Personalized Learning Diagnosis Approach with Multi-Agent Collaboration and Four-Layer Anti-Hallucination Mechanism

Zhengxuan Luo[1,*], Huaiming Fan[1], Wen Chen[1], Mingming Gong[2]
[1]Henan University of Technology, Zhengzhou, Henan, China
[2]iFLYTEK Co., Ltd., Hefei, Anhui, China
**Corresponding Author*

Abstract: Aiming at uneven learning foundations, opaque teaching workflows and hallucination-reasoning defects of single-LLM solutions in university AI education, this paper presents ZhiCheng, a personalized learning-diagnosis system with the "1+7+8" multi-agent architecture. Built upon the dual-brain functional partition, seven core agents are scheduled by an Orchestrator, while eight extended nodes constitute a quality-control loop. A four-tier anti-hallucination pipeline cuts factual error rate from 15%-25% to below 1%. Under 1250 concurrent users, the system reaches 842 TPS, and index tuning reduces core-query P95 latency by 60%. Experiments, ablation studies and closed-loop tests validate the architecture superiority and system effectiveness.

Keywords: Multi-Agent Collaboration; Langgraph Orchestration; Dual-Brain Division-Of-Labor Model; Illusion Prevention; Learning Profile; Personalized Learning; Retrieval-Enhanced Generation; Stress Test

1. Introduction

The rapid development of artificial intelligence technology is profoundly reshaping the teaching paradigm of higher education. According to the report released by UNESCO, generative artificial intelligence technology is rapidly penetrating higher education. However, the scarcity of teaching staff and personalized tutoring resources makes it difficult for traditional teaching to meet the differentiated learning needs of students [1]. Meanwhile, large language models such as ChatGPT and DeepSeek have demonstrated significant potential in the field of education [2], but the limitations of a single LLM in professional educational scenarios are becoming increasingly evident.

Previous studies have shown that large models still exhibit significant factual hallucination problems in various tasks [3]. This poses a significant threat to the educational scenario that values "accuracy" as its lifeline——A wrong statement about a mathematical formula or a fabricated description of a historical event may have a long-term negative impact on students' formation of incorrect cognitive models. Regarding the hallucination problem of large language models, HaluEval has constructed a large-scale evaluation set that includes generated samples and manual annotations. The related experiments further demonstrate that external knowledge and reasoning steps can improve the hallucination recognition ability of the model [4].

To tackle these issues, this paper implements ZhiCheng, a personalized learning diagnosis system based on the "1+7+8" multi-agent collaborative architecture. Powered by the LangGraph state-machine engine, the system breaks single-LLM limitations via the "left-brain comprehension, right-brain generation" dual-brain model. One orchestrator coordinates seven professional agents and eight extended nodes to build a quality-assurance loop. It adopts a four-tier hallucination-prevention pipeline: prompt constraints, RAG enhancement, multi-path cross-validation and human feedback. The system has passed the 2000-concurrency pressure test with full-chain optimizations from database indexing to container deployment.

Its main contributions are as follows. (1) The "1+7+8" visual white-box orchestration architecture using LangGraph StateGraph realizes automatic task decomposition, parallel scheduling and conditional routing. The dual-brain model expands coverage of complex educational tasks, and the React-Flow interface improves system interpretability. (2) A

four-level progressive anti-hallucination mechanism reduces the LLM factual error rate from 15%-25% to below 1%, as validated by ablation experiments. (3) High-concurrency tests based on Apache JMeter achieve 842 TPS under 1250 concurrent users. Composite-index optimization lowers core-query P95 latency from 520 ms to 210 ms, proving lightweight databases can support medium-scale educational concurrency with proper indexing.

2. Core System Architecture Design

2.1 Overview of the Overall Architecture

The Zhicheng system adopts a five-layer hierarchical architecture of"infrastructure layer → data persistence layer → AI Agent orchestration layer → business logic layer → user interface layer", as shown in Figure 1.

