TECHNICAL PLATFORM REPORT

Docere

Memory Infrastructure for Education

Adaptive AI Tutoring • LMS Integration • Bayesian Knowledge Tracing

Youdahe Asfaw

Founder & Sole Architect

Computer Science • Gustavus Adolphus College
Distributed Systems • Infrastructure • Reliability Engineering

youdaheasfaw@gmail.com • linkedin.com/in/youdaheasfaw • youdahe.dev

Abstract

Docere is an AI tutoring platform that integrates directly into existing learning management systems (Canvas and Moodle) to provide every student with a personalized tutor that maintains persistent memory across sessions. The platform combines a three-source memory retrieval architecture—teacher context via semantic search, student profiles with Bayesian Knowledge Tracing, and interaction history with diversity-filtered recall—with an adaptive teaching strategy system powered by UCB1 multi-armed bandit optimization and evolutionary strategy mutation. Every response is scored across four dimensions (helpfulness, clarity, understanding delta, engagement), creating a closed feedback loop that continuously improves tutoring quality. The system is live across 2 schools and 1 library serving 100+ students, with sub-second response latency and built-in ablation study infrastructure for rigorous evaluation. This report details the system architecture, memory infrastructure, adaptive

100+	< 1s	$10K	3
Students Served	Response Latency	Funding Secured	Pilot Deployments

Table of Contents

1. Introduction & Problem Statement

The central challenge in K–12 education is scale: teachers are responsible for dozens or hundreds of students, and the individual attention that produces the deepest learning—one-on-one tutoring—is economically infeasible for most schools. Existing AI tools have attempted to fill this gap, but they suffer from a fundamental limitation: they are **stateless**. Every conversation starts from zero. The AI does not know what a student struggled with last week, which explanations clicked, or what their grades look like. The result is generic responses that answer questions without teaching.

Docere addresses this by providing an AI tutoring agent that integrates directly into the learning management systems schools already use—Canvas and Moodle—and maintains **persistent, structured memory** of every student across sessions. The system adapts its teaching approach to each individual through a self-improving strategy engine, tracks concept mastery via Bayesian Knowledge Tracing, and gives instructors visibility into where their class is actually struggling. The platform has been piloted across two schools and one library, serving over 100 students, with $10,000 in funding secured.

2. System Architecture Overview

Docere is organized around a six-node LangGraph state machine that orchestrates every student interaction, supported by a three-source memory retrieval system, an async background processing pipeline, and an abstract LMS adapter layer. The architecture is designed for low-latency synchronous response generation while deferring expensive analytical work to background tasks.

The backend is a **FastAPI** asynchronous server backed by **PostgreSQL** for relational data (student profiles, mastery scores, strategy metadata), **Qdrant** as a vector database for semantic search over course materials and interaction history, and **Redis** for caching and background task queues managed through ARQ. The frontend is **React 19** with TypeScript and Tailwind CSS. The LLM backbone is **Anthropic Claude**, invoked through the LangGraph agent pipeline.

