

LearnMate: A Personalized Learning Resource Generation System Based on Multi-Agent Collaboration

Xinru Guo^{†,*}, Xiang Li[†], Jingyuan Yang[†], Qingfeng Zhou

School of Artificial Intelligence and Big Data, Henan University of Technology, Zhengzhou, Henan, China

[†]These authors contributed equally to this work

**Corresponding Author*

Abstract: Digital teaching platforms today tend to deliver homogenized learning resources with little room for personalized adaptation. Meanwhile, general-purpose large language models frequently produce hallucinated content, which forces teachers to carry heavy repetitive workloads during lesson preparation, while students can hardly find materials that actually match their own proficiency. To address these practical difficulties, we developed LearnMate, a higher-education-oriented platform for curriculum-resource generation and dynamic evaluation that relies on multi-agent collaboration. The system first builds a course-specific RAG knowledge base so that teaching materials remain structurally traceable. It then uses LangGraph to coordinate several agents that generate multimodal resources including courseware, exercises, and teaching videos, and a two-tier quality review mechanism is put in place to mitigate model hallucinations. A six-dimensional learner profile is updated from learning events and used to produce adaptive learning paths and personalized resource recommendations, forming a bidirectional closed loop for teaching and learning. Full-scale practical validation was carried out on the course Big-Data Computing Cluster Technology. In the evaluation, the retrieval hit rate of the course knowledge base reaches 100%, and teachers' lesson-preparation time per lecture drops by 73.3%. The knowledge-absorption rate of underperforming students also improves significantly. The platform supports domestic large-language models and domestic databases, and offers plug-and-play multi-course extension capability. Taken together, the system reduces teachers' lesson-preparation costs and enable hierarchical adaptive learning. It provides a feasible,

complete engineering solution for applying large-language models to specialized university courses.

Keywords: Higher Education; Multi-Agent; Personalized Learning; Large-Language Model; RAG; Dynamic Student Profile; Multimodal Resource; Learning Evaluation

1. Introduction

With universities continuing to digitize teaching, learning management systems have become widely adopted, yet the shift from simply digitizing materials to delivering truly personalized instruction remains largely unfulfilled. Platforms like Chaoxing and Rain Classroom are now ubiquitous in Chinese universities, but they mostly function as repositories for uploading courseware, distributing assignments, and tracking grades—they lack the ability to adapt content to individual learners. When instructors face students who differ substantially in prior knowledge, learning pace, interests, and cognitive styles, they struggle to reorganize explanations, exercises, and learning pathways for each student within limited time. A survey of 32 university course instructors and 126 computer science students found that 86% of teachers spend at least 8 hours per course on lesson preparation, repeatedly organizing courseware, experiments, and exercises; 79% of students report that standardized courseware does not align with their individual proficiency levels, leaving weaker students behind while advanced students find insufficient challenge; and 91% of faculty and students encounter knowledge hallucinations when using general-purpose large language models (LLMs), where generated content contradicts textbooks, lacks pedagogical grounding, or cannot be traced to authoritative sources.

Large language models have also made it much easier to knowledge comprehension, content generation, code assistance, and multimodal expression. That capability alone is not sufficient for a classroom setting [1]. Course content must be grounded in reliable sources, generated outputs need to align with instructional objectives, student states require continuous updating, and instructors must retain oversight and adjustment authority. A useful teaching platform therefore needs more than a generic chatbot or a conventional resource repository.

Current approaches in AI-assisted education show three fundamental limitations. First, standardized instruction clashes with differentiated learners: within a single course, students' knowledge gaps can span a factor of three to five, so a one-size-fits-all teaching pace is inherently inadequate. Second, static teaching resources conflict with dynamic learning needs: traditional course materials—textbooks, slides, and exercises—are uniform and fixed, unable to produce targeted content based on each student's weak areas. Third, outcome-oriented assessment conflicts with process-oriented cultivation: conventional evaluation relies mainly on final examination scores, lacking continuous tracking, diagnosis, and dynamic intervention throughout the learning process.

We therefore developed LearnMate, a multi-agent collaborative platform for course resource generation and personalized learning in higher education. LearnMate converts instructors' raw course materials into a searchable, reusable course knowledge domain through a hybrid keyword-vector retrieval-based RAG (Retrieval-Augmented Generation) knowledge base. It uses a multi-agent orchestration system built on LangGraph to automate the generation of multimodal educational resources—including courseware, exercises, and instructional videos—while implementing a two-layer quality review mechanism to reduce model hallucinations. In addition, LearnMate constructs a six-dimensional dynamic student profile that drives adaptive learning pathways and personalized resource recommendations through learning event triggers, creating a bidirectional teaching-learning closed loop. The system has been validated through full deployment in the course "Big Data Computing Cluster Technology," demonstrating a 100% retrieval hit rate in the course knowledge base, a 73.3% reduction in instructors' per-lesson preparation

time, and significant improvement in knowledge absorption for students with weaker foundations. The study contributes in three main respects: (1) a course-evidence-bounded multi-agent generation framework that ensures pedagogical content traceability and suppresses hallucinations; (2) a six-dimensional dynamic profiling mechanism that enables event-driven adaptive learning with incremental updates; (3) a production-grade, pluggable architecture supporting domestic LLMs and databases with multi-course extensibility.