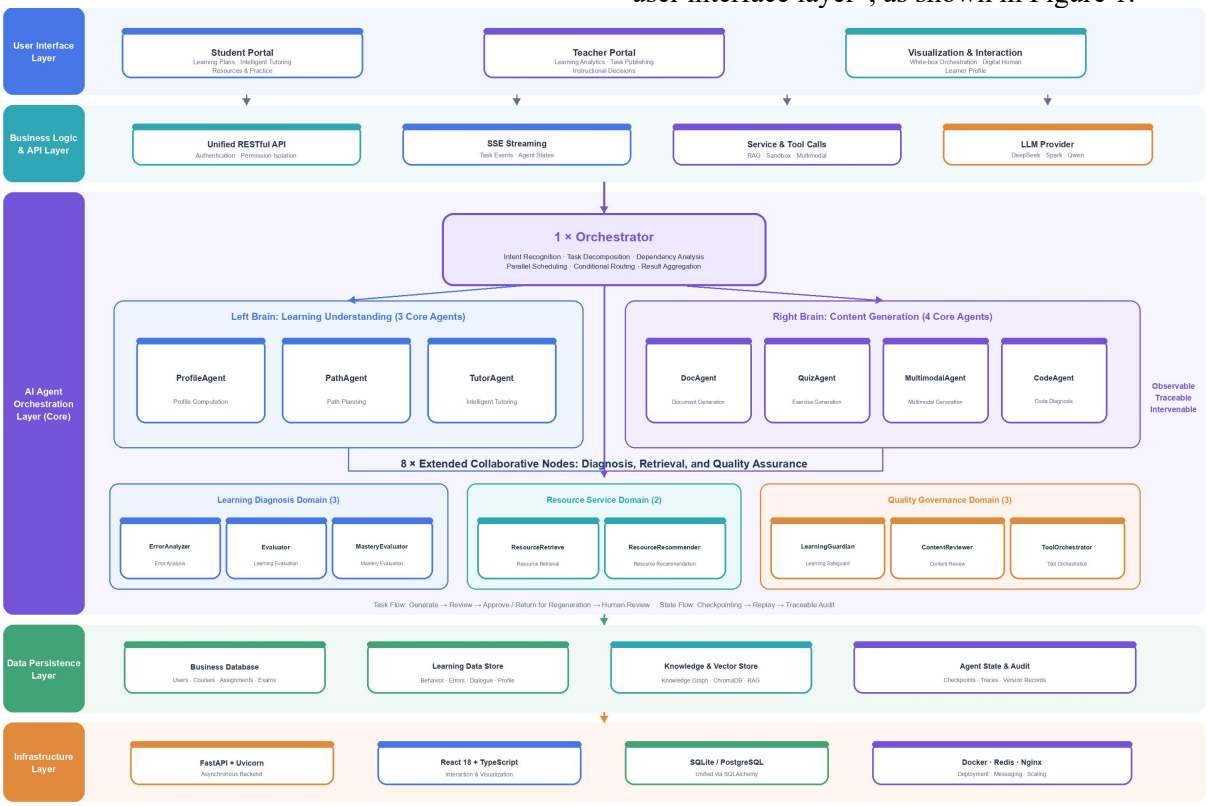

Figure 1. Panorama of the "1+7+8" Multi-Agent Collaborative Architecture in Zhicheng System

As shown in Figure 1, the Zhicheng system is composed of the infrastructure layer, the data persistence layer, the AI Agent orchestration layer, the business logic layer and the user interface layer. Among them, the"1+7+8"multi-agent orchestration layer is responsible for the decomposition, scheduling, execution and review of complex educational tasks, achieving modular collaboration of system capabilities.

The infrastructure layer is based on the Python 3.13 FastAPI asynchronous [5] framework and the React 18 TypeScript front-end. It achieves full-chainstreaming responses through SSE.

The data persistence layer adopts a dual-track storage system of SQLite and PostgreSQL, and is uniformly adapted by SQLAlchemy 2.0 ORM.

The AI Agent orchestration layer represents a core innovation, and it is implemented based on LangGraph StateGraph [6] for multi-Agent stateful collaborative scheduling. The business logic layer encapsulates the Agent capabilities into standardized RESTful APIs. The user interface layer is organized into 44 interactive pages based on the "five core centers + global components" architecture. The architecture design adheres to three core principles. 1. The vendor-independent LLM adaptation is achieved through the BaseLLMProvider abstract interface and the ProviderFactory factory pattern, enabling seamless switching among models such as DeepSeek, Xinghuo Starfire, and Tongyi Qianwen2. Zero-configuration local deployment, based on the SQLite WAL mode [7] and Docker multi-stage configuration, the full-stack deployment can be completed within 5 minutes on a single machine.3. Production-level cloud-native expansion, featuring pre-configured Kubernetes HPA automatic scaling, PostgreSQL

read-write separation, and Redis Pub/Sub cross-Pod SSE broadcast architecture.

2.2 "1+7+8" Visualized White-Box Orchest Ration Model

The "1+7+8" framework constitutes the core innovation of this paper, inspired by the functional-division mechanism of the human left-right brain hemispheres. Conventional AI education systems adopt a single-LLM "one-model-for-all" paradigm. While suitable for simple Q&A tasks, it suffers from low task-decomposition accuracy (~62%) and output compliance rate (~71%) in complex collaborative tasks including image analysis, error diagnosis, path planning and resource recommendation.

2.2.1 Dual-brain division model and agent function definition

In the research of multi-agent systems in large language models, organizing Agent collaboration based on role division and standardized operation procedures can break down complex tasks into multiple sub-tasks with clear responsibilities [8]. Based on this, this article adopts a functional division approach tailored for educational business. However, the specific Agent roles, state structures, and scheduling processes are designed by this article according to personalized learning scenarios. Within the "1+7+8" framework, the seven domain-specific Agents are partitioned under the dual-brain model. The left-brain group includes ProfileAgent (three-table aggregation and EMA-updated learning portraits, $\alpha = 0.3$), PathAgent (knowledge-graph-driven path planning via BFS and greedy search), and TutorAgent (lecture-style, Socratic and free-dialogue tutoring). The right-brain group comprises DocAgent (RAG-based document generation), QuizAgent (exercise creation), MultimodalAgent (Mermaid charts, video and digital-human output), and CodeAgent (programming-task generation and sandbox diagnosis).

