http://dx.doi.org/10.62762/NGCST.2026.601800

RESEARCH ARTICLE

A Course-Specific Agentic RAG Chatbot for IT Student Support: Architecture, Local Deployment, and Preliminary Evaluation at Hai Phong University

Tran Huu Van[1], Nguyen Phu Vinh[1], Vu Thi Tuyet Ngan[1], Luong Ha Phuong[1] and Dac-Nhuong Le[1,*]

[1] Faculty of Information Technology, Hai Phong University, Hai Phong 180000, Vietnam

Abstract

This paper introduces a course-specific agentic retrieval-augmented generation (RAG) chatbot developed to support Information Technology students at Hai Phong University. The proposed system addresses the limitations of static FAQ bots, which are unable to manage open-ended academic tasks, and model-only large language model (LLM) assistants, which may generate fluent yet insufficiently grounded responses. The prototype combines local LLM deployment, semantic retrieval of course materials, and constrained, tool-oriented orchestration to perform four key tasks: question answering, document summarization, study planning, and quiz generation. The primary contributions include a six-layer privacy-aware architecture, a semi-agentic workflow for task routing and source-grounded response generation, and an evaluation protocol that compares LLM-only, RAG, and RAG-plus-orchestration scenarios using a five-dimensional academic rubric. Preliminary local testing demonstrates response times ranging from approximately 2 to 45 seconds for inputs of 1,000-20,000 words, with summary outputs compressed to 10–30% of the original source length. These findings suggest that department-level, course-grounded academic assistance is achievable with modest infrastructure, provided that retrieval, logging, and bounded orchestration are prioritized as core design requirements.

Keywords: agentic AI, educational chatbot, retrieval-augmented generation, local large language model, academic support, information technology education.

1 Introduction

Artificial intelligence is transforming higher education by facilitating adaptive content delivery, intelligent tutoring, automated feedback, and advanced human-computer interaction. Recent reviews indicate that AI enhances personalization, engagement, feedback, and learning support, but also necessitates careful consideration of trust, ethics, transparency, and alignment with pedagogical objectives [1–3]. In

Submitted: 30 April 2026
Accepted: 20 May 2026
Published: 30 May 2026

Vol. 2, **No.** 2, 2026.
10.62762/NGCST.2026.601800

***Corresponding author:**
✉ Dac-Nhuong Le
nhuongld@dhhp.edu.vn

Citation
Tran, H. V., Nguyen, P. V., Vu, T. T. N., Luong, H. P., & Le, D. N. (2026). A Course-Specific Agentic RAG Chatbot for IT Student Support: Architecture, Local Deployment, and Preliminary Evaluation at Hai Phong University. *Next-Generation Computing Systems and Technologies*, 2(2), 21–34.

Information Technology programs, the demand for such support is especially pronounced, as students engage with complex technical concepts, algorithms, programming syntax, database models, and multi-step problem-solving processes.

Despite the increasing availability of general-purpose LLMs, their direct use in academic settings remains problematic. A model-only assistant may provide a plausible answer that is correct in a broad sense but still misaligned with the syllabus, terminology, level of detail, or examples used in a specific course. This risk is not only technical but pedagogical: generative AI in engineering and computer science education can support learning, but it also raises concerns about hallucination, overreliance, academic integrity, and insufficient source transparency [4, 5].

The student research report on which this article is based addresses these concerns in the context of the Faculty of Information Technology at Hai Phong University. The target courses include Python Programming, Data Structures and Algorithms, and Database Management Systems, with possible extension to systems analysis and introductory artificial intelligence. The report reframes the prototype from a simple document summarizer into a course-grounded academic assistant that can classify student intent, retrieve relevant materials, call task-specific tools, and return structured learning support.

This manuscript extends the original report into a journal-style system paper. It emphasizes the scientific rationale underlying the architecture, the design decisions required for controllable agentic behavior, and the development of a robust evaluation framework. Rather than introducing a new learning algorithm, the contribution lies in integrating local LLM deployment, retrieval-augmented generation (RAG), metadata-aware knowledge management, and bounded orchestration into a practical model for student academic support.

This paper offers three primary contributions. First, it introduces a six-layer architecture that delineates interface, orchestration, tools, retrieval, model inference, and logging. Second, it establishes a semi-agentic workflow for four academic tasks: question answering, summarization, study planning, and quiz generation. Third, it presents preliminary observations from the prototype and outlines an evaluation protocol that integrates content quality, operational performance, and user-centered assessment. Collectively, these contributions position the prototype as a controllable academic support system rather than a generic conversational application.

2 Related Work

2.1 Educational Chatbots and LLM-based Learning Support

Educational chatbots have progressed from rule-based and menu-driven systems to data-driven conversational agents. While early systems provided administrative guidance and scripted tutoring, they were constrained by manually curated question-answer pairs and demonstrated limited capacity to address open-ended disciplinary questions. Recent studies of artificial intelligence in education emphasize that effective chatbots should be evaluated not only on conversational fluency but also on their alignment with learning objectives and institutional contexts [1, 3, 6].

Reviews of artificial intelligence in education highlight that system quality should not be assessed solely based on language fluency or user satisfaction. Educational AI must also demonstrate alignment with learning objectives, transparency, trustworthiness, and include safeguards against overreliance. In computer science education, these considerations are particularly critical because minor conceptual errors in algorithmic complexity, program behavior, or database constraints can mislead students during practice [4, 5, 7].

2.2 Retrieval-Augmented Generation for Source-Grounded Assistance

RAG combines a generative model with an external knowledge source. Instead of relying solely on information implicitly stored in model parameters, the system retrieves relevant passages and conditions its generation on them. This mechanism is well-suited to academic support because course materials are bound, curated, and frequently updated; it also supports provenance and verification, which are central requirements for educational RAG systems [8, 9].

In the proposed system, RAG is not treated as a single plug-in but as a knowledge engineering pipeline. The pipeline includes document ingestion, text cleaning, chunking, embedding generation, vector indexing, metadata filtering, optional reranking, context construction, prompt conditioning, response generation, and post-processing. Semantic

embeddings and approximate nearest-neighbour indexing make this pipeline practical for course-level retrieval over heterogeneous materials [10–12].

