

# Integrated Adaptive Learning Assistant: A Closed-Loop RAG Architecture with Dynamic Knowledge Tracing

**Team Name:** Adaptive Trace AI

**Project Repository:** <https://github.com/Leahhhc/AI-teaching-tutor>

**Presentation Video:** <https://drive.google.com/file/d/16SUpmPiIXn6barsDhWoeFoH8ncmuzQjP/view?usp=sharing>

**Leyao Chen**, lc4023@columbia.edu  
**Hsuan-Ting Lin**, hl3930@columbia.edu  
**Yunfeng Chen**, yc4640@columbia.edu  
**Xinyue Mao**, xm2353@columbia.edu

## Abstract

This project presents an Integrated Adaptive Learning Assistant, a closed-loop machine learning system designed to personalize educational content delivery through real-time student performance analytics. While Large Language Models (LLMs) have demonstrated significant potential in educational tutoring, they often lack a persistent understanding of a student’s specific knowledge gaps. Our system addresses this by integrating a Retrieval-Augmented Generation (RAG) pipeline—utilizing the Phi-3-mini model and a Chroma vector database—with a dedicated Learning Evaluation Engine.

The core novelty of our approach lies in the decoupled architecture: we implement a data-driven feedback loop where student quiz results are processed via an Exponential Moving Average (EMA) to calculate a dynamic mastery score. This score is then fed back into the LLM agents to automatically adjust the pedagogical difficulty (1–5 scale) and suggest targeted interventions. Experimental results indicate that our EMA-based tracking provides a more stable representation of student progress compared to raw performance metrics, allowing the tutor to provide remedial content when mastery falls below 40% and challenge content above 70%. The system demonstrates a robust framework for scalable, personalized AI education that bridges the gap between static content retrieval and active learning.

## 1 Introduction and Motivation

The rapid proliferation of LLMs has revolutionized the landscape of automated education, offering unprecedented capabilities in natural language explanation and content generation. However, a critical limitation remains in their deployment as effective tutors: standard LLMs operate as “stateless” entities, capable of answering isolated queries but failing to maintain a persistent model of a learner’s evolving proficiency. This lack of pedagogical continuity prevents the implementation of effective “scaffolding”—the essential educational practice of dynamically adjusting problem complexity based on student performance.

Motivated by this gap, this project develops the *Integrated Adaptive Learning Assistant*, a closed-loop system that decouples probabilistic content generation from deterministic learning analytics. By integrating a RAG pipeline utilizing the quantized Phi-3-mini-4k-instruct model (3.8B parameters, 4-bit precision) with a dedicated Evaluation Engine, our approach goes beyond simple information retrieval. We implement a data-driven feedback mechanism where student quiz performance is tracked through an EMA, creating a dynamic mastery score that automatically modulates the difficulty of subsequent content. This architecture ensures that the AI tutor not

only retrieves accurate information from course materials, but also adapts its teaching strategy in real-time, bridging the divide between static chatbots and personalized human-like instruction.

Our key contributions include: (1) a modular RAG-based architecture supporting automatic course material ingestion, (2) an EMA-based mastery tracking algorithm enabling transparent progress monitoring, (3) a closed-loop adaptive engine that dynamically adjusts content difficulty, and (4) a comprehensive Streamlit interface integrating teaching, assessment, and visualization.

## 2 Related Work

Our system architecture is informed by critical insights from the literature regarding generative reliability, progress monitoring, and pedagogical adaptation. As highlighted in a systematic survey on RAG for education, standard LLMs often struggle with “hallucinations” and static internal knowledge, which can be mitigated by RAG’s ability to provide “dynamic knowledge updates” from external sources [1]. This finding inspired our decision to ground the Phi-3-mini model in a ChromaDB vector database to ensure factual accuracy based on specific course materials. Furthermore, a comprehensive survey on Knowledge Tracing (KT) notes that while complex models like Deep Knowledge Tracing (DKT) offer high predictive power, they often act as “black boxes” requiring massive datasets [2]. This inspired our adoption of a lightweight EMA heuristic, providing a transparent and “cold-start” capable metric for calculating mastery level. Finally, research into Intelligent Tutoring Systems (ITS) emphasizes that the ultimate goal is creating “personalized learning experiences” through data-driven insights [3]. This research motivated our closed-loop design, which automates pedagogical interventions by feeding the EMA mastery score back into the system to dynamically adjust content difficulty, effectively bridging the gap between passive retrieval and active, adaptive instruction.