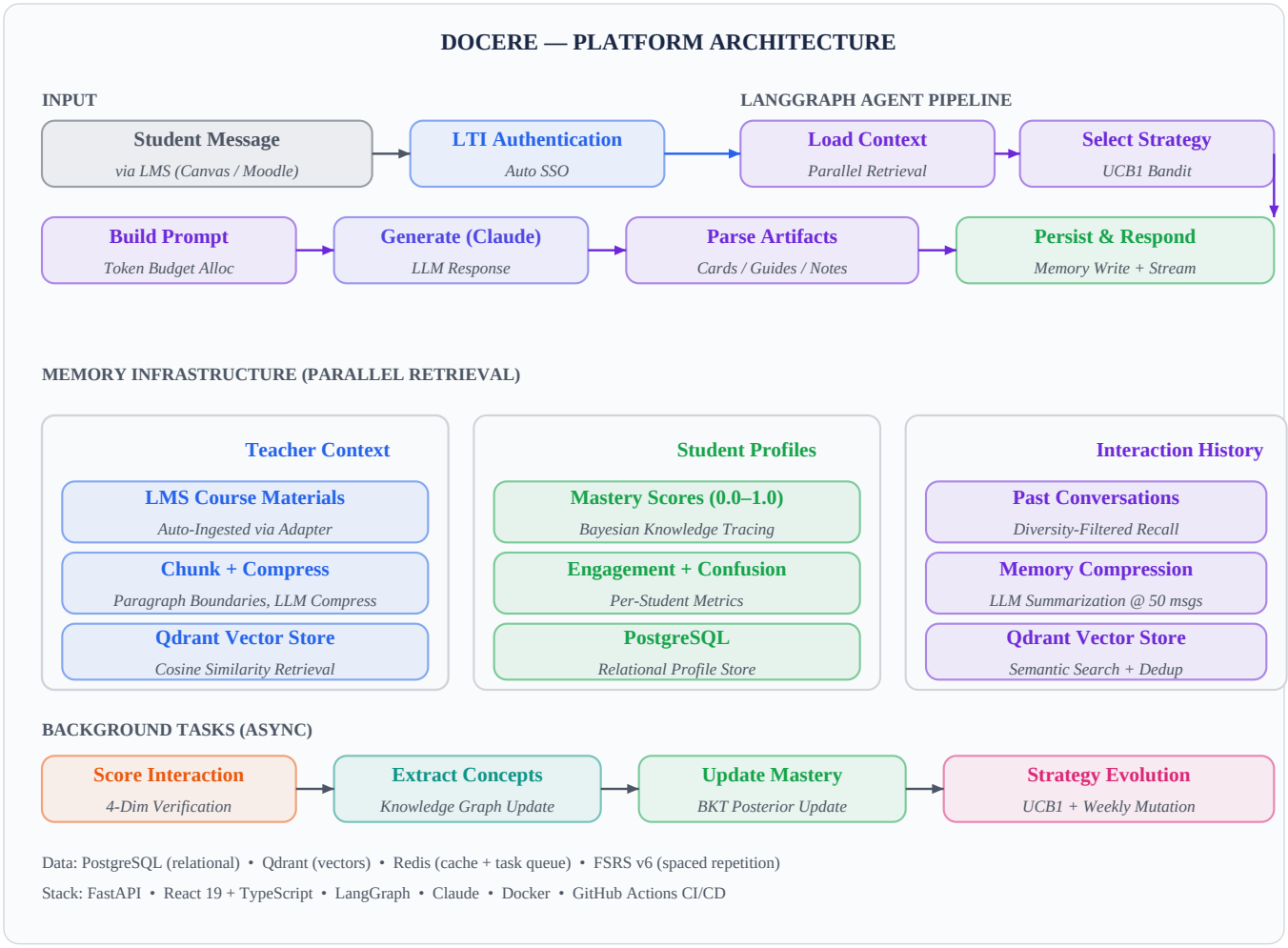

Figure 1 — Full platform architecture: agent pipeline, memory infrastructure, and background processing.

The critical path for a student message—memory retrieval, strategy selection, prompt construction, LLM generation, and response streaming—executes synchronously and returns in under one second. After the response is sent, background tasks fire asynchronously via the Redis-backed ARQ queue: scoring the interaction, extracting concepts, updating mastery levels, and compressing stale memory. This separation ensures that analytical overhead never impacts the student experience.

3. Memory Infrastructure

The memory infrastructure is the core differentiator of Docere. Before generating every response, the system retrieves context from three independent sources in parallel, allocating a shared token budget of up to 4,000 tokens with priority-based ranking. This ensures the LLM always has the most relevant context without exceeding context window constraints.

3.1 Teacher Context (Semantic Course Materials)

Course materials are automatically ingested from the LMS through the adapter layer. Documents are chunked at paragraph boundaries to preserve semantic coherence. Chunks exceeding 3,000 characters are

compressed via LLM summarization to reduce embedding noise while preserving instructional intent. Each chunk is embedded and stored in **Qdrant**, the vector database. At retrieval time, the student's message is embedded and the top-k most relevant chunks are retrieved via cosine similarity, ensuring the tutor's response is grounded in the actual curriculum the teacher assigned.