2.3 Agentic Workflows and Tool Use

The recent literature on LLM agents describes systems that can inspect a user request, plan intermediate steps, use tools, and revise outputs [13, 14]. For educational deployment, however, unrestricted autonomy is not necessarily desirable. A fully autonomous agent can call inappropriate tools, generate unstable reasoning paths, or hide the source of its decisions. This paper, therefore, adopts a semi-agentic design: the workflow includes task classification, routing, and tool invocation, but the policy is explicit and bounded. When revision or verification is needed, iterative self-feedback mechanisms can be adopted as controlled post-processing rather than unconstrained self-correction [15].

The distinction is important because the assistant should behave differently when a student asks for a definition, a chapter summary, a seven-day revision plan, or quiz questions. A single prompt may handle some cases, but a controllable workflow with specialised tools is more transparent, easier to debug, and better suited to institutional governance.

2.4 Research Gap

Three gaps motivate the present work. First, many educational chatbot studies emphasize perception measures while providing limited content-level evaluation. Second, many systems are either generic LLM assistants or narrow tools rather than integrated academic support platforms. Third, there is a practical need for privacy-aware local deployment in institutions that cannot expose internal teaching materials to public cloud services. The proposed assistant is positioned as an applied systems study that addresses these gaps through architecture, grounding, orchestration, and evaluation design [3, 6, 9].

3 Research Questions and Methodology

3.1 Research Questions and Hypotheses

The research questions are designed to isolate three mechanisms in the proposed system: grounding, orchestration, and deployability. RQ1 examines the knowledge mechanism by asking whether retrieved course evidence reduces unsupported or generic answers. RQ2 examines the process mechanism by asking whether task classification and tool routing improve multi-step learning support. RQ3 examines the deployment mechanism by asking whether the prototype remains usable when the LLM and course corpus are hosted locally rather than through a public cloud service (see Table 1).

The hypotheses are therefore operational rather than merely descriptive. H1 is evaluated through the difference between S1 and S2, H2 through the difference between S2 and S3, and H3 through timing, compression, and error observations. This structure makes the contribution testable: retrieval, orchestration, and local deployment are treated as separable design factors rather than as a single undifferentiated chatbot configuration

The hypotheses should be interpreted with appropriate caution, as the current evidence is at the prototype level. H1 is supported if

Table 1. Research questions, hypotheses, and evidence used in the evaluation design.

RESEARCH QUESTION	OPERATIONAL HYPOTHESIS	EVALUATION EVIDENCE
RQ1. Does retrieval improve factual accuracy and source grounding compared with a model-only assistant?	H1. RAG responses will obtain higher accuracy and grounding scores than LLM-only responses, while maintaining comparable coherence.	S1 versus S2; mean rubric score; accuracy and grounding subscores; unsupported-claim count; qualitative source-alignment check.
RQ2. Does semi-agentic orchestration improve multi-step academic tasks?	H2. RAG plus explicit task routing will improve planning and quiz-generation outputs over a single-step RAG pipeline.	S2 versus S3; task groups C and D; completeness and usefulness subscores; routing-error count; qualitative error analysis.
RQ3. Is privacy-aware local deployment feasible under ordinary department-level infrastructure?	H3. Compact local LLMs can meet acceptable response-time and compression requirements for bounded course-support tasks.	Timing by document length; compression ratio; model and hardware notes; error logs; resource footprint and interface responsiveness.

the RAG configuration improves accuracy and grounding without substantially reducing coherence or usefulness. H2 is supported if the semi-agentic configuration produces more complete study plans and better-grounded quiz items, particularly for compound prompts. H3 is supported if local processing remains within an acceptable latency range for academic support and if the system degrades gracefully when documents are long, ambiguous, or poorly formatted.

3.2 Design-Science Workflow

The study follows a design-science and prototype-evaluation methodology. The workflow consists of six phases: literature review, requirement analysis, architecture design, prototype implementation, evaluation set construction, and reflective assessment. This structure is appropriate because the research output is both an artifact - a working chatbot prototype - and a design framework that can be reused or extended by similar departments. It also aligns with the applied orientation of educational AI studies, which emphasise institutional fit, usability, and trustworthy deployment rather than model accuracy alone [1, 4].

The input data are grouped into three categories. The first category contains course materials such as syllabi, lecture notes, textbooks, slides, review questions, and practice documents in PDF, DOCX, and TXT formats. The second contains evaluation prompts and reference answers prepared from the course materials. The third contains future user feedback, including Likert-scale ratings and open comments from students or lecturers. Because the prototype uses institutional materials, the knowledge base is bounded and versioned rather than openly crawled from the web.

3.3 System Scope and Functional Requirements

The scope of the proposed system is defined as a course-specific academic support assistant for selected Information Technology courses at Hai Phong University, rather than a general-purpose conversational chatbot. The system is intended to operate on a bounded, institutionally curated knowledge base comprising syllabi, lecture notes, textbooks, slides, review questions, and other course-related documents in PDF, DOCX, and TXT formats. Its primary users are students who need timely support for understanding concepts, reviewing learning materials, preparing for assessments, and practising course-specific questions, while lecturers or administrators act as secondary users responsible for uploading, updating, and validating learning resources. Within this scope, the chatbot must support four core academic tasks: source-grounded course question answering, document summarisation, study-plan generation, and quiz generation. These functions are implemented through a semi-agentic workflow in which the system classifies the user's intent, retrieves relevant course materials when needed, invokes task-specific tools, and generates structured responses through a locally served

Table 2. Functional requirements of the system.