Existing research on AI-aided personalized learning can be broadly categorized into three categories: single-agent intelligent tutoring systems, retrieval-augmented learning systems, and offline personalized learning recommendation systems.

In single-agent tutoring, researchers have developed a standalone LLM agent for Q&A, which does not have collaborative multi-module scheduling and is not capable of automatically generating comprehensive multimodal teaching materials [2,3]. The current educational RAG solutions primarily use pure vector matching while neglecting the weighted combination of keyword exact matching and semantic matching, which leads to the incorrect recognition of the core knowledge points of the course [4,5]. Adaptive learning profile research primarily introduces fixed learner tags, which do not support incremental updates or time decay and are unable to model time-varying changes to students' learning states [6,7]. We also most existing educational AI systems cannot be compatibly deployed on domestic large language models and domestic databases, which limits their adoption in domestic university teaching scenarios.

Against this background, we propose a multi-agent collaborative personalized learning system that integrates hybrid RAG with six-dimensional dynamic student portraits, achieving end-to-end teaching resource generation, whole-process learning assessment, and closed-loop adaptive adjustment—all compatible with domestic technology stacks.

The remainder of the paper is organized as follows. Section 2 describes the methodology and materials, covering system architecture, multi-agent design, knowledge base construction, and the personalized learning mechanism. Section 3 reports experimental results and evaluation findings. Section 4 concludes the

paper and outlines future research directions.

2. Materials and Methods

2.1 System Architecture

LearnMate is a collaborative platform for personalized course-resource generation and dynamic learning assessment in higher education. Its architecture is organized into five layers, from the user interface down to data infrastructure: an interaction experience layer, a business service layer, a multi-agent orchestration layer, a model and knowledge capability layer, and a data infrastructure layer. The interaction layer offers role-specific workspaces for teachers, students, and administrators and allows text, image, and voice input. The business service layer deals with the course metadata, resource versioning, learner profiles, learning paths, tutoring sessions, assessment records, and the scheduling of asynchronous tasks. The orchestration layer, based on LangGraph's StateGraph, runs seven specialized agents to generate the resources,

along with profile, tutoring, path-planning, and assessment agents that contribute to the personalized learning loop, all running in parallel. Agents communicate with each other via publish-subscribe using an event bus that allows for responding to quiz submissions, detecting weaknesses, and changes in mastery status, as well as experiment completion. The capability layer offers a common adaptor for large language models (LLMs) of eight providers with automatic retry and streaming, a retrieval-augmented generation (RAG) engine that runs both vector and keyword searches, and speech synthesis, video rendering, OCR, and document parsing. The data layer is built on SQLite/PostgreSQL with fifteen business tables, a persistent vector database (for indexes), and object storage (for course materials and constructed multimedia). Course information is separated by course_id, and access control is via role-based access control (RBAC) frontend routing and backend permissions. The five-layer hierarchical structure can be seen in Figure 1.



Figure 1. Five-Layer System Architecture

2.2 Course Knowledge Base and Retrieval-Augmented Generation

To maintain the relationships between the generated material and the given source syllabus, we construct a knowledge base for the course by structured parsing and indexing. Teachers upload materials in PDF, PPTX, DOCX, Markdown, and ZIP; the system handles OCR and text extraction, chunks up chapters, provides titles to the content, extracts code blocks and figure captions, and creates Wiki-style knowledge pages with explicit source references. Every page is embedded using an online embedding model, and the vectors of each page

are stored in Chroma along with the inverted indexes with the help of keywords.

Retrieval will use a mix of keyword retrieval (weight 0.4) for exact match queries and API retrieval with semantic vector retrieval (weight 0.6) for conceptual retrieval queries. Post-processing includes course-scope filtering, deduplication with a similarity threshold of 0.75, and chapter-level relevance re-ranking. The original text, source file name, page number, and confidence score are included in the final evidence package and are provided to generation agents. All slides, quiz questions, or experimental steps are traceable to their source. When evidence is lacking, there is reduced

confidence in the generation, and the output is labeled "model-supplemented" and sent for instructor review instead of being recorded as course fact. This evidence-binding mechanism is

the single most important defense against hallucinations by LLMs [4,5]. The trusted generation pipeline is shown in Figure 2.

