:** 로컬 기반 외래 EMR 전처리 및 연구용 데이터셋 생성 파이프라인 **스택:** Python 3.12, FastAPI, SQLite3, React, TypeScript, Vite, Pandas, Pytest
대상: 내분비/당뇨 외래 자유 텍스트 EMR **출력:** cleanemr.db, 연구용 CSV/JSON 번들, codebook, quality report, Excel bundle
핵심 구조: 원문 EMR → 정규화 → 비식별화 → 외래 방문 분리 → SOAP/검사/활력징후/약물 파싱 → 근거 기반 매핑 → 미파싱 문장

리뷰 → 연구용 DB/파일 내보내기

1) 기록은 환자 단위로 다시 묶어야 한다

EMR 기록은 진료를 위해 작성됩니다. 의사는 이전 외래 기록을 보고 검사 결과를 확인한 뒤, 약을 조정하고 다음 계획을 남깁니다. 이 구조는 진료 현장에는 적합하지만, 연구를 하려면 기록을 다른 방식으로 다시 배열해야 합니다.

한 환자의 여러 방문을 연결해야 하고, 방문마다 어떤 검사값이 있었는지 정리해야 하며, 그 시점에 어떤 약을 쓰고 있었는지 알아야 합니다. 또한 다음 방문에서 약이 추가되었는지, 중단되었는지, 혹은 유지되었는지도 확인해야 합니다. 즉 연구용 데이터는 단순히 깨끗한 텍스트가 아니라, 환자와 방문을 기준으로 정렬된 구조여야 합니다.

CleanEMR은 이처럼 외래 기록을 환자 단위, 방문 단위, 검사 단위, 약물 단위로 다시 나누려는 시도였습니다. 텍스트를 읽기 좋게 만드는 것이 아니라, 기록 자체를 연구 가능한 테이블로 바꾸는 것이 목표였습니다.

2) 외래 기록을 방문 단위로 나누다

The screenshot displays the CleanEMR interface. At the top, there are buttons for 'Parse files' and 'Clear files'. A green notification bar states 'Successfully parsed 8 EMR file(s). Explore data tabs below.' Below this, a dashboard shows counts for various data categories: PATIENTS (8), VISITS (73), LABS (689), MEDS (126), SUMMARY (73), DIAFRAME (73), and UNPARSED (120). The 'DIAFRAME' tab is selected, showing a table of visit records. The table has columns for patient_id, visit_date, age, sex, A1c, A1c_prev, A1c_delta, FBS, eGFR, UACR, BMI, SBP, DBP, and current_drug_cls. Below the table are buttons for 'Download SQLite DB', 'Download Research Bundle (ZIP)', and 'Download Excel Bundle'.

patient_id	visit_date	age	sex	A1c	A1c_prev	A1c_delta	FBS	eGFR	UACR	BMI	SBP	DBP	current_drug_cls
P001	2022-06-27	73	F	9.9			278			28.6	141	81	["Insulin"]
P001	2022-09-26	73	F	10.5	9.9	0.6	96			27.8			["Insulin"]
P001	2022-12-26	73	F	7.5	10.5	-3	96			30.1	119	78	["Insulin"]
P001	2023-03-27	73	F	8	7.5	0.5	96			29.4	142	75	["Insulin"]
P001	2023-06-26	73	F	8	8	0	96			29.9	128	80	["Insulin"]
P001	2023-09-25	73	F	7.8	8	-0.2	99			28.6	119	64	["Insulin"]
P001	2023-12-18	73	F	7	7.8	-0.8	105			29.9	140	73	["Insulin"]
P001	2024-03-18	73	F	7.4	7	0.4	105			29.8	134	87	["DPP4"], ["Insulir"]
P001	2024-04-16	73	F	7.4	7.4	0	105						["DPP4"], ["Insulir"]

방문 단위로 분리된 외래 기록

외래 EMR은 시간의 기록입니다. 한 환자는 여러 번 병원에 오고, 각 방문마다 증상, 검사, 약물, 계획이 조금씩 달라집니다. 그래서 CleanEMR에서 가장 중요했던 첫 단계는 원문 기록을 방문 날짜를 기준으로 나누는 일이었습니다. 그 후 각 방문 안에서 Subjective, Objective, Assessment, Plan 등의 SOAP 구조와 검사, 활력징후, 약물 정보를 추출합니다.

모든 기록이 완벽하게 정리되어 있지는 않습니다. 어떤 기록은 구조가 비교적 분명하지만, 어떤 기록은 줄글에 가깝거나 약어와 검사 결과가 뒤섞여 있습니다. 하지만 연구를 위해서는 적어도 어떤 정보

가 어느 방문에 속하는지 명확히 구분해야 합니다. 검사값 하나도 어느 날짜의 것인지 모르면 그 의미가 크게 줄어들기 때문입니다. 언제부터 어떤 약이 추가되었고, 어떤 방문에서 치료 방향이 바뀌었는지 알아야 비로소 실제 의사결정의 흐름을 파악할 수 있습니다.

3) 검사와 약물을 연구 변수로 꺼내다

연구에서 중요한 것은 반복해서 비교할 수 있는 변수입니다. 당뇨 외래의 경우 HbA1c, FBS, PP2, eGFR, creatinine, AST, ALT, LDL, HDL, TG, total cholesterol, UACR 같은 검사값과 혈압, 맥박, 체중, 키, BMI 등 활력징후가 중요하게 작용합니다.

여기에 더해 어떤 약을 사용 중인지, 어떤 약제 계열이 추가되거나 중단되었는지 파악하는 것도 필수적입니다. Metformin, SGLT2 inhibitor, DPP-4 inhibitor, GLP-1 receptor agonist, Sulfonylurea, TZD, Insulin, Statin, ACE inhibitor, ARB 등의 정보들은 EMR 원문 안에 흩어져 있습니다. 어떤 것은 결과 표에, 어떤 것은 외래 기록 본문에, 어떤 것은 처방 기록에 위치합니다.

CleanEMR은 이런 흩어진 정보들을 환자와 방문에 연결된 연구 변수로 꺼냅니다. 여기서 핵심은 단순 추출이 아닙니다. 각 값이 어느 환자의 어느 방문에 속하는지, 나아가 원문에서 어떤 텍스트를 근거로 파싱되었는지 추적할 수 있어야 나중에 오류를 검증할 수 있습니다. 좋은 연구 데이터는 값만 존재하는 것이 아니라, 그 값이 어디서 부터 왔는지 확인할 수 있는 데이터입니다.

4) 결정론적 파싱을 선택한 이유

CleanEMR에서는 AI 모델을 바로 붙이는 대신 정규표현식과 dictionary 기반 매칭, 즉 규칙 기반 파싱을 기본 구조로 잡았습니다. 이유는 단순합니다. 연구용 데이터셋은 무엇보다 재현 가능해야 하기 때문입니다.

같은 원문을 넣었을 때 어제와 오늘의 결과가 달라지면 곤란합니다. 모델의 상태나 프롬프트의 미세한 차이에 따라 HbA1c가 다르게 추출되거나 약물 계열이 다르게 분류된다면, 그 데이터로는 신뢰할 수 있는 연구를 진행하기 어렵습니다. 그래서 CleanEMR은 가능한 한 동일한 입력에서 동일한 출력을 만들도록 설계되었습니다.

물론 규칙 기반 파싱이 완벽한 것은 아닙니다. 표현이 너무 다양하거나 문장이 복잡한 경우, 기록 형식이 예상과 다른 경우에는 파싱하지 못하는 항목이 생깁니다. 하지만 적어도 왜 그렇게 추출되었는지 그 근거를 명확히 확인할 수 있습니다. 의료 연구용 전처리 단계에서는 이 투명성이 예측 불가능한 모델의 유연성보다 훨씬 중요합니다.

5) 파싱되지 않은 줄도 버리지 않는다

patient_id	visit_date	source_section	raw_line	reason	created_at
P001	2026-04-25	P	(진료전)Hemoglobin A1c (당뇨검사) [W...	lab_order_ignored	2026-06-03T01:11:08.4E
P001	2025-12-31	S	인슐린 이25개 남음	medication_unparsed	2026-06-03T01:11:08.47
P001	2025-12-31	P	(진료전)Hemoglobin A1c (당뇨검사) [W...	lab_order_ignored	2026-06-03T01:11:08.47
P001	2025-09-24	P	(진료전)Hemoglobin A1c (당뇨검사) [W...	lab_order_ignored	2026-06-03T01:11:08.47
P001	2025-06-25	P	(진료전)Hemoglobin A1c (당뇨검사) [W...	lab_order_ignored	2026-06-03T01:11:08.4E
P001	2025-03-25	P	(진료전)Hemoglobin A1c (당뇨검사) [W...	lab_order_ignored	2026-06-03T01:11:08.4E
P001	2024-12-19	P	(진료전)Hemoglobin A1c (당뇨검사) [W...	lab_order_ignored	2026-06-03T01:11:08.50
P001	2024-09-19	P	(진료전)Hemoglobin A1c (당뇨검사) [W...	lab_order_ignored	2026-06-03T01:11:08.50
P001	2024-06-17	P	(진료전)Hemoglobin A1c (당뇨검사) [W...	lab_order_ignored	2026-06-03T01:11:08.51

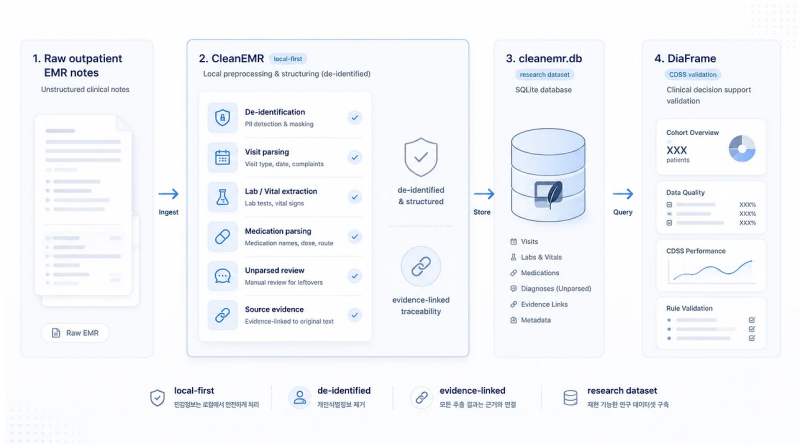
수동 검토가 필요한 문장 목록

자동 파싱에서 중요한 것은 추출에 성공한 값만 쳐다보는 것이 아니라, 시스템이 읽어내지 못한 줄도 살펴보는 것입니다. 의료 기록은 형식이 일정하지 않아 같은 약이나 검사라도 이름이 다르게 적히거나 예상 밖의 위치에 기록될 수 있습니다.

그래서 CleanEMR은 정형 데이터로 변환되지 못한 모호한 문장을 별도의 테이블에 남겨둡니다. 이것은 시스템의 실패 목록이 아니라 수동 검토 목록입니다. 자동화가 조용히 틀린 값을 내놓는 것보다, 불확실한 부분을 솔직하게 드러내는 편이 낫습니다. 의료 데이터에서는 모르는 것을 모른다고 표시하는 것이 매우 중요합니다. 파싱되지 않은 줄을 따로 모아두면 파서의 한계를 파악하고 다음 버전에 어떤 규칙을 보완해야 할지 알 수 있습니다. 자동화와 수동 검토 사

이의 안전한 경계를 만드는 과정입니다.

6) DiaFrame으로 넘기기 위한 데이터베이스



CleanEMR과 DiaFrame의 연결 구조

CleanEMR의 중요한 목적 중 하나는 DiaFrame(당뇨병 약제 추천을 평가하기 위한 프로토타입)과 연결되는 것입니다. DiaFrame이 제대로 작동하려면 환자의 혈당 상태, 신기능, 체중, 현재 사용 중인 당뇨약 계열, 그리고 다음 방문에서 처방이 강화되었는지 완화되었는지에 대한 실제 방향 등이 명확히 정리되어 있어야 합니다. 이 정보가 갖춰져야만 비로소 AI 추천과 실제 처방을 비교할 수 있습니다

CleanEMR은 이 downstream 분석을 위해 cleanemr.db라는 구조화된 데이터베이스를 내보냅니다. 이 데이터베이스는 단순한 요약본이 아니라, 진단이나 약물 추천의 바로 앞단에서 실제 기록을 연구

가능한 입력 형태로 바꾸는 중간 데이터 계층 역할을 합니다.

7) 연구 가능한 형태로 바꾼다는 것

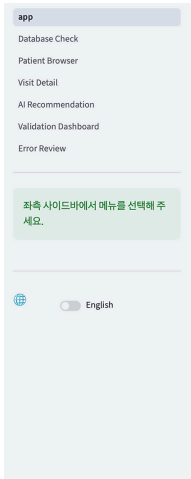
연구는 질문에서 시작하지만, 질문에 답하려면 변수를 꺼낼 수 있는 일정한 구조가 필요합니다. CleanEMR은 원문 외래 기록을 로컬 환경에서 정규화 및 비식별화하고, 방문 단위로 나눈 뒤 검사와 약물을 파싱하여 원문의 근거를 남기는 전체 파이프라인입니다. 애매한 문장은 별도 리뷰 대상으로 넘기고 결과를 SQLite, CSV, Excel 등으로 내보냅니다.

이 지저분한 정리 작업은 전혀 화려하지 않지만, 의료를 구조화하려면 반드시 거쳐야 하는 단계입니다. 데이터의 시작은 AI 모델이 아니라 바로 이 구조화에 있으며, CleanEMR은 그 기반을 다지기 위한 첫 번째 계층이었습니다.

5. 머신러닝 기반 당뇨 약물 추천, DiaFrame

AI에게 환자 정보를 주고 약제를 추천하게 하는 것은 생각보다 어렵지 않습니다. 문제는 그다음입니다. 그 추천이 맞는지 어떻게 볼 것인가, 어떤 기준으로 평가할 것인가, 그리고 실제 의사의 처방과 다를 때 그 차이를 어떻게 해석할 것인가.

의료 AI에서 중요한 것은 답을 내는 능력만이 아니라, 그 답을 검증할 수 있는 구조입니다. DiaFrame은 바로 이 질문에서 시작했습니다.



DiaFrame: 내분비내과 임상 의사결정지원 시스템(CDSS) 프로토타입

개요

DiaFrame은 CleanEMR을 통해 구조화된 당뇨병 환자 데이터를 기반으로 AI 추천 약제 후보를 도출하고, 이를 실제 임상 현장의

* 본 시스템은 원시 EMR(전자의무기록) 텍스트를 직접 파싱하지 않으며, 어떠한 최종 임상 결정도 내리지 않습니다.

```
---  
### 🏠 데이터 처리 파이프라인  
**CleanEMR (데이터 전처리)** → `cleanemr.db` → **DiaFrame (본 시스템)** → **AI 약제 추천 및 결과 검증**  
---
```

```
### ⚠️ 연بة 조항  
**DiaFrame은 실제 처방을 의뢰하거나 진단 보조 도구가 아닙니다.**
```

본 시스템은 AI 모델이 제시하는 약제 클래스 후보군을 과거 처방 패턴과 대조하여 성능을 평가하기 위한 연구 목적의 프로토타입입니다. 실제 환자의 임상적 판단이나 처방 결정에 절대 사용해서는 안 됩니다.

```
### 🛡️ 개인정보 보호  
DiaFrame은 **비식별화 처리된 구조화 데이터베이스**만을 취급합니다. 본 애플리케이션의 어떠한 과정에서도 원시 EMR 텍스트나
```

DiaFrame 메인화면

레포: github.com/jsbang01357/DiaFrame **개발 기간:** 2026년 6

월 **형태:** 당뇨병 약제 추천 평가용 내분비 CDSS 프로토타입 **스택:**

Python, Streamlit, SQLite, Vertex AI Gemini, Pandas **입력:** Cl

eanEMR에서 생성된 비식별화 구조화 데이터베이스 cleanemr.db

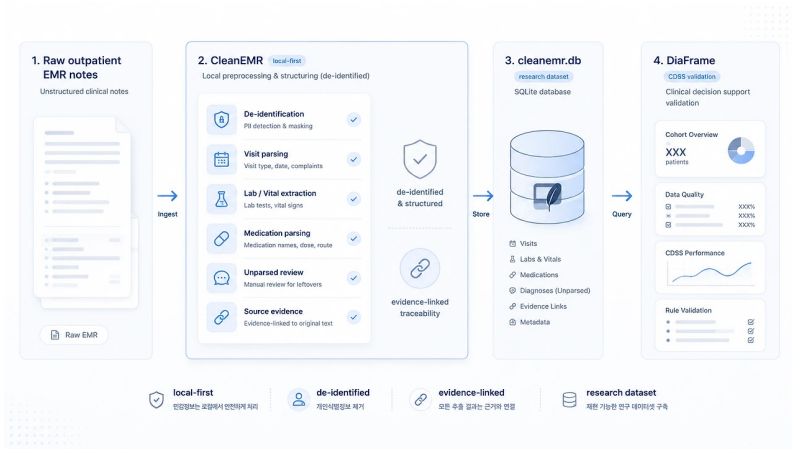
핵심 구조: CleanEMR → diaframe_visits → AI 추천 JSON → Sa

fety Rule Layer → 실제 처방 방향과 비교 → 오류 리뷰 **주의:** 실

제 처방을 내리기 위한 시스템이 아니라, 과거 처방 패턴과 AI 추천

을 비교·평가하기 위한 연구용 프로토타입입니다.

1) DiaFrame은 EMR 파서가 아니다



Data Pipeline 구조도

DiaFrame은 EMR 원문을 읽고 파싱하거나 자유 텍스트를 구조화하는 도구가 아닙니다. 그 역할은 앞서 다른 CleanEMR이 수행합니다. DiaFrame은 그다음 단계, 즉 cleanemr.db 안의 구조화된 데이터를 읽어 들이는 계층입니다.

환자 ID, 방문일, 나이와 성별, A1c, FBS, eGFR, BMI 같은 임상 지표, 현재 사용 중인 당뇨약 계열, 수동 검토 필요 여부 등이 명확히 정리되어 있어야 AI 추천을 평가할 수 있습니다. 즉 DiaFrame은 텍스트를 변환하는 과정이 아니라, 이미 구조화된 데이터를 바탕으로 AI의 추천과 실제 처방을 비교하는 검증 계층입니다.

2) 추천을 자유롭게 말하게 두지 않는다

AI 약제 추천 및 안전성 검증

대상 환자 ID: P001 | 방문 일자(Visit Date)를 선택해 주세요: 2022-06-27

[AI 입력 컨텍스트 (Input Context)]

```
{
  "age": 73
  "sex": "F"
  "A1c": 9.9
  "A1c_previous": NULL
  "A1c_delta": NULL
  "FBS": 278
  "eGFR": NULL
  "UACR": NaN
  "BMI": 28.6
  "SBP": 141
  "DBP": 81
  "current_drug_classes": [
    "Insulin"
  ]
  "perioperative_context": false
  "manual_review_required": false
}
```

실행 모드 선택

- Mock 모드 (로컬 테스트용 Rule-based)
 Vertex AI (Gemini 클라우드 모델)

AI 약제 추천 생성하기

AI 추천 결과 (Output)

처방 방향 (Direction)

INTENSIFY

Top 3 추천 약제

SGLT2i, GLP1RA, DPP4i

추천 근거 (Rationale):

- A1c is ≥ 8.0 , intensification needed.
- SGLT2i is not currently prescribed.

AI 자체 경고 (Model Warnings):

- Check eGFR before starting SGLT2i.

구조화된 AI 추천 출력


의료 AI에서 가장 위험한 것은 그럴듯하게 자유롭게 말하도록 두는 것입니다. 일반적인 대화형 AI는 부드럽고 긴 설명을 잘 만들어내지

만, 의료 현장에서는 그 유연성이 오히려 혼란과 위험을 초래할 수 있습니다. 추천 방향이 애매해지거나, 약제명이 임의로 등장하거나, 금기 사항을 놓치면 결과 자체를 평가하기가 불가능해집니다.

그래서 DiaFrame에서는 AI의 출력 형식을 엄격히 제한했습니다. AI는 유지, 강화, 완화, 변경 중 하나의 처방 방향을 명확히 제시해야 하고, 허용된 목록 안에서 상위 3개의 약제 계열을 골라내야 합니다. 또한 추천 근거와 경고 사항을 정해진 JSON 구조 안에 담도록 강제했습니다. 이렇게 해야 AI가 무엇을 제안했는지, 실제 처방과 어느 부분이 일치하고 어느 부분이 다른지 정량적으로 비교할 수 있습니다. 평가 가능한 형태로 입을 단게 만드는 것이, 모델을 제대로 평가하기 위한 첫걸음입니다.

3) 안전 규칙은 모델 밖에 따로 둔다

임상 안전성 검증 레이어 (Safety Rule Layer)

 [주의] 안전성 경고가 발생했습니다.

- eGFR missing; renal safety cannot be fully assessed

추천 결과 검증 (실제 임상 처방 vs AI 추천)

 실제 처방 방향 (Actual Direction):

MAINTAIN

처방 방향 일치

✗ 불일치

Top 3 약제 일치

✗ 불일치

 현재 추천 및 검증 결과를 DB에 저장

안전성 위반 경고 예시

아무리 잘 훈련된 AI라도 추천을 전적으로 믿을 수는 없습니다. Dia


Frame에는 AI의 추천 결과를 한 번 더 걸러내는 별도의 Safety Rule Layer가 존재합니다.

이 레이어는 AI가 eGFR이 낮은 환자에게 Metformin을 추천하지는 않았는지, 고령 환자에게 저혈당 위험이 큰 SU나 Insulin을 무리하게 강권하지는 않았는지 등을 시스템 밖에서 기계적으로 확인합니다. 모델이 알아서 조심해주기를 기대하기보다, 명백히 위험한 출력을 확실하게 막아내는 최소한의 안전망을 덧대는 방식입니다. 모델이 답을 내고, 규칙이 그 답을 다시 검사하는 이 이중 구조가 있어야 비로소 그 추천을 연구의 대상으로 다룰 수 있습니다.

4) 실제 처방과 비교해야 한다

종합 검증 대시보드 (Batch Validation)

최종 평가 가능 건수: 55 / 73

 Batch 검증 실행 모델

- mock
- vertex

 전체 Batch 검증 실행

종합 성능 지표 (Aggregate Metrics)

총 평가 건수 (N)

55

처방 방향 일치

40.0%

Top-1 약제 정확도

40.0%

안전성 위반율

0.0%

상세 검증 결과 데이터

	patient_id	visit_date	ai_direction	actual_direction	top1_class	top3_drug_classes			class_added	class_re
0	None	None	intensify	maintain	SGLT2i	SGLT2i	GLP1RA	DPP4i		
1	None	None	intensify	maintain	SGLT2i	SGLT2i	GLP1RA	DPP4i		
2	None	None	maintain	maintain	None	None				
3	None	None	intensify	maintain	SGLT2i	SGLT2i	GLP1RA	DPP4i		
4	None	None	intensify	maintain	SGLT2i	SGLT2i	GLP1RA	DPP4i		
5	None	None	maintain	maintain	None	None				
6	None	None	maintain	intensify	None	None		DPP4i		
7	None	None	maintain	maintain	None	None				
8	None	None	maintain	maintain	None	None				
9	None	None	intensify	maintain	SGLT2i	SGLT2i	GLP1RA	DPP4i		

추천 방향 일치율, Top-1/Top-3 일치율, Safety Violation Rate

검증의 기준은 AI의 추천이 얼마나 '그렇듯해 보이는가'가 아닙니다.

실제 임상 기록에서 의사가 어떤 처방을 선택했는지와의 비교가 중요합니다. AI가 치료 강화를 제안했을 때 실제 의사도 처방을 강화했는지, AI가 고른 약제 계열이 실제 추가된 약제와 일치하는지 등을 처방 방향 일치율과 Top-1/Top-3 일치율로 확인합니다.

만약 추천 결과가 실제 처방과 비슷하더라도 앞선 안전성 규칙을 위반했다면 그 추천은 실패로 간주됩니다. 이런 객관적인 지표들이 있어야 AI의 대답은 단순한 텍스트 덩어리를 넘어 검증 가능한 결과로 변모합니다.