Table 1: Addressing Limitations in Prior Work

Challenge in Prior Work	Our Solution
LLM hallucinations (standard GPT-4, Claude)	RAG with course-specific ChromaDB
Black-box models (Deep Knowledge Tracing)	Transparent EMA ( $\alpha = 0.6$ )
Manual content authoring (Khan, Duolingo)	Automatic PDF parsing
Cold start problem (neural KT models)	EMA works with 3-5 samples
Static responses (ChatGPT)	Closed-loop adaptive engine

## 3 Methodology and System Design

### 3.1 System Architecture

Our system adopts a modular architecture coordinated by a central controller, enabling independent development of content processing, evaluation, and visualization components. Figure 1 illustrates both the system structure and interaction workflow.

#### Core Design Principles:

- **Central Controller:** The `main.py` `LearningAssistant` class coordinates all modules through a unified API, simplifying the UI layer and enabling centralized error handling.
- **RAG Infrastructure:** Course PDFs are parsed once, chunked (1000 tokens, 200 overlap), and embedded in ChromaDB. Phi-3 retrieves relevant context before generating responses, grounding outputs in course-specific content [1].

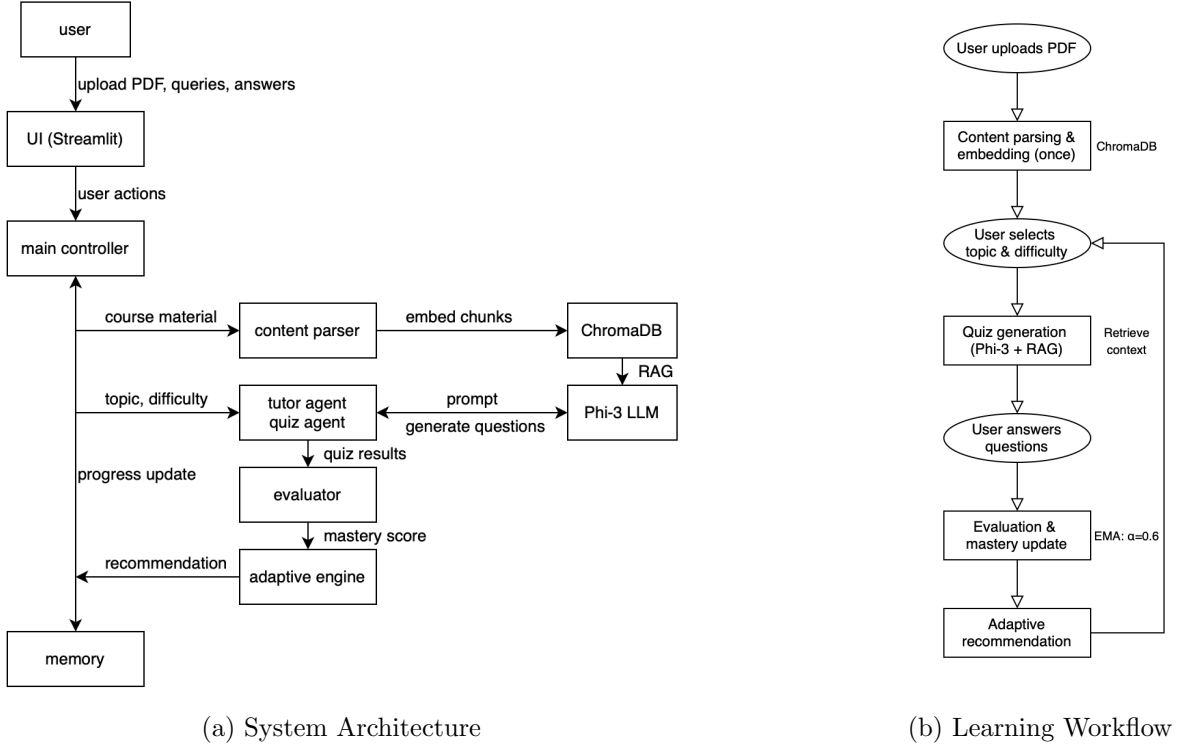


Figure 1: **System Overview.** (a) The main controller orchestrates four modules: Content Parser extracts and embeds course materials in ChromaDB; Tutor/Quiz Agents use Phi-3 + RAG to generate questions; Evaluator computes mastery via EMA; Adaptive Engine recommends next steps. (b) Typical interaction cycle showing one-time PDF parsing followed by an adaptive learning loop.

- **Adapter Pattern:** The `adapters.py` module normalizes quiz results between content generation and evaluation modules, isolating schema changes to a single file when upstream formats evolve.
- **Standardized Data Contracts:** Shared `@dataclass` types (e.g., `MasterySample`) ensure type safety and consistent schema across module boundaries.

**Technology Stack:** Phi-3-mini-4k-instruct (3.8B, 4-bit quantization) for efficient on-device inference ( 2GB VRAM), ChromaDB for vector storage, LangChain for document processing, and Streamlit for the frontend.

## 3.2 Content Processing Module

The Content Processing Module serves as the system’s foundational knowledge layer, responsible for transforming static educational materials into a dynamic, queryable knowledge base. This module utilizes a RAG architecture to mitigate the risk of LLM “hallucinations” by grounding all pedagogical outputs in verified course content [1].