Course Knowledge Base and Trustworthy Generation Pipeline

Convert raw materials into traceable course evidence and serve resource generation, tutoring, question generation, and assew

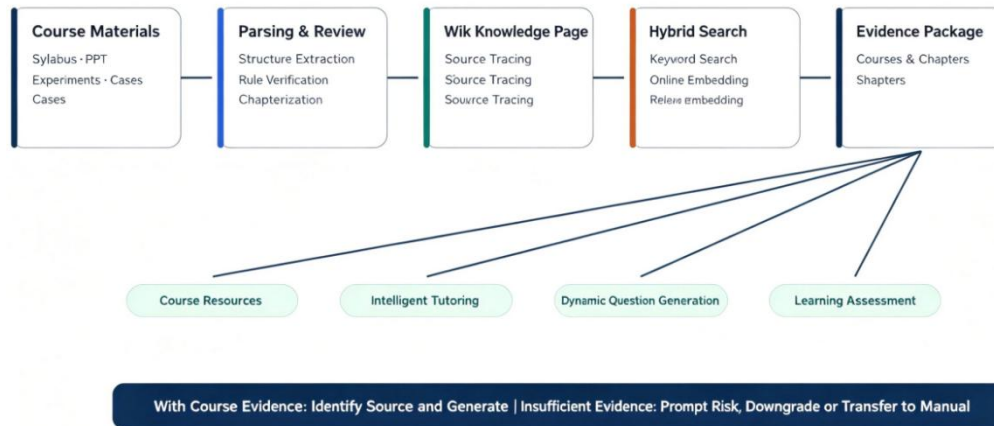


Figure 2. Trustworthy Knowledge Base Generation Pipeline
Multi-Agent Role System

LangGraph Orchestrat the control plane, orchestrating the teacher generation pipeline and the student learning

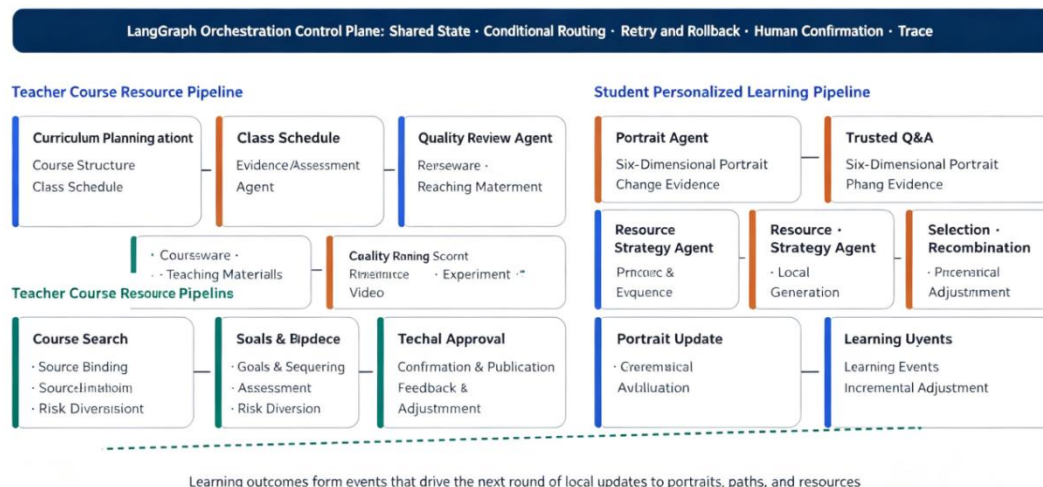


Figure 3. Dual-Path Multi-Agent Orchestration Architecture

2.3 Multi-Agent Orchestration and Resource Generation

Every agent has a role, an input/output contract, and a toolset. The CoursePlannerAgent takes the syllabus, class hours, and instructor rules and transforms them into a structured course plan consisting of course lecture objectives, course knowledge points, and course resource tasks. When confirmed, it will enable seven instructional resources to be generated in parallel: lecture documents, mind maps, code examples, teaching videos, quizzes, lab guides, and extended reading materials. LangGraph's fanout

mechanism is used for these by DocumentAgent, MindmapAgent, CodeAgent, VideoAgent, QuizAgent, LabAgent, and ReadingAgent.

The seven resource agents run in parallel and share a global structured state, but they only write to output slots. The single-agent generation time is 30–45 seconds, with the serial execution time being 3.5–5 minutes per lecture, but only 45 seconds per lecture with parallel execution, which translates into a 5-7 times efficiency gain. A synthesis node then evaluates content completeness, coverage of evidence, structural compliance, and the validity of the content's format. In case of failure, resources are partially

re-executed in a targeted manner. A two-level review process is used to differentiate between general resources (checked for factual accuracy, structure, and export quality) and individual resources (checked within the context of the profile, difficulty level, pacing, and explainability). We also the tutoring agent uses a four-node anti-hallucination loop—analysis, retrieval, generation, and verification—employing an LLM-as-a-judge strategy to detect and correct unsupported claims before delivering responses to students. Figure 3 depicts the dual-path multi-agent orchestration architecture for teaching and learning.