REQUIREMENT GROUP	FUNCTIONS IMPLEMENTED OR SPECIFIED
Knowledge management	Upload or register documents; extract text from PDF, DOCX, and TXT files; clean text; split into chunks; assign metadata; generate embeddings; update the vector index. The index also supports document-level incremental upsert, deletion, and re-embedding of changed chunks without rebuilding the entire corpus, except when the embedding model, chunking policy, or metadata schema changes.
Course question answering	Receive natural-language questions; infer course or topic; retrieve relevant passages; generate a concise explanation; show or mention source evidence.
Document summarization	Summarize a document, section, or chapter as paragraphs, bullet points, revision notes, or concept-example tables.
Study planning	Use the learner goal, available time, and perceived level to create a daily revision schedule with reading, practice, and self-check activities.
Quiz generation	Generate multiple-choice or short-answer items, plausible distractors, answers, hints, and self-check criteria grounded in retrieved content.
Monitoring and control	Log task type, model, retrieval results, source chunks, response time, errors, and user feedback for later evaluation.

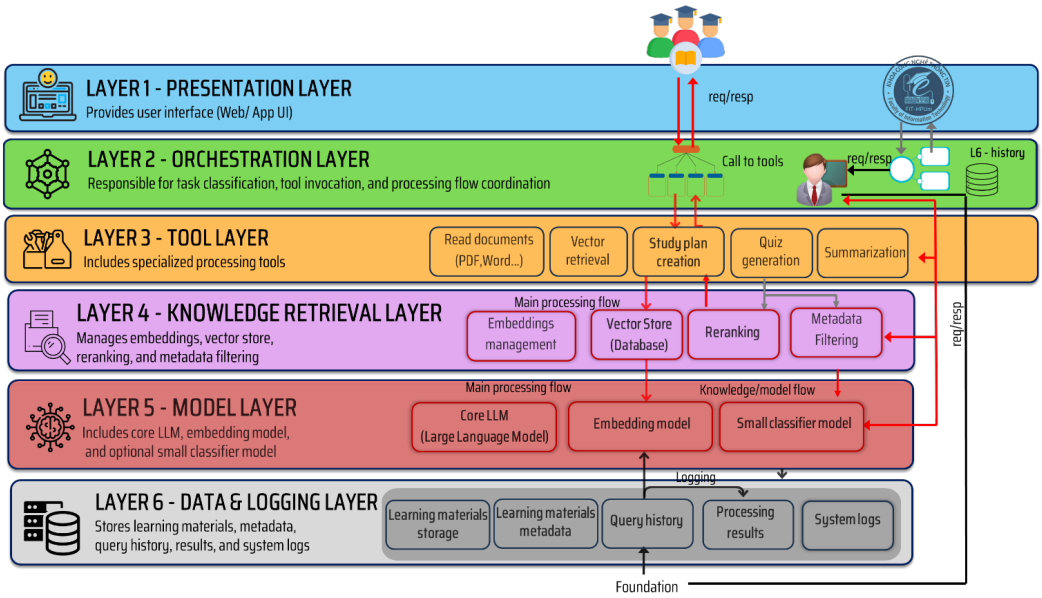

Figure 1. Six-layer semi-agentic architecture for the academic support chatbot.

language model. The system deliberately excludes unrestricted open-domain answering and autonomous decision-making beyond the defined academic workflows, because the educational setting requires controllability, source fidelity, and privacy-aware operation. Therefore, the functional requirements summarised in Table 2 translate the assistant's pedagogical goals into concrete system capabilities, covering knowledge management, learning support functions, and monitoring mechanisms for later evaluation and improvement.

4 Proposed System Architecture

4.1 Six-layer architecture

The proposed architecture comprises six layers, as illustrated in Figure 1. The Presentation Layer provides the user interface, while the Orchestration Layer is responsible for request classification, conversation context management, tool selection, and workflow coordination. The Tool Layer implements specialized academic operations, including search, summarization, planning, quiz generation, and document reading. The Knowledge Retrieval Layer handles embeddings, vector indexing, metadata filtering, and reranking. The Model Layer hosts the local LLM, embedding model, and an optional lightweight classifier. Finally, the Data and Logging Layer manages the storage of documents, metadata, conversation history, generated outputs, and operational logs. This layered separation enhances system maintainability, as individual layers can be updated or replaced independently without requiring a redesign of the overall architecture.

4.2 Agentic Routing and Tool Policy

At runtime, a student request is first normalized and classified into one of the supported task types: course question answering, document summarisation, study planning, quiz generation, material navigation, or out-of-scope request. The orchestrator then selects the corresponding toolchain. For knowledge questions, retrieval must precede generation. For summarisation, the document reader and summarizer are invoked. For study planning, the planner uses the goal, days available, and learner level. For quiz generation, the system retrieves source chunks before generating questions and answers. This bounded tool-use policy draws on agentic LLM concepts while avoiding unnecessary autonomy in an educational setting [13, 14].

The tool policy is intentionally constrained. The prototype does not allow the LLM to freely choose arbitrary tools. Instead, each tool has an explicit schema, a description, and conditions of use. This design reduces unpredictable behaviour and supports academic governance. When retrieval confidence is low, the assistant can ask the student to clarify the course, chapter, or concept rather than generate an unsupported answer.

To avoid ambiguity, the lightweight classifier in the current prototype is not a separate generative small language model and has not been trained as a fine-tuned BERT classifier. It is implemented as a lightweight rule-based and semantic-similarity router:

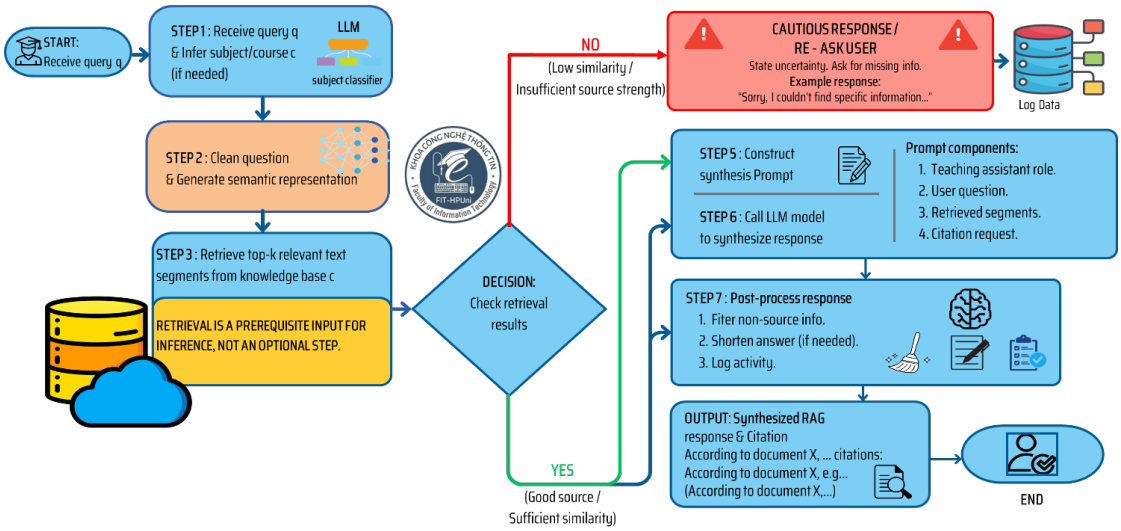

Figure 2. RAG-based orchestration algorithm for course question answering.

keyword and regular-expression cues detect explicit tasks such as summarise, plan, or generate quiz; metadata cues identify course and document scope; and optional sentence-embedding similarity to task prototypes provides an intent confidence score. A fine-tuned BERT or a compact intent model could replace this component when a sufficiently large, labelled interaction log is available, but the preliminary system uses a transparent rule-based/embedding approach because it is easier to audit in an educational setting.

4.3 RAG Pipeline for Course Question Answering

Let D be the set of institutional documents and C be the set of chunks generated from those documents. For a student query q, the system computes an embedding $E(q)$ and retrieves top-k chunks according to semantic similarity, optionally constrained by metadata such as course, chapter, or document type. The response is generated from the query, retrieved context, task policy, and system prompt. This formulation follows the core RAG principle of conditioning generation on retrieved evidence while using semantic sentence representations and vector indexes for scalable retrieval [8, 10–12].

$$y = \mathrm{LLM}(q, \mathrm{TopK}(C, E(q)), \mathrm{policy}) \qquad (1)$$

In this formulation, retrieval is a precondition for academic reasoning rather than an optional improvement step.

The route-selection method is therefore fixed in the set of permitted workflows, but dynamic in its execution. The orchestrator computes an intent confidence score P_{intent} and a retrieval confidence score $\text{Cert}_{\text{retrieval}}$. The retrieval score is estimated from the top-k similarity scores, the margin between the first and subsequent chunks, metadata matches, and optional reranking results. If P_{intent} is high, the request is sent to the corresponding fixed tool chain. If P_{intent} is low or two task types have similar scores, the assistant asks for clarification rather than allowing the LLM to choose an arbitrary path. For course question answering, generation is performed only when $\text{Cert}_{\text{retrieval}}$ exceeds a predefined threshold; otherwise, the system broadens metadata filters, increases k within a safe limit, or asks the student to specify the course, chapter, or concept. This policy means the system does not use a single static method for all questions; it combines fixed governance rules with confidence-based routing and fallback decisions.

The vector index is designed to support incremental updates. Each chunk is stored with a document identifier, a chunk identifier, a content hash, metadata, and an embedding model version. When a new document is uploaded, only its chunks are embedded and upserted into the index. When an existing document is modified, the system compares the new content hashes with stored hashes, re-embeds only changed chunks, deletes or deactivates obsolete chunk IDs, and preserves unchanged vectors. A full re-ingestion is required only when the embedding model, chunking policy, metadata schema, or vector database backend changes, or when the index is corrupted and must be rebuilt.

The RAG-based orchestration algorithm for course question answering is shown in Figure 2. This workflow emphasizes retrieval, confidence checking,

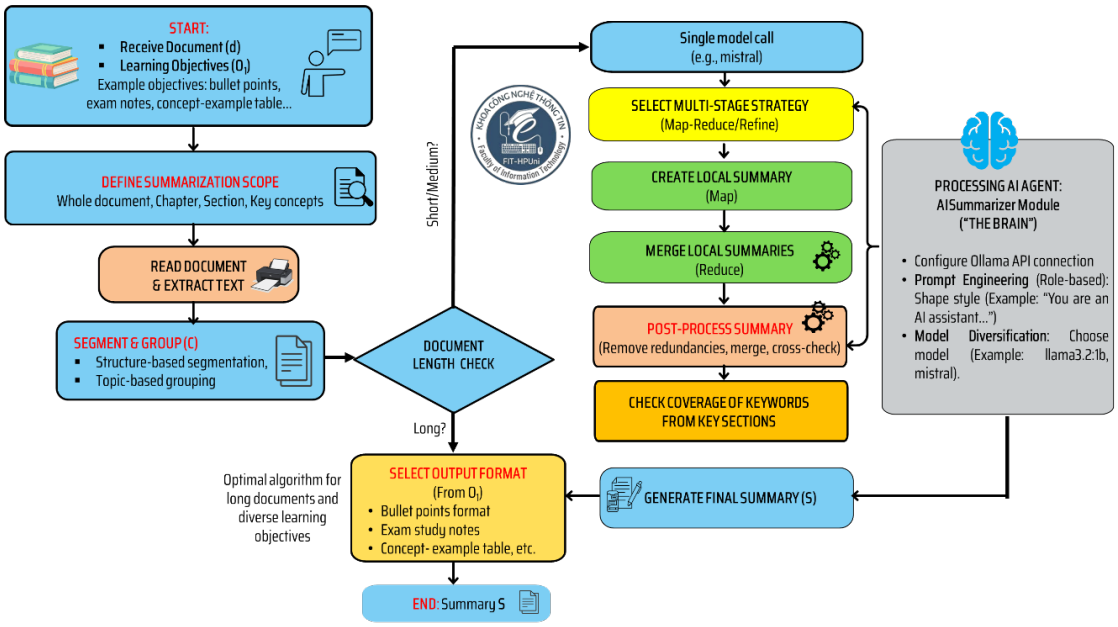

Figure 3. Algorithm flowchart for document summarization according to learning objectives.