3.2 Student Profiles (Bayesian Knowledge Tracing)

Each student has a structured profile stored in PostgreSQL that tracks engagement level, average confusion, current grade, and **per-concept mastery scores** ranging from 0.0 to 1.0. Mastery is updated after every interaction using **Bayesian Knowledge Tracing (BKT)**—a probabilistic model that estimates the likelihood a student has learned a concept based on their sequence of correct and incorrect responses. The posterior probability of mastery is updated using observed evidence, accounting for slip (knowing the answer but getting it wrong) and guess (getting it right without understanding) parameters.

Additionally, an LLM-generated narrative summary captures qualitative patterns that are difficult to represent numerically—for example, "this student tends to understand concepts visually but struggles with algebraic notation" or "responds well to real-world analogies." The summary is regenerated periodically as more interaction data accumulates.

3.3 Interaction History (Diversity-Filtered Recall)

Past tutoring conversations are stored as embeddings in Qdrant and retrieved via semantic similarity to the current message. A critical innovation here is the **diversity filter**: naive similarity retrieval often fills the context window with near-duplicate memories (e.g., multiple conversations about the same homework problem). The diversity filter ensures that retrieved memories span distinct topics and timepoints, providing the LLM with a richer, more representative view of the student's history.

3.4 Memory Compression

To prevent unbounded memory growth, a compression system activates after a student accumulates 50 uncompressed interactions. LLM summarization distills older conversations into compact representations that preserve key patterns—what topics were discussed, what the student struggled with, what strategies worked—while discarding redundant detail. Compressed memories are stored alongside raw interactions, and the retrieval system blends both sources to maintain historical awareness without consuming excessive token budget.

4. LangGraph Agent Pipeline

The tutoring agent is implemented as a **six-node LangGraph state machine** that executes on every student message. LangGraph provides deterministic execution flow with explicit state transitions, ensuring that every interaction follows the same rigorous pipeline regardless of input.

The six nodes execute in sequence: (1) **Load Context**—retrieves all three memory sources in parallel and assembles the context window within the token budget; (2) **Select Strategy**—the UCB1 bandit chooses the optimal teaching strategy for this student and context; (3) **Build Prompt**—constructs the full system prompt by combining the strategy instructions, retrieved context, student profile, and conversation history; (4)

Generate—invokes Claude with the assembled prompt and streams the response; (5) **Parse Artifacts**—extracts any structured output embedded in the response, such as flashcards, study guides, meeting actions, or notes; and (6) **Persist**—writes the interaction to the memory store and dispatches background tasks.

