The Clin- Ecosystem

From "patients understand" to "clinicians can use"

Chia-Yu Chiang

Biomedical Engineering, Ming Chuan University · 2026

The problems I saw

Consumer side: health-check reports full of numbers and jargon — people don't know what to worry about.

- Information asymmetry ・ Privacy concern (pasting values into ChatGPT is unsafe)

Clinician side: a wide digitization gap among primary clinics in Taiwan.

- Large hospitals have HIS/EMR; small clinics can't afford them or find them a poor fit

Clin- Ecosystem: one interlocking system

Product	Role	One line
ClinCalc (consumer)	Entry	Make reports **understandable**
ExClinCalc Pro (clinician)	Workflow	Full clinic flow; security **in the database**
clinconvert	Interop (PoC)	FHIR data-conversion proof of concept
Kaizei	Cross-domain	Same engineering method, reused elsewhere

Shared stack: **Cloudflare Workers + Supabase + PostgreSQL RLS + TypeScript**

Product 1: ClinCalc

Consumer health self-check + AI interpretation

ClinCalc: two core design decisions

1. Local-first interpretation

- 35 lab indicators, KDIGO 2024 CKD staging
- Interpretation runs **entirely in the browser**; raw values never leave the device

2. AI as assistant, not decision-maker

- Classify locally ("normal / high / low") → send **only structured results** to Gemini
- The model needn't memorize reference ranges → lower hallucination and privacy risk

ClinCalc: rules first, LLM second

```
User enters 35 indicators
      ↓
Local knowledge base referenceRanges.ts  ← KDIGO 2024 / ADA / ACC-AHA
      ↓
checkAbnormal(): normal / high / low / severe…
      ↓
Send ONLY the structured verdict as prompt → Gemini for overall advice
```

Traceable: every verdict maps back to a source guideline — not AI guesswork.

Product 2: ExClinCalc Pro

Clinical decision-support workstation

ExClinCalc: a full clinic-workflow loop

Not a note editor — **five interlocking stages**:

Check-in → Nurse triage → Physician SOAP (7 steps) → Drug-interaction check → Pharmacist dispensing

- SOAP auto-expands guided questions by chief complaint
- Prescribing triggers real-time drug-interaction alerts
- Six roles: doctor / nurse / pharmacist / staff / admin / super-admin

★ The core: security in the database layer

Access control is **not** written in the backend code — it lives in PostgreSQL.

> **41 Row-Level-Security policies**
> Even if the front end is compromised, an attacker can't read data outside their role —
> because PostgreSQL checks identity and permission **before every query**.

Defense in depth: backend code will always have bugs; pushing authorization into the DB means "app layer breached, data still safe."

Three depths of the security design

- **Enforced MFA**: any PHI access requires a TOTP code (AAL2)
- **RESTRICTIVE gate**: a PostgreSQL restrictive policy ANDs MFA on → **no policy can bypass it**
- **Role capability matrix**: pharmacists dispense but can't alter diagnoses; nurse-written vitals are doctor-read-only

Built via **8 migrations**, verified by **40+ RLS integration tests**.

How do I know the security is correct?

On a disposable Postgres, **simulate each role's session** and test RLS directly:

Scenario	Expected	Result
Doctor without MFA (aal1)	PHI denied	✅
Doctor with MFA (aal2)	PHI allowed	✅
Pharmacist alters diagnosis	Blocked by trigger	✅
Anonymous access	Denied	✅

Security isn't asserted — it's proven by tests.

Stack & deployment

- **Front end**: Next.js 16 (App Router) · React 19 · TypeScript · Tailwind v4
- **Backend / DB**: Supabase (PostgreSQL + Auth + RLS + TOTP MFA)
- **AI**: Google Gemini (rules-then-LLM)
- **Deploy**: Cloudflare Workers (OpenNext), global edge, very low cost

Both ends **share one Supabase**, isolated by RLS:
patients see only their own records; a doctor needs **one-time consent** to view them.

What I learned / research directions

1. **A design methodology for RLS in multi-tenant medical systems**
   Can we **auto-derive** policies from requirements and **formally verify** their completeness?
2. **A tiered architecture for safely embedding LLMs in clinical flow**
   How to allocate the rules-vs-LLM ratio, and quantify hallucination vs coverage?
3. **Real-world evaluation methods for CDSS**
   From "feature-complete" to "actually adopted" — measuring acceptance and alert fatigue.

Thank you

Chia-Yu Chiang · yuyulsc881209@icloud.com

Live demos: clincalc / exclincalc `.yuyulsc881209.workers.dev`

GitHub: github.com/yu8812 · Site: jiayuselfweb.pages.dev