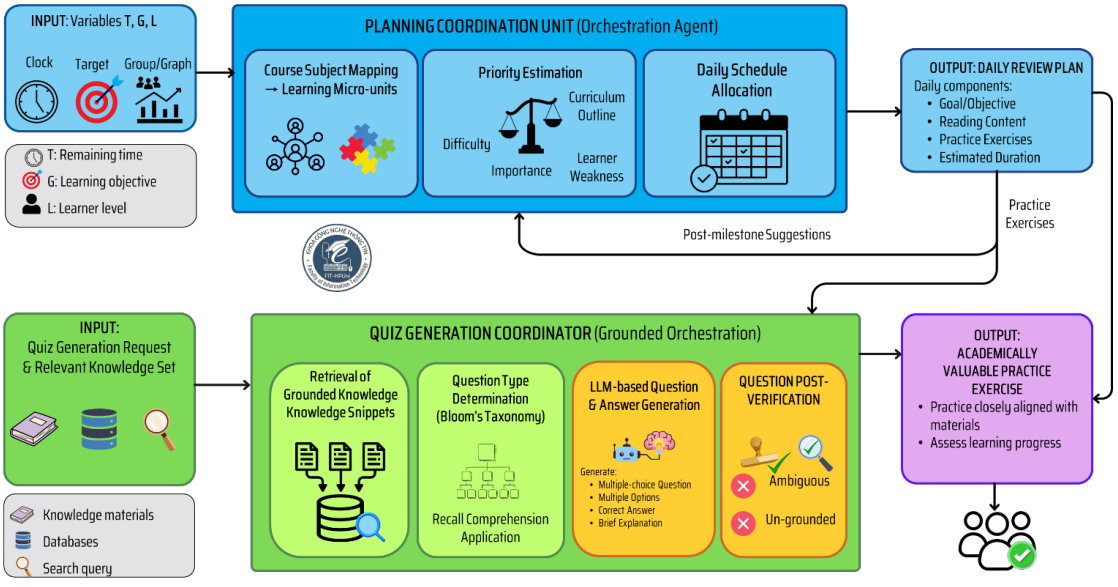

Figure 4. Algorithm flowchart for review planning and practice question generation.

source-conditioned prompting, and post-processing.

4.4 Goal-Oriented Summarization, Planning, and Quiz Generation

The summarisation workflow is goal-oriented rather than purely length-oriented, as shown in Figure 3. The student may request bullet points, exam notes, a concept-example table, or a chapter summary. For short and medium documents, the system can call the local model directly. For longer documents, a map-reduce or refine strategy is preferable: local summaries are created for chunks, merged, de-duplicated, and checked against important headings. This approach is more stable than sending a long document to the model in a single call and is consistent with iterative refinement strategies for improving generated outputs [7, 15].

Study planning and quiz generation are treated as structured academic tasks. Planning uses three variables: available time t, learning goal g, and learner level l. Topics are mapped to smaller learning units, weighted by difficulty, importance, and weakness, and then scheduled across days. Quiz generation retrieves source passages, selects the cognitive level of the question, generates items and answers, and applies post-verification to remove vague

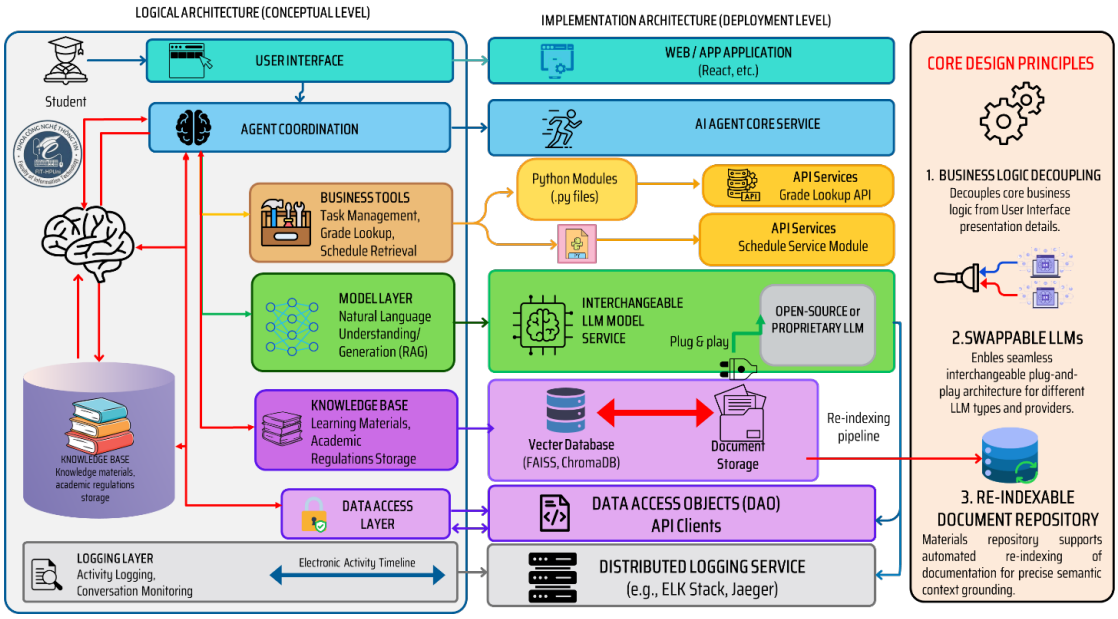

Figure 5. The agent-based chatbot system implementation.

or unsupported questions. This design reflects the need for pedagogically grounded and course-aligned AI support in computer science education [4, 5].

The study-planning and quiz-generation workflow is shown in Figure 4. The design combines grounded retrieval, orchestration, generation, and post-verification.

