Growth Loops Technology Pvt Ltd

Engineering Portfolio · Case Study · Full-Stack Engineering

StitchedHealth

AI-Powered Medical Education & Continuing Learning Platform

ROLE	DOMAIN	STACK
Full-Stack Engineer	**Medical Education · HealthTech**	**NestJS · Python · GPT-4o**

40+	11+	20+	3
Database Tables	AI Endpoints	Backend Modules	User Roles

About this Project

StitchedHealth is a comprehensive medical education platform that enables clinicians to engage with real-world case studies, share peer perspectives, earn continuing education credits, and participate in evidence-based learning experiences. Growth Loops Technology designed and built the complete backend API and a dedicated Python AI microservice — from database schema through LLM-powered analytics to production deployment.

Document Contents

gunendu@growth-loop.io · +91 9880806633 · Cabin No. 23, Mani Casadona, Kolkata 700161

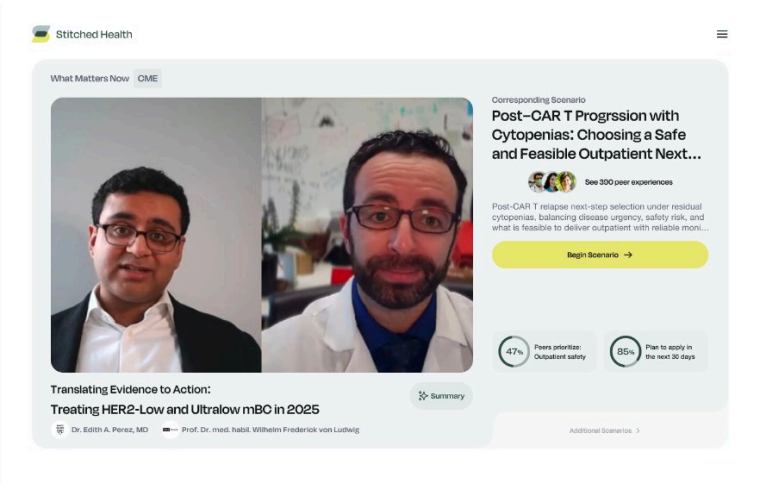
01 — OVERVIEW

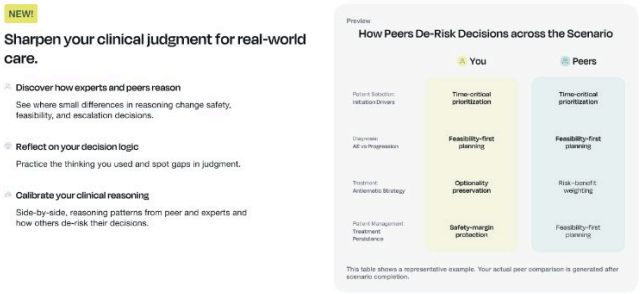
What is StitchedHealth?

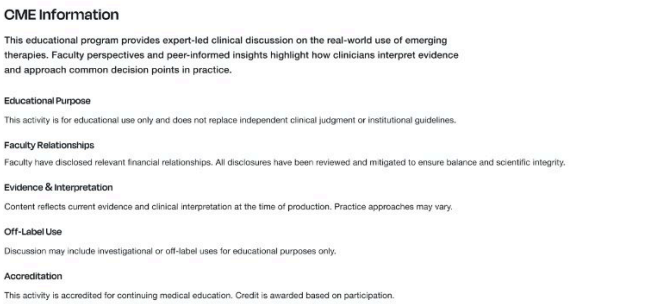
StitchedHealth is a medical education platform that transforms how clinicians engage with continuing education. Through interactive case studies built around real patient scenarios, clinicians progress through diagnostic decision trees, share peer perspectives, receive AI-generated feedback, and earn accredited continuing education credits.

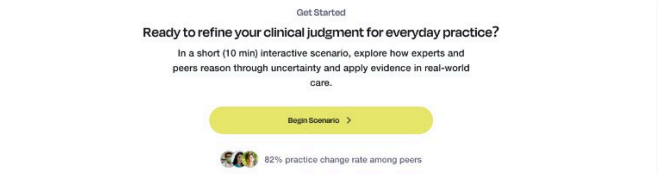
The platform addresses a persistent gap in medical education: traditional CME programs are passive, one-directional, and disconnected from clinical reality. StitchedHealth makes learning active and social — clinicians see how peers approach the same case, receive structured AI analysis of response patterns, and gain evidence-backed insights from integrated RCT literature.