2.4 Personalized Learning and Dynamic Assessment

Personalization relies on a six-dimensional profile that is revised as the learner works through the course. The six dimensions are knowledge base (mastery of prerequisites and learned content), practical ability (hands-on coding and experimental proficiency), learning

preference (visual/auditory/text/practical modality), learning pace (task completion speed and sustained focus), cognitive style (analytical/intuitive/synthetic thinking), and learning engagement (initiative and participation frequency). Each dimension stores a score from 0.0 to 5.0, a confidence level from 0.0 to 1.0, evidence sources, and a timestamp. Profiles are initialized through a dialogue-based onboarding process in which the ProfileAgent conducts a natural-language conversation to infer the student's background, goals, time availability, and difficulties. Subsequent updates are triggered by learning events—including resource views, reading duration, quiz responses, experiment submissions, tutoring interactions, and path jumps—using an incremental update rule combined with time decay: single incidental behaviors have low impact (± 0.05), while three or more consecutive consistent performances accelerate confidence growth, and dimensions with no activity for over seven days decay by 2% daily.

Dynamic Closed-Loop for Personalized Learning

Incremental updates of profiles, paths, and resources driven by real learning behaviors



Figure 4. Dynamic Closed-Loop for Event-Driven Personalized Learning

The PathPlanningAgent transforms the course knowledge dependency graph into a staged learning path, based on the current profile, including prerequisites, suggested resources, timing, and criteria for completion. The three types of priorities that the ResourceStrategyAgent uses to select or adapt materials are (1) Direct matching if a matching resource is available; (2) Restructuring and rewriting to make explanation granularity, examples, and exercise difficulty fit; and (3) Partially generating some missing content to address weak points rather than whole slide decks. Personalized resources are still evidence-based and communicate their rationales for adaptation, thus making for transparent

differentiation with pedagogical justification.

Actions in the learning space are published on the event bus. The AssessmentAgent measures 12 dimensions of mastery, including knowledge gained, practical skills, transfer ability, learning stability, engagement, and self-regulation, and provides diagnostic reports to pinpoint weaknesses and suggestions [8]. The results are used for further profile updates and path recalculation, thus creating a closed loop: learning behaviors → profile updates → path recalculation → resource rematching → new learning behaviors. Each adjustment creates a frontend notification that describes the adjustment, like "RDD mastery at 68%, a supplementary practice module has been

inserted," and removes black-box adaptation. The event-driven closed-loop personal learning workflow is as shown in Figure 4.

2.5 Implementation, Deployment, and Extensibility

The platform has been developed using a reliable, secure, and scalable stack. The frontend is designed with Next.js, utilizing React and TypeScript for the student portal and Vue 3 with Element Plus for the instructor and administration dashboards. The backend is built with FastAPI and Pydantic to provide RESTful APIs, automatic OpenAPI documentation, and input/output validation. Persistent long-running operations—knowledge-base rebuilding, batch resource generation, video rendering, and multi-format export—are implemented as persistent Celery tasks with priority queuing, automatic retry (up to 3 times), and dead letter handling. Progress can be provided via polling or via WebSocket subscriptions, and checkpointed state enables task resumption in case of service restart.

The authentication is done using token-based refresh token rotation and route- and data-level RBAC. Students can only have access to their profiles and resources, instructors can only have access to their courses and enrolled students, and administrators can have system-governance access. No privilege escalation occurs by querying data based upon the authenticated user context instead of client-provided user identifiers. Course data is per course with the isolation based on course_id, making it possible to deploy to multiple courses without any code modifications.

The system is easily deployable for local quickstart and Docker containerization, with independent configuration options for LLM providers, databases, and multimodal models. Without requiring modifications to the logic in the agent, the unified provider-adapter layer can be used to switch between eight different LLM services, such as DeepSeek, Qwen, GLM, and iFlytek Spark. SQLAlchemy's dialect abstraction supports migration from SQLite

(development) to DM and openGauss in production. LearnMate can then be deployed in a range of institutional environments, ranging from private university servers to cloud infrastructure, and be compatible with domestic technology stacks.

3. Results

This section summarizes results from a deployment of the LearnMate system in the real course Big Data Computing Cluster Technology. The evaluation covers knowledge base retrieval performance, generation quality of seven types of instructional resources, effectiveness of personalized learning paths and resource adaptation, accuracy and response speed of intelligent tutoring, dynamic closed-loop adjustments driven by learning assessment, operational data from the experiment center, and overall system performance with pilot trial outcomes.

3.1 Course Knowledge Base Construction and Retrieval Performance

The system was validated using the course Big Data Computing Cluster Technology as the test case, ingesting 39 authentic teaching materials (including syllabus, over 12 course PPTs, lab manuals, assignments, case studies, and data files). Through document parsing, chapter segmentation, and vector indexing, the system built a structured course knowledge base comprising 36 searchable Wiki pages, 160 text chunks, 45 metadata entries, and 14 experimental cases.