4.5 Local Deployment and Implementation Choices

The prototype is implemented in Python 3.10 or later. Text extraction uses libraries that support PDF, DOCX, and TXT formats. Local inference is provided through Ollama or an equivalent backend, with lightweight models such as llama3.2:1b and TinyLlama used during preliminary testing; Mistral or other models can be substituted when hardware permits. The initial interface is a desktop application built with Tkinter, with a planned migration path to web or messaging platforms such as Telegram or Messenger. Local deployment aligns with the broader need for privacy-aware, controllable AI support in university environments [4]. The agent-based chatbot system implementation is shown in Figure 5.

Two engineering decisions are central to usability. First, model inference is executed in a separate thread to prevent the interface from freezing during long processing. Second, a cache stores repeated summaries or generated outputs when the same document and configuration are used. These choices are modest but important for educational environments where hardware varies, and students expect stable interaction (see Figure 6).

5 Evaluation Design

5.1 Task Groups and Comparison Scenarios

The evaluation design contains four task groups. Group A covers course question answering, including

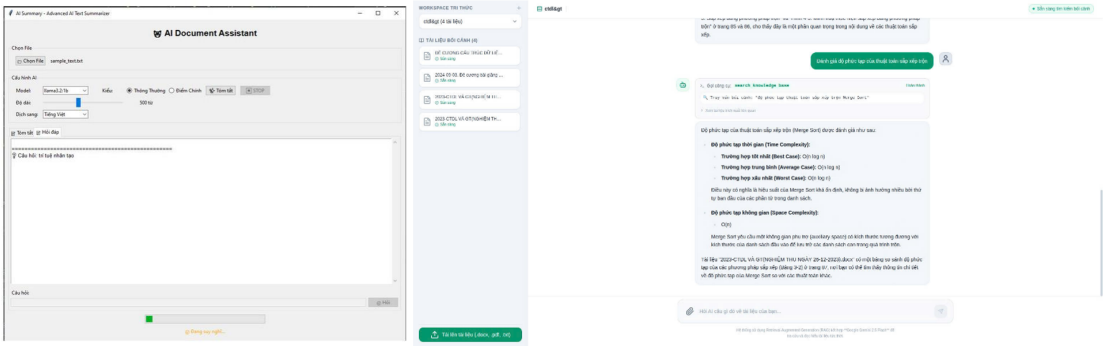

Figure 6. Prototype desktop interfaces: (a) question-answering interaction and (b) summarization interaction.

definitions, explanations, comparisons, examples, and algorithmic reasoning. Group B covers summarisation of chapters, sections, and long documents. Group C covers study planning for three-day, seven-day, and fourteen-day revision cycles. Group D covers quiz generation, including multiple-choice, short-answer, hints, and self-check criteria. The report recommends at least 20 prompts per group and separating development prompts from final evaluation prompts to reduce prompt overfitting (see Table 3). This task-oriented design responds to calls for educational AI evaluation that combines technical, pedagogical, and user-centred evidence [3, 5].

Table 3. Comparison scenarios for isolating the effects of retrieval and orchestration.

SCENARIO	CONFIGURATION	PURPOSE OF COMPARISON
S1	LLM only, without retrieval	Tests the weakness of model-only answers in course-specific contexts.
S2	LLM plus RAG, without explicit tool routing	Isolates the effect of source grounding and semantic retrieval.
S3	LLM plus RAG plus semi-agentic orchestration	Tests whether task classification and specialized tools improve multi-step academic tasks.

5.2 Academic Quality Rubric

The academic quality rubric translates the general idea of educational usefulness into five observable response properties. Knowledge accuracy and source grounding are treated as foundational dimensions because a fluent answer that is unsupported or inconsistent with course materials should not be considered academically reliable. Completeness, coherence, and usefulness then capture whether the response is sufficiently informative, understandable, and actionable for student learning.

Each criterion is scored from 0 to 2, producing a maximum of 10 points per response (see Table 4). For content-sensitive tasks, the rubric should be applied by at least two independent raters, preferably a lecturer and a teaching assistant or trained reviewer. A disagreement greater than one point on any dimension should trigger adjudication using the source material and the intended learning objective. This scoring protocol aligns with the broader view that educational AI should be assessed on content quality and learning value, rather than solely on user satisfaction [3, 5].

A suggested interpretation of the total score is as follows: 0-3 indicates that the response is unsuitable for student use; 4-6 indicates that the response is usable only with revision or instructor checking; 7-8 indicates an acceptable learning-support response; and 9-10 indicates a high-quality answer that is accurate, grounded, complete, coherent, and useful. The rubric should be slightly adapted for each task group: source grounding for summarisation means faithful preservation of source ideas, whereas source grounding for quiz generation means that each item and answer can be traced to retrieved course content.

The rubric also helps diagnose failure modes. Low accuracy suggests model or retrieval errors; low grounding suggests missing citations or weak retrieval; low completeness suggests insufficient context or over-compression; low coherence suggests prompt or formatting problems; and low usefulness suggests that the answer does not match the student's learning goal. Thus, the rubric serves as both an evaluation instrument and a debugging guide to improve the RAG and orchestration pipeline.

5.3 Operational Metrics

Operational evaluation records document reading and indexing time, retrieval latency, model inference latency, total response time, error rate, number of retrieved chunks, context-token usage, number of

Table 4. Five-dimensional rubric for evaluating chatbot responses. Each answer can receive up to 10 points.

CRITERION	0 POINTS	1 POINT	2 POINTS
Knowledge accuracy	Incorrect, misleading, or too vague for learning	Partially correct but contains omissions or imprecision	Correct, precise, and consistent with course materials
Source grounding	No source support or unsupported claims	Source support is present but incomplete or unclear	Clear, relevant, and traceable source grounding
Completeness	Important ideas, steps, or constraints are missing	Covers some expected ideas but leaves gaps	Covers the core ideas needed for the learning task
Coherence	Fragmented, illogical, or difficult to study	Mostly clear with minor organization problems	Clear, logical, and easy to follow
Usefulness	Limited learning value or not actionable	Some support for revision or practice	Clearly supports revision, practice, or conceptual understanding

Table 5. Preliminary processing time and compression targets of the local summarization module.