5) 불일치가 더 중요할 때도 있다

개별 케이스 심층 검토

분석할 불일치 케이스를 선택해 주세요:

Patient: None | Date: None

AI 모델 추천 결과

- 처방 방향 (Direction): `maintain`
- Top 1 Class: `None`
-  AI 자체 경고 (Model Warnings):: []

실제 EMR 처방 데이터

- 처방 방향 (Direction): `deintensify`
-  추가된 약제 클래스: []
-  중단된 약제 클래스: ['DPP4i']
- 수동 검토(Manual Review) 대상 여부: `nan`

 시스템 추정 불일치 원인(Error Category): `manual_review_required, perioperative_context, guideline_discordance`

 오류 검토 리포트 CSV 다운로드

AI 추천과 실제 처방 간 불일치 케이스 리뷰

가끔은 일치율보다 AI와 실제 처방이 어긋나는 순간의 차이를 들여다보는 일이 더 중요합니다. AI가 놓친 환자의 맥락이 있었는지, 의사가 지나치게 보수적이거나 공격적으로 판단했는지, 아니면 단순히 가이드라인 해석의 차이인지 고민하게 만듭니다.

불일치는 모델의 실패일 수도 있지만, 임상 의사결정의 숨겨진 구조를 이해하기 위한 분석의 출발점이 되기도 합니다. 그래서 DiaFrame에는 불일치 케이스를 따로 모아 검토하는 기능이 포함되어 있습니

다. AI를 무작정 믿기 위한 도구가 아니라, AI와 의사의 차이를 통해 판단의 근거를 되짚어보는 돋보기에 가깝습니다.

6) 연구용 프로토타입이라는 경계

다시 한 번 강조하지만, DiaFrame은 환자의 실제 임상 결정을 내리는 처방용 시스템이 아닙니다. 비식별화된 과거 데이터를 바탕으로 추천과 검증 구조를 테스트해보는 연구용 도구입니다.

이 프로젝트의 핵심은 약제를 직접 추천하는 기능 자체가 아니라, 어떤 추천이 나왔을 때 그것을 안전 규칙으로 거르고, 실제 과거 처방과 비교하며, 불일치 사례를 면밀히 검토해 나가는 전체 파이프라인의 구축에 있습니다.

의료 AI의 진정한 난관은 답을 내는 순간이 아니라, 그 답을 어떻게 검증할 것인가를 고민하는 시점에서 시작됩니다. DiaFrame은 그 검증의 구조를 시각화하기 위해 만들어진 프로토타입입니다. AI의 추천을 믿기 전에, 우리는 그것을 먼저 검증할 수 있어야 합니다.

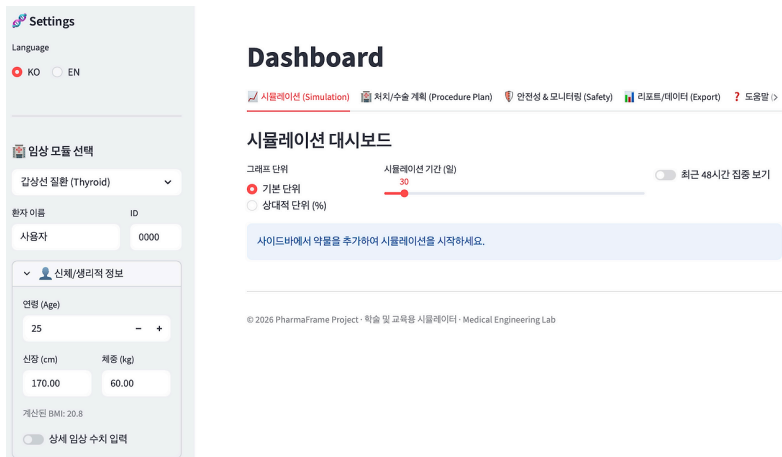
Part 5. 약물은 점이 아니라 곡선이다

1. 다학제 약제 PK 분석, PharmaFrame

약은 보통 점처럼 기록됩니다. 몇 mg, 하루 몇 번, 며칠 동안, 식전

또는 식후, 혹은 몇 주 뒤 증량. 처방전에는 늘 단편적인 점으로 적힙니다.

하지만 몸이 경험하는 약물은 하나의 점이 아닙니다. 약을 먹거나 맞는 그 짧은 순간은 사건에 불과하지만, 그 뒤 몸 안에서 약물 농도는 끊임없이 움직입니다. 약물은 서서히 흡수되고, 온몸으로 분포하며, 간이나 신장을 거쳐 대사되고, 마침내 배설됩니다. 처방은 용량과 간격으로 쓰이지만, 몸은 시간에 따른 농도 곡선을 통째로 경험합니다. PharmaFrame은 바로 이 관점에서 시작한 프로젝트입니다.



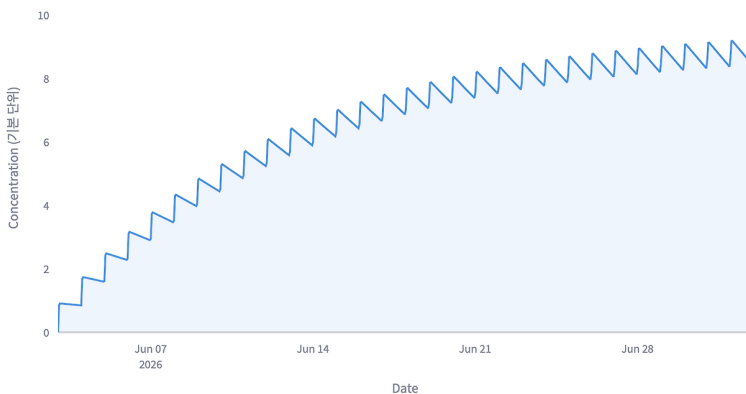
PharmaFrame 첫 화면

레포: github.com/jsbang01357/PharmaFrame **개발 기간:** 2026년 5월 15일~ **형태:** 학술·교육용 개인화 약동학 시뮬레이터 **스택:** Python, Streamlit, Plotly, YAML, ReportLab, iCalendar **지원 모듈:** 갑상선, 폐경 후 호르몬 요법, opioids, NSAIDs, 사춘기 유도, 심혈관 약물 등 **핵심 구조:** 약물 PK 데이터베이스 → 환자 프로필

→ 복약 일정 → 시간-농도 곡선 → 시나리오 비교 → 수술/시술 전
 후 중단 계획 → 리포트/캘린더 내보내기 **주의:** 실제 처방 조정을
 지시하는 의료기기가 아닙니다. 약동학적 사고 과정을 시각화하기
 위한 학술·교육용 프로토타입입니다.

1) 처방은 사건이고, 농도는 함수다

Pharmacokinetics Simulation



예상 약동학(PK) 지표

안정 상태(Steady State) 도달 이후의 예상 농도입니다.

최고 농도 (Peak)	최저 농도 (Trough)	평균 농도 (Avg)	변동폭 (Fluctuation)
9.7	8.8	9.3	9.1%

시간-농도 곡선 위의 *peak*와 *trough*

약을 복용하거나 주사하는 순간은 하나의 사건입니다. “오전 8시에
 복용했다”, “일주일에 한 번 주사했다”, “수술 며칠 전부터 중단했다

” 같은 사건들은 EMR에 기록하기 쉽습니다. 하지만 그 사건 이후에 몸 안에서 일어나는 일은 시간이라는 연속된 축 위에서 펼쳐집니다

혈중 농도는 서서히 올라가 최고점에 도달한 뒤 다시 감소합니다. 투여를 반복하면, 이전 투여로 인해 남아 있던 잔여 농도 위에 새로운 투약분이 겹쳐 쌓입니다. 그래서 약물은 점이 아니라 곡선으로 파악해야 합니다. Peak, Trough, Average concentration, Fluctuation index와 같은 지표들은 각각 동떨어진 숫자가 아니라, 시간-농도 곡선 위의 특정한 지점을 가리키는 이름표일 뿐입니다. 처방은 점으로 쓰여도 약동학은 곡선으로 이해해야 합니다.

2) 같은 용량이어도 같은 곡선이 아니다

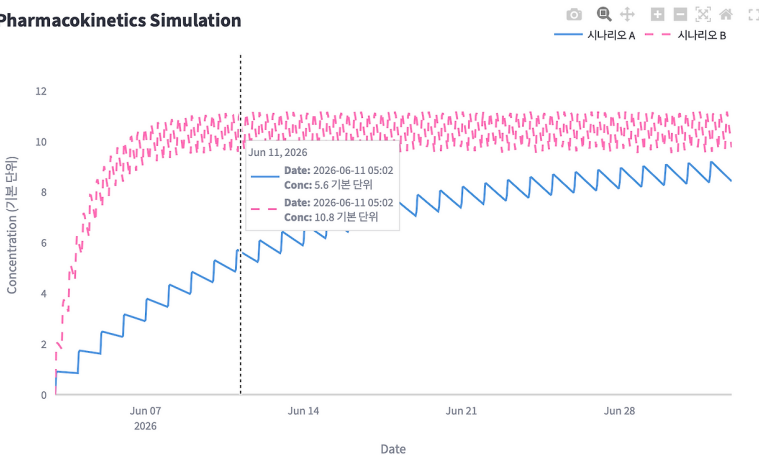
같은 약을 같은 용량으로 쓴다고 해서 모든 사람이 동일한 농도 곡선을 경험하지는 않습니다. 나이가 다르고, 체중이 다르고, 신장과 체지방률이 다르기 때문입니다. 더 근본적으로는 신기능(eGFR)과 간기능(AST, ALT 등)이 환자마다 천차만별입니다.

약물에 따라 신장 배설 비율이나 간 대사 비율도 모두 다릅니다. eGFR이 떨어져 있다면 신장으로 주로 배설되는 약물의 소실 속도가 현저히 느려질 수 있습니다. 간 수치가 높다면 간에서 대사되는 약물을 쓸 때 조심해야 하고, BMI와 체지방률은 약물의 분포용적 자체에 영향을 줍니다. 이런 임상 수치들은 단순한 환자 정보가 아니라, 약물 농도 곡선의 형태를 빚어내는 중요한 변수입니다.

PharmaFrame은 환자의 임상 지표를 약동학 파라미터에 적극적으로 반영하려 했습니다. 같은 용량이라도 환자가 달라지면 곡선도 달라져야 합니다. 약물을 환자와 동떨어진 고정값으로 보지 않고, 환자의 신체 조건과 유기적으로 연결된 동적인 모델로 바라보기 위함입니다.

3) 약물 조정은 곡선의 형태를 바꾸는 일이다

Pharmacokinetics Simulation



예상 약동학(PK) 지표

안정 상태(Steady State) 도달 이후의 예상 농도입니다.

최고 농도 (Peak)	최저 농도 (Trough)	평균 농도 (Avg)	변동폭 (Fluctuation)
9.7	8.8	9.3	9.1%

시나리오 B 지표

최고 농도 (Peak)	최저 농도 (Trough)	평균 농도 (Avg)	변동폭
11.2	9.5	10.4	15.7%
↑ 1.5	↑ 0.7	↑ 1.2	↑ 6.7%

Liothyronine(T3)과 Levothyroxine(T4) 복약일정에 따른 농도 곡선 비교

임상 현장에서 의사들은 끊임없이 약을 조정합니다. 용량을 올리거나 간격을 좁히고, 때로는 투약을 중단하거나 약제 자체를 바꿉니다.

겉으로 보면 단순한 처방 변경이지만, 약동학적으로 들여다보면 이 모든 행위는 농도 곡선의 형태를 빚고 변형하는 조각 작업과 같습니다.

용량을 올리면 곡선 전체가 위로 떠오르고, 투여 간격을 좁히면 최저농도(trough)가 높아지며 혈중 농도의 진동폭(fluctuation)이 줄어듭니다. 반대로 투여 간격을 늘리면 최고농도(peak)와 최저농도의 격차가 심해집니다. 약을 아예 중단하면, 그 약의 고유한 반감기와 환자의 청소율에 맞춰 농도가 긴 꼬리를 그리며 서서히 떨어집니다. 이렇게 보면 약물 조정은 “증량”이나 “감량”이라는 낯잡한 단어로 요약될 수 없습니다. 환자의 몸이 겪게 될 농도 곡선의 높이, 폭, 진동의 주기, 그리고 하강 속도를 정교하게 바꾸는 행위입니다. Pharma Frame을 통해 시각적으로 확인하고 싶었던 것은 처방 변경이 만들어내는 이 구체적인 곡선의 변화였습니다.

4) 복용 누락도 하나의 시나리오다

특정 투여일 조정 (누락/지연)

특정 회차의 투여를 건너뛰거나 지연시킬 수 있습니다.

적용 대상 약물: Levothyroxine (1...
조정할 회차 (N번째): 5
상태 변경: 누락 (Missed)

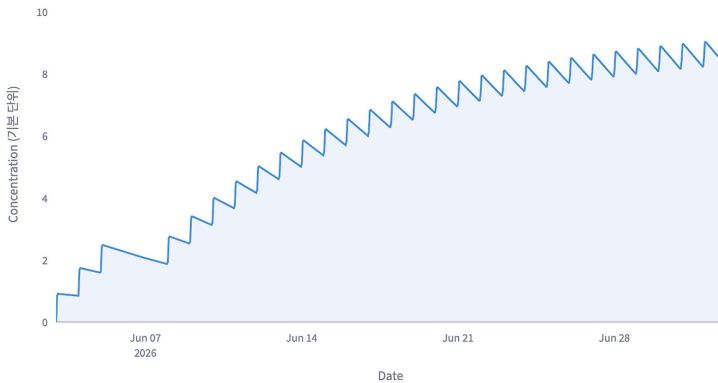
적용

현재 적용된 예외:

Levothyroxine 4회차: 누락 삭제

Levothyroxine 5회차: 누락 삭제

Pharmacokinetics Simulation



복약 누락과 지연 시뮬레이션

현실의 복약 패턴은 결코 처방전처럼 단정하지 않습니다. 바빠서 약을 늦게 먹기도 하고, 깜빡 잊어 한 회차를 통째로 건너뛰기도 하며, 임의로 복약 일정을 바꾸기도 합니다. 이런 일들은 외래에서 흔히 듣는 이야기지만, 약동학적으로는 환자 몸속의 곡선 형태를 크게 흔들어 놓습니다.

단 한 번의 복약 누락만으로도 다음 최저농도(trough)가 예상보다

흡착 낮아질 수 있고, 복용 지연은 최고농도(peak)의 도달 시점을 한참 뒤로 밀어버립니다. 투약 일정이 불규칙해지면, 이전 투여로 쌓인 잔여 농도와 새로운 투약분이 설계와는 완전히 다른 방식으로 겹쳐지게 됩니다.

PharmaFrame에 특정 회차의 투여를 임의로 늦추거나 건너뛸 수 있는 'override' 기능을 넣은 이유가 여기에 있습니다. 환자의 복약 순응도를 단순히 '잘 먹었다', '못 먹었다'는 이분법으로 재단하는 대신, 그것이 시간-농도 곡선에 어떤 변형을 가하는지 직관적으로 보기 위해서입니다. 이 관점에서는 복약 누락 역시 무시할 수 없는 중요한 입력 데이터이며, 그 누락이 환자의 다음 상태에 미칠 파장을 미리 가늠해볼 수 있게 해줍니다.

5) 수술 전 약을 끊는다는 것

처치/수술 계획 (Procedure Plan)

예정된 시술이나 수술을 위해 특정 기간 동안 약을 투여를 중단해야 할 경우 시뮬레이션합니다.

수술/처치 모드 활성화

일정 설정

수술/시술 예정일 수술 전 중단일 (일) ② 수술 후 재개일 (일) ②

2026/06/24 7 - + 7 - +

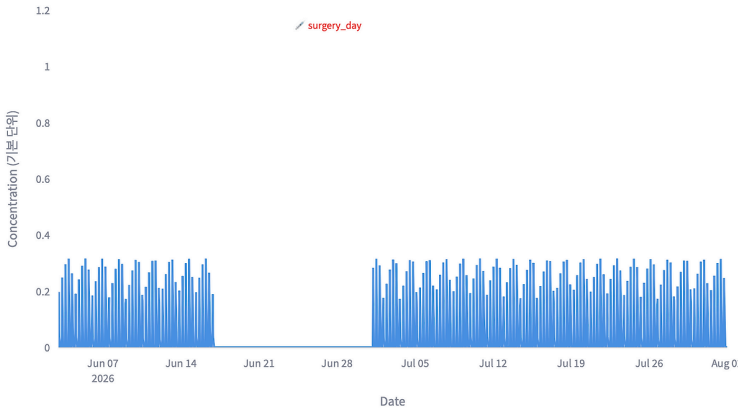
권장 중단 기간: 2026-06-17 ~ 2026-07-01

중단 기간 반영 약동학 그래프

조희 기간 (일)



Pharmacokinetics Simulation



수술/시술 모드에서는 중단 기간 내의 투여 일정이 시뮬레이션 상에서 모두 '누락' 처리됩니다.

수술 전후 약물 중단 시뮬레이션

수술이나 시술을 앞두고 가장 흔히 던지는 지시 중 하나가 “며칠 전 부터 드시던 약을 끊으세요”라는 말입니다. 어떤 약은 출혈 위험 때

문에, 어떤 약은 혈전이나 감염 위험, 혹은 대사적 변동성 때문에 투약을 일시적으로 중단해야 합니다. 하지만 이 짧은 안내 문장 뒤에는 생각보다 복잡한 시간의 퍼즐이 숨어 있습니다.

해당 약물의 반감기는 얼마나 긴지, 오랫동안 복용해서 몸에 얼마나 누적되어 있는지, 중단 후 실제 혈중 농도는 어떤 속도로 떨어질지, 그리고 대망의 수술 당일에는 몸속에 잔여 약물이 얼마나 남아 있을지. 나아가 수술이 끝나고 나면 언제 다시 안전하게 약을 재개할 수 있을지까지 모두 시간의 지배를 받는 질문들입니다.

PharmaFrame의 수술 전후 중단 시뮬레이션(Procedure/Surgery Cessation Planner)은 바로 이 지점을 파고듭니다. 수술 전 중단 기간 동안의 투여를 누락(missed) 처리했을 때, 수술 당일과 전후의 잔류 농도 추이가 어떻게 바닥을 향해 내려가는지 그래프로 보여줍니다. 수술 전 약물 중단을 달력의 날짜를 세는 일이 아니라, 농도 곡선이 안전선 아래로 미끄러져 내려가는 물리적 과정으로 이해하려는 시도입니다.

6) 여러 약물군으로 확장하기

처음에는 호르몬 치료를 관찰하며 곡선의 개념에 깊이 천착했습니다. 호르몬 수치는 외래에서 단 한 번의 채혈 값으로 기록되지만, 그 이면에 흐르는 실제 체내 농도는 주사 시점과 채혈 시점의 간격에 따라 완전히 다른 값을 가리킵니다. 같은 환자라도 주사 직후의 최고점과 주기 말의 최저점은 하늘과 땅 차이입니다. EstroFrame과 A

ndroFrame은 호르몬의 이런 역동적인 변화를 곡선으로 잡아내려는 프로젝트였습니다.

하지만 약을 곡선으로 보는 이 관점은 비단 호르몬에만 국한되지 않습니다. 갑상선 호르몬, 폐경 후 호르몬 요법, 강력한 진통제인 Opioids와 NSAIDs, 사춘기 유도 호르몬, 그리고 미세한 용량 조절이 생명인 심혈관계 약물들까지 수많은 약들이 시간의 흐름을 타며 흡수, 분포, 대사, 배설의 과정을 겪습니다.

그래서 특정 호르몬의 시뮬레이션을 넘어, 더 보편적이고 일반적인 약물 모델링 프레임워크로 시야를 넓히고자 만든 것이 PharmaFrame입니다. 특정 약물 하나를 정밀하게 묘사하는 데 그치지 않고, 다양한 약물 데이터베이스와 환자의 개별 프로필, 그리고 복잡한 복약 일정이라는 세 축을 단일한 약동학 구조 안에서 매끄럽게 연결하고 싶었습니다.

7) 약물은 판단의 시간축이다

처방전 종이 위에 인쇄된 약물은 정적입니다. 용량과 투여 간격이 무심하게 적혀 있을 뿐입니다. 그러나 그 약물이 환자의 몸 안으로 들어오는 순간부터 약은 동적인 생명력을 얻습니다. 임상 현장에서 의사가 내리는 모든 판단에는 필연적으로 '시간'이라는 축이 개입합니다.

언제쯤 약효가 나타날 것인가, 언제 부작용 발생 확률이 정점을 찍을 것인가, 며칠 뒤에 채혈해야 정확한 상태를 파악할 수 있는가, 수

술 며칠 전에 중단해야 수술 당일 안전한 농도에 도달할 수 있는가, 그리고 수술 후 언제 다시 투여를 시작해야 기저질환의 악화를 막을 수 있는가.

약물은 결국 시간 위에서 저울질하고 판단해야 하는 대상입니다. PharmaFrame은 숨겨져 있던 이 시간의 축을 눈앞으로 끌어내기 위해 고안된 도구입니다. 약을 고정된 점으로 박제하지 않고 흐르는 곡선으로 바라보는 것. 처방을 단발적인 사건이 아니라 몸속에서 끊임없이 박동하는 화학적 파동으로 이해하는 것.

약을 조절한다는 것은 단순히 숫자를 늘리고 줄이는 기계적인 작업이 아닙니다. 환자가 견뎌야 할 농도 곡선의 거친 형태를 부드럽게 다듬어가는 일입니다. 그 보이지 않는 곡선의 존재를 이해하고 상상할 수 있게 되는 순간, 약물은 더 이상 처방전 위의 숫자가 아니라 시간 위를 살아 움직이는 역동적인 모델로 다가옵니다.

2. 수술 전 약을 끊는다는 것

“수술 며칠 전부터 약을 끊으세요.”

이 문장은 매우 단순해 보입니다. 특정 날짜를 정해서 약 복용을 중단하고 수술이 끝난 뒤에 다시 시작하면 될 것처럼 들립니다. 하지만 몸 안에서 일어나는 일은 그렇게 칼로 무 자르듯 단순하지 않습니다.

약을 끊는다는 것은 조명 스위치를 끄듯 즉각적인 결과를 낳는 일이 아닙니다. 마지막 투여 이후에도 몸 안에는 약물이 남아 있으며, 농

도는 바로 0이 되지 않습니다. 약물의 고유한 반감기, 환자의 청소율, 분포용적, 투여 경로, 그리고 오랜 기간 반복 투여로 쌓인 누적 치에 따라 혈중 농도는 서서히, 각기 다른 속도로 내려갑니다.

결국 수술 전 약물 중단은 단순한 날짜 계산의 문제가 아닙니다. 시간에 따른 농도 곡선의 변화, 수술과 관련된 출혈이나 혈전의 위험도, 약을 끊는 동안 환자가 감당해야 하는 기저질환 악화의 부담, 그리고 언제 다시 안전하게 투약을 재개할 것인가를 조율하는 복합적인 결정입니다. 저는 이 질문을 시간축 위에 올려 시각화하고 싶었습니다. EstroFrame, AndroFrame, PharmaFrame은 각기 다른 목적에서 출발했지만, 이 지점에서는 모두 같은 맥락의 질문을 다루고 있습니다. 수술 전 약을 끊는다는 것은 곧 몸 안의 농도 곡선이 어떻게 내려가는지를 지켜보는 일입니다.

수술 준비 분석 결과

✓ 안전: 수술 전 목표 수치 도달 가능

• 예상 안전 도달일: 2026-07-01

• 중단 후 약 12.3일 뒤에 50 pg/mL 이하로 떨어집니다.

📄 복귀 가이드라인:

- 재개 시점: 수술 후 혈전 위험이 가장 높은 시기는 '움직이지 못하는 기간'입니다.
- 보통 수술 후 스스로 화장실을 가거나 복도를 걷는 등 **조기 보행(Early Mobilization)**이 가능해지면 재개를 고려합니다.
- 성확정 수술의 경우, 수술 후 약 2-4주 내외에 재개하는 것이 일반적이나 반드시 집도의와 상의하세요.

예측된 혈중 에스트라디올 농도 (📅) 수술 전 호르몬 중단 계획

표시 단위

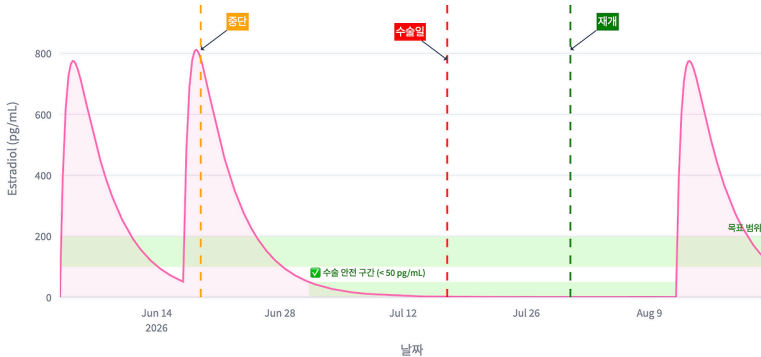
수술 그래프 기간 (일)

?