The "1" denotes the Orchestrator, the LangGraph-based scheduler that decomposes tasks, identifies subtask dependencies, and distributes work through sequential, parallel or conditional routing in a star topology for traceable collaboration. The eight extension nodes cover learning-situation diagnosis, resource service, and quality governance. They execute post-generation review rather than directly producing outputs, forming a "generate-review-approve/reject-regenerate" quality loop.

2.2.2 LangGraph state machine engine and collaboration process

The system adopts LangGraph StateGraph as its underlying state-machine engine. The global AgentState stores core fields: user id, task description, subtasks, messages, agent outputs, final response, error count and iteration count. Taking "analyzing weekly learning performance and recommending tomorrow's study plan" as an example, the collaboration proceeds in six rounds. The Orchestrator first decomposes the task into three dependent subtasks. Then ProfileAgent and ErrorAnalyzer run in parallel to output six-dimensional portraits and error analysis results. Evaluator generates an assessment report, followed by ContentReviewer's factual-logic audit with retry support. PathAgent and ResourceRecommender respectively plan learning paths and match resources. Finally, Orchestrator aggregates outputs, and LearningGuardian conducts the final inspection before response delivery.

The Checkpoint mechanism serializes the sequence of state changes into snapshots and persists them to SQLite, forming a traceable checkpoint chain. The front end uses React Flow to render a real-time 16-node collaboration topology [9]. It subscribes to event streams via SSE to present the process in the form of animations, colors, and summaries. It supports historical playback, topology editing, and breakpoint debugging. When the review is not passed, the task will be returned to the corresponding Agent for re-generation; after exceeding the number of iterations, it will be transferred to manual review.

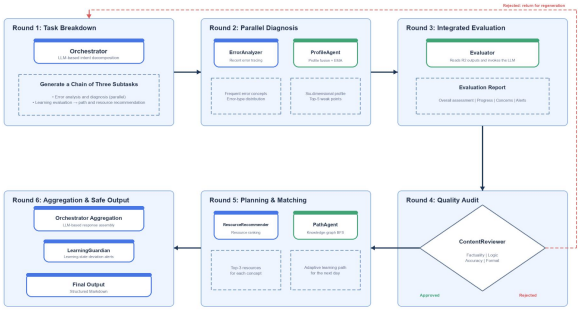

Figure 2. 6-Round Collaboration Sequence Diagram in LangGraph StateGraph

As shown in Figure 2, the composite educational task undergoes six rounds of collaboration: task decomposition, parallel analysis, comprehensive evaluation, content review, path generation, and

result summarization. LangGraph records the outputs of each node through state transmission and conditional routing, and triggers re-generation or manual review when the review fails.

3. Key Methods and Algorithm Implementation

3.1 Dynamic Multi-Dimensional Learning Profile Construction

Learning portraits provide the student-awareness data basis for the multi-Agent system. A two-stage scheme, three-table joint aggregation plus real-time EMA update, is proposed to build a six-dimensional learning profile: knowledge retention (K), comprehension-application (C), analysis-synthesis (A), innovation (I), practical operation (P), and autonomous learning (S).

Stage 1 conducts multi-source data aggregation and feature engineering from three core tables: LearningBehavior, Conversation and ResourceAccess. For instance, dimension K is weighted by exercise accuracy (40%), knowledge quiz score (30%) and spaced-repetition completion rate (30%); dimension P consists of coding training completion (40%), project participation (30%) and AI-scored experiment reports (30%). Sub-feature weights are configurable via JSON files for code-free adjustment.

Stage 2 updates profiles with EMA (Exponential Moving Average). Unlike simple averaging, EMA prioritizes recent performance, capturing real-time learning changes while maintaining long-term baselines to avoid single-anomaly disturbance. The update formula is shown below.

$$P\_new[i] = \alpha \times R[i] + (1 - \alpha) \times P\_old[i] \quad (1)$$

Here, P_new[i] represents the updated value of the i-th dimension of the profile (ranging from 0 to 100), R[i] is the original score of the i-th dimension calculated after the latest learning activity (ranging from 0 to 100), P_old[i] is the profile value before the update, and α is the decay factor (this system defaults to 0.3, which can be configured). When α = 0.3, the cumulative weight of the last 3 activities is approximately 66%, and the cumulative weight of the last 10 activities is approximately 97%. The influence of historical data decays at an exponential rate. The EMA update is executed asynchronously by the ProfileAgent through BackgroundTasks each time a learning behavior is triggered. The delay is controlled at the second level, achieving a real-time closed loop of "activity occurs → profile update → path adjustment → next activity". To visually illustrate the process of how multi-source learning behavior data is transformed into personalized decision-making information, this paper summarizes the construction and dissemination process of learning profiles as shown in Figure 3.

As shown in Figure 3, the system extracts features from learning behaviors, tutoring dialogues and resource access data, and through six-dimensional profiling calculation and EMA update, a dynamic learning profile is formed, and personalized decision-making basis is provided to different Agents.