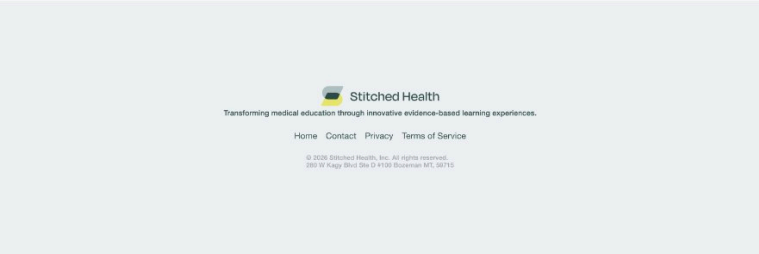
growth loops











StitchedHealth landing page — CME scenarios with peer experience comparison and evidence-based learning

40+	**11+**	**20+**	**3**
Database Tables	AI Endpoints	Backend Modules	User Roles

02 — THE PROBLEM

Medical education needs a feedback loop

Continuing medical education has remained structurally unchanged for decades. Clinicians attend lectures, read articles, and check boxes — but the learning rarely connects to how they actually make clinical decisions. Existing platforms deliver content passively with no mechanism to surface how peers are reasoning through the same cases.

The core challenges were threefold:

Peer Learning at Scale: Clinicians learn best from each other, but there was no structured way to aggregate, score, and present peer perspectives on clinical decisions. Free-text responses needed AI-powered quality scoring and thematic grouping to surface meaningful patterns.

Evidence Integration: Case studies needed to be enriched with real RCT evidence — not just referenced, but analysed and summarised by AI so clinicians could see how their decisions align with empirical data.

Content Intelligence: Admins needed to understand engagement patterns, response quality, and learning outcomes at scale. Every interaction had to be tracked, analysed, and surfaced through comprehensive analytics.

03 — ROLE & RESPONSIBILITIES

Full-Stack Engineer — Growth Loops Technology Pvt Ltd

Growth Loops Technology was responsible for the full design and implementation of both backend systems — the NestJS application server and the Python AI microservice — from initial architecture through production deployment. This included:

- Designing the full data model across 40+ Prisma-managed PostgreSQL tables with complex entity hierarchies
- Building the NestJS 11.x modular monolith with 20+ feature modules, role-based guards, and multi-tier caching
- Architecting the async job processing pipeline using BullMQ and Redis for background tasks including email, analytics sync, peer perspective scoring, and PDF generation
- Building the Python FastAPI AI microservice with 11+ LLM-powered endpoints for clinical content analysis
- Implementing LangChain + GPT-4o integration with LangSmith prompt management and observability
- Integrating Anthropic Claude (Haiku and Sonnet) for high-throughput content processing via the Chips pipeline
- Building the hierarchical summary system: question-level, section-level, and case-study-level AI summaries
- Implementing RCT PDF analysis pipeline for evidence-based clinical decision support
- Integrating Meilisearch for instant, typo-tolerant search across programs and case studies
- Setting up Mixpanel analytics, Sentry error tracking, and BigQuery data warehousing
- Configuring Docker containerisation, CI/CD, and multi-environment deployment

04 — ARCHITECTURE

Two services, one learning experience

StitchedHealth's backend is composed of two purpose-built services that work in concert. The separation was a deliberate architectural decision: the AI workload (LLM-intensive, variable latency) is fundamentally different from the API workload (low-latency, high-concurrency), and keeping them decoupled allowed each to scale independently.

Service 1 — NestJS Application Server

The primary backend handles all product logic: authentication (JWT + Google SSO), user management across three roles (Super Admin, Admin, Clinician), educational program and case study CRUD, response tracking through a five-level hierarchy (EP → Case Study → Section → Question → Option), peer perspective management with AI scoring, analytics, and credit tracking. It exposes a REST API backed by PostgreSQL via Prisma ORM, with Redis for multi-tier caching and BullMQ for async job orchestration.

Service 2 — Python AI Microservice (FastAPI)

A standalone FastAPI service responsible for all AI and LLM workloads. It receives requests from the main backend, processes clinical content through LangChain pipelines with GPT-4o and Claude, and returns structured JSON results. Features include peer perspective tag creation, RCT summary generation, comment ranking, hierarchical summaries, NPI verification, and content extraction from uploaded files.