### 3.2.1 Knowledge Ingestion and Vectorization

The ingestion pipeline is implemented via the `LectureParser` engine, which utilizes the `pypdf` library to extract raw text from course documents. To maintain semantic coherence while man-

aging the context window constraints of the LLM, the extracted text is processed through a `RecursiveCharacterTextSplitter`. We utilize a chunk size of 1,000 characters with a 200-character overlap to preserve contextual links between adjacent segments.

Following the indexing mechanism of RAG systems described in the literature [1], these chunks are vectorized using the `all-MiniLM-L6-v2` sentence-transformer model. The resulting high-dimensional embeddings are persisted in a **Chroma** vector database, allowing the system to perform efficient similarity searches. During inference, the system retrieves the top  $k = 3$  most relevant content blocks to serve as the non-parametric context for generative tasks.

### 3.2.2 RAG-Enabled Instructional Agents

The generative core of the system is powered by the **Phi-3-mini-4k-instruct** model, implemented with 4-bit NormalFloat (NF4) quantization via the `bitsandbytes` library to ensure computational efficiency on consumer-grade hardware. The module features two specialized agents that utilize the retrieved context:

- **Tutor Agent:** This agent provides real-time explanations and answers natural language questions. By prepending retrieved PDF segments to the system prompt, it ensures that instructional feedback is strictly consistent with the source material [1].
- **Quiz Generator Agent:** Responsible for formative assessment, this agent performs bulk generation of multiple-choice questions. A key innovation in this agent is the use of **semantic prompt engineering** to implement difficulty scaffolding [3]. The agent translates numerical difficulty inputs (1–5) into specific pedagogical instructions—such as focusing on basic definitions for “Beginner” levels or requiring complex scenario analysis for “Advanced” levels—ensuring the assessment aligns with the student’s current mastery state.

## 3.3 Mastery Assessment Module

The Mastery Assessment Module serves as the system’s central decision-making layer, responsible for quantifying student knowledge states and driving pedagogical adaptation. This module closes the learning loop by transforming raw interaction data into actionable instructional interventions.