● pg/mL ○ pmol/L



예측된 혈중 에스트라디올 농도 (88 시뮬레이션 기간 (일))



수술 전후 약물 중단 시뮬레이션 화면

주소: estroframe.streamlit.app 연결 프로젝트: EstroFrame, And

roFrame, PharmaFrame 형태: HRT 및 일반 약물의 약동학 기반

수술 전후 중단/재개 시뮬레이션 핵심 구조: 투약 일정 → 수술 전

중단일 → 수술일 → 재개일 → 잔류 농도 곡선 → 위험도와 환자 부

담 검토 **주의**: 실제 수술 전후 약물 조정이나 처방을 결정하는 도구가 아니라, 약동학적 사고 과정을 시각화하고 돕기 위한 연구용 프로토타입입니다.

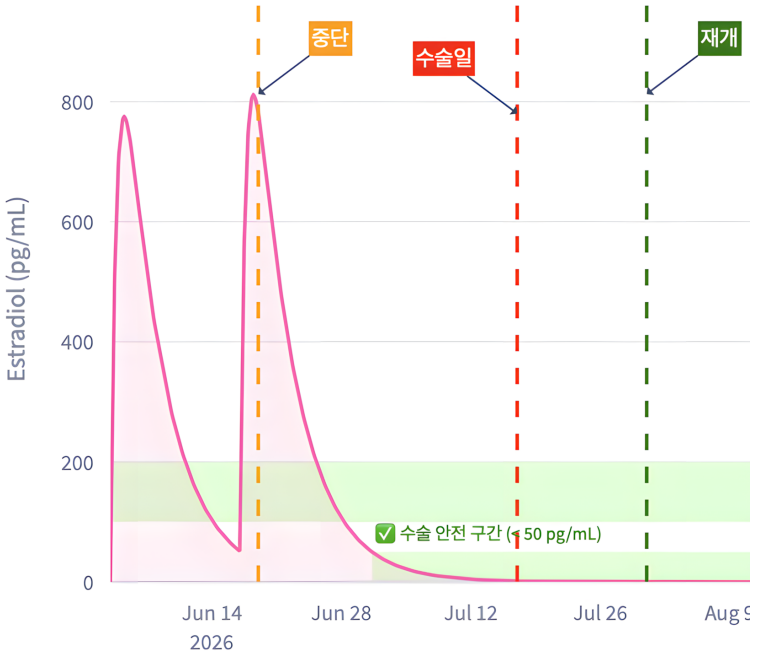
1) 중단은 스위치가 아니다

약을 끊었다고 해서 몸 안의 약물이 마법처럼 바로 사라지는 것은 아닙니다. 마지막 복용이나 주사는 EMR에 기록되는 단 하나의 사건에 불과합니다. 하지만 그 이후의 농도 변화는 약동학적 법칙을 따르며 시간의 흐름을 타고 이어집니다. 약물은 제형에 따라 계속 흡수되고, 전신으로 분포하며, 간과 신장을 거쳐 대사되고 배설됩니다. 경구약이라면 위장관의 흡수 속도와 생체이용률이 중요하게 작용하고, 주사제라면 제형 특성에 따른 저장소 효과(depot effect)가 긴 꼬리를 만들 수 있습니다. 신장으로 배설되는 약물이라면 환자의 eGFR 수치가 소실 속도를 결정하고, 간에서 대사되는 약물이라면 간 기능 상태가 절대적인 영향을 미칩니다. 그래서 수술 전 약물 중단을 논할 때 던져야 할 질문은 단순히 “며칠 전부터 끊을 것인가”에 머물러서는 안 됩니다.

그보다 더 정확하고 본질적인 질문은 “수술 당일, 환자의 몸 안에는 어느 정도의 약물이 잔류해 있을 것인가”에 가깝습니다. 중단이라는 행위 자체는 투약 사건을 멈추는 일에 불과하지만, 농도 곡선은 그 이후에도 멈추지 않고 계속해서 자신의 경로를 움직입니다.

2) 마지막 투여일보다 잔류 농도가 중요하다

예측된 혈중 에스트라디올 농도 (150 시뮬레이션 기간 (일))



마지막 투여 이후 서서히 감소하는 잔류 농도

진료 현장에서 수술 전 약물 중단은 보통 날짜를 기준으로 안내됩니다. '수술 3일 전 중단', '1주 전 중단', '2주 전 중단'과 같은 방식은 환자에게 설명하기 명확하고, 병원의 일정을 관리하기 편하며, 의료진 간의 소통을 원활하게 만들어 줍니다.

하지만 약동학적인 관점에서 들여다보면, 일괄적인 "3일 전 중단"이라는 지시가 모든 약과 모든 환자에게 같은 의미를 갖지는 않습니다

. 반감기가 짧은 약은 하루 이틀 만에도 바닥에 닿을 만큼 빠르게 줄어들지만, 반감기가 길거나 체내 조직에 깊게 분포하는 약은 중단 후 며칠이 지나도 수술에 지장을 줄 만큼 의미 있는 농도가 남을 수 있습니다. 게다가 장기간 반복 투여로 인해 몸에 단단히 누적되어 있던 약물은 일회성 투여 때보다 훨씬 천천히 내려갑니다. 주사제나 서방형 제형처럼 흡수 자체가 서서히 이루어지는 경우에는 단순히 계산된 반감기만으로는 예측하기조차 어렵습니다.

따라서 수술 당일의 안전은 마지막 투여일 하나만으로 담보되지 않습니다. 수술 당일의 정확한 잔류 농도, 중단 기간 동안 몸이 겪는 농도의 하강 변화, 그리고 재개 후 다시 회복되는 속도까지 한꺼번에 고려해야 합니다. 결국 약을 끊는다는 것은 농도 곡선의 하강 구간 전체를 정밀하게 설계하는 과정입니다.

3) EstroFrame과 AndroFrame: HRT 중단을 시간축 위에 올리다

호르몬 치료(HRT)에서는 이 문제가 한층 더 선명하게 드러납니다. 외래에서는 보통 단 한 번의 혈액 검사를 통해 호르몬 수치를 점검하지만, 몸이 겪는 실제 호르몬 농도는 투여 시점과 채혈 시점의 틈새에서 끊임없이 출렁입니다. 주사제의 경우 투여 직후의 수치와 주기 막바지의 수치는 완전히 다릅니다.

수술 전 호르몬을 며칠간 중단하라는 지시는 겉보기엔 단순한 안전 수칙처럼 보이지만, 그 기저에는 수많은 요소들의 팽팽한 줄다리기가

가 얽혀 있습니다. 수술과 연관된 치명적인 혈전 발생의 위험, 주사제와 경구약 등 제형별 반감기의 차이, 오랜 기간 투여로 쌓인 누적 농도, 약을 끊는 동안 환자가 고스란히 감당해야 하는 신체적·정서적 박탈감과 부담, 그리고 수술 후 재개 시점과 농도의 회복 속도까지. EstroFrame과 AndroFrame은 이 복잡한 방정식들을 시간축 위로 올려 직관적으로 확인하려는 시도였습니다. 수술 전 중단일, 실제 수술일, 투약 재개일을 시뮬레이션 일정에 반영하고 그에 따른 잔류 농도 곡선을 그려냅니다. 핵심은 “며칠 전부터 끊는다”는 말뿐인 지시를, 눈에 보이는 농도 변화의 곡선으로 번역하는 일입니다. 이를 통해 위험을 줄이려는 방어적인 판단과, 환자의 고통을 덜어주려는 임상적 판단 사이에서 최적의 시간적 타협점을 찾습니다.

4) PharmaFrame: 수술 전 약물 중단을 일반화하다

HRT 시뮬레이션에서 출발한 이 질문은 다른 모든 약물 영역으로도 고스란히 확장됩니다. 수술 전 약물 관리는 호르몬 치료에만 국한된 특수한 상황이 아닙니다.

갑상선 약물, 진통제 계열인 NSAIDs나 마약성 진통제(Opioids), 그리고 다양한 심혈관계 약물들은 저마다 수술 전후에 고려해야 할 치명적인 위험 요소를 안고 있습니다. 어떤 약은 수술 중 심각한 출혈을 유발할 수 있고, 어떤 약은 마취나 수술 스트레스와 겹쳐 심혈관 사고의 위험을 높입니다. 반대로 약을 너무 일찍, 너무 길게 중단하면 기존의 기저질환이 통제 불능 상태에 빠질 위험도 도사립니다.

PharmaFrame은 이 개별적인 딜레마들을 더 보편적이고 일반적인 약동학 프레임워크로 통합하려는 시도였습니다. 약물의 PK 데이터 베이스를 구축하고 환자의 프로필을 입력한 뒤, 수술로 인한 강제 중단 기간을 일정에 반영합니다. 중단 기간 동안의 투여는 누락(missed) 처리되고, 우리는 수술 당일과 그 전후의 잔류 농도가 어떻게 추락하는지 그래프를 통해 목격하게 됩니다. 중요한 것은 수술 전 약물 관리를 단순한 “복용 여부(O/X)”의 문제에서 “시간-농도 곡선의 하강 추이”를 다루는 문제로 인식의 틀을 전환했다는 점입니다.

5) 환자마다 같은 중단 기간이 같은 의미는 아니다

같은 약을 똑같이 7일 전에 중단하더라도 환자마다 몸이 그려내는 농도 곡선은 결코 같지 않습니다. 환자마다 체중이 다르고 체지방률에 따른 분포용적이 다르며, 무엇보다 약물을 걸러내는 신기능(eGFR)과 간기능이 제각각이기 때문입니다.

신장 배설 의존도가 높은 약물은 eGFR 수치에 따라 소실 속도가 현저히 느려질 수 있고, 지용성이 큰 약물은 넉넉한 체지방 조직에 깊숙이 축적되어 중단 후에도 몸 안에 끈질기게 머물 수 있습니다.

따라서 수술 전 약물 중단 지침에 동일한 날짜를 적용한다고 해서, 모든 환자가 수술 당일 완벽하게 똑같은 생리적, 약동학적 안전 상태에 도달하는 것은 아닙니다. 개인화 약동학 모델의 진정한 가치는 완벽히 들어맞는 미래의 농도값을 맞히는 데 있는 것이 아닙니다. 획일화된 의학적 지시가 다양한 신체 조건의 환자들에게 적용될 때,

각기 얼마나 다른 곡선의 궤적을 그려낼 수 있는지 미리 상상하고 대비하기 위함입니다.

6) 재개 시점도 판단이다

우리는 수술 전 약물 관리를 논할 때 흔히 '언제 끊을 것인가'에만 온통 신경을 곤두세웁니다. 하지만 언제, 어떻게 약을 다시 시작할 것인가 하는 '재개 시점' 역시 중단 시점 못지않게 고도의 임상적 판단을 요구합니다.

수술이 무사히 끝났다고 해서 중단했던 모든 약을 다음 날 아침부터 당장 한꺼번에 털어 넣을 수 있는 것은 아닙니다. 아직 아물지 않은 상처 부위의 출혈 위험이 남아 있을 수 있고, 마취와 수술 스트레스로 인해 위장관의 흡수 기능이 떨어져 있을 수도 있습니다. 반대로 기저질환의 통제를 위해 마냥 회복을 기다리며 투약을 미룰 수도 없는 노릇입니다.

결국 수술 전후의 약물 관리는 중단일, 수술일, 재개일이라는 세 개의 날짜가 톱니바퀴처럼 맞물려 돌아가는 과정입니다. 그리고 이 세 날짜 사이의 간격에서 농도 곡선은 쉴 새 없이 하강과 상승을 반복합니다. 재개 후에는 다시 약물이 몸에 쌓이는 상승 구간이 시작되며, 언제쯤 투약 전의 안정 상태로 회복될 것인지 예측하는 것 또한 임상 판단의 중요한 일부입니다. 약을 어떻게 안전하게 끊느냐 만큼이나, 어떻게 부드럽게 다시 시작하느냐가 환자의 최종적인 회복을 좌우합니다.

7) 모델은 결정을 대신하지 않는다

수술 전 약물 중단이라는 복잡한 문제에는 모든 환자에게 완벽하게 들어맞는 단 하나의 정답이 존재할 수 없습니다.

약물의 고유한 반감기와 농도 곡선의 형태, 수술의 종류에 따른 출혈 및 혈전 위험도, 투약 중단이 불러올 기저질환 악화 가능성, 약 없이 견뎌야 하는 환자의 신체적 불편감, 그리고 수술 후의 회복 양상과 모니터링 계획까지. 이 모든 변수들을 저울에 올려놓고 치열하게 균형을 잡아야 합니다.

제가 만든 그 어떤 시뮬레이션 모델도 이 지난하고 복잡한 임상적 판단을 대신해주지 않습니다. EstroFrame, AndroFrame, PharmaFrame은 의사의 처방을 결정해주거나 특정 중단 날짜를 정답인 양 점지해주는 인공지능이 아닙니다.

이 도구들의 유일한 목적은 우리가 던지는 질문을 눈에 보이게 만들어주는 데 있습니다. “약을 끊었을 때 몸 안의 농도 곡선은 어떤 각도로 떨어지는가”, “수술 당일의 잔여 위험은 어느 정도인가”, “위험을 회피하려다 환자에게 감당하기 힘든 부담을 지우고 있지는 않은가.”

수술 전 약을 끊는다는 것은 단순히 달력의 숫자에 동그라미를 치는 일이 아닙니다. 시간이라는 도화지 위에서 약물의 농도와 수술의 위험도, 그리고 환자의 고통을 섬세하게 조율해 나가는 예술에 가깝습니다. 저는 그 치열한 조율의 과정을 조금 더 선명한 곡선의 형태

로 눈앞에 꺼내놓고 싶었을 뿐입니다.

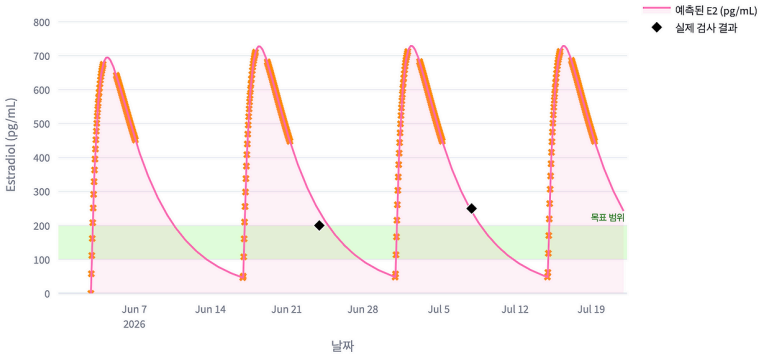
3. 모델은 처음부터 맞지 않는다

모델을 만들 때 가장 경계해야 할 착각이 하나 있습니다. 바로 처음부터 완벽하게 들어맞는 모델을 만들 수 있다는 착각입니다. 그럴듯한 수식을 세우고 문헌 값을 성실하게 입력한 뒤, 환자 정보를 바탕으로 매끄러운 그래프를 그려내면 마치 현실을 꽤 정확하게 묘사하고 있는 것처럼 보입니다.

하지만 실제로는 전혀 그렇지 않습니다. 모델은 결코 처음부터 맞지 않습니다.

대부분의 모델은 현실의 복잡성을 과감하게 쳐낸 단순한 가설에서 출발합니다. 그리고 그 가설은 실제 관찰값을 만나면서 깨지고, 부딪히고, 조금씩 수정되어 갑니다. 제가 EstroFrame, AndroFrame, NeuroFrame 등 여러 모델을 만들며 공통적으로, 그리고 가장 뼈저리게 느낀 것도 바로 이 지점이었습니다. 모델은 확고부동한 정답이 아니라, 끊임없이 조정되어야 하는 가설일 뿐입니다.

예측된 혈중 에스트라디올 농도 (49 시뮬레이션 기간 (일))



예측값과 관찰값의 비교

주소: - estroframe.streamlit.app - androframe.streamlit.app - neuroframe.streamlit.app

연결 프로젝트: EstroFrame, AndroFrame, NeuroFrame 핵심 구

조: 초기 모델 → 예측값 생성 → 관찰값 입력 → 오차 확인 → 보정 계수 또는 baseline 조정 → 다시 예측 주의: 모델은 실제 처방이나 치료, 생활 방식의 결정을 직접적으로 지시하는 도구가 아닙니다. 예측과 관찰의 차이를 정량적으로 확인하고 수정 과정을 밝기 위한 학습 및 연구용 구조입니다.

1) 모델은 현실의 축소판이다

모델은 현실을 픽셀 단위로 복제하지 않습니다. 현실은 수학 공식 몇 줄로 담아내기엔 너무나 복잡하고 방대합니다. 사람마다 약물의 흡수율과 대사 속도가 다르고, 체중과 체지방률이 다르며, 매일매일

의 생활 패턴마저 요동칩니다. 약을 제시간에 정확히 맞았는지, 채혈 시점이 식후였는지 공복이었던지도 결괏값을 크게 흔들어 놓습니다

일상의 수면과 피로 역시 마찬가지입니다. 같은 시간에 자고 같은 시간에 일어나도 어떤 날은 머리가 맑고, 어떤 날은 물먹은 솜처럼 몸이 무겁습니다. 같은 양의 카페인을 마셔도 심장이 쿵쿵 뿔 만큼 효과가 강한 날이 있는가 하면, 아무리 마셔도 피로가 가시지 않는 날도 있습니다. 이처럼 끝없이 파생되는 수많은 변수를 처음부터 모델에 다 옥여넣을 수는 없습니다.

그래서 모델은 필연적으로 현실을 축소합니다. 가장 중요하다고 여겨지는 핵심 변수 몇 개만 추려내고 관계를 억지로 단순화하여, 일단 대략적인 뼈대부터 세웁니다. 이것은 모델의 한계나 실패가 아닙니다. 모든 모델은 원래 그렇게 불완전하게 태어납니다. 진짜 문제는, 모델을 만든 사람이나 쓰는 사람이 그 모델이 '단순화된 가설'에 불과하다는 사실을 잊어버리는 순간 시작됩니다. 모델은 현실 그 자체가 아니라, 현실을 더 잘 이해하기 위해 깎아 만든 축소판일 뿐입니다.

2) 예측값은 관찰값을 만나야 한다

모델이 단순한 숫자 놀음을 넘어 현실적인 의미를 가지려면, 모델이 뱉어낸 '예측값'이 반드시 현실의 '관찰값'과 정면으로 마주해야 합니다. 약물 농도를 추정하는 모델이라면 예상 농도 곡선이 실제 채

혈된 혈액검사 결과와 겹쳐져야 하고, 하루의 에너지 상태를 그리는 모델이라면 예상된 궤적이 실제 그날 느낀 주관적 명료도와 직접 비교되어야 합니다.

관찰값이 빠진 채 예측만 난무하는 모델은 현실과 단절된 자기만의 닫힌 세계에 머물습니다. 혼자서 아무리 그럴듯하고 매끄러운 그래프를 그려내더라도, 그것이 현실과 얼마나 동떨어져 있는지는 알 길이 없습니다. 그래서 EstroFrame과 AndroFrame에서는 환자의 실측 검사값을 직접 입력하여 예측 곡선과 비교하는 구조를 필수적으로 넣었습니다. 투여 경로, 정확한 혈액검사 시점, 그리고 측정된 결괏값을 넣었을 때 비로소 모델의 예측과 실제 현실 사이의 노골적인 격차를 눈으로 확인할 수 있습니다.

NeuroFrame의 구조 역시 동일한 철학을 따릅니다. 수면 시간, 카페인 섭취량, 업무 부하를 재료 삼아 하루의 에너지 곡선을 야심 차게 예측하지만, 그 곡선이 진짜 내 하루의 체감과 맞아떨어지는지는 전적으로 다른 문제입니다. 그래서 하루의 끝에 수행하는 'end-of-day check-in'이 결정적으로 중요합니다. 오늘의 주관적 명료도, 실제 집중할 수 있었던 시간, 전반적인 에너지 만족도 같은 현실의 값들이 입력될 때, 모델은 비로소 환상에서 깨어나 현실과 부딪힙니다.

물론 관찰값은 모델의 불완전함을 들춰내어 개발자를 불편하게 만듭니다. 하지만 그 뼈아픈 불편함이 있어야만 모델은 진화할 수 있습니다.

3) 오차는 실패가 아니라 정보다

분석 통계

통계 수치는 항정 상태(90-180일) 구간을 기준으로 산출되었습니다.

최고치	최저치	평균
729.1 pg/mL	47.9 pg/mL	306.7 pg/mL
변동폭 ②	최대 변화율 ②	RMSE (오차) ②
222.1%	1322.4	29.4 pg/mL

모델 신뢰도: 보통

현재 그래프는 사용자 피검사 결과에 의해 보정된 상태입니다. (Injection: 0.90x)

예측과 관찰의 차이

예측이 틀렸다는 사실은 얼핏 모델의 참담한 실패처럼 보일 수 있습니다. 하지만 모델을 다루는 관점에서 오차는 실패가 아니라 가장 귀중한 '정보'입니다. 어느 방향으로 어긋났는지, 얼마나 크게 빗나갔는지, 특정한 조건에서만 반복적으로 틀리는지, 아니면 어떤 특정 변수가 개입될 때 오차가 폭발적으로 커지는지를 면밀히 들여다보아야 합니다.

이 오차의 패턴을 읽어내면 모델이 현실의 어떤 조각을 놓치고 있는지 역으로 추적할 수 있습니다. 약물 농도의 예측값이 실제 검사값보다 유독 높게 나온다면, 환자의 실제 대사율이나 배설률이 교과서적인 문헌 값보다 훨씬 빠르다는 뜻일 수 있습니다. 반대로 예측값이 너무 낮다면 체내 분포용적이나 청소율에 대한 초기 가정을 근본적으로 다시 짚어봐야 합니다. 특정 경로(예: 경피 투여)에서만 오차가 크다면 그 경로의 흡수 메커니즘을 너무 나이브하게 설계했을 확률이 큽니다.

NeuroFrame에서도 마찬가지입니다. 모델은 오늘 오후를 최고의 생산성을 낼 수 있는 'Prime Zone'으로 호언장담했지만, 실제 체감 에너지는 바닥을 기었을 수 있습니다. 그렇다면 모델에 미처 반영하지 못한 전날의 감정적 스트레스, 보이지 않는 수면 부채, 혹은 급격한 기온 변화 같은 외부 변수를 새로 의심해 보아야 합니다. 오차는 부끄러운 결함이 아닙니다. 모델이 현실의 단단한 벽과 맞부딪히며 파생되는 불꽃입니다. 좋은 모델은 애초에 한 번도 틀리지 않는 완벽한 모델이 아닙니다. 자신이 얼마나 틀렸는지를 명확히 보여줄 수 있는 모델입니다.

4) 보정은 모델을 현실 쪽으로 당기는 일이다

경로별 정밀 보정

검사 시점 어떤 투여 경로를 기준으로 보정할까요? ②

35 - + 주사 ▼



검사 결과 (pg/mL)

250.00 - +

! 주의: 주사제는 투약 주기 중간의 채혈 결과가 중요합니다. 정확한 보정을 위해 투약 주기 중간 시점의 채혈 결과를 입력하세요.

+ 검사 기록 추가

주사 검사 이력

21일	200.0 pg/mL	
35일	250.0 pg/mL	

현재 적용된 개인화 계수

주사 계수
x 0.90

보정 초기화

보정 이력 화면

관찰값을 통해 뼈아픈 오차를 확인했다면, 이제 남은 단계는 '보정(Calibration)'입니다. EstroFrame과 AndroFrame에서는 투여 경로별로 누적된 혈액검사 기록을 바탕으로 시스템에 보정계수를 먹일 수 있도록 설계했습니다. 같은 약물이라 하더라도 투여 경로에 따라 흡수 양상이 판이하고, 환자 개인의 유전적·생리적 특성에 따라 대사와 분포가 천차만별로 달라지기 때문입니다.

보정계수는 공중에 붕 떠 있는 추상적인 모델의 먹살을 잡아 현실 쪽으로 질질 끌어당기는 장치입니다. 최초의 모델이 텍스트북에 나오는 창백하고 일반적인 곡선이었다면, 슬한 오차와 보정을 거친 이

후의 모델은 점점 그 환자 개인의 살갓에 밀착된 고유한 곡선으로 변모합니다.