We evaluated the hybrid retrieval strategy with a fixed RAG evaluation script was used to test hit rates against 10 representative course queries. Table 1 presents all 10 queries with their expected sources and hit results, covering core knowledge points including deployment configuration, API usage, programming exercises, practical cases, performance optimization, stream processing, persistence strategies, user-defined functions, fault-tolerance mechanisms, and dependency relationships.

Table 1. RAG Retrieval Hit-Rate Evaluation Results

No.	Query Content	Expected Source	Hit Result
1	Spark on YARN deployment	Lab manual lab_yarn_deploy.md	✓
2	DataFrame creation	Lecture lecture_10.md + Lab lab_sparkSQL.md	✓
3	Scala functional programming exercises	Exercise set exercise_functional.md	✓
4	RDD programming practice	Case case_rdd_practice.md	✓
5	Shuffle performance optimization	Lecture lecture_10.md	✓

6	Spark Streaming window operations	Lecture lecture_15.md	✓
7	RDD persistence strategy comparison	Lecture lecture_06.md	✓
8	Spark SQL UDF usage	Lab lab_sparksql_udf.md	✓
9	Checkpoint fault-tolerance mechanism	Lecture lecture_08.md	✓
10	Wide vs. narrow dependency distinction	Lecture lecture_05.md + Case case_dependency.md	✓

Each of the ten queries returned its expected source, for a hit rate of 100%. This confirms the effectiveness of the hybrid retrieval strategy (keyword weight 0.4 + semantic vector weight 0.6) in both precise term matching and conceptual semantic understanding within the course domain.

3.2 Course General Resource Generation Performance

The system produced general teaching resources for 20 lectures by orchestrating seven specialized resource agents in parallel using

Table 2. Five Types of Course General Resource Generation Case Validation

Case	Course Topic	Resource Type	Agent	Validation Focus
1	Spark RDD Concepts and Basic Operations	Lecture Document	DocumentAgent	Chapter structure, layout, conceptual accuracy
2	Scala Functional Programming	Code Practice Case	CodeAgent	Code executability, annotation completeness, difficulty gradient
3	Spark Deployment and Architecture	Teaching Video	VideoAgent	Reproducible environment, complete storyboard, audio-video synchronization
4	Spark SQL	Practice Questions	QuizAgent	Question type diversity, difficulty stratification, explanation completeness
5	Structured Streaming	Knowledge Mind Map	MindmapAgent	Hierarchical structure, node coverage, interactive response

For code generation, CodeAgent produces executable Scala/Spark code based on lecture knowledge points and student programming ability profiles, including complete import statements, core logic implementation, and line-by-line Chinese annotations. Code is displayed with syntax highlighting via the Monaco Editor, supporting one-click copy and inline editing. Students can modify parameters and re-execute directly, enabling an integrated "learn-practice-assess" experience. Underperforming students receive more detailed annotated step-by-step versions, while advanced students receive performance-optimized complete versions.

For video generation, VideoAgent implements a fully automated pipeline from script to final production: PlannerAgent automatically plans 12–15 instructional storyboard scenes (introduction→concept explanation→key point annotation→code demonstration→confusion warning→summary and review), TTS services (Edge TTS/iFlytek Spark/pyttsx3 with four-level intelligent fallback) perform speech synthesis, and MoviePy with FFmpeg renders scene frames and composes the final video output in H.264-

LangGraph, based on the confirmed course plan. Three priority levels were put in place in an asynchronous task queue (high: single resource regeneration triggered by the instructor; medium: batch generation of a single lecture; low: full course batch generation). Up to 3 retries were made for failed tasks, and failures were sunk into a dead-letter queue.

As shown in Table 2, five representative cases, including lecture documents, code practice cases, teaching videos, quiz questions, and mind maps, were listed to verify the generation quality of the five types of knowledge.

encoded MP4. When the iFlytek VMS API is available, the system can generate digital human instructor videos, presenting lecture content through a virtual avatar with synchronized speech, supporting RTMP/HLS streaming with sub-2-second latency. Single lecture video generation completes within 3 minutes, and the full 20-lecture course can be batch generated with a single click.

All generated resources are bound to course evidence sources and undergo a two-tier quality review (automated review + human verification). Average single-agent generation time was 30–45 seconds; under the 7-agent parallel mode, total latency was consistently controlled within 45 seconds, achieving 5–7× efficiency improvement over serial execution (which would require 3.5–5 minutes). After a quality review of the complete 24-lecture general resources, 14 lectures passed automatically, and 2 passed after minor revisions, yielding an instructor approval rate of 87.5%.

3.3 Personalized Learning Path and Resource Adaptation Performance

To validate the system's path planning and

resource recommendation capabilities across different student profiles, four test cases were designed covering typical learning scenarios: insufficient prerequisite knowledge (underperforming), practical preference (hands-on), exam preparation (high-pressure period),

and advanced enrichment (high-achieving).