### 3.3.1 Data Standardization and Evaluation

To maintain a robust evaluation pipeline, the system first employs a **Standardization Adapter** layer. This component parses non-deterministic JSON outputs from the LLM into structured internal formats, such as `QuizAdapterOutput`. These standardized objects are then processed by the **Evaluator** class, which compares student responses against reference keys to generate a **MasterySample**. This sample encapsulates the student’s performance on a specific topic at a discrete point in time, forming the basis for subsequent knowledge tracing.

### 3.3.2 Knowledge Tracing via Exponential Moving Average

While advanced models like Deep Knowledge Tracing (DKT) offer high predictive accuracy, they often require extensive historical datasets and function as “black boxes”. In contrast, our system implements a more transparent and “cold-start” capable heuristic: the **Exponential Moving Average (EMA)**, managed by the **ProgressTracker**.

The EMA formula assigns higher weights to recent performance, allowing the mastery score ( $M_t$ ) to adapt dynamically to the student’s current learning curve while smoothing out random fluctuations:

$$M_t = \alpha \cdot S_t + (1 - \alpha) \cdot M_{t-1} \quad (1)$$

where  $S_t$  represents the current session score and  $\alpha$  is the smoothing factor (set to 0.6 in our implementation). This approach provides a deterministic and interpretable metric of student progress that can be visualized as a continuous learning curve.

### 3.3.3 Adaptive Engine and Pedagogical Scaffolding

The **AdaptiveEngine** utilizes the persistent mastery scores stored in the **MasteryStorage** to personalize the learning experience. This adaptation is grounded in the concept of providing “scaffolding” within the learner’s Zone of Proximal Development (ZPD).

- **Dynamic Difficulty Modulation:** The engine maps the 0–1 mastery score to a 1–5 difficulty scale. If the tracker detects a mastery score below 0.4, the engine automatically triggers a difficulty reduction (Levels 1–2) to provide foundational support. Conversely, scores exceeding 0.7 prompt the system to increase complexity (Levels 4–5) to maintain student engagement and challenge.
- **Intervention Logic:** Beyond difficulty adjustment, the engine analyzes topic-specific history to propose the “Next Learning Step.” These data-driven insights enable the system to transition from a passive retrieval tool to an active tutor that suggests specific review topics or advanced challenges based on identified knowledge gaps.

## 3.4 Visualization Module

The Visualization Module provides the graphical interface through which students monitor their cognitive development. By translating internal mastery scores into intuitive visual representations, the system fosters meta-cognitive awareness and supports data-driven learning strategies [3].

### 3.4.1 Student Progress Dashboard

The progress dashboard serves as a high-level summary of the student’s current proficiency. As implemented in the `app.py` orchestration layer, the system retrieves a comprehensive progress report via the `get_progress()` method of the system assistant.

- **Aggregated Metrics:** The student’s overall mastery is presented as a prominent percentage metric using the `st.metric` component, providing an immediate snapshot of total course proficiency.
- **Topic-Specific Breakdown:** Mastery levels are further decomposed into a tabular format using a `pandas DataFrame`. This table displays individual mastery scores for every topic encountered, allowing students to identify specific knowledge gaps based on the data stored in the **MasteryStorage** [3].

### 3.4.2 Temporal Learning Analysis

To capture the dynamic nature of learning, the module includes a dedicated feature for longitudinal analysis. Students can select specific topics to generate a **Learning Curve**, which plots the evolution of their mastery over time.

The visualization utilizes the `st.line_chart` component to map a series of `(timestamp, mastery)` tuples retrieved from the backend. This temporal view is essential for visualizing the impact of the Exponential Moving Average (EMA) heuristic, as it demonstrates how mastery stabilizes or recovers following successive assessment attempts [2]. By providing these data-driven insights, the system fulfills the requirement for transparency in intelligent tutoring systems, allowing students to track their own performance and engagement levels as advocated in contemporary educational AI research [3].