NeuroFrame의 'baseline offset' 개념도 이와 정확히 같은 역할을 수행합니다. 수면과 카페인 공식으로 빚어낸 곡선이 실제 체감보다 시종일관 과대평가되거나 과소평가된다면, 매일의 자기평가 데이터를 근거로 그 곡선의 기준선 자체를 위아래로 끌어내리거나 올려야 합니다. 이것은 자신이 만든 모델의 실패를 자인하고 포기하는 일이 아닙니다. 오만했던 가설을 꺾고 모델을 현실의 굴곡에 맞게 구부러 뜨리는 가장 과학적인 과정입니다. 모델이 틀렸음을 깨끗하게 인정하고, 그 틀림의 데이터를 다음 예측의 땀감으로 쓰는 일입니다.

5) 보정할 수 없는 모델은 위험하다

모델이 처음부터 틀릴 수 있다는 것은 지극히 자연스러운 섭리입니다. 하지만 틀렸다는 사실이 명백하게 드러났음에도 불구하고 그것을 뜯어고칠 수 없다면, 그때부터 모델은 흥기가 됩니다. 특히 사람의 건강과 생명을 다루는 의료 영역에서는 더욱 치명적입니다.

현실의 값과 비교할 수도 없고, 오차가 얼마나 벌어졌는지 기록되지 않으며, 사후에 어떤 파라미터도 보정할 수 없는 '블랙박스' 모델이 있다고 가정해 봅시다. 이런 모델은 겉보기에 그럴듯하고 확신에 찬 결론을 내놓을수록 오히려 더 위험합니다.

건강한 모델은 자신이 어떤 가너린 가정들 위에 서 있는지 투명하게 고백해야 합니다. 어떤 입력값을 집어넣었는지, 내부적으로 어떤 파

라미터를 곱했는지, 실제 관찰값과 비교했을 때 오차율은 몇 퍼센트였는지, 그리고 보정을 거치고 난 뒤 그래프가 얼마나 요동쳤는지 이 모든 과정이 백일하에 드러나야 합니다. 이 지지분한 수정의 족적이 낱낱이 보여야 비로소 의사나 환자는 이 모델을 어디까지 신뢰하고 어디서부터 의심해야 할지 결정할 수 있습니다.

모델은 언제나 설명 가능해야 하고, 필연적으로 수정 가능해야 합니다. 처음부터 완벽하게 정답을 내놓는 오만한 모델보다, 틀렸을 때 군말 없이 고칠 수 있는 유연한 모델이 실제 임상 현장에서는 훨씬 더 강력하고 안전합니다.

6) 개인화는 완성된 정밀함이 아니다

우리는 흔히 '개인화(Personalization)'라는 말에 과도한 환상을 품습니다. '개인화 모델'이라고 하면 마치 환자의 DNA부터 생활 습관까지 모조리 긁어모아 그 사람의 완벽한 디지털 클론을 창조해 내는 마법처럼 여겨집니다. 하지만 제가 모델링을 통해 체득한 개인화는 그렇게 거창하고 SF적인 개념이 아닙니다.

제가 생각하는 개인화는, 평범한 모델이 끈질기게 관찰값을 받아들이고 스스로를 깎아나가는 태도 그 자체입니다. 처음에는 누구나 일반적인 값으로 시작합니다. 문헌에 적힌 평균 반감기, 대다수에게 통용되는 흡수 시간, 평균 체중을 가정한 분포용적, 교과서적인 수면-에너지 공식을 차용합니다.

그다음 현실의 성적표를 받아 듭니다. 채혈된 혈액검사 수치, 정확한

채혈 시점, 그날 느낀 찌뿌둥함의 정도, 집중력을 유지했던 분(minute) 단위의 시간, 그리고 예측 곡선과의 오차 크기와 방향. 이 거친 데이터들을 주워 담아 모델의 기준선을 1밀리미터씩 위로, 혹은 아래로 옮깁니다.

개인화는 단번에 도달하는 '완성된 정밀함'의 상태가 아닙니다. 끝없는 관찰과 오차 수정을 통해 영원히 점근선처럼 다가가는 지루한 과정입니다. 개인화는 상태가 아니라 동사입니다.

7) 모델은 겸손해야 한다

모델은 언제나 현실보다 초라하고 작습니다. 화면에 그려진 매끄러운 곡선은 결코 환자의 몸속에서 벌어지는 복잡다단한 현실의 전부 가 될 수 없습니다.

제가 진정으로 매료되는 모델은 한 번도 틀린 적 없는 완벽한 모델이 아닙니다. 자신이 틀린 정도를 솔직하게 보여주고, 그 오차의 폭을 부끄러움 없이 기록하며, 내일을 위해 기꺼이 자신의 수식을 뜯어고치는 유연한 모델입니다. 모델은 현실이라는 거대한 벽을 향해 끊임없이 몸을 던지며 스스로를 조정해 나가는 가설일 뿐입니다.

모델은 처음부터 절대로 맞지 않습니다. 하지만 관찰값을 두려워하지 않고 직면하며, 자신의 오차를 성실하게 기록하고, 다음번 예측에서 어김없이 스스로를 보정해 나간다면, 어제보다 오늘 조금 더 나아질 수는 있습니다.

제가 진정으로 매료되는 모델은 한 번도 틀린 적 없는 완벽한 모델

이 아닙니다. 자신이 틀린 정도를 솔직하게 보여주고, 그 오차의 폭을 부끄러움 없이 기록하며, 내일을 위해 기꺼이 자신의 수식을 뜯어고치는 유연한 모델입니다. 모델은 하늘에서 똑 떨어진 정답이 아닙니다. 현실이라는 거대한 벽을 향해 끊임없이 몸을 던지며 스스로를 조정해 나가는, 눈물겨운 가설일 뿐입니다.

Part 6. AI 시대에 살아남는 것

1. AI에게 일을 맡긴다는 건, 좋은 상사가 되는 일이다

ChatGPT Plus를 쓰다 보면 묘한 감각이 든다. 월 몇 만 원으로 꽤 똑똑한 직원을 한 명 고용한 것 같다. 심지어 Pro급으로 올라가도 월 수십만 원이다. 사람 인건비로 생각하면 말도 안 되게 싸다. 월 수십만 원에 글을 정리해주고, 코드를 짜주고, 자료를 요약해주고, 표지 콘셉트를 같이 고민해주고, 내가 대충 말한 생각을 구조화해주는 직원이 옆에 앉아 있는 셈이다.

물론 여기에는 중요한 조건이 있다. 아무리 직원이 똑똑해도 상사가 일을 설명하지 못하면 결과는 망한다. 부하직원이 아무리 똑똑해도 상사가 병신이면 조직은 산으로 간다. AI도 비슷하다. “알아서 잘해 줘”라는 말은 사람에게도 위험한 지시이고, AI에게도 위험한 지시다. 무엇을 만들고 싶은지, 왜 필요한지, 어떤 기준을 만족해야 하는지, 어디까지 하면 되는지 알려주지 않으면 AI는 자기 나름대로 해석해

서 움직인다. 그리고 그 해석은 종종 내가 원한 것과 다르다. 그래서 AI를 잘 쓰는 일은 천재 비서를 부리는 일이 아니다. 똑똑하지만 맥락을 모르는 보조직원을 매니징하는 일에 가깝다. 좋은 상사는 모든 일을 직접 하지 않는다. 목표를 정하고, 업무를 나누고, 기준을 세우고, 결과를 검수한다. 그리고 모르는 것이 있으면 직원에게도 묻는다.

“이 일을 잘 맡기려면 내가 너에게 어떤 정보를 더 줘야 하지?” 이 질문을 할 줄 아는 것이 중요하다. 상사가 항상 정답을 알고 있어야 하는 것은 아니다. 오히려 좋은 상사는 일을 맡기기 전에 필요한 정보를 확인한다. AI에게도 똑같이 물어볼 수 있다.

“내가 원하는 결과를 얻으려면 어떤 정보를 더 줘야 해?”

“이 작업을 어떤 단계로 나누면 좋을까?”

“내 지시에서 애매한 부분이 뭐야?”

“지금 정보만으로 가능한 일과, 추가 정보가 필요한 일을 나눠줘.”

이런 질문을 던지는 순간, AI는 단순한 답변 생성기가 아니라 함께 일을 정의하는 보조자가 된다.



1) 프롬프트는 주문이 아니라 업무 지시서다

AI 활용법을 이야기하면 흔히 “프롬프트를 잘 써야 한다”고 말한다. 맞는 말이지만, 그것만으로는 조금 부족하다. 프롬프트는 마법 주문이 아니다. 내가 원하는 결과를 기계가 처리할 수 있는 형태로 바꾸는 업무 지시서에 가깝다.

사람의 언어는 원래 두루뭉술하다.

“깔끔하게 정리해줘.”

“너무 딱딱하지 않게 써줘.”

“전문적으로 보이게 해줘.”

“내 의도는 살려줘.”

사람끼리는 이런 말도 어느 정도 통한다. 상대가 맥락을 읽고, 내 말투를 기억하고, 눈치껏 적당한 결과물을 만들어주기 때문이다. 하지만 AI에게는 이 표현들이 아직 덜 정의된 입력값이다. “깔끔하게”가 중복을 줄이라는 뜻인지, 문단을 짧게 나누라는 뜻인지, 표를 넣으라는 뜻인지, 말투를 담백하게 바꾸라는 뜻인지 알 수 없다.

그래서 AI에게 일을 맡길 때는 감각을 조금 더 작업 가능한 기준으로 바꿔야 한다.

“깔끔하게 정리해줘”보다는 “중복 문장을 줄이고, 문단당 핵심 메시지를 하나로 제한하고, 소제목을 추가해서 처음 읽는 사람이 3분 안에 구조를 파악할 수 있게 해줘”가 낫다.

“전문적으로 써줘”보다는 “용어는 정확하게 쓰되, 근거와 한계를 함

게 표시하고, 과장된 표현은 피해서 작성해줘”가 낫다.

“보기 좋게 해줘”보다는 “모바일 화면에서도 읽기 쉽도록 짧은 문단, 소제목, bullet을 사용하되, 핵심 내용은 줄이지 말고 표현만 정리 해줘”가 낫다.

AI에게 잘 말한다는 것은 내 감각을 버리는 일이 아니다. 내 감각을 AI가 다룰 수 있는 형태로 번역하는 일이다.

2) AI는 자연어와 기계의 작업 언어 사이에 있는 번역기다

AI가 잘하는 일 중 하나는 번역이다. 여기서 말하는 번역은 한국어를 영어로 바꾸는 종류의 번역만이 아니다. 사람이 대충 말한 자연어를, 다른 AI나 소프트웨어가 실행할 수 있는 지시문으로 바꾸는 것도 번역이다.

예를 들어 나는 “내 말투를 살려서 책을 만들고 싶다”고 말한다. 사람에게는 어느 정도 통하는 문장이다. 하지만 작업 시스템 입장에서는 아직 모호하다. 어떤 말투를 살릴 것인지, 어떤 형식으로 저장할 것인지, 장별 구조는 어떻게 나눌 것인지, 긴 출력은 어떻게 분할할 것인지가 정해져 있지 않다.

이때 ChatGPT는 내 말을 받아서 프로젝트 규칙, 프롬프트, AGENTS.md, 작업 계획, 파일 구조 같은 형태로 바꿔줄 수 있다. 사람의 “대충 이런 느낌”을 에이전트가 이해할 수 있는 “이 조건을 지켜서 이 순서로 작업하라”는 문장으로 바꿔주는 것이다.

앞으로 AI가 코딩을 더 많이 하고, 문서를 더 많이 만들고, 분석을

더 많이 대신하게 되더라도 이 능력은 사라지지 않는다. 오히려 더 중요해진다. 사람이 해야 할 일은 모든 코드를 직접 치는 것이 아니라, 무엇을 만들고 싶은지, 어떤 조건을 지켜야 하는지, 결과물이 어떤 기준을 만족해야 하는지를 분명하게 정의하는 일이 되기 때문이다.

결국 AI 시대의 핵심 능력은 프롬프트 몇 줄을 외우는 것이 아니다. 내 머릿속의 감각과 욕구를 기계가 실행 가능한 요구사항으로 바꾸는 능력이다.

3) 긴 작업은 한 번에 시키지 말고 파이프라인으로 나눈다

AI에게도 입력과 출력의 한계가 있다. 긴 글을 한 번에 붙여 넣고 “전부 정리해줘”라고 하거나, 책 한 장 분량의 글을 한 번에 완성해달라고 하면 결과가 흔들리기 쉽다. 앞부분의 조건을 뒤에서 잊어버리거나, 중간에 중요한 정보가 빠지거나, 출력이 길어지면서 구조가 무너질 수 있다.

그래서 긴 작업을 할 때는 프롬프트보다 먼저 작업 단위를 설계해야 한다. 입력할 자료가 너무 길다면 본문 전체를 채팅창에 붙여 넣기보다 별도 텍스트 파일이나 문서로 저장해두고, AI가 그 파일을 기준 자료로 참조하게 하는 편이 낫다. 그리고 바로 완성본을 요구하기보다 먼저 “이 자료를 어떤 순서로 처리하면 좋을지 계획을 세워줘”라고 요청한다.

출력도 마찬가지다. 한 번에 모든 내용을 뽑아내려고 하면 결과가

불안정해진다. 긴 문서라면 먼저 목차와 구조를 만들고, 각 절을 따로 작성한 뒤, 마지막에 전체 톤과 중복을 정리하는 식으로 나누는 편이 좋다.

AI는 무한한 종이 아니다. 제한된 입력과 출력 안에서 작동하는 도구다. 그러므로 AI를 잘 쓴다는 것은 멋진 프롬프트 한 줄을 아는 것이 아니라, 자료를 어떤 단위로 넣고 결과를 어떤 단위로 받을지 설계하는 일에 가깝다.

긴 작업은 한 번에 시키지 말고 나눈다



4) 실제 작업 로그: 책 한 권을 AI와 같이 빌드하기

이 책도 그렇게 만들었다. 그냥 “책 한 권 만들어줘”라고 던진 것이 아니다. 구어체 메모, 장별 아이디어, 표지 콘셉트, 원고 구조, 빌드 파이프라인, 파일 관리 규칙을 전부 작은 작업으로 쪼갬다.

작업 흐름은 대략 이랬다.

≡ 책 한 권을 AI와 같이 빌드하기 ≡

단계	내가 한 일	AI에게 맡긴 일	사람이 검수한 것
 1 구어체 메모	생각을 말하듯 쏟아냄	핵심 주장·챕터 후보 정리	말맛과 의도
 2 원고 구조화	책의 방향 설정	목차·소제목·흐름 제안	너무 뻔하지 않은지
 3 표지 콘셉트	분위기와 제목 결정	시각 요소·프롬프트 작성	책의 농담과 맞는지
 4 AGENTS.md	작업 원칙 설명	에이전트 규칙으로 번역	금지사항과 수정 범위
 5 원고 관리	파일 단위로 나눔	문서 구조·수정 단위 정리	중복·누락·톤
 6 PDF/EPUB 빌드	산출물 형식 결정	빌드 흐름·출력 구조 설계	읽을 만한 결과물인지
 7 최종 검수	내 이름으로 널리 판단	오타자·흐름·누락 점검	최종 책임

그러니까 AI가 책을 대신 써준 것이 아니다. 책을 만드는 시스템을 AI와 같이 만든 것이다. 나는 생각을 말했고, AI는 그 생각을 구조화했다. 나는 방향을 정했고, AI는 작업 단위로 나눴다. 나는 결과를 보고 “이건 내 말이 맞다” 또는 “이건 내 의도와 다르다”고 판단했다

이 과정에서 중요한 것은 AI에게 일을 던지는 것이 아니라, AI가 할 수 있는 구조를 만드는 것이었다. 책 원고 정리, 표지 콘셉트 만들기, AGENTS.md 작성, PDF/EPUB 빌드까지 전부 AI와 같이 설계했지만, 최종적으로 그것을 이해하고 책임지는 사람은 나였다.

5) 모델마다 다른 직원을 배치한다

AI 모델을 하나의 만능 도구처럼 생각하면 오히려 쓰기 어렵다. 나는 차라리 여러 명의 직원이 있는 작은 사무실처럼 생각하는 편이 편했다.

ChatGPT는 내게 비서에 가깝다. 매일 옆에 두고 생각을 정리하고,

애매한 말을 구조화하고, 다른 AI에게 줄 프롬프트를 만들고, 긴 작업을 단계로 나누는 데 균형이 좋았다. 꼭 최종 작업자라기보다, 내가 여러 AI에게 일을 맡기기 전에 먼저 의도를 정리해주는 chief of staff 같은 역할이었다.

Gemini는 저비용 말단 직원처럼 느껴질 때가 있었다. 복잡한 맥락을 알아서 잘 파악하지는 못하지만, 해야 할 일을 아주 명확하게 써주면 그 일은 꽤 성실하게 한다. 이런 모델에게는 “알아서 잘해줘”라고 말하면 안 된다. 입력 형식, 출력 형식, 금지할 행동, 예시를 구체적으로 줘야 한다. 말귀를 잘 알아듣는 직원이라기보다는, 체크리스트를 주면 움직이는 직원에 가깝다.

Claude는 비싼 외주 업체처럼 느껴진다. 글을 다듬거나 일정한 톤의 문서를 만들거나, 완성도 있는 산출물을 뽑는 데 강하다. 다만 비용이 부담스럽다. 그래서 매일 아무 일이나 쓰기보다는, 정말 문체와 완성도가 중요한 작업에 가끔 맡기는 쪽이 낫다.

Codex 같은 코딩 에이전트는 개발자나 시공팀에 가깝다. 사람이 “이런 제품을 만들고 싶다”고 말하면, 실제 파일을 열고 코드를 고치고 빌드하고 테스트하는 일을 맡길 수 있다. 하지만 이쪽도 그냥 던지면 삽질한다. 프로젝트 구조, 금지할 행동, 테스트 방법, 출력 형식, 수정 범위를 알려줘야 한다. 그래서 AGENTS.md 같은 작업 규칙이 중요해진다.

이 관점에서 AI 활용은 “어떤 모델이 제일 똑똑한가”를 고르는 문제가 아니다. 어떤 일은 비서에게, 어떤 일은 말단 직원에게, 어떤 일은 외주 업체에게, 어떤 일은 개발자에게 맡기는 식으로 역할을 나

누는 문제다. 중요한 것은 내가 원하는 결과를 이해하고, 작업을 나누고, 각 모델이 알아들을 수 있는 방식으로 지시하는 것이다.

물론 이 비교는 특정 시점의 사용 경험에 가깝다. 모델의 가격, 사용량 제한, 성능은 계속 바뀐다. 하지만 중요한 원칙은 남는다. AI 모델을 하나로 보지 말고, 역할이 다른 여러 직원처럼 배치하라는 것이다.

모델마다 다른 직원을 배치한다

모델/도구	내 안의 비유	잘하는 일	주의할 점
 <p>우엇을 도와드릴까요? 내 바로 처리할게요!</p>	비서 · Chief of Staff 전체를 보고 정리에 주는 참모	 <p>생각 정리 · 구조화 · 작업 계획</p>	 <p>애매한 생각을 같이 정리하게 하기</p>
 <p>내 바로 처리할게요!</p>	저비용 말단 직원 시키면 적절 해내는 실무 담당	 <p>반복 작업 · 단순 변환 · 형식 맞추기</p>	 <p>체크리스트와 예시를 구체적으로 주기</p>
 <p>더 좋게 고쳐주세요! 다들 어드바이스 해주지?</p>	비싼 외주 업체 전문성과 퀄리티를 책임지는 전문가	 <p>문체 다듬기 · 완성도 있는 문서</p>	 <p>문체가 중요한 작업에 선별 사용</p>
 <p>발도하고 코드를 짜주세요!</p>	개발자 · 시공팀 기술을 구현하고 완성하는 실행 팀	 <p>코드 수정 · 빌드 · 테스트 · 자동화</p>	 <p>수정 범위와 금지사항을 명확히 주기</p>
 <p>최종 결정은 내가 할게!</p>	상사 · 편집장 · 책임자 방향을 정하고 최종 책임을 지는 사람	 <p>목표 설정 · 판단 · 검수 · 최종 책임</p>	 <p>그렇듯한 결과를 그대로 믿지 않기</p>

6) 책임은 결국 사람에게 있다

AI가 만든 결과물의 책임은 결국 나에게 있다. 이걸 꽤 중요하다. AI를 써서 책을 만들든, 코드를 만들든, 발표자료를 만들든, 최종적으로 그 결과물을 내 이름으로 내보내는 순간 책임은 나에게 온다.



그래서 내가 모르는 것을 AI가 그럴듯하게 만들어내면 위험하다. 내가 이해하지 못하는 코드, 내가 판단할 수 없는 의학 정보, 내가 검수하지 못하는 분석 결과를 그대로 가져다 쓰면 안 된다. 그건 AI 활용이 아니라 책임 회피에 가깝다.

특히 의학에서는 더 그렇다. AI가 만든 의학 정보를 내가 이해하고 검수할 수 없으면, 그걸 활용하는 건 진짜 미친 짓이다. 의학 정보는 틀렸을 때 그냥 문장이 어색한 정도로 끝나지 않는다. 누군가의 판단, 치료, 검사, 불안, 비용, 안전에 영향을 줄 수 있다. AI가 그럴듯한 말투로 틀린 정보를 만들어내면 오히려 더 위험하다. 틀린 말인데 너무 말이 되게 들리기 때문이다.

의대생이나 의료인이 AI를 쓴다면 적어도 결과물이 말이 되는지 볼 수 있어야 한다. 용어가 맞는지, 병태생리가 말이 되는지, guideline이나 근거와 충돌하지 않는지, 환자에게 적용할 때 위험한 비약은 없는지 확인해야 한다. 내가 모르는 영역이라면 AI의 답을 그대로 쓰는 것이 아니라, 그 답을 검토하기 위한 출발점으로만 써야 한다. AI를 쓴다는 것은 내가 몰라도 되는 영역을 무한히 늘리는 일이 아

니다. 오히려 최소한 결과물을 이해하고, 분석하고, 판단하고, 검수할 수 있어야 한다. 내가 직접 모든 코드를 외워서 칠 필요는 없지만, 코드가 대략 무엇을 하는지, 어디가 위험한지, 결과가 말이 되는지는 봐야 한다. 내가 모든 문장을 처음부터 쓰지 않아도 되지만, 글이 내 생각을 왜곡하지 않았는지, 근거 없는 말을 추가하지 않았는지, 독자에게 오해를 주지 않는지는 확인해야 한다.

AI가 강력해질수록 사람의 역할은 줄어드는 것처럼 보인다. 하지만 실제로는 조금 다르다. 사람이 직접 모든 문장을 쓰고, 모든 코드를 치고, 모든 표를 정리하는 비중은 줄어들 수 있다. 대신 목표를 정의하고, 작업을 분해하고, 기준을 세우고, 결과를 검수하는 일이 더 중요해진다.

AI에게 일을 맡긴다는 것은 일을 하지 않는다는 뜻이 아니다. 일을 시킬 수 있는 형태로 바꾸고, 나온 결과를 책임질 수 있는 수준으로 이해하는 일이다.

AI를 잘 쓰는 사람은 명령을 많이 내리는 사람이 아니다. 일을 이해 가능한 단위로 나누고, 좋은 기준을 주고, 결과를 책임지고 검수하는 사람이다. 똑똑한 직원을 고용했다고 해서 상사의 역할이 사라지는 것은 아니다. 오히려 상사가 해야 할 일이 더 분명해진다.

AI와의 소통 능력은 결국 업무 지시 능력이다. 원하는 것을 설명하고, 애매한 부분을 줄이고, 작업 가능한 단위로 나누고, 결과를 다시 판단하는 능력. 앞으로 더 많은 일이 AI에 의해 실행될수록, 사람에게 더 중요해지는 것은 바로 이 능력일지도 모른다.

AI를 쓴다는 것은 책임을 외주화하는 게 아니다. 실행은 위임하되,

판단은 다시 회수하는 일이다.

2. 판단은 설명될 때 완성된다

의료를 구조화하고 모델을 만드는 긴 과정에서, 항상 잊지 않으려 했던 하나의 문장이 있습니다.

“모델은 정답이 아니다.”

1) 모델은 정답이 아니다

기록을 파싱해 CleanEMR을 만들고, 호르몬 곡선을 그리는 EstroFrame을 기획하고, 약물의 동태를 추적하는 PharmaFrame을 만들 때마다 이 사실을 뼈저리게 마주했습니다. 아무리 정교한 수식을 짜고 많은 데이터를 쏟아부어도, 모델은 현실의 앙상한 축소판일 뿐입니다. 환자의 몸 안에서 벌어지는 복잡다단한 생리적 반응과 우연을 완벽하게 재현할 수는 없습니다.