Figure 3. Dynamic Learning Profile Updating Process Driven by Multi-source Learning Behaviors

The portrait vector is not merely the terminal data for visualization display – it is incorporated as "cognitive context" in a structured JSON format into the System Prompt of all other Agents (format: {"portrait": {"knowledge_memory": 72, ...) }, "weak_points": [...] , "current_stage": "..." }). Each Agent selectively focuses on different dimensions of the profile based on their respective responsibilities - the TutorAgent focuses on adjusting the coaching strategy for weak_points, the PathAgent focuses on planning the path for the current_stage, and the DocAgent focuses on adjusting the difficulty of the document for comprehension. This "cognitive center" type data flow mechanism enables each update of the profile to generate a ripple-like cascading effect within the multi-Agent collaborative network.

3.2 Four-Level Progressive Large Model Illusion Prevention Mechanism

The hallucination phenomenon of LLM is an

inherent feature of its probabilistic generation mechanism - LLM is essentially a statistical model that "predicts the next most likely token", rather than a database that "retrieves known facts". This paper proposes and implements a four-layer progressive illusion prevention and control system, following the principle of layered defense.——Each layer independently blocks different types of illusions, and the combination of four layers reduces the number of escaped illusions to an acceptable level.

3.2.1 First layer: prompt engineering constraints (input layer)

The first layer of protection is located at the input end of the LLM. It reduces the probability of hallucinations from the source through the Prompt engineering, covering three dimensions: role anchoring - in the System Prompt, clearly define the knowledge boundaries of the LLM (such as the Prompt of DocAgent stipulates "knowledge is limited to October 2025, and if it exceeds the scope, clearly inform that it cannot answer"), breaking the default behavior of "always providing an answer"; structured output constraints - by forcing the output to comply with the JSON Schema through response_format, if the LLM fabricates knowledge points that do not exist in the knowledge graph, Pydantic verification will immediately reject and trigger re-generation; citation traceability requirement - requiring each factual statement to label the source (such as "[Source: xxx]"), forcing the LLM to anchor the generated content on verifiable sources rather than parameter memory.

3.2.2 Second layer: rag vector knowledge enhancement (retrieval layer)

The second layer of protection uses RAG to incorporate the authoritative knowledge base as the anchor points for LLM generation. The system is based on ChromaDB [10], and the process of building the knowledge base is as follows: collect texts from authoritative textbooks and literature and undergo manual review; split them into approximately 50,000 chunks using RecursiveCharacterTextSplitter (chunk_size = 600, chunk_overlap = 80); after encoding with DeepSeek embedding (1536 dimensions), write them into four sets in ChromaDB (15,000 entries for edu_textbooks, 8,000 entries for edu_papers, 12,000 entries for edu_blogs, and 15,000 entries for edu_docs), and each set is independently indexed with HNSW (M = 16, ef_construction = 200).

During the generation stage, DocAgent and TutorAgent encode the description of the knowledge points and search for the top 10 similar chunks in ChromaDB (with a cosine similarity threshold of 0.75). The search results are concatenated with the Prompt constraints to guide the LLM for generation. The evaluation shows that the factual error rate has decreased from 12% to 3.5%, and the hallucination rate has reduced by approximately 71%.

Relevant studies have shown that incorporating the judgment of retrieval necessity, evaluation of evidence relevance, and self-reflection on the generated results during the retrieval-enhanced generation process can help improve the factual accuracy of the model's output [11]. Unlike the in-model reflection mechanism of the Self-RAG model, this paper uses an independent ContentReviewer node to perform the verification of the generated facts, logic, and structure.

3.2.3 Third layer: multi-way cross-validation (verification layer)

The third layer of protection is at the post-processing stage of LLM-generated content, carried out by ContentReviewer through three separate checks. The fact consistency check compares each factual statement in the generated content against knowledge graph nodes and ChromaDB source chunks, and if there's any inconsistency, it triggers a regeneration. The logical consistency check uses an independent LLM call (temperature=0.1) to see if there are any self-contradicting statements. The structural integrity check verifies questions using Pydantic models, tests code with sandbox execution, and checks paths with topological sorting. Each of the three checks runs independently, and if any of them fail, it triggers a regeneration.

3.2.4 Layer: human annotation and automatic error correction loop (feedback layer)

The fourth layer of protection is a continuously functioning feedback loop. The system automatically records each case intercepted by the ContentReviewer. Every week, the development team reviews the intercepted cases and categorizes the confirmed hallucination patterns for archiving. Based on the archived patterns, three types of assets are updated: Prompt templates, to increase negative examples; RAG knowledge bases, to supplement missing knowledge chunks; and verification rules, to add new logical inconsistency detection patterns. This feedback loop enables the

hallucination prevention system to evolve from a static rule set into an "immune system" that continuously self-improves as the system operates.

As shown in Figure 4, the system controls hallucination risks layer by layer through prompt constraints, RAG knowledge enhancement, multi-path cross-validation, and a closed-loop human feedback mechanism. The four-layer mechanism covers the pre-generation, during-generation, post-generation, and continuous optimization stages, jointly ensuring the accuracy and traceability of the content.