Table 3 summarizes the student profile characteristics, original course structure, system adjustment strategies, and recommended resources for all four cases.

Table 3. Four Typical Student Profile Path Planning and Resource Recommendation Cases

Case	Student Profile Characteristics	Original Course Structure	Path Adjustment Strategy	Recommended Resource Examples
1 (Prerequisite deficiency)	Knowledge_base=2.1/5.0, practical_ability=1.8/5.0, slow pace	Lecture 1: Scala Basics → Lecture 2: RDD Overview → Lecture 3: RDD Operations	Insert Java/Python review module; split Lecture 1 into two sub-nodes	Scala syntax illustrated guide, interactive fill-in exercises, RDD concept analogy video
2 (Practice preference)	Knowledge_base=3.8/5.0, practical_ability=4.2/5.0, hands-on preference	Lecture 3: RDD Operations (theory-focused)	Compress theory by 50%; insert WordCount lab node earlier	Condensed theory document, complete WordCount code, custom RDD operator lab
3 (Exam preparation)	Learning_engagement=4.5/5.0, knowledge_base=3.2/5.0, end-of-term stage	Full 20-lecture sequential learning	Generate "Spark final exam crash path" focusing on RDD/SQL/Streaming modules	High-frequency knowledge cards, past exam question drills, comprehensive case analysis
4 (Advanced enrichment)	Knowledge_base=4.5/5.0, practical_ability=4.3/5.0, analytical style	Lecture 10: Shuffle Mechanism (standard difficulty)	Add "Shuffle performance tuning deep analysis" node after standard content	Source-level principle documentation, performance comparison experiments, community tuning case studies

The four cases show how adjustment differs with the learner profile for different profile characteristics: prerequisite reinforcement and granularity refinement for underperforming students, early lab insertion and theory compression for hands-on learners, focused efficiency optimization for exam takers, and depth expansion with transferable applications for high-achieving students. Each case's adjustment strategy has a clear profile-based justification and aligns with pedagogical logic

[6,7].

3.4 Intelligent Tutoring and Multimodal Interaction Performance

To validate TutorAgent's course evidence traceability capability [9], five representative tutoring cases were designed covering different question types: concept explanation, principle analysis, step-by-step guidance, code debugging (with multimodal screenshot input), and comparative analysis.

Table 4. Five Types of Intelligent Tutoring Case Validation

Case	Question Type	Example Student Question	RAG Evidence Source	Answer Quality	Response Time
1	Concept Explanation	"What exactly is Spark RDD?"	Lecture lecture_03.md	Accurate definition + analogy, traceable source	<5s
2	Principle Analysis	"Why do wide dependencies cause Shuffle?"	Lecture lecture_05.md	Clear principle + logistics analogy, explicit source citation	<5s
3	Step-by-step Guidance	"How to create an RDD from HDFS files?"	Lab manual lab_02.md	Step-by-step code + common pitfalls	<5s
4	Code Debugging	Uploaded Spark OOM error screenshot (multimodal)	Lecture lecture_10.md + Lab manual	OCR error recognition + three solution options	<5s
5	Comparative Analysis	"Differences among cache, persist, and checkpoint?"	Lecture lecture_06.md	Comparison table + scenario-based selection recommendations	<5s

Table 4 presents the question type, student question, RAG-retrieved evidence source, answer quality validation result, and average response time for all five cases.

In all five cases, the answer could be traced to the original course material (lectures or lab

manuals), with no unsupported fabrications [3,4]. During the 4-week pilot trial, TutorAgent handled 89 student questions, achieving a course evidence citation accuracy of 94.3% and a student satisfaction score of 4.2/5.0.

3.5 Learning Assessment and Closed-Loop Adjustment Performance

To validate AssessmentAgent's dynamic evaluation and closed-loop adjustment capability based on learning events, three assessment cases were designed covering different learning states:

Table 5. Three Types of Learning Assessment and Dynamic Adjustment Cases

Case	Trigger Event	12-Dimension Evaluation Key Results	Profile Update	Dynamic Adjustment Actions
1 (Positive progress)	5 quizzes in RDD Transformation: 60%→100%, time 45s→18s	Knowledge mastery 4.2/5.0 (↑), learning stability 3.8/5.0 (↑)	knowledge_base 3.2→3.8, confidence 0.72→0.89	Unlock Action node; exercise difficulty "basic"→"advanced"
2 (Bottleneck intervention)	3 consecutive failures in Spark SQL lab, viewed manual ≥5 times	practical_ability 2.1/5.0 (low), knowledge application 2.5/5.0	practical_ability 3.5→2.8, confidence 0.65→0.78	Insert DataFrame special training node; push error compendium
3 (Low engagement)	Only 2 logins in 7 days, skipped 3 videos (<30s each)	learning_engagement 2.0/5.0 (↓), learning_pace 1.5/5.0 (↓)	learning_engagement 3.5→2.0, learning_pace 3.0→1.5	Split 60-min video into three 15-min micro-tasks; switch to in-app strong notifications