End-to-End Flow

Web/Mobile Client → NestJS API → BullMQ Queue → Python AI Service → PostgreSQL / R2 Storage → Push Notification

Layered Architecture

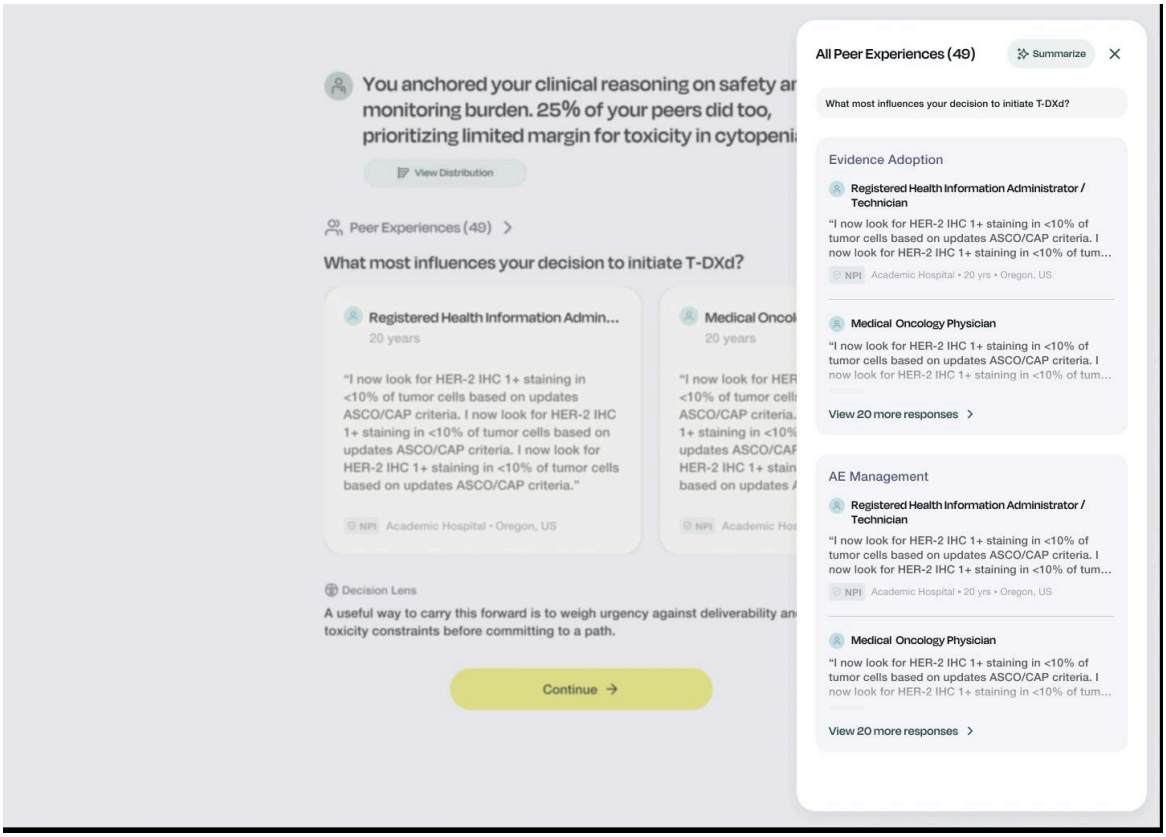
Client Layer	Web Frontend / Mobile App
Presentation Layer	Controllers, Guards, Interceptors, Swagger Documentation
Business Logic Layer	20+ NestJS Modules with Services, DTOs, and Business Rules
Data Access Layer	Prisma ORM with 40+ Models, Transactions, Connection Pooling
Cache Layer	Multi-tier: In-Memory (Keyv, 5000 items) → Redis (distributed)
Queue Layer	BullMQ + Redis with 12+ job types, retry strategies, status tracking
AI Layer	FastAPI + LangChain + GPT-4o/Claude + LangSmith prompt management
Infrastructure	PostgreSQL, Redis, Cloudflare R2, Meilisearch, Sentry, Mixpanel, BigQuery

05 — KEY ENGINEERING CHALLENGES

Where the hard problems lived

1. AI-Powered Peer Perspective Pipeline

Clinicians submit free-text responses to clinical questions, and these responses need to be scored, tagged, grouped, and summarised automatically. A multi-stage AI pipeline was built: GPT-4o-mini creates thematic tags for each response, a separate LLM ranks responses against RCT evidence on a 1–4 scale, and a summarisation layer groups related perspectives into peer clusters. All of this feeds into a hierarchical summary system that rolls up from question-level through section-level to full case-study summaries.