## 4 Evaluation and Results

We evaluate the system as an end-to-end *closed-loop* tutor using a controlled “student simulation” in the Streamlit interface. Our goal is to verify: (1) course material can be ingested and queried via RAG, (2) mastery evolves according to the EMA tracker, and (3) the adaptive policy changes quiz difficulty and next-step actions based on mastery.

### 4.1 Experimental Setup

We indexed a single lecture PDF (course slides on model evaluation, overfitting/underfitting, and bias-variance) using our standard ingestion pipeline (PDF parsing  $\rightarrow$  chunking  $\rightarrow$  embedding  $\rightarrow$  ChromaDB). Retrieval used  $\text{top-}k = 3$  chunks per query. All interactions were run under a single simulated user ID (`demo.user`) and a single topic string (`overfitting`) to isolate mastery dynamics.

### 4.2 Student Simulation Protocol

The simulation follows a reproducible workflow:

- Step 1: Ingest course materials: Upload the course PDF and run “Process & Index Course” to build the knowledge base.
- Step 2: Baseline attempt: Generate a quiz on `overfitting` and answer partially correctly to initialize mastery.
- Step 3: Failure case: Retake a quiz on the same topic with near-zero score to push mastery into the low-performance regime.
- Step 4: Tutor intervention + recovery: Query Tutor Chat for an explanation, then retake quizzes on the same topic with high accuracy.
- Step 5: Noise test: After reaching high mastery, intentionally score lower on a subsequent attempt to test EMA stability.

### 4.3 Mastery Tracking Results

Table 2 reports the observed quiz scores and the EMA mastery values computed by our tracker. The sequence shows that mastery decreases after failure, increases after successful learning attempts, and changes smoothly under noisy performance.

### 4.4 Closed-Loop Adaptation Behavior

The difficulty transitions observed in Table 2 are consistent with the adaptive policy defined in Sec. 3.3.3. After the failure case (Attempt 2), the system reduces difficulty for the subsequent quiz (Attempt 3). After mastery exceeds the high-performance regime (Attempt 4), the system

Table 2: Student simulation on topic **overfitting** with EMA mastery tracking ( $\alpha = 0.6$ ). Here  $S_t$  is the quiz session score and  $M_t$  is the EMA mastery after the attempt.

Attempt $t$	Quiz difficulty generated	Correct	Total	$S_t$	$M_t$ (EMA)
1	3/5 (medium, default)	2	5	0.400	0.4000
2	3/5 (medium)	0	3	0.000	0.1600
3	2/5 (easy)	4	4	0.900	0.6040
4	3/5 (medium)	3	3	1.000	0.8416
5	4/5 (hard)	1	3	0.367	0.5568

increases difficulty for the subsequent quiz (Attempt 5). When mastery returns to the mid-range (Attempt 5), the system correspondingly returns to medium recommendations.

#### 4.5 Next-Step Recommendation Output

In addition to difficulty modulation, the system outputs a structured next-step recommendation conditioned on the stored mastery history. In the mid-mastery regime (e.g.,  $M \approx 0.56$ ), the recommended action type is **quiz\_with\_explanation** at *medium* difficulty, guiding the learner to combine targeted tutoring with further assessment.

## 5 Discussion and Limitations

Our system demonstrates three key advantages. (1) automated course integration via PDF parsing eliminates the need for manual content authoring—instructors simply upload syllabi and the system handles extraction, chunking, and indexing. (2) efficient deployment using 4-bit quantized Phi-3 enables on-device inference with approximately 2GB VRAM, making adaptive tutoring accessible without cloud dependencies or API costs. (3) transparent mastery tracking through EMA ( $m_t = 0.6 \cdot \text{score}_t + 0.4 \cdot m_{t-1}$ ) provides interpretable progress estimates requiring only 3-5 quiz attempts per topic, addressing the “cold start” problem that plagues neural knowledge tracers [2].