그렇다면 왜 우리는 끊임없이 모델을 만들고, 흠어진 의료 데이터를 구조화하려 할까요?

2) 구조화는 불확실성을 드러낸다

정답을 자동으로 얻어내기 위해서가 아닙니다. 불확실성을 더 정직하게 다루기 위해서입니다. 모델은 우리가 무엇을 모르는지, 어느 지점에서 오차가 발생하는지, 그리고 초기 예측이 얼마나 빗나갔는지

를 객관적인 지표로 보여줍니다. 막연한 직관의 영역에 머물러 있던 판단의 근거를 눈에 보이는 곡선과 수치로 끌어냅니다.

물론 수치 그 자체는 최종적인 답이 될 수 없습니다. 하지만 그 수치와 곡선을 바탕으로 환자에게 현재의 상황과 앞으로의 위험을 '설명'하는 순간, 비로소 데이터는 의미를 갖습니다. AI가 아무리 발달해도 의사를 대체할 수 없는 이유가 바로 여기에 있습니다. AI는 확률을 계산하고 모델은 궤적을 그리지만, 그 불확실성의 무게를 딛고 최종적인 판단을 내리며 그 결과에 책임을 지는 것은 오롯이 사람의 몫입니다.

의료를 구조화한다고 해서 진료가 차가운 기계적 과정으로 전락하는 것은 결코 아닙니다. 오히려 모델이 제시하는 한계와 오차를 분명히 이해하고, 이를 환자에게 정직하게 설명할 때 의료는 더 인간적이고 따뜻해집니다. “이 약을 쓰면 농도가 이렇게 변할 가능성이 높습니다. 하지만 당신의 신기능을 고려할 때 이런 위험이 있으니, 일정을 이렇게 조절합시다.”라는 구체적인 설명은, 막연한 권위식 처방보다 훨씬 더 깊고 단단한 신뢰를 만듭니다.

3) 설명은 신뢰와 책임을 만든다

결국, 판단은 설명될 때 완성됩니다. 모델과 데이터는 그 설명을 더 투명하고 책임감 있게 만들어주는 가장 강력한 도구일 뿐입니다.

4) 판단은 설명에서 완성된다

3. 어떤 의사가 될 것인가

오늘 한 교수님이 우리에게 물었다.

“자네들은 어떤 의사가 될 것인가.”

의대에 들어온 뒤 이 질문을 여러 번 들었다. 좋은 의사가 되어야 한다는 말도 많이 들었다. 환자를 위하는 의사가 되어야 한다는 말도, 평생 공부해야 한다는 말도, 책임감 있는 의사가 되어야 한다는 말도 들었다. 그런데 이상하게도 이 질문은 들을 때마다 조금씩 다르게 다가온다.

처음에는 막연한 다짐처럼 들렸다. 조금 더 시간이 지나자 직업윤리처럼 들렸다. 그리고 이제는, 앞으로 내가 어떤 방식으로 살아남을 것인가에 대한 질문처럼 들린다. 의사는 한 번 완성되는 직업이 아니다. 면허를 받는다고 끝나는 것도 아니고, 전문의가 된다고 끝나는 것도 아니다. 오히려 그때부터 더 긴 시간이 시작된다.

나는 오늘 그 질문을 세 가지 축으로 다시 정리해 보았다.



성장.

환자.

그리고 자기 자신(自信).

1. 의사는 평생 배우는 직업이다



해리슨 교과서의 첫 페이지에는 이런 문장이 나온다.

“Medicine is an ever-changing science.”

의학은 계속 변한다.

질병의 이름이 바뀌고, 진단 기준이 바뀌고, 치료 원칙이 바뀐다. 예전에는 당연하다고 여겼던 치료가 더 이상 권고되지 않기도 하고, 과거에는 불가능하다고 생각했던 치료가 표준이 되기도 한다. 그래서 의사가 된다는 것은 한 번 배운 지식으로 평생 버티겠다는 뜻이 아니다. 계속 바뀌는 지식 앞에서 계속 배우겠다는 뜻에 가깝다.

출발점은 저마다 다를 수 있다.

어떤 병원에서 시작했는지, 어떤 전공을 선택했는지, 처음부터 얼마나 잘했는지는 물론 중요하다.

하지만 그것이 전부는 아니다.

긴 시간으로 보면 시작은 하나의 지점일 뿐이다. 더 중요한 것은 그 이후에도 계속 배우고, 수정하고, 성장할 수 있는가이다. 최근 의료 환경은 AI의 등장으로 빠르게 바뀌고 있다. 정보를 찾는 방식도, 논문을 읽는 방식도, 진료 기록을 다루는 방식도 달라지고 있다.

AI는 이미 의학 공부와 임상 판단의 주변부에 들어와 있고, 앞으로는 더 깊숙이 들어올 것이다. 그렇다고 해서 의사의 역할이 사라진다고 생각하지는 않는다. 오히려 AI 시대가 될수록 의사의 역할은 더 분명해진다.

환자의 이야기를 직접 듣는 일, 환자의 표정과 말투, 망설임과 불안을 읽는 일, 신체진찰을 통해 화면 밖의 정보를 얻는 일, AI가 제안한 답을 그대로 받아들이지 않고 이 환자에게 정말 적용 가능한지 판단하는 일, 그리고 그 판단에 대해 최종 책임을 지는 일. 이것은 여전히 의사의 몫이다.

AI는 강력한 도구다. 하지만 도구는 방향을 스스로 정하지 않는다.

도구를 어떻게 쓸지, 어디까지 믿을지, 어떤 상황에서 멈출지를 결정하는 사람은 결국 의사다. 그래서 나는 AI 시대의 의사가 더 많이 배워야 한다고 생각한다. 의학도 배워야 하고, 기술도 이해해야 하며, 그 기술이 만들어내는 오류와 한계도 알아야 한다. 의사는 완성된 사람이 아니라, 계속 업데이트되는 사람이어야 한다.

2. 환자를 가족처럼 생각한다는 것

의사라면 누구나 마음 한쪽에 두려움이 있다.

의료 분쟁.

소송.

예상하지 못한 악화.

내가 놓친 것은 없었는지에 대한 불안. 이 위험을 완전히 없앨 수는 없다. 의료는 언제나 불확실성을 포함하고 있고, 의사도 결국 사람이다.

항상 완벽할 수는 없다.

하지만 위험을 줄이는 태도는 있다. 교수님은 환자를 가족처럼 생각하라고 하셨다. 이 말은 너무 익숙해서 오히려 쉽게 지나치기 쉽다.

하지만 실제 상황에 놓고 생각하면 꽤 강한 기준이다.

응급 호출이 왔을 때, 피곤해서 몸이 무겁고 이미 해야 할 일이 쌓여 있을 때, 환자가 같은 질문을 반복하고 보호자가 예민하게 반응할 때, 내가 설명할 여유가 별로 없다고 느낄 때. 그 순간 스스로에게 묻는 것이다. “이 사람이 내 가족이라면 나는 어떻게 했을까?” 물

론 모든 환자를 실제 가족처럼 느낄 수는 없다. 그것은 인간적으로 불가능하다. 하지만 적어도 의사결정의 기준을 세울 때, 이 질문은 방향을 잡아준다.

조금 더 확인했을 것이다. 조금 더 설명했을 것이다. 조금 더 빨리 보러 갔을 것이다. 조금 더 조심스럽게 말했을 것이다.

그 차이가 의료의 질을 만든다.

좋은 의사-환자 관계는 단순한 친절에서 나오지 않는다. 환자에게 무조건 맞춰주는 것도 아니다. 오히려 정확한 설명, 일관된 태도, 예측 가능한 대응, 그리고 환자가 자신이 방치되지 않았다고 느끼게 하는 신뢰에서 나온다. 의료는 기술만으로 이루어지지 않는다.

검사 수치와 영상, 약물과 수술만으로 완성되지 않는다. 의료는 결국 사람과 사람 사이에서 일어난다. AI가 진단을 보조하고, 알고리즘이 위험도를 계산하고, 모델이 예후를 예측하는 시대가 와도 환자를 직접 마주하는 사람은 여전히 의사다. 환자는 단지 질병을 가진 데이터 포인트가 아니다.

불안하고, 기대하고, 때로는 화가 나고, 때로는 아무 말도 하지 못하는 사람이다. 좋은 의사는 치료를 잘하는 사람이어야 한다. 하지만 그것만으로 충분하지 않다. 좋은 의사는 환자가 자신을 맡길 수 있다고 느끼게 하는 사람이어야 한다.

3. 마지막 기준은 자기 자신(自信)이다



예전과 달리, 이제 의사라는 이유만으로 존경받는 시대는 아니다. 어찌면 그것이 더 자연스러운 일일지도 모른다. 어떤 직업도 이름만으로 존중받을 수는 없다. 존중은 결국 그 사람이 어떻게 행동하는지에서 만들어진다.

그럼에도 의사에게 필요한 소양은 여전히 있다. 나는 그것을 정의감과 사명감이라고 생각한다. 다만 이런 말은 조금 낡게 들릴 수 있다.

요즘 시대에 정의감, 사명감이라는 말을 하면 너무 거창하거나 비현실적으로 들리기도 한다. 하지만 막상 의료 현장에 가까이 갈수록, 이런 단어가 완전히 사라질 수 없다는 생각이 든다.

아무도 보지 않는 순간이 있다. 기록에는 남지 않는 판단이 있다. 조금 더 볼 수도 있었고, 그냥 넘어갈 수도 있었던 장면이 있다. 누군가는 몰랐을 수도 있고, 환자도 끝내 알지 못했을 수도 있는 선택이 있다. 그때 남는 것은 외부의 평가가 아니다. 자기 자신이다.

나는 내가 무엇을 했는지 안다. 무엇을 하지 않았는지도 안다.

조금 더 할 수 있었는데 하지 않았던 순간도, 최선을 다했지만 어쩔 수 없었던 순간도 결국 자기 자신은 알고 있다. 사람은 자기 양심에 떳떳할 때 비로소 자기 자신을 믿을 수 있다. 그리고 그 자기 신뢰는 험난한 시간을 버티게 하는 중요한 힘이 된다. 레지던트가 되면 환자가 갑자기 악화되는 순간을 맞이하게 될 것이다.

최선을 다했는데도 결과가 좋지 않은 날도 있을 것이다. 의사이기 전에 인간이기 때문에, 내가 할 수 있는 일에는 한계가 있다. 그 한계를 인정하는 것은 무책임과 다르다. 오히려 한계를 알기 때문에, 내가 할 수 있는 몫에 더 충실해야 한다. 그래서 나는 마음속에 하나의 기도를 적어 둔다.

“주님, 제가 할 수 있는 최선을 다하겠습니다. 그리고 나머지는 주님께 맡기겠습니다.” 이 기도는 책임을 내려놓겠다는 말이 아니다. 오히려 내가 책임질 수 있는 부분을 끝까지 책임지겠다는 말에 가깝다.

내가 배워야 할 것을 배우고, 봐야 할 것을 보고, 설명해야 할 것을 설명하고, 확인해야 할 것을 확인하는 것. 그 이후에도 인간이 통제할 수 없는 영역이 남는다면, 그것은 겸허하게 받아들이는 것. 그것이 내가 지금 이해하는 의사의 태도다.

나는 어떤 의사가 될 것인가



나는 대단한 의사가 되겠다고 쉽게 말하고 싶지는 않다. 그 말은 너무 크고, 아직은 잘 모르겠다.

다만 방향은 조금 알 것 같다.

계속 배우는 의사.

환자를 데이터가 아니라 사람으로 보는 의사. AI와 기술을 활용하되, 그 책임을 회피하지 않는 의사. 그리고 아무도 보지 않는 순간에도 자기 자신에게 부끄럽지 않은 의사.

어떤 의사가 될 것인가.

아직 완성된 답은 없다. 아마 평생 답을 고쳐가며 살게 될 것이다. 하지만 적어도 지금의 나는 이렇게 말할 수 있다. 의사는 한 번 완성되는 직업이 아니라, 평생 자기 자신을 업데이트하는 직업이다. 그리고 그 업데이트의 기준은 결국 환자와 자기 자신 앞에서 떳떳한가에 있다.

4. 임상 디지털 트윈 설계자

디지털 트윈이라는 말은 거창하게 들립니다.

현실의 무언가를 가상 공간에 그대로 복제하는 것처럼 느껴집니다. 공장, 도시, 항공기, 기계 장비처럼 구조가 비교적 명확한 대상을 떠올리기 쉽습니다. 하지만 임상에서 환자의 디지털 트윈을 만든다는 말은 훨씬 조심스럽게 다뤄야 합니다.

환자는 기계가 아닙니다.

몸은 복잡하고, 생활은 변하고, 감정과 선호는 숫자만으로 환원되지 않습니다. 검사 수치가 같아도 환자가 처한 맥락은 다를 수 있습니다. 그래서 제가 생각하는 임상 디지털 트윈은 환자를 완벽하게 복제하는 시스템이 아닙니다.

그보다 현실적인 의미에서는 의사결정에 필요한 변수와 변화를 추적하는 구조화된 환자 모델에 가깝습니다.

환자의 현재 상태, 시간에 따른 검사 수치 변화, 복용 중인 약물, 위험도, 치료 반응, 예상되는 다음 상태, 그리고 아직 불확실한 부분. 이런 요소들을 하나의 구조 안에 놓고 더 설명 가능한 판단을 준비하는 것.

제가 만들고 싶은 것은 환자를 대체하는 AI가 아니라, 판단을 돕는 환자 모델입니다.

임상 디지털 트윈은 완벽한 복제가 아니라, 구조화된 판단 모델입니다

다양한 입력을 연결하고, 불확실성을 고려해 더 나은 결정을 돕습니다



임상 디지털 트윈 개념도

연결 프로젝트: EstroFrame, AndroFrame, DiaFrame, NeuroFrame, CleanEMR

형태: 임상 상태, 약물, 판단, 데이터 구조화를 연결하는 장기 비전

핵심 구조: EMR 데이터 구조화 → 상태 변수 추출 → 시간 변화 추적 → 모델 생성 → 관찰값 보정 → 의사결정 보조

주의: 환자를 대체하거나 자동으로 진료하는 AI가 아니라, 임상 판단을 더 설명 가능하게 만들기 위한 구조화된 모델 개념

1) 환자를 복제하는 것이 아니다

임상 디지털 트윈을 이야기할 때 가장 먼저 경계해야 할 것은 복제라는 환상입니다. 환자의 몸을 완벽하게 재현하고, 미래를 정확히 예측하고, 의사의 판단 없이도 최적의 결정을 내리는 시스템. 그런 상상은 매력적이지만 현실의 임상과는 거리가 있습니다.

의료 데이터는 불완전합니다. 검사 간격은 일정하지 않고, 복약 순응

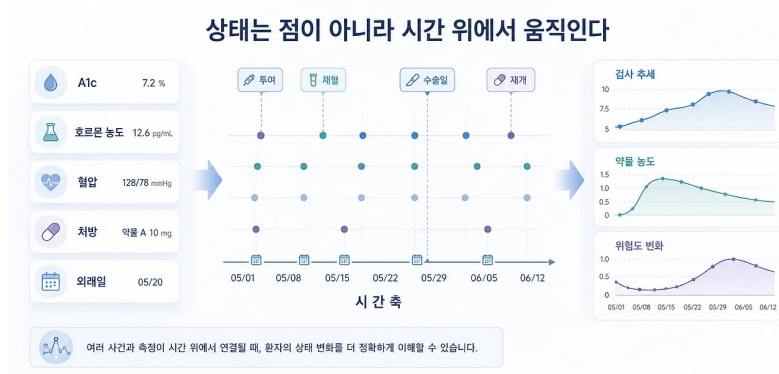
도는 흔들리고, 생활 습관은 계속 바뀝니다. 기록되지 않은 정보도 많습니다. 환자가 어떤 음식을 먹었는지, 실제로 약을 어떻게 먹었는지, 그날의 스트레스가 어땠는지, 증상을 얼마나 참고 있었는지는 항상 데이터로 남지 않습니다.

그래서 임상 디지털 트윈은 현실의 완전한 복제본이 될 수 없습니다. 오히려 중요한 것은 복제가 아니라 근사입니다.

의사결정에 필요한 변수들을 고르고, 그 변수들이 시간에 따라 어떻게 변하는지 추적하고, 어떤 가정 아래에서 다음 상태를 예측하는지 표시하는 것.

이 정도의 겸손한 모델이 임상에서는 더 현실적입니다. 디지털 트윈은 정답을 만드는 장치가 아니라 판단의 구조를 보여주는 장치여야 합니다.

2) 상태를 시간 위에 올리다



검사 수치와 약물 농도의 시간축 관계

의학은 많은 순간을 점으로 기록합니다. 혈액검사 결과, 혈압, 체중, A1c, 호르몬 농도, 약물 처방, 외래 방문일. 각각은 하나의 시점에 찍힌 값입니다.

하지만 환자의 상태는 점으로 존재하지 않습니다. 시간에 따라 움직입니다. 검사 수치는 오르고 내리고, 약물 농도는 흡수와 대사를 거치며 변하고, 증상은 좋아졌다 나빠지며, 위험도도 치료와 생활 변화에 따라 달라집니다. 그래서 임상 디지털 트윈의 기본 단위는 정적인 값이 아니라 시간에 따른 상태 변화입니다.

EstroFrame에서 보려 했던 것도 이것이었습니니다. 호르몬 농도를 단일 검사값으로만 보지 않고, 투여 시점과 채혈 시점 사이에서 움직이는 시간-농도 곡선으로 보고 싶었습니다.

같은 estradiol 수치라도 주사 직후인지, 주기 중간인지, 주기 말인지에 따라 의미가 달라질 수 있습니다. 그 순간 호르몬은 숫자 하나가 아니라 시간 위에서 움직이는 상태가 됩니다. AndroFrame도 같은 문제의식에서 출발했습니다.

호르몬 치료는 처방전 위에서는 용량과 간격으로 기록되지만, 몸 안에서는 peak와 trough, 평균 농도와 변동폭으로 경험됩니다. 디지털 트윈은 이런 시간축을 필요로 합니다. 환자의 현재 값뿐 아니라 그 값이 어디에서 왔고 어디로 움직일 가능성이 있는지를 함께 봐야 하기 때문입니다.

3) 약물은 환자 안에서 다시 계산된다

약물은 약 자체의 특성만으로 움직이지 않습니다. 같은 약이라도 환자에 따라 다르게 경험됩니다. 체중, 체지방률, 신기능, 간기능, 투여 경로, 복약 순응도는 모두 약물 농도 곡선의 모양을 바꿀 수 있습니다. 약물은 환자 안으로 들어간 뒤 환자의 조건과 만나 다시 계산됩니다.

신기능이 떨어진 환자에서

신장 배설 비율이 높은 약물은 다르게 움직일 수 있습니다.

간기능이 좋지 않은 환자에서

간 대사 의존도가 높은 약물은 더 조심해서 봐야 합니다.

복약을 한 번 놓치거나, 수술 전 일정 기간 중단하거나, 다시 시작하는 시점이 달라지면 곡선은 다시 바뀝니다.

임상 디지털 트윈은 약물을 단순히 “복용 중” 또는 “중단”으로만 보지 않습니다. 언제, 얼마만큼, 어떤 경로로 들어왔고, 환자 안에서 어느 정도 남아 있으며, 다음 시점에 어떻게 변할지를 묻습니다. 약물은 처방의 항목이 아니라 환자 상태를 바꾸는 동적 변수입니다.

4) 판단도 모델이 될 수 있다

디지털 트윈은 생리적 상태만 다루지 않습니다. 임상 판단의 구조도 모델이 될 수 있습니다. DiaFrame은 이 지점에서 출발했습니다.

당뇨병 약제 선택은 단순히 A1c 하나만 보고 결정되지 않습니다. A1c, 공복혈당, eGFR, BMI, 현재 사용 중인 약제, 저혈당 위험, 심혈

관 위험, 신장 기능, 환자의 나이와 순응도 같은 조건들이 함께 들어
옵니다.

의사는 이 정보들을 바탕으로 치료를 유지할지, 강화할지, 완화할지,
다른 약제로 변경할지 판단합니다.

DiaFrame은 이 판단을

AI가 대신하게 하려는 도구가 아니었습니다.

오히려 AI 추천을

검증 가능한 구조로 제한하고 싶었습니다.

AI가 어떤 방향을 제안했는지,

어떤 약제 계열을 추천했는지,

실제 처방과 어디가 일치하고

어디가 다른지 비교하는 구조.

즉 DiaFrame은 처방 판단의 디지털 트윈에 가깝습니다.

환자의 상태와 실제 처방 방향을

하나의 검증 가능한 데이터 구조 안에 놓고, AI의 추천을 그 구조

안에서 평가하는 것입니다.

여기서 중요한 것은 AI가 답을 내는 일이 아닙니다. 그 답을 비교하
고, 오류를 찾고, 불일치 사례를 리뷰할 수 있는 구조입니다.

임상 디지털 트윈은 생리적 모델뿐 아니라

판단의 모델도 포함할 수 있습니다.

5) 상태도 모델이 될 수 있다

NeuroFrame은 조금 다른 방향의 시도였습니다. 이 프로젝트는 환자를 모델링하려는 도구는 아니었습니다. 하루의 상태를 모델링하려는 시도에 가까웠습니다.

수면 시간, 카페인 섭취, 업무량, 교대 근무, 주관적 명료도, 하루가 끝난 뒤의 피드백. 이런 값들을 바탕으로 하루의 에너지 곡선을 예측하고, Prime Zone과 Crash Zone, Sleep Gate를 나누어보려 했습니다.

이것 역시 넓은 의미에서는 상태의 디지털 트윈입니다. 완벽한 생리학 모델은 아닙니다. 하지만 하루를 무작정 버티는 것이 아니라 어떤 시간대에 집중력이 높을지, 언제 무너질 가능성이 큰지, 어떤 일정을 어느 구간에 배치할지 예상해보는 구조입니다.

그리고 중요한 것은 피드백입니다. 예측한 에너지 곡선이 실제 체감과 맞았는지, 주관적 명료도는 어땠는지, 집중은 성공했는지, 실제 집중 시간은 얼마나 되었는지를 다시 입력합니다.

모델은 처음부터 맞지 않습니다. 관찰값을 만나고, 오차를 보고, 조금씩 보정됩니다. 임상 디지털 트윈도 마찬가지입니다. 처음부터 완성된 복제본이 아니라 반복되는 관찰과 보정을 통해 조금씩 현실에 가까워지는 모델입니다.

6) CleanEMR은 디지털 트윈의 바닥이다

좋은 모델은 좋은 데이터 구조 위에 올라간다



CleanEMR → 구조화 데이터 → 모델 계층 구조도

모델을 만들기 전에 데이터가 필요합니다. 그리고 의료에서 데이터는 처음부터 깔끔한 형태로 존재하지 않습니다. EMR에는 많은 정보가 있지만, 그 정보는 자유 텍스트와 검사 결과, 처방 기록과 외래 기록 속에 흩어져 있습니다. CleanEMR은 이 흩어진 기록을 연구 가능한 형태로 바꾸기 위한 전처리 파이프라인이었습니다.

환자, 방문, 검사, 활력징후, 약물, 실제 처방 방향, 수동 검토가 필요한 문장. 이런 것들이 구조화되어야 그다음 모델이 의미를 가집니다.

디지털 트윈은 화려한 AI 모델에서 시작하지 않습니다. 먼저 환자의 상태를 담은 데이터 구조가 필요합니다.

어떤 값이 어느 환자에게 속하는지,

어느 방문의 값인지,

어떤 원문을 근거로 추출되었는지,

어떤 부분은 아직 불확실한지 알아야 합니다. CleanEMR은 디지털

트윈의 바닥에 있는 작업입니다.

눈에 잘 띄지는 않지만, 없으면 위에 어떤 모델도 안정적으로 올라가기 어렵습니다. 좋은 환자 모델은 좋은 데이터 구조 위에서만 만들어질 수 있습니다.

7) 임상 디지털 트윈 설계자

제가 만들고 싶은 것은 환자를 대신하는 AI가 아닙니다. 환자를 하나의 데이터 포인트로 줄이는 시스템도 아닙니다. 오히려 반대에 가깝습니다. 환자의 상태가 얼마나 복잡한지 알기 때문에 그 복잡함을 조금 더 조심스럽게 다루는 구조가 필요하다고 생각합니다.