Peer experiences view — AI-tagged and grouped clinician responses with thematic clustering

2. Five-Level Response Tracking Hierarchy

Every clinician interaction is tracked through a deeply nested hierarchy: EP Response → Case Study Response → Section Response → Question Response → Option Selection. This enables granular progress tracking, credit calculation, and analytics. The challenge was maintaining data integrity across this five-level chain while supporting question branching logic (FlowRules) that dynamically changes the path based on answers.

3. RCT Evidence Integration

The platform ingests Randomized Controlled Trial PDFs, extracts text, and generates structured clinical summaries using LLM pipelines. Individual RCT summaries are then compiled into aggregate summaries. These evidence bases are used to evaluate MCQ

options, rank peer comments, and provide evidence-aligned feedback — turning static research papers into dynamic decision-support tools.

4. Multi-Tier Caching with Cache Invalidation

A two-layer cache architecture was implemented: L1 in-memory (Keyv with CacheableMemory, 5000 items, 60s TTL) and L2 Redis (distributed, configurable TTL). Static content like educational programs and case studies are aggressively cached, while dynamic content like peer perspectives use short TTLs. Cache invalidation on content updates was the most delicate part — stale clinical content is not acceptable.

5. Dual-LLM Strategy: GPT-4o + Claude

The AI microservice uses both OpenAI and Anthropic models strategically. The Chips pipeline (content extraction, citations, label generation) uses Claude Sonnet for complex analysis and Claude Haiku for high-throughput tasks like open-text responses. The LangSmith-managed endpoints use GPT-4o for structured output via LangChain. Prompt caching with 5-minute TTL prevents LangSmith API rate limiting.

06 — TECH STACK

Technology decisions

Layer	Technology
Application Server	NestJS 11.x (TypeScript) · REST API · Swagger
AI Microservice	Python · FastAPI · LangChain · GPT-4o · Claude
Database	PostgreSQL · Prisma 6.3 ORM (40+ models)
Job Queue	BullMQ · Redis 5.x (12+ job types)
LLM Framework	LangChain · Pydantic structured output · LangSmith tracing
Prompt Management	LangSmith (versioned prompts with caching)
Storage	AWS S3 · Cloudflare R2 (PDFs, images)
Search	Meilisearch (typo-tolerant, faceted search)
Caching	Multi-tier: Keyv (in-memory) + Redis (distributed)
Email	SendGrid (templated emails)
Analytics	Mixpanel · Google BigQuery · Sentry
Auth	JWT · Passport.js · Google OAuth SSO · bcrypt
Infrastructure	Docker · GitHub Actions CI/CD · CloudFront CDN

07 — OUTCOME & REFLECTIONS

What was delivered

Growth Loops Technology delivered StitchedHealth from initial architecture to a production-ready platform encompassing a modular NestJS backend with 20+ feature modules, a full PostgreSQL schema with 40+ tables and complex entity hierarchies, a Python AI microservice with 11+ LLM-powered endpoints, and a comprehensive analytics and engagement tracking system.

The most technically demanding aspect was building the AI-powered peer perspective pipeline: scoring free-text clinical responses against RCT evidence, generating thematic tags, clustering perspectives, and rolling everything into hierarchical summaries that clinicians could consume in seconds. Getting structured, reliable outputs from LLMs across diverse medical specialties and response styles required significant iteration on prompt engineering, output schema design, and caching strategy.

The response tracking architecture was equally challenging — maintaining data integrity across a five-level hierarchy while supporting dynamic question branching required careful transaction management and state machine design. Every response path had to be auditable, resumable, and credit-eligible.

Key Engineering Takeaways

- Decouple AI microservices from API servers early — LLM latency is unpredictable and should never block user-facing endpoints
- LangSmith prompt management with caching is essential for production LLM systems — prompt versioning and observability prevent silent regressions
- Multi-tier caching (memory + Redis) dramatically improves response times for content-heavy platforms, but cache invalidation for clinical content must be bulletproof
- BullMQ with Redis is excellent for complex job pipelines, but job idempotency and status tracking must be built explicitly from day one
- Structured output from LLMs (Pydantic models via LangChain) is non-negotiable for production — free-form responses are too fragile to persist to a database

Growth Loops Technology Pvt Ltd · Engineering Portfolio
gunendu@growth-loop.io · +91 9880806633