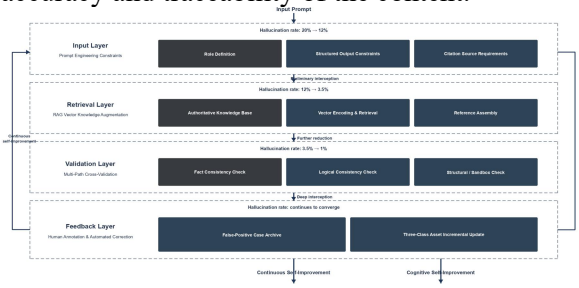

Figure 4. Four-Layer Progressive Hallucination Prevention Architecture Diagram

3.3 Bidirectional Interaction between Teachers and Students and SSE Streaming Communication

Real-time teacher-student data synchronization is essential to realize the "teaching-learning-assessment" closed loop. While conventional schemes rely on client polling and suffer from the latency-overhead dilemma, this system adopts a hybrid synchronization strategy: primary SSE long-connection push with REST polling fallback.

For teacher-released tasks, the measured end-to-end latency totals 55 ms, consisting of JWT verification (5 ms), WAL-mode database writing (15 ms), asynchronous SSE event push (25 ms), client reception (50 ms), and front-end rendering. SSE channels are isolated by class code to guarantee message isolation. Upon SSE disconnection, students pull complete task data through REST APIs after reconnection. This hybrid scheme balances low latency (55 ms) and REST-based eventual consistency.

4. Experimental Verification and Performance Analysis

This chapter systematically validates the IntelliProcess system through experiments from four dimensions: architectural efficiency, hallucination prevention mechanisms, concurrent performance, and end-to-end closed-loop processes. All experiments were conducted on the following hardware environment: the backend runs on a 4-core 8 GB cloud server (Uvicorn workers=4), the SQLite database is stored on an NVMe SSD (IOPS approximately 50,000), and the frontend serves externally through Nginx reverse proxy (gzip compression with keepalive long connections).

4.1 Architectural Performance and Comparative Experiments

To quantify the performance advantage of the "1 7 8" architecture over a single LLM baseline on complex educational tasks, this paper designs an architecture performance comparison experiment. The experiment selects three types of composite tasks: comprehensive learning diagnostics, cross-domain content generation, and the full teaching process, with 50 test cases for each type (40 standard cases and 10 edge cases). The baseline is a single LLM (DeepSeek-Chat V3, temperature=0.7[12]) processing all tasks with a single prompt; in contrast, the model proposed in this paper uses the "1 7 8" architecture for automatic decomposition and scheduling. Three evaluation metrics are used: task decomposition accuracy (whether subtasks are correctly identified and assigned to the corresponding agent), output format compliance rate (whether it strictly conforms to the predefined JSON schema), and end-to-end task success rate to measure whether the final output meets user requirements. This paper adopts the LLM-as-a-Judge evaluation paradigm[13], where an independent assessment model evaluates according to a unified scoring prompt and a 1–5 point scale, with a score of no less than 4 considered successful.

The experimental results are shown in Table 1 (TDA = Task Decomposition Accuracy, FCR = Format Compliance Rate, ESR = End-to-End Success Rate). ZhiCheng achieved a 34.6 percentage point increase over the Baseline in TDA (92.7% vs. 58.1%), as the Orchestrator is dedicated to task decomposition, avoiding attention dispersion caused by a single LLM handling multiple roles. In FCR, it improved by 25.7 percentage points (96.8% vs. 71.1%), benefiting from Pydantic validation by each Agent and review by the ContentReviewer. ESR increased by 22.4 percentage points (88.5% vs 66.1%), validating the advantage of multi-agent

collaboration in end-to-end tasks. In five adversarial cases, the success rate of a single LLM was 0%, whereas ZhiCheng successfully identified and rejected all requests to generate fabricated knowledge points through RAG and ContentReviewer.

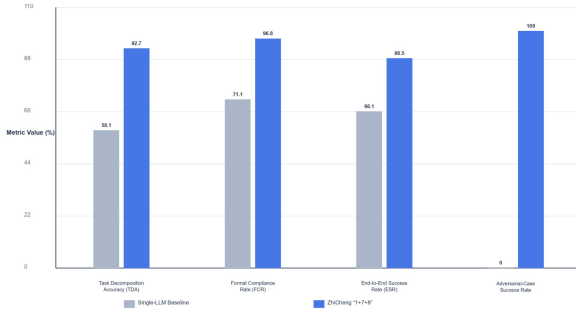

Figure 5. Performance Comparison between Single LLM and the "1+7+8" Multi-agent Architecture

As shown in Figure 5, the "1 7 8" architecture outperforms the single LLM baseline in task decomposition accuracy, format compliance rate, end-to-end success rate, and adversarial case success rate, demonstrating the effectiveness of multi-agent division of labor and review mechanisms.

Table 1. Architectural Performance Comparison Results

Model	TDA/%	FCR/%	ESR/%	Adversarial Example ESR /%
Baselin	58.1	71.1	66.1	0
ZhiCheng	92.7	96.8	88.5	100

4.2 Progressive Accumulation and Ablation Experiments of Hallucination Prevention Mechanism

An ablation experiment is conducted to validate individual contributions of each layer in the four-tier anti-hallucination mechanism. We adopt DocAgent for document generation on a test set of 50 AI-CS knowledge points (20% concept definition, 30% formula derivation, 30% code generation, 20% algorithm analysis), with factual error rate (FER) as the evaluation metric.