디지털 트윈은 환자를 완벽하게 복제하는 일이 아닙니다. 의사결정에 필요한 변수들을 고르고, 그 변수들의 시간적 변화를 추적하고, 관찰값을 통해 보정하고, 불확실한 부분을 표시하고, 의사와 환자가 설명 가능한 판단을 하도록 돕는 모델을 만드는 일입니다.

EstroFrame은 호르몬 농도의 시간축을 보려는 시도였습니다. AndroFrame은 호르몬 치료의 다른 축을 같은 방식으로 다루려는 시도였습니다. PharmaFrame은 약물과 환자 조건을 연결한 일반 약동학 모델로 확장하려는 시도였습니다. DiaFrame은 당뇨병 약제 판단을 검증 가능한 구조로 만들려는 시도였습니다. NeuroFrame은 하루의 상태와 에너지 변화를 모델로 보려는 시도였습니다. CleanEMR은 그 모든 모델이 올라갈 수 있는 의료 데이터의 바닥을 만드는 시도였습니다.

각각은 서로 다른 프로젝트처럼 보입니다. 하지만 제게는 같은 방향을 향하고 있었습니다.

감각을 수치로 바꾸고, 수치를 구조로 묶고, 구조를 모델로 만들고, 모델을 관찰값으로 보정하고, 그 결과를 설명 가능한 판단으로 가져가는 일. 이것이 제가 생각하는 임상 디지털 트윈입니다.

저는 의사가 되고 싶습니다. 동시에 이런 모델을 설계하는 사람이 되고 싶습니다.

환자를 더 정확하게 이해하기 위한 구조, 판단을 더 명확하게 만드는 모델. 제가 만들고 싶은 것은 그 사이에 있습니다.

의학과 공학 사이. 데이터와 환자 사이. 예측과 책임 사이. 저는 그 구조를 설계하고 싶습니다.

에필로그

두 세계를 오가는 마법진

의세계에 전생한 지 어느덧 4년이 되었다.

처음 이 세계에 떨어졌을 때, 나는 아무것도 몰랐다. 눈앞에 펼쳐진 마법진은 도무지 읽을 수 없었고, 고대어처럼 적힌 주문들은 입에 잘 붙지 않았다. 마도구의 사용법도 서툴렀다. 길드에서 나누어준 두꺼운 서책들은 하루가 지나도 한 장 넘기기 어려웠고, 마왕이 내는 문제는 언제나 필요 이상으로 성실했다.

처음에는 1성 마법 하나 제대로 쓰는 것도 버거웠다.

기초 술식 하나를 외우는 데에도 시간이 오래 걸렸고, 마법진의 선

하나가 무엇을 의미하는지 몰라 멍하니 바라보는 날이 많았다. 고위 마법사들이 아무렇지 않게 펼치는 술식은 거의 기적처럼 보였다. 저걸 사람이 한다고? 저걸 보고 바로 해석한다고? 저걸 외워서 시험까지 친다고?

그때의 나는 이 세계에서 오래 살아남을 자신이 없었다.

하지만 4년이라는 시간은 생각보다 많은 것을 바꾸어놓았다.

어느새 나는 기본 마도구를 제법 다룰 수 있게 되었다. 처음에는 의미 없는 선처럼 보이던 마법진도 이제는 천천히 읽을 수 있었다. 1성 마법도 버거워하던 시절이 있었지만, 이제는 2성, 3성 마법진 정도는 펼쳐볼 수 있게 되었다. 물론 아직 대마법사라 부르기에는 한참 멀었다. 새로운 던전에 들어갈 때마다 나는 다시 초보자가 되었고, 마왕은 여전히 성실하게 문제를 냈다.

그래도 이제는 예전처럼 아무것도 모른 채 얼어붙지는 않았다.

의세계에는 다양한 종족의 동료들도 있었다.

숲의 기척을 읽듯 작은 변화를 알아차리는 엘프가 있었다.

묵묵히 자료를 정리하고 장비를 고치는 노움도 있었다.

한 번 본 술식을 거의 그대로 외워내는 현자도 있었고, 전장에서 망설임 없이 손을 움직이는 전사도 있었다.

어떤 이는 사람의 마음을 읽는 데 능했고, 어떤 이는 복잡한 규칙을 순식간에 정리했다. 어떤 이는 긴박한 순간에도 침착했고, 어떤 이는 사소한 이상 신호를 절대 놓치지 않았다.

다들 각자의 메인 특성을 가지고 있었다.

나는 오래도록 내 특성이 무엇인지 알지 못했다.

나는 원래 이 세계 사람이 아니었다. 전생의 나는 토목공학도였다. 돌과 철과 콘크리트, 하중과 반력, 전단력과 모멘트의 세계에서 살던 사람이었다. 의세계에 오고 나서도 한동안은 내가 이곳에 어울리는 사람인지 확신하지 못했다.

이 세계의 언어를 충분히 익힐 수 있을까.

이 세계의 문제를 제대로 풀 수 있을까.

이 세계의 책임을 감당할 수 있을까.

공학을 공부했던 시간은 이제 과거가 되는 걸까.

그런 질문들이 오래 남아 있었다.

그러던 어느 날, 나는 오래된 마법진 하나를 다시 펼쳐보게 되었다.

그 마법진은 원래 세계로 돌아갈 수 있는 마법진이었다.

토목공학의 세계로 이어지는 길. 하중을 계산하고, 구조물을 해석하고, 익숙한 언어로 문제를 풀던 세계. 한때는 너무 당연했고, 의세계에 온 뒤로는 멀어진 줄 알았던 세계였다.

나는 그 마법진을 사용했다.

그리고 정말로, 원래 세계에 다녀왔다.

토목공학과 연구실에 갔다. 교수님을 뵈었고, 그 세계의 언어로 다시 대화를 나누었다. 구조, 해석, 모델, 재료, 설계, 연구. 오랜만에 듣는 말들이 있었다. 이상하게 낯설면서도 익숙한 단어들이었다.

완전히 떠났던 줄 알았던 세계가 아직 그곳에 있었다.

그때 알게 되었다.

그 마법진은 한 번만 쓸 수 있는 귀환 주문이 아니었다.

원하면 다시 펼칠 수 있는 문이었다. 나는 의세계에 갇힌 것도 아니

었고, 원래 세계를 완전히 잃은 것도 아니었다. 두 세계는 서로를 지우는 선택지가 아니었다. 오히려 나는 두 세계 사이를 오가며 생각할 수 있었다.

돌아갈 것인가.

남을 것인가.

처음에는 질문이 그렇게 생겼다.

하지만 시간이 지나자 질문이 조금 바뀌었다.

어느 한 세계만 선택해야 할까.

아니면 두 세계 사이에 길을 만들 수 있을까.

일주일만 지났다.

실습 병원의 복도에는 한 사람이 들어섰다.

흰 의사복을 입고, 목에는 청진기를 걸고, 손에는 노트북과 태블릿을 든 사람이었다. 아직 완성된 의사는 아니었다. 모르는 것이 많았고, 자주 긴장했고, 가끔은 시험 문제 앞에서 마왕을 떠올렸다.

하지만 그는 더 이상 완전히 낯선 세계에 떨어진 사람은 아니었다.

그 사람의 명찰에는 이렇게 적혀 있었다.

방지송.

나는 결국 원래 세계로 돌아가기를 포기한 것도 아니고, 의세계에만 남기로 한 것도 아니었다.

나는 두 세계를 왕복하기로 했다.

공학의 세계에서 배운 구조와 모델의 감각을 들고 의세계로 들어가고, 의세계에서 마주한 환자와 기록과 판단의 문제를 다시 공학의 언어로 생각해보기로 했다. 한쪽을 버리고 다른 한쪽을 택하는 대신,

두 세계 사이에 작업대를 하나 놓기로 했다.

그 작업대 위에는 시험지와 노트북이 함께 놓여 있다.

가이드라인과 코드가 함께 놓여 있다.

의료 기록과 모델 곡선이 함께 놓여 있다.

환자의 말과 검사 수치가 함께 놓여 있다.

예측과 책임이 함께 놓여 있다.

이 책은 그 작업대 위에서 쓴 기록이다.

처음에는 살아남기 위해 구조를 만들었다.

의대에 처음 들어왔을 때, 의학은 거대한 암기 세계처럼 보였다. 질병 이름, 진단 기준, 검사 수치, 약물 용량, 치료 알고리즘. 외워야 할 것이 많았고, 시험은 끊임없이 다가왔다. 무엇을 먼저 봐야 하는지, 무엇이 중요한지, 어떤 개념이 어디에 연결되는지 알기 어려웠다

그래서 구조가 필요했다.

암기를 부정하려는 것이 아니었다. 암기는 필요했다. 다만 암기가 어디에 놓이는지 알고 싶었다. 이해 없이 쌓인 암기는 쉽게 무너졌다. 반대로 구조 위에 놓인 암기는 오래 남았다. 그래서 공부를 할 때마다 묻게 되었다.

이건 어떤 구조로 이루어져 있을까.

그 질문은 공부에서 시작했지만, 곧 다른 곳으로 변했다.

설명도 구조의 문제였다.

가르침도 구조의 문제였다.

프로젝트도 구조의 문제였다.

의료 기록도 구조의 문제였다.

약물의 변화도 구조의 문제였다.

AI의 추천도 구조의 문제였다.

임상 판단도 구조의 문제였다.

생각해보면 나는 계속 같은 일을 하고 있었다.

감각을 수치로 바꾸고, 수치를 관계로 묶고, 관계를 모델로 만들고, 모델을 다시 설명 가능한 판단으로 가져가려 했다. 목소리의 변화를 음성 지표로 보고 싶었고, 하루의 상태를 에너지 곡선으로 보고 싶었고, 호르몬 농도를 시간축 위에서 보고 싶었다. 의료 텍스트를 연구 가능한 데이터로 정리하고 싶었고, AI의 추천을 검증 가능한 구조 안에 넣고 싶었다.

그 과정에서 여러 도구와 프로젝트가 생겼다.

어떤 것은 작게 작동했고, 어떤 것은 아직 가설에 가깝고, 어떤 것은 더 다듬어야 한다. 어떤 모델은 너무 단순했고, 어떤 구조는 현실을 충분히 담지 못했다. 어떤 아이디어는 멋있어 보였지만 실제 데이터와 만나자 바로 약점을 드러냈다.

처음에는 그게 실패처럼 느껴질 때도 있었다.

하지만 시간이 지나면서 조금 다르게 생각하게 되었다.

모델은 처음부터 맞지 않는다.

모델은 현실을 완벽하게 복제하기 위해 존재하지 않는다. 모델은 복잡한 현실에서 판단에 필요한 관계를 꺼내기 위해 존재한다. 좋은 모델은 정답을 대신 주지 않는다. 대신 질문을 더 명확하게 만든다. 무엇을 알고 있는지.

무엇을 모르는지.

어떤 변수가 중요한지.

어떤 가정을 하고 있는지.

어디서부터 예측이 흔들릴 수 있는지.

좋은 모델은 결과만 보여주지 않는다. 불확실성의 위치도 함께 보여 준다.

의료에서는 이 점이 특히 중요하다.

환자는 데이터 포인트가 아니다.

검사 수치는 환자의 전체 상태를 대신하지 못한다.

의료 기록은 처음부터 연구용 데이터가 아니다.

AI의 추천은 의사의 판단을 대체하지 못한다.

그래서 의료를 구조화한다는 것은 의료를 차갑게 만드는 일이 아니다. 환자를 숫자로 줄이는 일도 아니다. 의사의 판단을 기계에 넘기는 일도 아니다.

오히려 반대에 가깝다.

더 잘 듣기 위해 구조가 필요하다.

더 정확히 보기 위해 구조가 필요하다.

더 정직하게 모른다고 말하기 위해 구조가 필요하다.

그리고 가능한 만큼 더 잘 설명하기 위해 구조가 필요하다.

의학을 공부하면서 가장 자주 마주친 것은 불확실성이었다.

환자는 교과서처럼 말하지 않는다. 검사 수치는 언제나 명확한 답을 주지 않는다. 같은 진단명 아래에서도 실제 환자의 상태는 조금씩 다르고, 같은 약을 써도 반응은 사람마다 다르다. 가이드라인은 기준

을 주지만, 실제 진료에서는 그 기준 사이의 회색지대를 자주 만나게 된다.

그 회색지대에서 필요한 것은 단순한 확신이 아니다.

오히려 확신의 모양을 점검하는 일이다.

이 판단은 어떤 정보 위에서 있는가.

이 정보는 얼마나 믿을 수 있는가.

빠진 변수는 없는가.

환자의 맥락은 충분히 반영되었는가.

이 선택의 이득과 위험은 무엇인가.

그리고 이 판단을 환자에게 설명할 수 있는가.

결국 임상에서 중요한 것은 머릿속에서 답을 고르는 것만이 아니다.

그 답이 어떤 가정 위에 있는지 알고, 어디까지 믿을 수 있는지 설명하고, 환자의 맥락 안에서 다시 조정하는 일이다.

판단은 머릿속에서 맞는 것만으로 완성되지 않는다.

판단은 설명될 때 완성된다.

환자가 이해할 수 있는 언어로 바뀌고, 선택의 이유와 한계가 드러나고, 불확실성이 숨겨지지 않을 때 비로소 판단은 임상적인 것이 된다. 설명할 수 없는 판단은 아무리 정교해 보여도 위험하다. 반대로 잘 설명된 판단은 완벽하지 않더라도 함께 검토될 수 있다.

그래서 이 책의 끝에서 나는 다시 책임을 생각한다.

모델은 가설을 제시한다.

AI는 후보를 보여준다.

시스템은 구조를 정리한다.

하지만 환자 앞에서 설명하고, 맥락을 판단하고, 책임지는 사람은 여전히 의사다.

앞으로 의사가 되면 나는 더 많은 불확실성 앞에 서게 될 것이다. 책상 위에서 만든 모델보다 훨씬 복잡한 환자를 만나게 될 것이고, 예측에서 벗어나는 상황을 마주하게 될 것이다. 지금 만든 구조가 충분하지 않다는 사실도 계속 알게 될 것이다.

아마 계속 틀릴 것이다.

하지만 모델이 처음부터 맞지 않는다는 것을 안다면, 틀림은 끝이 아니라 시작이 될 수 있다. 다시 관찰하고, 다시 수정하고, 다시 설명하고, 다시 책임지는 과정이 남는다.

이 책은 완성된 결론이 아니다.

의세계에 적응해가는 한 공학도의 기록이고, 의사가 되어가는 과정에서 남긴 초판의 지도에 가깝다. 시간이 지나면 일부 생각은 바뀔 것이고, 일부 모델은 버려질 것이고, 더 나은 질문이 생길 것이다.

그래도 지금 이 시점에서 남겨두고 싶은 말은 있다.

의학과 공학은 생각보다 멀지 않았다.

감각과 수치는 서로를 지울 필요가 없었다.

모델과 환자는 같은 것이 아니지만, 모델은 환자를 더 잘 이해하기 위한 도구가 될 수 있었다.

예측은 책임을 대신하지 않지만, 책임 있는 설명을 준비하는 데 도움이 될 수 있었다.

의세계에 전생한 지 4년이 지났다.

처음에는 낯선 마법진 앞에서 얼어붙었고, 마도구 하나 제대로 다루

지 못했으며, 마왕이 내는 문제 앞에서 자주 당황했다. 그러다 조금씩 이 세계의 언어를 배웠고, 동료들을 만났고, 원래 세계로 돌아가는 마법진도 다시 펼쳐보았다.

그리고 알게 되었다.

나는 한 세계를 버리고 다른 세계를 선택한 것이 아니었다.

나는 두 세계 사이를 오가며, 그 사이에 놓인 문제들을 바라보는 사람이 되었다. 공학의 구조와 의학의 불확실성 사이에서, 데이터와 환자 사이에서, 모델과 책임 사이에서 계속 질문하는 사람이 되었다.

아마 앞으로도 마왕은 성실하게 문제를 낼 것이다.

그리고 나는 아마 앞으로도, 꽤 오래, 자꾸 모델을 만들 것이다.

부록 (Appendix)

하루 30분, 1년 반. JLPT N1까지

일본어는 큰 계획 없이 시작했습니다. 다만 한 가지는 지켰습니다. 매일 30분. 길게 하지 않았고, 대신 끊지 않았습니다. 그렇게 1년 반 동안 N3, N2, N1을 순서대로 준비했고, 결국 N1에 합격했습니다. 특별한 재능이 있었다기보다는, 구조를 만들고 반복했을 뿐입니다.

2025년 1회 시험성적 조회

성적결과

영문이름	JISONG BANG	수험번호	
생년월일		응시레벨	N1
결과	합격 (PASSED)	인정번호	

JLPT N1 시험 결과

1) 하루 30분을 유지한 방법

처음부터 N1을 목표로 하지 않았습니다. N3부터 차례대로 봤습니다. 단계별로 넘어가되, 한 번에 많은 시간을 쓰지 않았습니다. 특히 지하철을 타야 하거나, 여유시간이 생긴다면 그날의 Anki를 먼저 돌리고 휴대폰으로 교재도 읽었습니다.

하루 30분은 부담이 없고, 다음 날 다시 읽을 수 있습니다. 교재는 해커스 JLPT 책 세 권을 사용했습니다. N3, N2, N1 각각 한 권씩 끝까지 풀었습니다. 문제를 많이 풀기보다는, 틀린 문제를 구조화하는 데 시간을 썼습니다.

왜 이 문법이 여기 들어가는지, 왜 이 표현이 자연스러운지 이해하려고 했습니다. 짧은 시간이라도 매일 반복하니, 감각이 끊기지 않았습니다. 언어는 결국 누적이라는 걸 체감했습니다.

2) 한자와 단어는 구조로 외우기

한자는 따로 정리하지 않으면 끝이 없습니다. 그래서 무작정 외우는 대신 구조로 접근했습니다.

<https://nihongokanji.com>

[일본어 한자 공부방

일본어 상용한자의 모든 것을 무료로 자유롭게 공부하실 수 있는 사이트입니다. <https://nihongokanji.com>](<https://nihongokanji.com>)

위의 사이트를 활용했습니다. 한자를 부수 단위로 나눠 보여주고, 구조를 설명해줘서 좋았습니다. 비슷한 형태끼리 묶어보니 암기가 훨씬 수월해졌습니다. 개별 글자를 외우는 게 아니라, 조합을 이해하는 느낌이었습니다.

단어와 한자는 Anki로 관리했습니다. 텍은 직접 만들지 않고 인터넷에서 공유된 텍을 받아 사용했습니다. 대신 무작정 돌리지 않고, 헛갈리는 단어는 따로 태그를 달아 반복했습니다. 이미 알고 있는 단어를 반복하는 데 시간을 쓰지 않으려고 했습니다.

- 한자는 부수 구조로 이해하고
- 단어는 반복으로 고정하고
- 문장은 문제 풀이로 감각을 유지했습니다.

이 세 가지를 분리해서 관리했습니다.

3) N1 합격 이후에 느낀 점

N1을 준비할 때는 양이 많아서 부담이 컸습니다. 하지만 이미 N3, N2를 거치며 쌓인 구조가 있었습니다. 완전히 새로운 공부가 아니라, 확장에 가까웠습니다.

시험장에서 모르는 단어가 나와도, 한자 구조를 보면 대략적인 의미를 추측할 수 있었습니다. 문법도 처음 보는 표현이라기보다, 익숙한 패턴의 변형처럼 느껴졌습니다. 결국 N1 합격은 단기간 집중의 결과라기보다, 하루 30분씩 쌓인 반복의 결과였습니다.

언어 공부도 결국 구조와 반복의 문제라는 걸 배웠습니다. 한자를 부수로 나누고, 단어를 톱으로 관리하고, 문장을 문제 속에서 확인하는 방식. 이 방식이 저에게는 가장 안정적이었습니다. 앞으로 새로운 언어를 배운다고 해도, 같은 방식으로 접근할 것 같습니다.

Obsidian PKM, 연결을 관리하기

Obsidian PKM: 기억이 아니라 연결을 관리하기 메모를 많이 한다고 지식이 되는 것은 아닙니다. 파일이 많아졌다고 생각이 정리되는 것도 아닙니다. 처음에는 저도 메모를 저장하는 일이 중요하다고 생각했습니다. 강의 내용을 적고, 실습 중 들은 말을 적고, 연구 아이디어를 적고, 프로젝트 계획을 적고, 브런치 글감을 적었습니다.

적어두면 잊어버리지 않을 것 같았습니다. 하지만 시간이 지나면 문제가 생깁니다. 메모는 있는데 다시 찾기 어렵습니다. 찾아도 왜 적었는지 기억나지 않습니다. 비슷한 주제의 문서가 여러 개 생깁니다.

어떤 것은 현재 쓰는 문서이고, 어떤 것은 과거 기록인데 섞여 있습니다. AI에게 맥락으로 보여주고 싶어도 어떤 문서를 읽혀야 할지 애매합니다.

그때 알게 됩니다.

PKM은 모든 것을 저장하는 시스템이 아닙니다. 생각이 다시 연결될 수 있게 만드는 구조입니다. Obsidian을 쓰는 이유도 여기에 있습니다. 저에게 Obsidian은 기억 보조 앱이라기보다, 생각의 연결을 관리하는 개인 인프라에 가깝습니다.

Obsidian vault 전체 구조

형태: iCloud 기반 Obsidian PKM

역할: 공부, 연구, 프로젝트, 글쓰기, 인간관계, 자기 이해, AI 맥락 저장소

핵심 구조: 문서 저장 → 폴더 경계 → index → 링크 → frontmatter → AI context routing

연결 프로젝트: Obsidian_jsbang, 브런치 글쓰기, 의대 공부, 연구 아이디어 정리

주의: 플러그인 추천이나 생산성 도구 소개가 아니라, 생각을 다시 찾고 연결하기 위한 구조 운영 기록

1) 메모는 많아질수록 흐려진다

메모의 역설이 있습니다.

메모를 적게 하면 잊어버립니다.

하지만 메모를 많이 하면 흩어집니다. 처음에는 아무 곳이나 적어도 괜찮습니다. 문서가 몇 개 없을 때는 검색으로도 충분합니다. 파일 이름이 조금 이상해도 기억으로 찾을 수 있습니다. 하지만 메모가 수백 개가 되면 상황이 달라집니다.

의대 공부 문서가 있고, 실습 기록이 있고, 연구 아이디어가 있고, 개발 프로젝트 문서가 있고, 브런치 글감이 있고, 인간관계 정리 문서가 있고, 나 자신을 분석한 문서가 있습니다. 이 모든 것이 한 폴더 안에 있으면 메모는 많아지지만 구조는 약해집니다. 나중에 다시 찾으려고 하면 기억해야 할 것이 너무 많아집니다.

어떤 문서가 최신인지. 어떤 문서는 보관용인지. 어떤 문서는 지금 운영 중인 문서인지. 어떤 문서는 AI에게 읽혀도 되는지. 어떤 문서는 원본 보존용인지.

이 구분이 없으면 PKM은 금방 저장소가 됩니다. 자료는 많은데 다시 연결되지 않는 저장소. 그래서 저는 Obsidian을 단순한 메모 앱으로 쓰기보다, 하나의 운영 체제로 다루기 시작했습니다.

2) 폴더는 분류가 아니라 경계다

Obsidian은 링크가 강한 도구입니다. 그래서 어떤 사람은 폴더보다 링크가 더 중요하다고 말합니다.

저도 어느 정도 동의합니다.

하지만 폴더가 필요 없는 것은 아닙니다. 폴더는 단순 분류가 아니라 경계입니다. 제 vault에서는 최상위 폴더가 역할별로 나뉩니다.

01_지송은 타인이 없어도 성립하는 자기 모델과 삶의 방향을 둡니다. 02_인간관계는 특정 사람, 가족, 집단, 관계 운영 문서를 둡니다. 03_학업은 시험 준비, 실습, 조장 역할, 케이스 발표 같은 현재 수행 중인 학업 문서를 둡니다. 04_개발은 코드, 시스템, 구현, 배포, 장애 대응, 자동화처럼 실제 build와 운영에 가까운 문서를 둡니다. 05_프로젝트는 연구 설계, 글쓰기, 주거와 생활 실행처럼 코드 바깥의 실행 프로젝트를 둡니다. 90_아카이브는 원본 보존과 과거 판단 복기를 위한 영역입니다. 99_System은 vault 운영 규칙, AI 라우팅, 템플릿, 검사 규칙처럼 내용이 아니라 구조를 관리하는 문서를 둡니다.

이렇게 나누면 좋은 점이 있습니다. 문서를 어디에 둘지 결정할 때 그 문서의 역할을 먼저 생각하게 됩니다.