DOCUMENT LENGTH	MODEL USED IN TESTING	OBSERVED PROCESSING TIME	COMPRESSION TARGET	INTERPRETATION
1,000 words	llama3.2:1b	2-5 s	10-20% of source length	Fast for routine support
5,000 words	llama3.2:1b	5-15 s	10-20% of source length	Moderate and usable
10,000 words	llama3.2:1b	10-30 s	10-20% of source length	Moderate; benefits from chunking
20,000 words	TinyLlama	15-45 s	10-20% of source length	Slow but acceptable for offline summarization
Medium texts	Local compact LLM	5-15 s average	70-80% shorter than source	Usable for study support

model calls, and approximate memory and storage requirements. These indicators are important because a department-level assistant must be judged not only by answer quality but also by its deployability and maintainability under realistic infrastructure constraints. Vector database and indexing studies further suggest that retrieval latency, metadata filtering, and scalability should be treated as system-level variables rather than implementation details [10, 11].

6 Prototype Observations and Evaluation Framework Validation

6.1 Preliminary Timing and Compression Results

The timing results should be read as preliminary engineering observations rather than as a controlled hardware benchmark. Their purpose is to estimate whether compact local models can support common academic tasks under department-level infrastructure constraints. Because latency depends on CPU/GPU configuration, memory, model quantisation, prompt length, retrieved context size, and whether caching is enabled, the values in Table 5 should be interpreted as feasibility ranges.

For consistency, compression can be computed as CR = (number of words in the generated summary / number of words in the source text) x 100%. A CR of 10-20% corresponds to an 80-90% reduction, while the reported 70-80% shorter output corresponds approximately to a CR of 20-30%. The variation is expected because different academic tasks require different levels of detail: exam notes may be concise, but concept explanations and algorithmic procedures often need examples and intermediate reasoning steps.

A key preliminary observation from these results is not that the selected lightweight models are optimal, but that local deployment appears technically plausible for small-to-medium academic support. The next evaluation stage should repeat each timing condition across multiple runs, record hardware specifications, log token counts and retrieved context length, and compare identical prompts under S1, S2, and S3. This would allow the prototype observations to be converted into a reproducible benchmark.

The timing results show that compact local models can support bounded academic tasks when latency expectations are calibrated to the learning context. Short inputs are suitable for near-interactive assistance, whereas long-document summarization should be treated as an asynchronous or progress-indicated task. The reported 10-20% summary length target is pedagogically meaningful because it produces revision notes rather than overly compressed fragments. In contrast, a 70-80% reduction for medium texts indicates a less aggressive compression setting, often appropriate when preserving examples, definitions, or procedural steps is more important than maximizing brevity.

These timing observations also show a clear trade-off between interactivity and depth. Short prompts and small documents are appropriate for near-real-time tutoring interactions, whereas long-document summarisation should be presented as a progress-indicating background operation within the interface. The practical value of the local deployment, therefore, depends not only on raw latency, but also on matching the task mode to student expectations: question answering and short revision support should be fast, while chapter-level summarisation can tolerate longer processing if the output is more structured and faithful to the source.

6.2 Qualitative Comparison of S1, S2, and S3

The qualitative comparison indicates that the LLM-only configuration can produce fluent but generic answers. Such answers may be useful for broad background explanation but are less reliable for course-specific support. When RAG is added, the assistant becomes more aligned with official materials and can better match local terminology and emphasis. This effect is especially important for topics such as algorithmic complexity, database normalization, and programming constructs, where course examples and notation matter.

The semi-agentic setting is most valuable for multi-objective requests. A prompt such as "summarize chapter 3, identify weak points, and create a three-day revision plan" contains more than one academic task. In S3, the orchestrator can decompose the request, retrieve relevant content, summarize the topic, and then invoke the planner. This decomposition makes the output more structured and easier to evaluate than a single monolithic generation step.

More specifically, the expected performance pattern differs by task group. In Group A, S2 should primarily improve accuracy and grounding, as retrieved chunks constrain conceptual explanations. In Group B, the advantage comes from chunk-wise summarization and consistency checking rather than routing alone. In Group C and Group D, S3 is expected to outperform S2 because planning and assessment generation require additional structure: decomposing goals, sequencing topics, selecting difficulty levels, and verifying that generated questions are supported by retrieved content. Thus, the contribution of orchestration is not uniform across all tasks; it is most evident when a prompt requires multiple operations or contains implicit constraints.

These observations should also be interpreted as prototype-level evidence rather than statistically conclusive results. The current evidence supports feasibility and plausibility, but it does not yet prove significant learning gains. The next benchmark should report mean rubric scores, standard deviations, inter-rater agreement, unsupported-claim counts, routing-error counts, and latency distributions for S1, S2, and S3. Presenting results in this way would make the discussion more reproducible and would show whether improvements come from retrieval quality, tool selection, prompt design, or model capacity.

6.3 Engineering challenges and responses

The engineering challenges identified during development can be grouped into four categories: model-inference constraints, knowledge-quality constraints, user-interface reliability, and local-service operations. These categories show that the main difficulty is not only choosing an LLM, but integrating model inference, document processing, retrieval, prompting, and interface behavior into a stable learning-support workflow. The responses in Table 6 are therefore pragmatic design decisions rather than final optimizations.

Three engineering principles follow from these responses. First, the system should use explicit orchestration rather than allowing the model to choose arbitrary actions, as this makes bounded routing easier to debug and audit. Second, the knowledge base should be treated as a versioned instructional asset: poor metadata, duplicated files, or noisy extraction directly reduce answer quality. Third, the interface should support graceful degradation via progress messages, cached outputs, clear error messages, and fallback modes when retrieval confidence or local model capacity is insufficient.

Table 6. Engineering challenges and solution strategies reported during prototype development. Effects are indicative and should be validated in a controlled benchmark.