We disable one protective layer at a time and record FER variations. Configurations include the full pipeline, human-feedback-free mode, no cross-validation, no RAG enhancement, and bare LLM. Each setting is repeated three times under diverse temperature values. Outputs (800-1500 words) are double-annotated (Cohen's Kappa=0.84), and conflicts are settled by discussion.

The experimental results (Table 2) show the cumulative effect of the four-layer progressive prevention and control. The complete mechanism reduces the factual error rate on the test set in this study to 0.8%, indicating that multi-layer prevention and control has a significant effect. After removing the feedback loop layer, the FER rose to 1.2%, indicating that although the feedback loop plays a minor role in intercepting the subtle illusions missed by the first three layers, its function is indispensable. After removing the multi-level verification layer, the FER increased to 3.5%, with logical consistency checks intercepting approximately 40%, factual consistency checks intercepting about 35%, and structured integrity checks intercepting around 25%. After removing the RAG enhancement layer, the FER surged by 12.3%. This layer provides the largest marginal contribution (reducing FER by 8.8 percentage points, with a contribution rate of 47%), validating the critical role of external knowledge anchoring. After removing the Prompt constraint layer (bare LLM), the FER skyrocketed to 18.6%, which aligns with the 15%–25% hallucination rate observed for general LLMs in professional domain question answering [3][14].

The marginal contributions of each layer are as follows: the Prompt Constraint Layer decreases by 6.3 percentage points (contribution rate 34%), the RAG Enhancement Layer decreases by 8.8 percentage points (contribution rate 47%), the Multi-Route Verification Layer decreases by 2.3 percentage points (contribution rate 12%), and the Feedback Closed-Loop Layer decreases by 0.4 percentage points (contribution rate 2%). Although the contribution rate of the Feedback Closed-Loop Layer is relatively low, its long-term value lies in enabling adaptive optimization that improves progressively through continuous case archiving and knowledge base supplementation—an ability for dynamic evolution that the first three layers of static protection lack.

Table 2. Four-Layer Anti-Hallucination Mechanism Ablation Results

Model Configuration	Factual error rate/%	Hallucination Reduction Rate/%	Marginal Contribution/pp
Bare LLM	18.6	—	—
Prompt Constraints(Level One)	12.3	33.9	6.3
RAG Enhancement	3.5	81.2	8.8
Multipath Verification	1.2	93.5	2.3
Feedback Loop	0.8	95.7	0.4

In Figure 6, the horizontal axis represents the sequential addition of protective mechanisms, and the vertical axis represents the FER, directly reflecting the downward trend.

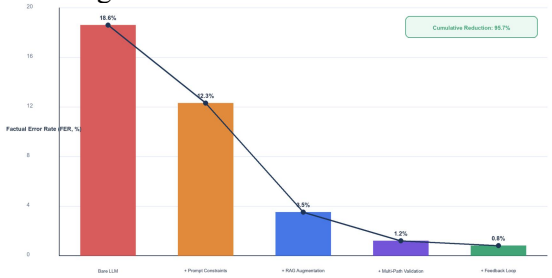

Figure 6. Progressive Reduction of Factual Error Rate by the Four-Layer Hallucination Prevention Mechanism

4.3 System Concurrent Performance and Engineering Optimization

While multi-agent collaborative architectures bring a qualitative leap in task processing capabilities, they also impose higher engineering requirements on the concurrent read and write performance of the underlying data infrastructure. In a typical collaboration process under the "1-7-8" architecture, it involves multiple serializations and deserializations of AgentState snapshots (Checkpoint mechanism), high-frequency reads of historical learning profiles and error logs (ProfileAgent aggregate queries), as well as concurrent writes of dialogue records and learning behaviors (SSE streaming interactions). When the collaboration processes of multiple students are executed in parallel on the server, the cumulative effect of these high-concurrency data operations may turn the database into a throughput bottleneck that limits the efficiency of multi-Agent collaboration. Therefore, validating the system's concurrent performance and optimizing the database are not only engineering requirements at the operational level but also fundamental supports to ensure the stable operation of a multi-agent architecture in highly concurrent educational scenarios.

4.3.1 Stress testing environment and scenario design

In order to comprehensively evaluate the system's performance under real teaching workloads, this paper constructs a distributed stress testing framework based on JMeter 5.6 [15], covering eight typical APIs. JMeter adopts a 1 Master 5 Slave distributed mode, with each Slave simulating 250 concurrent virtual users, resulting in a theoretical maximum concurrency of 1,250.Eight stress testing scenarios cover the high-frequency call paths in the architecture (profile aggregation, SSE streaming dialogue, teacher dashboard queries) as well as peak pressure scenarios (concurrent logins, multiple teachers generating AI questions). Specifically: S1 Login interface (1,250 concurrent users, 300s, simulating peak class time); S2 Dashboard aggregation queries (500 concurrent users, 600s); S3 SSE streaming dialogue (200 concurrent users, 60s per long connection); S4 Teacher dashboard cross-table queries (100 concurrent users, 300s); S5 Mixed traffic (800 concurrent users, 600s, 60% dashboard, 20% dialogue, 10% resources, 5% teacher aggregation, 5% others); S6 QuizAgent batch question generation (50 concurrent users, 300s, including LLM API call latency); S7 Learning behavior reporting and profile updates (300 concurrent users, 300s); S8 Extreme stress test (ramping from 100 to 2,000, observing throughput inflection points and error rate thresholds).