5. Adaptive Teaching Strategy Engine

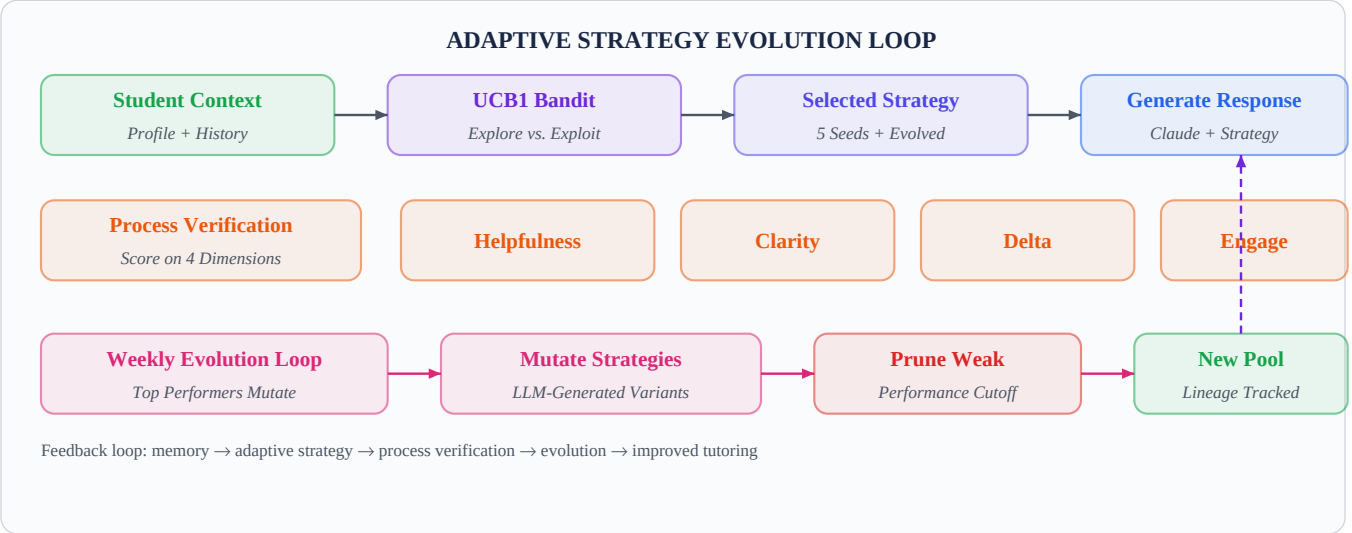

Figure 2 — Closed-loop strategy evolution: selection, scoring, mutation, and pruning.

5.1 UCB1 Multi-Armed Bandit Selection

At the core of the adaptive strategy system is a **UCB1 (Upper Confidence Bound) multi-armed bandit**. The system maintains a pool of teaching strategies, starting with five seed strategies and expanding through evolutionary mutation. For each student interaction, UCB1 selects the strategy that maximizes the sum of its average reward (exploitation) and an exploration bonus proportional to the square root of the log of total trials over the strategy's trial count (exploration). This balance ensures that well-performing strategies are used more frequently while underexplored strategies receive sufficient trials to establish reliable performance estimates.

5.2 Process Verification Scoring

Every tutoring response is scored across four dimensions by an independent evaluation pass: **helpfulness** (did the response address the student's actual question?), **clarity** (was the explanation understandable at the student's level?), **understanding delta** (did the student's demonstrated understanding increase after the response?), and **engagement** (did the student remain actively involved?). Crucially, the scoring prioritizes whether the student's *understanding actually changed*, not whether they rated the response positively—a distinction that prevents the system from optimizing for student satisfaction at the expense of learning outcomes.

5.3 Evolutionary Strategy Mutation

On a weekly cadence, an evolution loop analyzes strategy performance across all interactions. Top-performing strategies are selected as parents for mutation—LLM-generated variants that modify aspects of

the teaching approach (e.g., adjusting the level of scaffolding, changing the use of analogies, modifying question frequency). Underperforming strategies are pruned from the pool. Full lineage tracking records which parent strategy produced each variant and its performance trajectory, enabling analysis of which mutations tend to improve outcomes. This creates a system that develops better teaching intuition over time, analogous to how an experienced human tutor refines their approach through years of practice.

6. LMS Integration Layer

Docere integrates with learning management systems through an **abstract adapter layer** with concrete implementations for Canvas and Moodle. This abstraction ensures that adding support for new LMS platforms requires only implementing the adapter interface without modifying core logic.

Authentication uses the **LTI (Learning Tools Interoperability)** standard—students click a link in their LMS and are automatically authenticated via single sign-on, with no separate account creation required. On first launch, a full synchronization pulls the student's courses, assignments, grades, and course materials. Subsequent sessions trigger **incremental syncs every two hours**, updating only changed data to minimize API calls and latency. The sync pipeline handles the mapping between LMS-specific data models (which differ significantly between Canvas and Moodle) and Docere's internal schema.

Google services—Calendar, Sheets, Drive, and Gmail—are integrated via OAuth2 with encrypted token storage, enabling features like office hours booking, gradebook sync from spreadsheets, and document access.

7. Student Experience & Product Features

Students access Docere directly from their LMS through an LTI link—there is no separate login or onboarding. They land in a conversational chat interface with their AI tutor, which remembers their history and adapts to their level. The tutor responds with conversational explanations and can generate structured artifacts inline: **flashcards** (automatically saved to per-student decks with FSRS scheduling), **study guides**, and **interactive notes**.

Beyond tutoring, the student workspace includes a **focus mode** with integrated Pomodoro timers for structured study sessions, a **Kanban task board** for organizing assignments and study plans, and **office hours booking** through Google Calendar integration. These features are designed to support the complete study workflow, not just the question-answering moment.