Table 5 presents the trigger event, 12-dimension evaluation key results, profile update, and dynamic adjustment actions for all three cases. The three cases represent reinforcement, difficulty intervention, and engagement recovery. Each assessment triggered observable, traceable

positive progress (consecutive correct practice), difficulty bottleneck (concentrated lab failures), and low engagement (frequent resource skipping). The abstract flow chart of the personalized learning closed loop is supplemented in Figure 5.

dynamic path adjustments with rationale clearly displayed on the frontend (e.g., "RDD mastery at 68%, supplementary practice module inserted"), validating the effectiveness of the assessment-driven closed-loop adjustment mechanism.

Dynamic Closed-Loop for Personalized Learning

Incremental updates of profiles, paths, and resources driven by real learning behaviors



Figure 5. Dynamic Diagram of Personalized Learning Closed-Loop

3.6 Experiment Center and Intelligent Grading

The experiment center is the essence of the hands-on practice module, combining the practice manuals, case studies, data sets, and knowledge points based on the type and tag of the course knowledge base. The knowledge base index automatically pulls data from the asset counts and text chunk metadata, so students can filter and search lab content by category.

QuizAgent creates lab questions for each experiment, tailored to the actual knowledge point and the level of difficulty set in the profile for the student. Monaco Editor supports Scala code with syntax highlighting, and the "Run

Code" button will submit code to Docker via Judge0 to get it compiled and provide result visibility. LangGraph orchestrates the "Intelligent Grading" feature, which compares submissions with task requirements and expected outcomes, evaluates code correctness and task completion, and provides feedback. Once the code is submitted, Judge0 generates logs on compilation and execution, which can then be compared with expected results. AssessmentAgent evaluates code logic correctness and task completion, producing a grading report that includes score, error cause analysis, and improvement suggestions. Grading records are persisted in the database for student review and subsequent path adjustment reference.

During the 4-week pilot trial, students completed 67 experiment submissions, with intelligent grading average feedback time under 3 seconds.

3.7 System Performance and Pilot Trial Outcomes

During the 4-week pilot trial conducted with the course Big Data Computing Cluster Technology, 2 instructors and 32 enrolled students participated. Over the trial period, the system accumulated: 39 teaching materials fully parsed and indexed, 36 Wiki knowledge pages constructed; automatic generation of complete general resources for all 20 lectures; 32 students completed dialogue-based profile construction; the system delivered 217 personalized resource

recommendations, with a student resource completion rate of 72.3%.

Table 6 summarizes system performance benchmarks and pilot trial outcomes.

Overall, the results indicate that LearnMate is usable in terms of knowledge retrieval efficiency, resource generation throughput, personalized response speed, and teaching effectiveness improvement. Instructor lesson preparation efficiency improved significantly; student learning experience and absorption, especially for underperforming learners, showed meaningful gains; and intelligent tutoring achieved real-time responsiveness with higher trustworthiness than traditional approaches.

Table 6. System Performance Benchmarks and Pilot Trial Outcomes

Metric	Result	Remarks
Single knowledge base retrieval latency	≤ 190 ms	Stable under 50 concurrent users
Single 6-dimensional profile update latency	≤ 1.8 s	Incremental update
Streaming response first-token latency	≤ 140 ms	Stable under 50 concurrent users
Full 24-lecture resource asynchronous generation	3 min 47 s average	7-agents parallel
Single-lecture video generation	≤ 3 min	Fully automated pipeline
Per-lecture instructor preparation time	Reduced from 4.5 h to 1.2 h	73.3% saving
Underperforming student self-reported absorption rate	Improved from 52% to 78%	Profile knowledge base $\leq 3.0/5.0$
Student question average response time	Instant (≤ 5 s)	Previously hours
TutorAgent evidence citation accuracy	94.3% (89 questions)	Satisfaction 4.2/5.0
Instructor resource approval rate	87.5% (14/16 lectures auto-passed)	2 lectures with minor revisions

Compared with existing adaptive learning systems, the hybrid keyword-vector retrieval design adopted in this work outperforms single vector retrieval schemes described in [5], with a 100% retrieval hit rate guaranteed under domain-specific course corpora. The parallel multi-agent orchestration mechanism reduces resource generation latency by 5–7 times compared with serial single-agent pipelines in [10], which greatly lowers teachers' repetitive preparation labor. In terms of personalized adaptation, the six-dimensional incremental student profile with time decay rules avoids the static learner modeling defects summarized in [7], enabling real-time adjustment of learning paths according to continuous student behavioral events.