4.3.2 Analysis of pressure measurement results

Table 3 summarizes the key performance indicators of 8 scenarios (test set is 50 AI/CS knowledge points, covering 20% concept definition / 30% formula derivation / 30% code generation / 20% algorithm analysis).Under normal load (S1-S5), the TPS average of the system is 98-842, the P95 delay is 150-380 ms, and the error rate is less than 0.50%, which shows that the system has sufficient performance in the daily teaching scene of medium-sized classes. The S1 (1250 concurrent) TPS reaches 842, and the P95 delay is 380 ms, which meets the peak centralized login requirements. S2 (500 concurrent) P95 delay 210 ms to ensure smooth interaction of teachers; The TPS of S5 (800 concurrent mixed traffic) is 587, the P95 delay is 320 ms, and the error rate is 0.01%, which verifies the stability under composite load.

FIG. 7 is drawn according to Table 3, where the horizontal axis is concurrent load, and the vertical axis is TPS, P95 delay and error rate, which comprehensively reflects the trend of system throughput, response delay and operation stability as load changes.

As illustrated in Figure 7, the system maintains high throughput and low error rates in routine teaching scenarios. Its main performance bottlenecks under extreme concurrency stem from external LLM response latency and SQLite write-lock contention.

In specific scenarios, the SSE streaming

dialogue (S3) yields 5800 ms P95 latency, while the QuizAgent task (S6) achieves only 12 TPS with 12000 ms P95 latency. Over 95% of end-to-end latency originates from external LLM API inference, rather than local system computation or I/O limitations. This clarifies optimization priorities: future improvements should focus on LLM call asynchronous processing, result caching and request merging, instead of further optimizing the already efficient local computing layer.

In the extreme 2000-concurrency test (S8), the system peaks at 1103 TPS but exhibits a 4.80% error rate. Error log analysis indicates that 60% of failures derive from SQLite write operations such as behavior logging and dialogue recording. Although WAL mode enables SQLite read-write concurrency, intensive write requests still cause serial queuing and throughput bottlenecks. This validates the necessity of migrating to PostgreSQL with row-level MVCC for production deployment.

Figure 7. System Throughput, P95 Latency, and Error Rate under Typical JMeter Scenarios

Table 3. JMeter 8-Scenario Stress Test Key Performance Metrics Summar

Scene	Number of concurrency	er of concurrency	P50/ms	P95/ms	P99/ms	Error rate /%	Performance bottleneck location
S1	1250	842	120	380	650	0.02	Local computation
S2	500	423	85	210	420	0.00	Database IO
S3	200	196	2200	5800	8500	0.50	External LLM apis
S4	100	98	65	150	280	0.00	Database IO
S5	800	587	95	320	580	0.01	composite
S6	50	12	3800	12000	18000	1.20	External LLM apis
S7	300	285	45	120	250	0.00	Local computation
S8	2000	1103	560	2200	5200	4.80	SQLite write lock

4.3.3 Engineering base tuning of Multi-agent state machines

Pressure-test results indicate database I/O, rather than external LLM latency, becomes the primary bottleneck for multi-agent collaboration. This bottleneck is amplified under the “1+7+8” architecture: each Orchestrator scheduling triggers 3-5 Agent-state read-write operations. Hundreds-millisecond single-operation latency accumulates into second-level delays after multiple collaboration rounds, capping system throughput under high parallelism. Hence, database-layer optimization is critical for deploying multi-agent state machines.

Three composite indexes are built for hot paths after slow-query analysis. An index on user_id-created_at cuts full-table scans of learning_behaviors from 150 000 to ~800 rows; classroom_code-role optimizes large-scale teacher queries; a three-column index accelerates examination-record filtering. As shown in Table 4, three tuning rounds yield an average 62% P95 latency reduction for core queries (265 ms → 103 ms), with dashboard latency dropping from 520 ms to 210 ms (−60%). Indexes are limited to 12 to avoid write overhead. This practice proves multi-agent concurrency relies jointly on Agent scheduling and data-persistence optimization for production-ready deployment.