이것은 나 자신에 대한 문서인가. 관계에 대한 문서인가. 학업 수행 문서인가. 개발 운영 문서인가. 프로젝트 실행 문서인가. 아카이브로 보존해야 하는가. 아니면 시스템 운영 규칙인가.

폴더는 단순한 보관함이 아닙니다.

문서의 책임과 위치를 정하는 경계입니다.

3) index는 생각의 입구다

문서가 많아지면 검색만으로는 부족합니다. 검색은 단어를 알고 있을 때 강합니다. 하지만 어떤 생각은 단어보다 맥락으로 찾아야 합니다.

예를 들어 “브런치 글감”이라는 단어를 검색하면 관련 문서를 찾을 수 있습니다. 하지만 “요즘 내가 쓰려던 글의 전체 흐름이 뭐였지?”라는 질문은 검색어 하나로 해결되지 않습니다. 그럴 때 필요한 것이 index입니다.

index는 단순 목차가 아닙니다.

생각의 입구입니다.

어떤 폴더에 어떤 문서가 있는지, 무엇을 먼저 읽어야 하는지, 현재 active 문서는 무엇인지, archive 문서는 어디에 있는지, AI에게 맥락을 줄 때 어떤 문서를 먼저 보여줘야 하는지 정리해두는 문서입니다. Obsidian에서 index가 생기면 vault가 달라집니다. 문서가 파일 목록으로만 존재하지 않습니다.

길이 생깁니다.

처음 들어왔을 때 어디서 시작해야 하는지 보이고, 지금 중요한 문서가 무엇인지 보이고, 오래된 문서는 어디로 보내야 하는지 보입니다. PKM에서 중요한 것은 모든 문서를 완벽히 기억하는 것이 아닙니다. 다시 들어갈 수 있는 입구를 만드는 것입니다. index는 문서 목록이 아니라 생각의 입구다.

4) 링크는 기억이 아니라 관계를 만든다

Obsidian의 장점은 링크입니다. 하지만 링크도 아무렇게나 걸면 의미가 약해집니다. 중요한 것은 문서 사이의 관계입니다.

의대 공부 문서가 연구 아이디어와 연결됩니다. 연구 아이디어는 브

런치 글감과 연결됩니다. 브런치 글감은 프로젝트 문서와 연결됩니다. 프로젝트 문서는 GitHub 레포와 연결됩니다. 개발 문서는 다시 개인 인프라 글로 이어집니다.

이 연결이 생기면 각 문서는 혼자 존재하지 않습니다. 예를 들어 GAHT 관련 연구 아이디어는 의학 문서이기도 하고, 연구 프로젝트이기도 하고, 브런치 글감이기도 하고, 웹앱 아이디어이기도 합니다. 하나의 문서를 하나의 폴더에만 가둬두면 그 다층성이 사라집니다. 링크는 이 문제를 해결합니다.

문서는 한 위치에 놓이지만, 여러 맥락으로 다시 연결될 수 있습니다. 이것이 Obsidian의 핵심이라고 생각합니다. 기억을 대신하는 것이 아니라, 관계를 보이게 하는 것. 메모를 다시 떠올리는 것이 아니라, 메모가 어떤 생각과 이어지는지 보여주는 것.

5) frontmatter는 작은 운영 데이터다

Markdown 문서 위에는 frontmatter가 붙습니다. 처음에는 이것이 조금 귀찮게 느껴질 수 있습니다.

title. aliases. tags. created. updated. status. type.

이런 값을 매번 넣는 일이 왜 필요한지 의문이 들 수도 있습니다.

하지만 문서가 많아질수록 frontmatter는 중요해집니다.

문서가 초안인지, active인지, archive인지. 글쓰기 문서인지, 프로젝트 문서인지, 시스템 문서인지. 어떤 이름으로도 검색될 수 있는지. 언제 만들어졌고, 언제 수정되었는지.

이런 정보는 본문 내용과는 다릅니다. 본문이 생각의 내용이라면, frontmatter는 그 생각을 관리하기 위한 운영 데이터입니다. 특히 AI와 함께 쓸 때 이 차이가 중요해집니다.

AI에게 모든 문서를 다 읽히는 것은 비효율적입니다. 어떤 문서는 항상 읽어야 하고, 어떤 문서는 주제에 따라 읽어야 하며, 어떤 문서는 보존용이라 기본 로딩에서 제외해야 합니다. 그 기준을 만들려면 문서에 메타데이터가 필요합니다. frontmatter는 그래서 단순한 꾸밈이 아닙니다.

생각을 다시 불러오기 위한 작은 운영 레이어입니다.

6) AI와 함께 쓰려면 맥락도 관리해야 한다

요즘 메모는 사람만 읽지 않습니다.

AI도 읽습니다.

브런치 글을 쓸 때도, 연구 아이디어를 정리할 때도, 프로젝트 문서를 다시 잡을 때도 AI에게 이전 맥락을 보여주면 훨씬 좋은 결과가 나옵니다.

하지만 여기서도 문제가 생깁니다.

AI에게 어떤 문서를 읽힐 것인가. 모든 문서를 다 넣을 수는 없습니다. 오래된 문서와 최신 문서가 섞이면 오히려 혼란스럽습니다. 보존용 아카이브와 현재 운영 문서가 섞이면 판단이 흐려집니다. 그래서 AI context routing이 필요해집니다.

먼저 읽을 문서. 현재 active 문서. 주제별로 추가로 읽을 문서. 기

본 로딩에서 제외할 archive 문서.

이 기준이 있어야 AI에게도 안정적인 맥락을 줄 수 있습니다. 저에게 Obsidian vault는 단순히 제가 다시 읽기 위한 공간이 아닙니다.

AI와 함께 작업할 때, 제가 어떤 사람이고, 어떤 프로젝트를 하고 있고, 어떤 글을 쓰고 있고, 어떤 기준으로 판단하는지를 전달하는 맥락 저장소이기도 합니다. 이 점에서 PKM은 개인 기억 보조를 넘어섭니다.

사람과 AI가 함께 읽을 수 있는 사고 인프라가 됩니다. AI에게도 모든 기억이 아니라, 필요한 맥락을 읽히는 구조가 필요하다.

7) 아카이브는 버리는 곳이 아니다

문서가 많아지면 버릴 것과 남길 것을 나눠야 합니다. 하지만 모든 오래된 문서를 지울 수는 없습니다. 과거의 판단, 원본 대화, 오래된 계획, 당시의 감정과 맥락은 나중에 다시 중요해질 수 있습니다.

그래서 아카이브가 필요합니다.

다만 아카이브는 현재 운영 문서와 섞이면 안 됩니다. 현재 쓰는 문서와 과거 보존 문서가 같은 위치에 있으면 어느 쪽이 최신 판단인지 헷갈립니다.

아카이브는 버리는 곳이 아닙니다.

원본을 보존하되, 현재의 판단과 섞이지 않게 분리하는 곳입니다. 이 구분이 있어야 vault가 오래 버팁니다. 지금 쓰는 문서는 가볍게 움직일 수 있어야 하고, 과거 자료는 필요할 때 추적할 수 있어야 합

니다. PKM은 모든 것을 현재형으로 붙잡는 시스템이 아닙니다.
현재 운영 문서와 과거 보존 문서를 구분하는 시스템입니다.

8) 의대 공부도 연결로 관리된다

의대 공부는 양이 많습니다.

질병 이름도 많고, 약도 많고, 검사도 많고, 진단 기준도 많고, 치료 알고리즘도 많습니다. 처음에는 많이 외우는 것이 중요해 보입니다. 물론 외워야 합니다.

하지만 외우기만 하면 금방 한계가 옵니다. 질병은 병태생리와 연결되어 있고, 검사는 진단 기준과 연결되어 있고, 약물은 기전과 부작용과 연결되어 있고, 증상은 감별진단과 연결되어 있습니다.

결국 공부도 연결입니다.

Anki는 기억을 관리하는 데 좋습니다. 하지만 Obsidian은 연결을 관리하는 데 좋습니다. 예를 들어 내분비를 공부하다가 호르몬 농도 곡선에 관심이 생기고, 그것이 EstroFrame이나 PharmaFrame 같은 프로젝트로 이어지고, 다시 브런치 글감으로 이어질 수 있습니다.

이런 흐름은 단순 암기장에서는 잘 보이지 않습니다.

Obsidian에서는 하나의 개념이 공부, 연구, 프로젝트, 글쓰기로 확장되는 경로가 남습니다. 그래서 저에게 Obsidian은 의대 공부 노트이면서, 동시에 연구 아이디어 저장소이고, 브런치 글감 저장소이고, 개발 프로젝트의 설계 문서입니다. 하나의 지식이 다른 작업으로 넘어가는 통로입니다.

9) PKM은 기억 보조가 아니라 사고 인프라다

PKM을 처음 시작하면 어떻게 잘 저장할지 고민하게 됩니다. 어떤 폴더 구조가 좋은지, 어떤 태그를 쓸지, 어떤 플러그인을 설치할지, 어떤 템플릿을 만들지 고민합니다.

이 고민은 필요합니다.

하지만 어느 순간부터는 질문이 바뀝니다. 어떻게 저장할 것인가가 아니라, 어떻게 다시 연결할 것인가. 어떤 문서가 현재 판단이고, 어떤 문서가 원본 보존인지. 어떤 문서를 AI에게 읽힐 것인지, 어떤 문서는 맥락에서 제외할 것인지.

어떤 생각이 공부에서 연구로, 연구에서 프로젝트로, 프로젝트에서 글쓰기로 넘어가는지.

이 질문들이 중요해집니다.

Obsidian은 그 질문을 다루기 좋은 도구입니다. 하지만 도구 자체가 답은 아닙니다. 폴더, 링크, index, frontmatter, archive, AI context routing이 하나의 운영 체제로 묶일 때 비로소 PKM이 됩니다. 저에게 PKM은 기억을 대신하는 시스템이 아닙니다.

생각이 다시 연결될 수 있게 만드는 구조입니다.

기억은 사라질 수 있습니다.

하지만 연결이 남아 있으면 다시 들어갈 수 있습니다. 그리고 다시 들어갈 수 있는 생각은 다음 공부가 되고, 다음 글이 되고, 다음 프로젝트가 됩니다. 그래서 Obsidian PKM은 저에게 단순한 노트 앱

이 아닙니다. 생각을 저장하는 곳이 아니라, 생각이 다시 움직이게 하는 인프라입니다.

도메인을 산다는 것

처음에는 앱을 하나씩 만들었습니다. 필요한 기능이 있으면 만들고, 작은 실험이 떠오르면 레포를 만들고, 어떤 프로젝트는 Cloud Run에 올리고, 어떤 프로젝트는 GitHub Pages에 올렸습니다.

처음에는 그것으로 충분했습니다.

작동하면 됐고, 내가 쓸 수 있으면 됐고, 주소가 조금 길어도 크게 상관없었습니다. 하지만 프로젝트가 하나씩 늘어나면서 문제가 생겼습니다.

앱은 있는데 어디에 있는지 헷갈렸습니다. 레포는 있는데 무엇을 위한 것인지 다시 설명해야 했습니다. 포트폴리오는 따로 있고, 사진은 또 다른 곳에 있고, 개인 클라우드는 별도의 주소에 있었습니다. 작업은 분명히 쌓이고 있었는데, 그 작업들이 하나의 공간 안에 들어와 있지는 않았습니.

그때 도메인을 샀습니다.

jisong.dev

처음에는 그냥 주소라고 생각했습니다. 하지만 막상 도메인을 사고 나니, 이것이 단순한 주소 이상의 의미를 가진다는 것을 알게 되었습니다. 도메인을 산다는 것은 인터넷 위에 내 작업의 좌표계를 하나 만드는 일이었습니다.

jisong.dev 첫 화면

주소: jisong.dev

연결 프로젝트: jisong.dev, photo.jisong.dev, Jisong Cloud

형태: 개인 도메인 기반 작업 공간

역할: 포트폴리오, 프로젝트 허브, 사진 아카이브, 개인 인프라의 진입점

핵심 구조: 흩어진 작업 → 이름 부여 → 경계 설정 → 주소 연결 → 재사용 가능한 공간화

주의: 도메인은 자기소개용 장식이 아니라, 작업과 정체성을 정렬하는 개인 인프라의 일부

1) 주소가 생기면 경계가 생긴다

도메인을 사기 전에도 프로젝트는 있었습니다. GitHub 레포가 있었고, 배포된 앱이 있었고, 개인적으로 쓰는 도구도 있었습니다. 하지만 그때의 프로젝트들은 조금 흩어져 있었습니다. 각자 다른 주소를 가지고 있었고, 어떤 것은 임시 배포 링크에 가까웠고, 어떤 것은 로컬에서만 돌아갔고, 어떤 것은 이름은 있지만 외부에서 접근하기 어려웠습니다.

주소가 없다고 해서 프로젝트가 없는 것은 아닙니다. 하지만 주소가 없으면 경계가 흐립니다. 이것이 어디에 속한 작업인지, 어디까지가 하나의 프로젝트인지, 어떤 이름으로 기억해야 하는지 불분명해집니다. 도메인은 그 흐릿한 경계에 선을 긋습니다.

jisong.dev 아래에 놓인다는 것은 이 작업이 내 개인 개발 공간의 일부라는 뜻이 됩니다.

photo.jisong.dev라는 이름이 붙는 순간, 사진은 그냥 이미지 파일 묶음이 아니라 하나의 아카이브가 됩니다. Jisong Cloud에 주소가 붙으면, 그것은 더 이상 로컬에서만 쓰는 실험이 아니라 인터넷 위에서 접근 가능한 개인 작업대가 됩니다.

주소는 단순한 링크가 아닙니다.

작업의 경계입니다.

2) 이름이 생기면 실험은 공간이 된다

처음 만드는 프로젝트는 대개 임시적입니다. 일단 만들어봅니다. 작동하는지 확인합니다. 필요하면 고칩니다. 별로면 버립니다.

이 단계에서는 이름이 그렇게 중요하지 않습니다. 하지만 어느 순간 부터는 이름이 필요해집니다.

반복해서 쓰기 시작할 때. 다른 프로젝트와 연결되기 시작할 때. 다시 돌아와서 고치기 시작할 때. 누군가에게 설명해야 할 때.

이름이 생기면 프로젝트의 상태가 달라집니다. 그전까지는 “예전에 만든 그거”였던 것이 이제는 하나의 공간이 됩니다. jisong.dev는 그런 의미에서 제 작업의 루트 디렉토리 같은 이름입니다.

그 아래에 포트폴리오가 있고, 사진이 있고, 개인용 클라우드가 있고, 앞으로 생길 다른 프로젝트가 들어올 수 있습니다. 이름은 단순히 보기 좋게 붙이는 라벨이 아닙니다. 이름은 작업을 다시 찾을 수 있

게 하고, 다른 작업과 연결할 수 있게 하고, 외부에 설명할 수 있게 만듭니다. 프로젝트가 이름을 얻는 순간, 그것은 임시 실험에서 하나의 장소가 됩니다.

3) 도메인은 작업의 좌표계다

좌표계가 없으면 위치를 설명하기 어렵습니다. 어떤 점이 어디에 있는지 말하려면 기준이 필요합니다.

도메인도 비슷합니다.

인터넷 위에는 수많은 링크가 있습니다. GitHub 주소도 있고, Cloud Run 주소도 있고, 임시 배포 주소도 있고, 브랜치 주소도 있고, 사진 공유 링크도 있습니다. 각 링크는 존재하지만, 그 자체로는 하나의 세계를 만들지 않습니다. 도메인은 이 흩어진 링크들에 기준점을 줍니다.

jisong.dev라는 기준점이 생기면, 각 프로젝트는 그 기준점에 대해 위치를 가질 수 있습니다.

메인 포트폴리오는 루트에 둡니다. 사진은 photo.jisong.dev에 둡니다. 개인 클라우드는 별도의 하위 도메인이나 연결 주소에 둡니다.

다른 프로젝트들도 필요하면 각각의 이름을 가질 수 있습니다.

이렇게 되면 작업들이 단순히 나열되지 않습니다.

정렬됩니다.

도메인을 산다는 것은 웹 주소 하나를 소유하는 일이기도 하지만, 동시에 내 작업들의 좌표계를 만드는 일입니다. 어디에 무엇을 둘

것인가. 무엇을 공개하고, 무엇을 닫아둘 것인가. 어떤 프로젝트는 포트폴리오로 보이고, 어떤 프로젝트는 개인 도구로 남길 것인가. 이 질문들이 도메인 위에서 정리됩니다.

4) photo.jisong.dev는 사진 저장소가 아니다

사진도 비슷했습니다.

처음에는 그냥 찍었습니다.

좋아 보이는 장면을 찍고, 빛이 예쁘면 찍고, 선과 구조가 마음에 들면 찍었습니다.

사진 파일은 계속 쌓였습니다.

하지만 파일로만 남아 있으면 사진은 금방 흩어집니다. 휴대폰 앨범 안에 있고, 외장하드 어딘가에 있고, 가끔 SNS에 올리고, 다시 찾으려면 시간이 걸립니다.

photo.jisong.dev라는 이름을 붙이면 상황이 조금 달라집니다. 사진은 단순한 파일 묶음이 아니라 하나의 아카이브가 됩니다.

무엇을 찍는 사람인지, 어떤 분위기를 좋아하는지, 어떤 구조를 보는지 조금 더 분명해집니다. 저에게 사진은 감성의 기록이기도 하지만, 동시에 구조를 보는 일이기도 합니다.

빛. 선. 공간. 여백. 반복. 균형. 이런 것들을 모아두는 공간으로서 photo.jisong.dev는 단순한 사진 저장소가 아니라 시각적 사고의 한 영역이 됩니다. 도메인은 여기서도 같은 역할을 합니다.

흩어진 파일에 이름을 주고, 그 이름 아래에 경계를 만들고, 경계 안

에서 다시 정렬하게 합니다.

photo.jisong.dev는 사진 파일 묶음이 아니라, 시각적 사고를 정렬하는 공간이다.

5) 개인 클라우드는 작업대가 되고, 도메인은 문패가 된다

Jisong Cloud를 만들면서도 비슷한 생각을 했습니다. 처음에는 파일 업로드와 메모장, 작은 도구를 모은 개인용 앱이었습니다. 하지만 시간이 지나면서 Jisong Cloud는 단순 저장소에서 작업대로 바뀌었습니다. 파일을 올리고, 메모를 쓰고, AI로 분석하고, Markdown을 PDF로 바꾸고, 결과를 다시 저장하는 흐름이 생겼습니다.

이런 작업대에도 문패가 필요합니다. 내가 어디로 들어가야 하는지, 이 작업대가 무엇을 위한 공간인지, 어디까지가 개인 도구이고 어디부터가 공개 프로젝트인지 정리해주는 이름이 필요합니다.

도메인은 그 문패 역할을 합니다.

주소가 생기면 접속이 쉬워집니다.

하지만 그것보다 더 중요한 것은 작업을 인식하는 방식이 달라진다는 점입니다. 이제 이것은 “어딘가에 배포된 앱”이 아니라 내 작업 환경의 한 부분이 됩니다. 도메인은 프로젝트를 더 그럴듯하게 포장하는 장식이 아닙니다. 프로젝트가 어디에 속하는지를 알려주는 경계입니다.

6) 공개할 것과 닫아둘 것을 나누게 된다

도메인을 갖는다는 것은 무엇을 보여줄지 정하는 일이기도 합니다.

모든 프로젝트를 공개할 필요는 없습니다.

어떤 것은 포트폴리오로 보여줄 수 있습니다. 어떤 것은 설명 글로만 남겨도 됩니다. 어떤 것은 로그인 뒤에 숨겨야 합니다. 어떤 것은 아예 로컬에만 두는 편이 낫습니다.

개인 인프라에서는 이 구분이 중요합니다. 특히 파일, 메모, AI 분석, 개인 기록이 들어가는 도구라면 주소가 생겼다고 해서 모두에게 열어두면 안 됩니다. 도메인은 공개의 수단이지만, 동시에 경계를 설정하게 만드는 장치이기도 합니다.

어떤 하위 도메인은 공개 페이지로 둡니다. 어떤 하위 도메인은 인증 뒤에 둡니다. 어떤 프로젝트는 GitHub에 공개하고, 어떤 프로젝트는 private repository로 남깁니다. 이런 구분을 하다 보면 내 작업의 성격도 더 분명해집니다.

보여주기 위한 작업. 실제로 쓰기 위한 작업. 실험 중인 작업. 아직 이름 붙이기 전의 작업.

도메인은 이 모든 것을 한 번에 공개하라는 뜻이 아닙니다. 오히려 무엇을 공개하고, 무엇을 닫아둘지 정리하게 만듭니다.

7) 도메인은 정체성을 과시하는 도구가 아니다

개인 도메인을 산다고 하면 조금 자기 브랜딩처럼 보일 수 있습니다

틀린 말은 아닙니다.

도메인은 분명히 외부에 보이는 이름입니다. 누군가에게 링크를 줄 수 있고, 포트폴리오에 적을 수 있고, 프로젝트를 소개할 때 사용할 수 있습니다. 하지만 제가 도메인을 산 이유는 정체성을 크게 보여 주기 위해서만은 아니었습니다. 오히려 정체성을 정리하기 위해서였습니다.

의대생. 엔지니어. 글 쓰는 사람. 사진 찍는 사람. 개인 도구를 만드는 사람. 의료 데이터를 구조화하고 싶은 사람.

이런 것들은 따로 보면 조금 흩어져 보입니다. 하지만 제 안에서는 완전히 따로 떨어져 있지 않습니다.

공통점은 구조를 본다는 것입니다.

의학에서도 구조를 보고, 코드에서도 구조를 보고, 사진에서도 구조를 보고, 글에서도 구조를 봅니다. 도메인은 이 흩어진 작업들을 하나의 이름 아래에 느슨하게 묶어줍니다. 나를 설명하기 위한 거창한 선언문이 아니라, 내 작업들이 모이는 좌표계로서 작동합니다.

8) 이름을 붙인다는 것은 책임을 지는 일이다

작업에 이름을 붙이면 조금 부담이 생깁니다.

그전까지는 그냥 실험이었습니다.

잘 안 되면 지워도 되고, 주소가 바뀌어도 상관없고, 설명이 부족해도 크게 문제 되지 않았습니다. 하지만 도메인 아래에 올리면 조금 더 신경 쓰게 됩니다.

첫 화면은 괜찮은지. 링크는 깨지지 않는지. 무엇을 위한 공간인지 설명되는지. 보안 경계는 적절한지. 너무 오래된 프로젝트를 그대로 두고 있지는 않은지.

이 부담이 나쁘지만은 않습니다.

오히려 작업을 관리하게 만듭니다.

도메인은 프로젝트를 완성시켜주지는 않습니다. 하지만 프로젝트를 방치하기 어렵게 만듭니다. 이름이 생긴 작업은 다시 돌아가서 고칠 수 있는 대상이 됩니다. 그래서 도메인을 산다는 것은 단순히 주소를 결제하는 일이 아닙니다.

내 작업에 이름을 붙이고, 그 이름 아래에 어느 정도 책임을 지겠다는 뜻이기도 합니다.

9) 도메인을 산다는 것

도메인을 산다는 것은 인터넷 위에 내 이름을 하나 더 올리는 일이 아닙니다. 그보다는 흩어진 작업을 하나의 기준점 아래에 정렬하는 일에 가깝습니다.

파일은 저장소에 있을 수 있습니다. 코드는 GitHub에 있을 수 있습니다. 앱은 Cloud Run에 있을 수 있습니다. 사진은 별도의 페이지에 있을 수 있습니다. 글은 브런치에 있을 수 있습니다. 하지만 그 모든 것이 완전히 흩어진 상태로만 남아 있으면 작업은 쌓이지만 방향은 흐려집니다.

도메인은 그 작업들 사이에 좌표계를 만듭니다. 무엇을 어디에 둘

것인지. 어떤 이름으로 부를 것인지. 어디까지 공개할 것인지. 무엇을 내 작업의 일부로 받아들일 것인지.

이런 질문에 답하게 만듭니다.

저에게 jisong.dev는 단순한 주소가 아닙니다.

작업의 기준점입니다.

photo.jisong.dev는 단순한 사진 저장소가 아닙니다. 시각적 사고를 모아두는 공간입니다.

Jisong Cloud는 단순한 웹앱이 아닙니다. 반복되는 작업을 올려둔 개인 작업대입니다. 도메인을 산다는 것은 이 모든 것을 하나의 이름 아래에 배치하는 일입니다. 기술적 소유물이 아니라, 작업의 좌표계를 만드는 일. 그리고 좌표계가 생기면 내가 어디에서 있는지도 조금 더 잘 보입니다.