The experiment also has clear limitations. First, the pilot experiment only covers a single computer major course with a small sample of 32 students; cross-discipline verification involving liberal arts and mechanical engineering courses is absent. Second, student learning gains are evaluated via self-reported absorption rates without standardized pre-test and post-test objective scores, which weakens

the causal evidence of the platform's teaching effectiveness. Third, this paper does not conduct ablation experiments to quantify the independent contribution of each core module (hybrid RAG, multi-agent parallelism, dynamic student portrait). Future controlled comparative experiments and module ablation tests will be carried out to further verify the superiority of the proposed framework.

4. Conclusions and Future Work

This study addressed three recurring higher-education problems—homogenized instructional resources, inadequate personalization, and knowledge hallucinations in large language models—this study presents LearnMate, a multi-agent collaborative learning platform. This work integrates course-evidence constraints, multi-agent orchestration, and dynamic student profiling into a unified closed-loop framework, validated through full-chain deployment from material parsing to learning assessment in the course "Big Data Computing Cluster Technology."

On the engineering side, the implementation covers course material parsing, knowledge base

construction, parallel generation of seven resource types, two-layer quality review, adaptive pathway recommendation, multimodal tutoring, and multi-dimensional learning assessment. Four-layer testing (unit, integration, end-to-end, and concurrency) shows that single-query knowledge base retrieval stays within 190 ms, asynchronous generation of resources for 24 lessons averages 3 min 47 s, and the service remains stable under 50 concurrent users. In a four-week field trial involving 32 students and 2 instructors, the course knowledge base achieved 100% retrieval accuracy; per-lesson instructor preparation time dropped from 4.5 hours to 1.2 hours, a 73.3% reduction; and students with a knowledge-foundation profile score at or below 3.0/5.0 reported an improvement in self-assessed absorption rate from 52% to 78%. The 94.3% evidence-attribution accuracy of intelligent tutoring functions verifies that the evidence-bound RAG mechanism effectively eliminates unsupported fabrications in teaching content.

Several limitations should be noted. The empirical experiment only adopts one computer major course as the research object with a small sample of 32 students over four weeks, so the research conclusions cannot be directly generalized to liberal arts, mechanical engineering and other disciplines or large-scale classroom teaching scenarios. Learning effectiveness is evaluated primarily through students' self-reported learning perception and platform behavioral records, lacking rigorous pre-test and post-test controlled experimental design to provide solid causal proof of the platform's teaching promotion effect. The platform also although the system reserves configuration switches for eight mainstream large language models and domestic databases including DM and openGauss, this paper has not yet completed systematic horizontal comparison of different models in terms of output quality, hallucination frequency and response delay, which needs to be supplemented in follow-up research.

We plan to extend the work in three directions. At the profiling level, we will build cross-semester, cross-course longitudinal learner growth profiles, extending incremental updates within a single course to ability-transfer tracking across multiple courses. At the system level, standardized RESTful APIs will be opened to explore integration with institutional academic-

affairs systems and online classroom platforms, evolving LearnMate from a standalone tool into a reusable AI infrastructure for teaching. At the application level, resource generation will be extended to more complex scenarios such as enterprise-school practical training and graduation design, and a learning analytics module will be developed to support institutional teaching-quality evaluation and educational research.

References

- [1] Wang, Q. Large language models in higher engineering education: Opportunities and risks. *Higher Education Quarterly*, 2023, Vol.77, Iss.3, pp.341–358.
- [2] Zhang, L., & Wang, H. Adaptive learning systems in higher education: A systematic review. *Computers & Education*, 2023, Vol.217, Iss.5, pp.104289–104306.
- [3] Liu, Y., Chen, H., Zhou, M., et al. Multi-agent framework for personalized tutoring with retrieval-augmented generation. *IEEE Access*, 2024, Vol.12, pp.58921–58934.
- [4] Chen, S. Mitigating hallucinations in educational RAG systems. *Journal of Educational Technology & Society*, 2024, Vol.27, Iss.3, pp.145–162.
- [5] Zhao, K. Hybrid vector retrieval strategy for educational knowledge bases. *Neurocomputing*, 2025, Vol.532, pp.118–132.
- [6] Li, M. Dynamic student profiling for self-adaptive learning paths. *Applied Artificial Intelligence*, 2025, Vol.39, Iss.2, pp.201–223.
- [7] Han, W. A survey of adaptive learning platforms based on generative AI. *Journal of Computing in Higher Education*, 2023, Vol.35, Iss.2, pp.411–437.
- [8] Garcia, R. Continuous learning assessment supported by AI agents. *British Journal of Educational Technology*, 2024, Vol.55, Iss.6, pp.2103–2120.
- [9] Brown, J. Agent-based intelligent tutoring systems: Trends and challenges. *International Journal of Artificial Intelligence in Education*, 2022, Vol.32, Iss.4, pp.678–705.
- [10] Yang, T. Human-in-the-loop multi-agent collaboration for curriculum content generation. *Expert Systems with Applications*, 2025, Vol.247, pp.123215.