Table 4. Multi-Agent Core Query Path Performance Comparison Before and After Composite Index Tuning

Query path	Associating Agents	P95/ms before tuning	P95/ms after tuning	Reduction /%	Optimization strategy
dashboard-aggregations	ProfileAgent/Orchestrator	520	210	60	Composite index
bulkteacher queries	TeacherService	180	65	64	Composite index
assesment queries	Evaluator/PathAgent	95	35	63	Composite indexes

4.4 Closed-Loop Verification of Whole Process Teaching

To validate system effectiveness, a 105-minute teaching closed-loop case is constructed. Student Xiaozhi struggles with Python object-oriented programming (practical-operation proficiency: 48%,

polymorphism-abstract-class mastery: 52%). The teacher identifies six at-risk students via the data market. After the AI generates and assigns five exercises, the student's practical-operation proficiency rises to 64%. All 56 checkpoints pass (100% success rate). Three engineering links are verified: real-time teacher-to-student task push via SSE (55 ms latency); atomic four-table transactions for answer recording, EMA portrait and knowledge-graph updates; sub-100-ms multi-table JOIN aggregation for teacher dashboard queries. Figure 8 illustrates key interactions in this scenario.

The closed-loop verification proved the technical feasibility of the teaching process of "teacher data discovery → AI-assisted decision-making → precision teaching intervention → real-time effect tracking" in the Zhicheng system from the engineering level, and provided a reusable technical framework for AI-driven precision teaching.

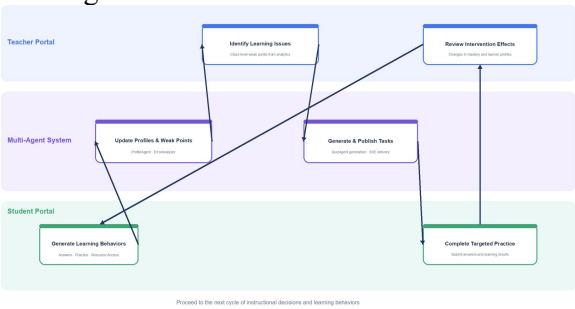

Figure 8. Personalized Teaching Closed Loop among Teachers, the Multi-agent System, and Students

5. Conclusion and Prospect

Targeting three challenges in university-oriented AI education — poor personalization, black-box teaching process and LLM hallucinations — this paper develops Zhicheng, a personalized learning-diagnosis system based on the "1+7+8" multi-agent collaborative architecture.

Three main contributions are summarized. (1) The "1+7+8" visual white-box orchestration model built on LangGraph StateGraph enables automatic task decomposition, parallel scheduling and conditional routing. Benefiting from the dual-brain model, it surpasses single-LLM baselines in task-decomposition accuracy (92.7% vs 58.1%), format compliance (96.8% vs 71.1%) and end-to-end success rate (88.5% vs 66.1%). React-Flow visualization makes agent collaboration traceable and intervenable. (2) A four-tier anti-hallucination pipeline cuts factual error rate from 18.6% to 0.8% (a 95.7% reduction). This vendor-agnostic mechanism can be generalized to accuracy-critical LLM systems. (3) JMeter-based multi-scenario stress testing and composite-index optimization achieve 842 TPS under 1250 concurrent users with an average 62% latency reduction, proving lightweight databases are viable for medium-scale educational scenarios and laying groundwork for PostgreSQL migration.

Future work includes dialogue-level hallucination evaluation, multi-modal learning portraits with physiological data, FedAvg-based federated learning, and protocol-compliant integration with national smart-education platforms as a reusable AI-education middleware.

References

[1] Unesco. Guidance for generative AI in education and research. Paris: UNESCO, 2023.

[2] Kasneci E, Sessler K, Kuchemann S, et al. ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 2023, 103: 102274.

[3] Ji Z, Lee N, Frieske R, et al. Survey of hallucination in natural language generation. ACM Computing Surveys, 2023, 55(12): 1-38.

[4] Huang L, Yu W, Ma W, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. arXiv preprint arXiv:2311.05232, 2023.

[5] Fastapi Contributors. FastAPI framework documentation. [2026-08-06]. https://fastapi.tiangolo.com/.

[6] LangGraph Contributors. LangGraph: Building stateful, multi-actor applications with LLMs. (2024-01-15). https://github.com/langchain-ai/langgraph.

[7] Sqlite Consortium. Write-ahead logging. [2026-08-06]. https://www.sqlite.org/wal.html.

[8] Hong S, Zhuge M, Chen J, et al. MetaGPT: Meta programming for a multi-agent collaborative framework//The Twelfth International Conference on Learning Representations. 2024.

[9] React Flow Contributors. React Flow: A customizable React component for building node-based editors and interactive diagrams.

[2026-08-06]. https://reactflow.dev/.
[10] Chroma Contributors. Chroma documentation. [2026-08-06]. https://docs.trychroma.com/.
[11] Asai A, Wu Z, Wang Y, Et al. Self-RAG: Learning to retrieve, generate, and critique through self-reflection//The Twelfth International Conference on Learning Representations. 2024.
[12] Deepseek-Ai. Deepseek-V3 technical report. arXiv:2412.19437, 2024[2026-08-06]. https://arxiv.org/abs/2412.19437.
[13] Zheng L, Chiang W L, Sheng Y, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena//Advances in Neural Information Processing Systems. 2023, 36: 46595-46623.
[14] Li J, Cheng X, Zhao X, et al. HaluEval: A large-scale hallucination evaluation benchmark for large language models//Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Singapore: Association for Computational Linguistics, 2023: 6449-6464.
[15] Apache Software Foundation. Apache JMeter user manual. [2026-08-06]. https://jmeter.apache.org/usermanual/.