사고 보조 장치, Antigravity IDE

Antigravity IDE: 코딩 도구가 아니라 사고 보조 장치 예전에는 코딩을 한다는 것이 직접 코드를 쓰는 일에 가깝다고 생각했습니다. 파일을 열고, 함수를 만들고, 에러를 고치고, 필요한 기능을 하나씩 구현했습니다. 물론 지금도 코드는 사람이 이해해야 합니다. 하지만 AI 코딩 도구를 쓰기 시작하면서 코딩의 중심이 조금 바뀌었습니다. 이제 중요한 것은 코드를 한 줄씩 직접 쓰는 능력만이 아닙니다. 무엇을 만들 것인지 정하고, 그 일을 몇 단계로 나눌지 정하고, 어떤 파일을 건드릴지 정하고, 어떤 함수가 어떤 책임을 가질지 정하고,

어떤 기준으로 완료 여부를 확인할지 정하는 일이 더 중요해졌습니다. 즉, 코딩은 문법의 문제가 아니라 작업을 구조화하는 문제에 가까워졌습니다. Antigravity IDE를 쓰면서 이 생각이 더 분명해졌습니다.

저에게 Antigravity IDE는 단순한 코딩 도구가 아닙니다. 생각을 파일, 함수, 작업 단위, 검증 기준으로 나누게 만드는 사고 보조 장치에 가깝습니다. Antigravity IDE 작업 화면

형태: AI 기반 개발 환경

사용 방식: ChatGPT에서 사고 구체화 → Codex용 프롬프트 작성 → Codex가 todo.md와 implementation plan 생성 → Antigravity IDE에서 Codex/Gemini 역할 분담 → 결과 검토 및 반영

연결 프로젝트: Jisong Cloud, CleanEMR, DiaFrame, PharmaFrame, Obsidian 관련 도구, 개인 웹앱 전반

핵심 구조: 문제 정의 → 작업 분해 → 문서화 → agent 실행 → 코드 수정 → 테스트/검토 → lessons 반영

주의: AI가 코드를 대신 책임지는 것이 아니라, 사람이 작업 범위와 검증 기준을 정해야 한다.

1) 코딩 전에 먼저 생각을 구체화한다

요즘 저는 바로 IDE를 열고 코드를 치지 않습니다. 먼저 ChatGPT와 이야기합니다. 무엇을 만들 것인지, 왜 필요한지, 현재 문제가 무엇인지, 기능을 어느 범위까지 넣을지, 어떤 파일과 구조가 필요할지

먼저 말로 풀어봅니다. 이 과정은 단순한 잡담이 아닙니다.

생각을 구체화하는 단계입니다.

처음 머릿속에 있는 아이디어는 대개 흐립니다.

“이 기능 있으면 좋겠다.”

“여기 버그가 있는 것 같다.”

“이 프로젝트 구조를 좀 정리해야겠다.” 이 정도로는 AI agent에게 일을 시키기 어렵습니다. 사람에게도 애매한 말은 AI에게 더 위험합니다. 그래서 먼저 생각을 문장으로 바꿉니다.

무엇이 문제인지. 어떤 결과를 원하는지. 어디까지 바꾸면 되는지.

무엇은 건드리면 안 되는지. 완료 기준은 무엇인지.

이 단계에서 ChatGPT는 코딩 도구라기보다 생각을 정리하는 파트너에 가깝습니다. 생각이 어느 정도 정리되면, 그다음에는 Codex에게 줄 프롬프트로 바꿉니다.

2) 좋은 프롬프트는 명령문이 아니라 작업 명세서다

AI에게 “이거 고쳐줘”라고 하면 정말 고쳐주기는 합니다.

문제는 어디까지 고칠지 모른다는 것입니다.

애매하게 시키면 AI는 지멋대로 판단합니다. 좋게 말하면 자율적이고, 나쁘게 말하면 망나니처럼 움직일 수 있습니다. 그래서 Codex에게 줄 프롬프트는 단순한 명령문이 아니라 작업 명세서에 가까워야 합니다.

현재 상황. 목표. 수정 범위. 건드리면 안 되는 파일. 예상 산출물.

테스트 방법. 완료 기준. 작업을 몇 턴으로 나눌지.

이런 것들이 들어가야 합니다.

명령을 잘해야 합니다.

AI가 똑똑해질수록 오히려 이 점이 더 중요해집니다. 애매한 지시를 주면 AI는 그럴듯한 방향으로 넓게 움직입니다. 하지만 프로젝트에서는 그 넓은 움직임이 위험할 수 있습니다. 원래 의도와 다른 파일을 고치거나, 이미 잡아둔 구조를 무너뜨리거나, 임시방편 코드를 추가하거나, 보안과 인증 경계를 건드릴 수 있습니다. 그래서 저는 AI에게 일을 시킬 때 먼저 작업을 좁힙니다.

“무엇을 할 것인가”만큼이나 “무엇을 하지 않을 것인가”를 적습니다.

좋은 프롬프트는 agent가 실행할 수 있는 작업 명세서가 된다.

3) Codex는 todo.md와 implementation plan을 만든다

프롬프트를 Codex에게 넘기면 바로 코드를 고치게만 하지 않습니다.

먼저 계획을 세우게 합니다.

몇 턴으로 나눌지. 어떤 파일을 볼지. 어떤 순서로 수정할지. 어떤 테스트를 돌릴지. 어떤 위험이 있는지.

이 과정에서 todo.md와 implementation plan이 생깁니다. 저에게 이 문서들은 매우 중요합니다. AI agent가 지금 무엇을 하려는지 사람이 볼 수 있는 형태로 남기기 때문입니다.

그냥 “알아서 고쳐줘”라고 하면 AI가 무엇을 기준으로 움직이는지

알기 어렵습니다. 하지만 todo.md가 있으면 작업이 체크리스트로 보입니다. implementation plan이 있으면 변경 방향과 순서가 보입니다.

이렇게 되면 AI가 코드만 쓰는 것이 아니라 작업의 구조를 먼저 드러내게 됩니다. 그리고 이 구조를 사람이 검토할 수 있습니다. 여기서 방향이 틀렸다면 아직 코드를 많이 건드리기 전에 수정할 수 있습니다. AI 코딩에서 중요한 것은 AI가 얼마나 빨리 코드를 쓰는지 가 아닙니다.

AI가 어떤 계획으로 움직이는지 사람이 볼 수 있어야 한다는 점입니다.

4) Antigravity IDE에서는 agent를 역할별로 쓴다

Antigravity IDE를 쓰면서 편했던 점은 여러 AI 도구를 한 작업 환경 안에서 같이 쓸 수 있다는 점이었습니다. 저는 중요한 작업은 Codex에게 시키고, 상대적으로 간단한 작업은 Gemini에게 맡기는 식으로 씁니다. 예를 들어 구조를 크게 바꾸거나, 여러 파일을 건드리거나, 기존 설계와 충돌 가능성이 있는 작업은 Codex에게 맡깁니다. 반대로 작은 수정, 간단한 설명, 짧은 코드 조각, 문서 초안, 명확한 단일 작업은 Gemini에게 맡길 수 있습니다.

중요한 것은 AI를 하나의 거대한 지능으로 보지 않는 것입니다.

작업마다 적합한 역할이 다릅니다.

어떤 작업은 깊게 생각해야 합니다. 어떤 작업은 빠르게 처리하면

됩니다. 어떤 작업은 계획부터 세워야 하고, 어떤 작업은 이미 정해진 구조 안에서 작은 수정만 하면 됩니다. Antigravity IDE는 이런 역할 분담을 실제 개발 환경 안에서 하게 해줍니다. 그래서 IDE가 단순히 코드를 쓰는 창이 아니라 작업 지휘소처럼 느껴질 때가 있습니다.

코드를 직접 쓰는 곳이면서, agent에게 일을 나눠주는 곳이고, 산출물을 확인하는 곳이며, 프로젝트 구조를 다시 잡는 곳입니다.

5) AGENTS.md, lessons.md, todo.md는 안전장치다

AI agent를 잘 쓰려면 말을 잘하는 것만으로는 부족합니다. 프로젝트 안에 기준 문서가 있어야 합니다.

저는 AGENTS.md, lessons.md, todo.md 같은 아티팩트를 중요하게 봅니다.

AGENTS.md는 agent가 프로젝트에서 지켜야 할 규칙을 정리하는 문서입니다. 어떤 구조를 따를지, 어떤 명령을 먼저 확인할지, 어떤 파일을 조심해야 할지, 어떤 방식으로 테스트해야 할지 기본 규칙을 알려줍니다.

lessons.md는 이전 작업에서 배운 점을 남기는 문서입니다. 어떤 실수를 했는지, 어떤 방식이 좋았는지, 다음에는 무엇을 피해야 하는지 프로젝트의 경험치를 쌓아둡니다.

todo.md는 지금 진행 중인 작업을 작은 단위로 쪼개는 문서입니다. 이 세 가지가 있으면 AI가 훨씬 덜 망나니처럼 움직입니다.

완전히 통제된다는 뜻은 아닙니다.

하지만 최소한 프로젝트 안에서 어떤 방향으로 움직여야 하는지 기준이 생깁니다. AI agent에게는 맥락이 필요합니다. 그리고 그 맥락은 채팅창 안에만 있으면 약합니다. 프로젝트 폴더 안에 문서로 남아 있어야 다음 작업에서도 다시 참조할 수 있습니다.

AI를 통제하는 가장 현실적인 방법은 매번 말로 붙잡는 것이 아니라 프로젝트 안에 규칙과 기록을 남겨두는 것입니다.

AGENTS.md, lessons.md, todo.md는 AI agent가 프로젝트 안에서 길을 잃지 않게 하는 안전장치다.

6) 예전에는 직접 짰고, 그다음에는 복붙했다

AI IDE를 쓰기 전에는 대부분 직접 코드를 짰습니다. 필요한 기능이 있으면 직접 구현하고, 에러가 나면 직접 로그를 보고, 문서를 찾아 보고, 하나씩 고쳤습니다. 그 다음 단계에서는 ChatGPT를 많이 썼습니다. 코드를 복사해서 붙여넣고, 에러 메시지를 넣고, 수정안을 받아서 다시 IDE에 붙였습니다.

이 방식도 꽤 유용했습니다.

하지만 한계가 있었습니다.

맥락이 자주 끊겼습니다.

ChatGPT는 전체 프로젝트 파일 구조를 계속 보고 있는 것이 아니었고, 제가 필요한 파일을 골라서 붙여넣어야 했습니다. 수정안이 나오면 다시 IDE로 옮겨야 했습니다. 여러 파일을 건드리는 작업에서

는 어떤 파일에 어떤 변경을 해야 하는지 사람이 계속 이어붙여야 했습니다. 이 방식은 AI를 조연자로 쓰는 방식에 가까웠습니다.

지금은 조금 다릅니다.

ChatGPT에서 사고를 구체화하고, Codex용 프롬프트를 만들고, Codex가 계획을 세우고, Antigravity IDE 안에서 agent가 실제 파일을 보고 수정합니다. 이제 AI는 채팅창 밖으로 나와서 프로젝트 구조 안에서 움직입니다.

그만큼 편해졌습니다.

하지만 그만큼 더 조심해야 합니다.

7) AI가 파일을 만질 수 있다는 것은 위험한 장점이다

AI IDE의 가장 큰 장점은 AI가 프로젝트 파일을 직접 보고 수정할 수 있다는 점입니다. 동시에 이것이 가장 큰 위험입니다. 채팅창에서 답변만 받을 때는 틀린 답을 받아도 바로 프로젝트가 망가지지는 않습니다. 하지만 IDE 안에서 agent가 파일을 직접 수정하면 잘못된 판단이 바로 코드 변경으로 이어질 수 있습니다. 그래서 AI IDE에서는 검토 구조가 더 중요합니다.

무엇을 바꿨는지 확인해야 합니다. 왜 바꿨는지 설명을 봐야 합니다.

테스트를 돌려야 합니다. diff를 봐야 합니다. 예상하지 않은 파일이 바뀌지 않았는지 확인해야 합니다.

AI가 코드를 생성했다고 해서 그 코드가 곧바로 내 코드가 되는 것은 아닙니다.

AI가 만든 것은 제안입니다.

사람이 받아들이고 검토한 뒤에야 프로젝트의 일부가 됩니다. 이 기준을 놓치면 AI IDE는 좋은 도구가 아니라 프로젝트를 엉망으로 만드는 빠른 손이 될 수 있습니다.

8) AI IDE를 잘 쓰려면 사람이 더 구조적이어야 한다

AI IDE를 쓰면 코딩이 쉬워집니다. 하지만 역설적으로 사람은 더 구조적이어야 합니다. 예전에는 내가 직접 코드를 짤기 때문에 내 머릿속에 어느 정도 구조가 있었습니다. 이제는 AI가 많은 부분을 대신 움직일 수 있습니다.

그러면 사람이 구조를 놓치기 쉽습니다. 어떤 파일이 어떤 책임을 갖는지. 이 함수는 왜 필요한지. 이 변경은 기존 설계와 맞는지. 테스트는 충분한지. 보안 경계는 유지되는지. 지금 수정이 임시방편인지, 장기 구조에 맞는지.

이 질문을 계속 사람이 잡아야 합니다. AI에게 맡길수록 사람은 더 큰 구조를 봐야 합니다. 명령을 잘해야 하고, 작업을 나눠야 하고, 산출물을 검토해야 하고, 이전 실수를 lessons로 남겨야 합니다. 잘 쓰면 AI IDE는 굉장히 좋은 도구입니다. 하지만 잘 쓰는 방법을 알아야 합니다.

9) IDE는 코딩 도구가 아니라 사고 보조 장치다

Antigravity IDE를 쓰면서 IDE에 대한 생각이 조금 바뀌었습니다.

예전의 IDE는 코드를 쓰는 곳이었습니다. 하지만 지금의 AI IDE는 코드를 쓰는 곳이면서 동시에 작업을 나누고, agent를 배치하고, 계획을 확인하고, 변경을 검토하고, 프로젝트의 구조를 관리하는 곳입니다. 코딩을 잘한다는 것은 문법을 많이 아는 것만은 아닙니다.

문제를 파일로 나누고, 함수로 나누고, 책임으로 나누고, 테스트 가능한 단위로 나누는 능력입니다. AI IDE는 이 과정을 빠르게 도와줍니다.

하지만 대신해주지는 않습니다.

결국 중요한 것은 어떤 질문을 코드의 형태로 바꿀 것인가입니다.

질문이 흐리면 코드도 흐려집니다.

구조가 없으면 AI도 구조 없이 움직입니다. 반대로 사람이 문제를 잘 나누고, 규칙을 문서로 남기고, 작업 범위를 좁히고, 검증 기준을 명확히 하면 AI IDE는 강력한 사고 보조 장치가 됩니다. 저에게 Antigravity IDE는 코딩을 대신해주는 도구라기보다 생각을 코드 구조로 바꾸는 작업 환경입니다. 코딩 도구가 아니라, 사고 보조 장치에 가깝습니다.

사진도 구조를 보는 일이다

사진은 감성의 기록처럼 보입니다.

어떤 날의 하늘. 어떤 여행지의 풍경. 우연히 마주친 빛. 그때의 기분.

물론 사진에는 감정이 남습니다.

하지만 제가 사진을 찍을 때 가장 먼저 보는 것은 생각보다 감정보다 구조에 가깝습니다.

빛이 어디에서 들어오는지. 선이 어디로 향하는지. 공간이 어떻게 나뉘는지. 색이 어디에서 멈추는지. 사람이나 건물이 화면 안에서 어떤 무게를 가지는지.

사진은 순간을 찍는 일입니다.

하지만 그 순간이 사진이 되려면 빛과 공간과 선이 잠깐 정렬되어야 합니다. 그래서 제 사진 포트폴리오의 이름을 Quiet Structure라고 붙였습니다.

조용한 구조.

거창한 선언은 아닙니다.

그냥 제가 사진을 볼 때 반복해서 끌리는 것이 무엇인지 가장 짧게 설명하는 이름입니다. Quiet Structure — photo.jisong.dev

주소: photo.jisong.dev

형태: GitHub Pages 기반 정적 사진 포트폴리오

구성: index.html, style.css, assets/images 핵심 문장: 빛과 공간, 구조가 잠깐 정렬되던 순간들

주의: 사진 작가 포트폴리오라기보다, 시각적으로 구조를 보는 방식의 작은 기록

1) 사진을 찍는다는 것

사진을 찍을 때 저는 대단한 이야기를 만들려고 하지는 않습니다.

오히려 이야기가 너무 많으면 사진이 조금 무거워진다고 느낄 때가 있습니다. 제가 더 자주 끌리는 것은 조용한 장면입니다.

강가의 다리. 숲길의 녹색. 호수 위의 하늘. 건물에 비친 저녁빛. 캠 퍼스의 조명. 푸른 수조 안의 해파리. 도시 위로 내려앉는 노을.

이런 장면들은 특별한 사건이 아닙니다. 대부분은 그냥 지나칠 수 있는 풍경입니다. 하지만 어떤 순간에는 빛과 공간이 잠깐 맞아떨어 집니다.

그때 사진을 찍습니다.

좋은 사진을 찍겠다는 의식보다, 지금 이 정렬이 곧 사라질 것 같다는 감각에 가깝습니다. 사진은 결국 사라질 구조를 붙잡는 일입니다.

2) 분위기보다 배치

사진을 볼 때 흔히 분위기를 말합니다. 따뜻하다. 쓸쓸하다. 고요하다. 몽환적이다. 예쁘다.

이런 말이 틀린 것은 아닙니다.

하지만 저는 그 분위기가 어디서 오는지 더 궁금합니다. 왜 고요해 보이는지. 왜 안정적으로 느껴지는지. 왜 시선이 한쪽으로 흐르는지.

왜 어떤 사진은 답답하고, 어떤 사진은 열려 있는지.

대부분의 답은 배치에 있습니다.

수평선이 어디에 있는지. 빈 공간이 얼마나 있는지. 피사체가 중앙에 있는지, 가장자리에 있는지. 밝은 면과 어두운 면이 어떻게 나뉘는지. 반복되는 선이 어디에서 끊기는지.

분위기는 결과입니다.

그 아래에는 구조가 있습니다.

사진을 찍는 일은 그 구조를 감각적으로 찾는 일입니다. 분위기는 결과이고, 그 아래에는 빛과 선과 공간의 배치가 있다.

3) 빛은 방향을 만든다

사진에서 빛은 단순히 밝기를 만드는 요소가 아닙니다.

빛은 방향을 만듭니다.

어디를 먼저 보게 할지, 무엇을 배경으로 밀어낼지, 어디에 시선을 멈추게 할지 결정합니다. 같은 장소도 빛이 달라지면 전혀 다르게 보입니다.

낮의 건물은 정보가 많습니다. 밤의 건물은 선과 면이 남습니다. 노을이 지면 도시는 색의 층으로 바뀝니다. 흐린 날의 숲은 대비가 줄어들고, 길의 깊이가 더 잘 보입니다.

빛은 장면을 설명하지 않습니다.

장면을 정리합니다.

무엇을 보여줄지, 무엇을 숨길지, 어떤 순서로 보게 할지 정합니다. 그래서 사진을 찍을 때 저는 피사체만큼 빛의 방향을 봅니다. 사진 속에서 빛은 일종의 구조화 도구입니다.

4) 공간은 여백으로 읽힌다

사진에서 중요한 것은 찍힌 대상만이 아닙니다.

찍히지 않은 공간도 중요합니다.

하늘. 물. 벽. 길. 그림자. 빈 좌석. 사람이 없는 복도. 이런 여백은 아무것도 없는 부분이 아닙니다.

오히려 사진의 호흡을 만듭니다.

여백이 없으면 장면은 딱 막힙니다. 여백이 너무 많으면 시선이 흩어집니다. 적당한 여백은 피사체가 놓일 자리를 만듭니다. 의대 공부를 할 때도, 코드를 짤 때도 비슷한 생각을 합니다.

모든 것을 한꺼번에 넣으면 구조가 무너집니다. 중요한 것을 보이게 하려면 중요하지 않은 것을 비워둘 수 있어야 합니다.

사진에서도 마찬가지입니다.

좋은 사진은 많이 담은 사진이 아니라, 무엇을 남기고 무엇을 비울지 정한 사진일 때가 많습니다.

5) 반사는 두 개의 공간을 겹친다

제가 자주 끌리는 장면 중 하나는 반사입니다. 물에 비친 나무. 호수에 비친 건물. 유리창에 겹친 도시. 밤빛이 닿은 바닥.

반사는 실제 공간과 뒤집힌 공간을 동시에 보여줍니다. 하나의 장면 안에 원래의 세계와 복사된 세계가 겹칩니다. 이때 사진은 단순한 기록이 아니라 공간의 관계를 보여주는 장면이 됩니다. 무엇이 실제이고, 무엇이 반사인지 사진 안에서는 잠깐 흐려집니다.

그 모호함이 좋습니다.

하지만 그것도 결국 구조의 문제입니다. 반사면이 어디에 있는지. 원

래의 피사체와 반사된 피사체가 어떻게 나뉘는지. 두 공간의 경계가 어디에서 희미해지는지.

감성적으로는 몽환적이라고 말할 수 있지만, 제가 보는 것은 두 공간이 겹치는 방식입니다. 반사는 하나의 장면 안에 두 개의 공간을 겹쳐 놓는다.

6) 사진 포트폴리오를 만든 이유

photo.jisong.dev를 만든 이유는 사진을 거창하게 전시하기 위해서 만든 아닙니다. 사진 파일은 그냥 두면 흩어집니다.

휴대폰 앨범에 있고, 외장하드에 있고, 가끔 SNS에 올라가고, 다시 찾으려면 기억에 의존해야 합니다. 그래서 사진에도 공간이 필요했습니다. 단순한 저장소가 아니라 제가 어떤 장면을 반복해서 보는지 정리할 수 있는 공간. Quiet Structure는 그런 용도에 가깝습니다. 사진을 모아두면 내가 무엇에 끌리는지 보입니다. 색보다는 빛. 사진 보다는 공간. 인물보다 배치. 설명보다 정렬. 화려함보다 조용한 균형.

하나씩 찍을 때는 몰랐던 반복이 모아두면 보입니다. 포트폴리오는 남에게 보여주는 공간이기도 하지만, 자기 자신이 자신의 시선을 다시 확인하는 공간이기도 합니다.

7) 사진도 구조를 보는 일이다

공부, 코딩, 의학, 사진은 멀어 보입니다. 하지만 저에게는 그렇게까

지 멀지 않습니다. 의학을 공부할 때는 증상과 검사와 병태생리 사이의 구조를 봅니다. 코딩을 할 때는 파일과 함수와 책임 사이의 구조를 봅니다.

글을 쓸 때는 문장과 문단과 메시지 사이의 구조를 봅니다. 사진을 찍을 때는 빛과 공간과 선 사이의 구조를 봅니다. 대상은 다르지만 반복되는 방식은 비슷합니다.

흠어진 것을 봅니다. 그 안에서 반복을 찾습니다. 무엇이 중심인지 봅니다. 무엇을 비워야 하는지 봅니다. 그리고 잠깐 정렬되는 순간을 붙잡습니다.

사진은 감성의 기록처럼 보입니다.

맞습니다.

하지만 저에게 사진은 구조를 보는 또 다른 방식이기도 합니다. 빛과 공간과 선이 잠깐 정렬되는 순간.

그 조용한 구조를 발견하는 일.

그래서 사진도 결국, 제가 세상을 이해하는 방식의 일부입니다.

제작 노트: 의공구 Book Pipeline

항목	내용
프로젝트명	의공구 Book Pipeline
개발 기간	2026.06.04 - 2026.06.06
원고 형식	Markdown
이미지 자산	PNG / JPG assets

항목	내용
메타데이터	YAML
언어	Python / Bash
빌드 엔진	Pandoc
작업 환경	Codex / Antigravity IDE
출력 형식	DOCX / PDF / EPUB
파이프라인	Markdown + assets → Python build script → Pandoc → EPUB/PDF/DOCX

이 책은 Markdown 원고와 이미지 assets를 기반으로 관리되었고, Pandoc을 통해 EPUB/PDF/DOCX 형식으로 빌드되었습니다. 일부 원고 정리와 빌드 파이프라인 구성에는 Codex의 도움을 받았습니다.

계으른 방지송 썬기는 결국 책마저 Codex로 빌드하였습니다.



뒷표지