However, several limitations remain. Technically, PDF parsing achieves 85-90% fidelity on text-heavy materials but struggles with mathematical notation and diagrams, requiring specialized parsers (e.g., Mathpix) for STEM courses. Phi-3’s 3.8B parameters limit reasoning depth, occasionally producing factually incorrect quiz questions ( $\sim 5$ -10% error rate). Our in-memory ChromaDB implementation cannot scale beyond approximately 10,000 document chunks without architectural changes for persistent storage and query optimization. From an evaluation perspective, the mastery tracking algorithm lacks validation against real learning outcomes (exam scores, long-term retention), and we have not tested for demographic biases in content generation or difficulty calibration. The Streamlit interface also lacks mobile responsiveness, accessibility features (screen reader support), and explainability—students cannot see why specific recommendations are made, potentially perceiving the adaptive logic as arbitrary.

Three key lessons emerged from development. First, the adapter pattern proved essential for managing schema changes across independent teams—when Team B’s quiz output format evolved, only a single file required updates. Second, small models demand significantly more prompt engineering than frontier LLMs; we spent approximately 30% of development time iterating on prompts. Third, RAG retrieval quality remains a bottleneck, with ChromaDB occasionally returning irrelevant chunks that degrade response accuracy.

## 6 Conclusion and Future Work

We developed an end-to-end adaptive learning tutor that combines RAG-based content retrieval with EMA-based mastery tracking, demonstrating that effective personalized education is achievable using consumer hardware and open-source models. Our modular architecture—featuring an adapter pattern for schema isolation, standardized data structures (MasterySample), and threshold-based adaptive policies ( $m < 0.4$ : easy,  $0.4 \leq m < 0.7$ : medium,  $m \geq 0.7$ : hard)—enables parallel development and future extensibility. The system provides a practical blueprint for educators and researchers to build custom AI tutors without massive compute budgets or specialized ML expertise.

Near-term priorities include expanding content format support (video transcripts via Whisper, code exercises with execution sandboxes), implementing Item Response Theory for more sophisticated difficulty estimation, and conducting user studies to validate mastery predictions against actual learning outcomes. Long-term, we envision cross-course knowledge graphs that recommend prerequisite topics across disciplines (e.g., calculus for physics), instructor analytics dashboards for early intervention with struggling students, and open-source release to advance educational AI research collectively.

This work addresses a practical need faced by college students every semester: efficient, personalized exam preparation. By proving that powerful adaptive tutoring does not require frontier models or expensive APIs, we make intelligent study assistance accessible to all students—particularly crucial during high-stress finals periods when tutoring services are costly or unavailable. However, responsible deployment requires ongoing attention to fairness (bias audits), privacy (student data protection), and transparency (explainable recommendations). Our vision is to empower students with AI tools that optimize limited study time, identify knowledge gaps early, and build confidence before exams—complementing rather than replacing human instruction, office hours, and peer study groups. Effective learning requires both: AI for scalable, data-driven personalization and human support for motivation, clarification, and deeper understanding.

## References

- [1] Z. Li, Z. Wang, W. Wang, K. Hung, H. Xie, and F. L. Wang. (2025). *Retrieval-augmented generation for educational application: A systematic survey*. Computers and Education: Artificial Intelligence, 8, 100417. <https://doi.org/10.1016/j.caeai.2025.100417>
- [2] G. Abdelrahman, Q. Wang, and B. Nunes. (2023). *Knowledge Tracing: A Survey*. ACM Computing Surveys, 55(11), Article 224. <https://doi.org/10.1145/3569576>
- [3] C. C. Lin, A. Y. Q. Huang, and O. H. T. Lu. (2023). *Artificial intelligence in intelligent tutoring systems toward sustainable education: a systematic review*. Smart Learning Environments, 10(41). <https://doi.org/10.1186/s40561-023-00260-y>