8. Instructor Dashboard & Analytics

Instructors have access to a comprehensive analytics dashboard providing visibility into student learning at both the individual and class-wide level. Key features include **engagement breakdowns** showing which students are actively using the tutor and how frequently, **confusion hotspots** identifying specific topics or assignments where students are consistently struggling, and **per-student concept mastery** visualizations showing each student's BKT-derived mastery scores across all tracked concepts.

A **3D concept graph** provides a spatial visualization of how students relate to topics, clustering related concepts and showing mastery gradients. Instructors can also interact with a **classroom agent**—a separate AI interface that synthesizes data-driven insights about class performance, answering questions like "which students are falling behind on linear equations?" or "what topics should I review before the next exam?" A guided **gradebook sync wizard** allows instructors to import grades from Excel or Google Sheets directly into their LMS, reducing manual data entry.

9. Spaced Repetition (FSRS v6)

Flashcards generated during tutoring sessions are automatically saved to per-student decks and scheduled for review using the **FSRS v6 (Free Spaced Repetition Scheduler)** algorithm. FSRS is a modern alternative to SM-2 (the algorithm behind Anki) that uses a three-parameter memory model to predict the probability of recall as a function of time since last review, current stability (how well the memory is consolidated), and difficulty. Cards are scheduled to appear at the optimal moment when recall probability drops to a target threshold (typically 90%), maximizing retention while minimizing review burden.

The integration with the tutoring system is seamless: when the tutor generates a flashcard during a conversation, it is automatically added to the student's deck with initial FSRS parameters. Review results feed back into the student's mastery scores, creating a connection between the spaced repetition engine and the Bayesian Knowledge Tracing system.

10. Experimental Framework & Ablation Studies

Docere includes built-in ablation study infrastructure designed for rigorous evaluation of the platform's components. The experimental framework supports **three arms**: a **control** group (baseline tutoring without memory or adaptive strategies), a **memory-only treatment** (full memory infrastructure but fixed teaching strategy), and a **full treatment** (memory infrastructure plus adaptive strategy engine). Students are assigned to arms using **deterministic hashing** based on their student ID, ensuring reproducible assignment without the need for a separate randomization service.

This design isolates the contribution of each component: comparing control to memory-only measures the impact of persistent context, while comparing memory-only to full treatment measures the additional impact of adaptive strategy selection. Primary outcome metrics include understanding delta (as measured by the process verification system), concept mastery trajectory (BKT scores over time), and engagement retention (session frequency and duration over weeks). The infrastructure is designed to produce publication-quality evidence of the system's educational efficacy.

11. Deployment & Infrastructure

The entire platform is containerized with **Docker** and deployed through **GitHub Actions CI/CD** pipelines to GitHub Container Registry. Separate staging and production pipelines ensure that changes are validated in a staging environment before reaching students. The deployment pipeline includes automated testing, container

building, registry pushing, and rolling updates to the production environment.

The async architecture ensures that the system scales gracefully under load. The critical synchronous path (memory retrieval, strategy selection, LLM call) is optimized for latency—returning responses in under one second—while all analytical and maintenance workloads (scoring, concept extraction, mastery updates, memory compression) are offloaded to the Redis-backed ARQ task queue and processed asynchronously.

12. Technology Stack Summary

Component	Technology / Approach
Backend	FastAPI (async Python), ARQ task queue
Frontend	React 19, TypeScript, Tailwind CSS
LLM	Anthropic Claude via LangGraph 6-node state machine
Relational DB	PostgreSQL (profiles, mastery, strategies, grades)
Vector DB	Qdrant (course materials, interaction history, embeddings)
Cache / Queue	Redis (caching, ARQ background task queue)
Knowledge Tracing	Bayesian Knowledge Tracing (per-concept mastery 0.0–1.0)
Spaced Repetition	FSRS v6 (flashcard scheduling with 3-param memory model)
Strategy Engine	UCB1 bandit + weekly evolutionary mutation with lineage
LMS Integration	Abstract adapter layer: Canvas + Moodle via LTI SSO
Google Services	Calendar, Sheets, Drive, Gmail via OAuth2
Scoring	4-dimension process verification (helpfulness, clarity, delta, engagement)
Deployment	Docker, GitHub Actions CI/CD, GitHub Container Registry
Experiments	3-arm ablation: control, memory-only, full treatment