CHALLENGE	CAUSE	ENGINEERING RESPONSE	REPORTED EFFECT OR RATIONALE
Slow model inference	LLMs require substantial computation	Use compact local models such as TinyLlama or llama3.2:1b	Reported as 50-70% faster in the prototype context; should be validated under controlled hardware conditions
Poor summarization quality	Vague prompt or weak output constraints	Use role-based prompts, explicit format requirements, and task-specific instructions	Reported as improving summary quality by 30-40%; indicates the value of explicit task constraints
Frozen interface	Long inference on the main UI thread	Run inference through threading.Thread	Improves perceived responsiveness and prevents the application from appearing unresponsive
Unsupported or unreadable files	Input format mismatch or corrupted documents	Support TXT, PDF, and DOCX; add exception handling	Improves robustness and reduces avoidable failure during document ingestion
Memory limitations	Model or context too large for local hardware	Use smaller models, chunking, and fallback modes	Allows normal operation under modest hardware by limiting context size and model load
Ollama service error	Local service not running	Check and start the service before inference	Enables early error detection and clearer user-facing diagnostic messages

The reported speed and quality gains are useful as prototype indicators but should not be overinterpreted as general performance claims. A stronger engineering evaluation should specify hardware, model version, quantization level, input length, retrieved context size, number of model calls, and repeated-trial variance. It should also include failure cases, because robust educational deployment depends on predictable behavior under imperfect documents and ambiguous student requests (see Table 6).

6.4 Error Analysis

Four recurring error categories were expected and were identified in the report. Retrieval errors occur when irrelevant chunks are selected, relevant chunks are omitted, or chunk size is poorly chosen. Generation errors include overgeneralization, verbosity, missing emphasis, and terminology inconsistent with the course. Orchestration errors occur when the wrong tool is invoked, for example, using the question-answering pipeline for a planning task. Data errors arise from poor document structure, OCR noise, duplication, or insufficient metadata. These categories provide an actionable improvement path: better chunking, metadata enrichment, reranking, stronger prompts, and a critique or verification step.

The confidence-based policy provides a practical mitigation for these error categories. A low intent-confidence score indicates a possible routing error, while a low retrieval-confidence score indicates weak evidence for generation. In both cases, the system should prefer clarification, source-filter adjustment, or a cautious partial answer over an unsupported fluent response. This behavior may make the assistant less conversational in some cases, but it improves auditability and is more appropriate for course-specific educational support.

7 Discussion

The strongest implication of the prototype is that educational usefulness depends more on grounding and orchestration than on model size alone. A larger model may produce more fluent text, but a smaller model connected to a well-curated knowledge base can be more appropriate for course-specific support. This observation is consistent with RAG research, which shows the value of retrieved evidence for knowledge-intensive generation, and with educational RAG surveys that emphasize source grounding and curriculum alignment [8, 9].

The six-layer architecture also supports governance. Lecturers or administrators can update the knowledge base without changing the model, replace the model without changing the interface, or refine tool policies without rebuilding the document store. This separation is important for a university department because curricula change, course documents are revised, and infrastructure budgets vary from semester to semester.

From a pedagogical perspective, the assistant should be positioned as a learning support tool rather than a replacement for teachers. It can help students ask questions, review materials, and practice, but it must also encourage source checking and independent reasoning. Guardrails should be in place for requests that seek complete answers to graded assignments, fabricate citations, or bypass academic integrity rules. Ethical and equity-aware governance is especially important in AI-supported education systems that handle institutional data and learner interactions [4, 16].

8 Limitations and Future Work

The present study has several limitations. The knowledge base currently covers only selected Information Technology courses. The evaluation remains preliminary and does not yet include a large, anonymised user dataset or a controlled assessment of learning outcomes. The figures and prototype interface come from a student research project and should be refined for production deployment. The system is also primarily text-centric and does not yet exploit diagrams, code notebooks, lecture videos, or multimodal slide content.

Future work should extend the corpus, standardize metadata, add reranking and answer verification, implement a human-in-the-loop review process for official course materials, and evaluate the system with students and lecturers over a semester. A larger benchmark should include reference answers, inter-rater agreement, blinded scenario comparison, latency measurements under different hardware configurations, and post-use learning analytics. The architecture can also evolve toward a multi-agent design with a Retrieval Agent, Tutor Agent, Planning Agent, Assessment Agent, and Critic Agent, but only after the single-orchestrator version is stable and well evaluated. This extension is compatible with broader work on AI-supported blended learning and intelligent training management in Vietnamese higher education [17].

9 Conclusion

This paper presents a comprehensive academic formulation of an agentic AI chatbot with retrieval-augmented generation to support students in Information Technology courses. The proposed system combines local LLM inference, course-grounded semantic retrieval, tool-oriented orchestration, and structured evaluation. Preliminary observations suggest that compact local models can provide usable response times for bounded academic tasks and that RAG substantially improves alignment with official course materials. The study contributes a practical architecture and evaluation framework for higher education institutions that require privacy, controllability, and pedagogical grounding. The next stage is to validate the prototype through a broader benchmark and an anonymised user study.

Data Availability Statement

Data will be made available on request.

Funding

This work was guided and supported by Associate Professor Dr Dac-Nhuong Le, the Principal Investigator of the project "Research on developing a search system (HPUmind) to support teaching and learning in Information Technology, integrated circuit design, and semiconductors at Hai Phong University" (ĐT.XH.2025.980).

Conflicts of Interest

Dac-Nhuong Le serves as an Associate Editor of *Next-Generation Computing Systems and Technologies*. To ensure the integrity of the peer-review process, Dac-Nhuong Le had no involvement in the editorial review, peer review, or decision-making process for this manuscript. The manuscript was handled independently by another editor in accordance with the journal's editorial policies. The remaining authors declare that they have no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Chiu, T. K. F., Xia, Q., Zhou, X., Chai, C. S., & Cheng, M. (2023). Systematic literature review on opportunities, challenges, and future research recommendations of artificial intelligence in education. *Computers and Education: Artificial Intelligence, 4*, 100118. [CrossRef]

[2] Garzón, J., Patiño, E., & Marulanda, C. (2025). Systematic review of artificial intelligence in education: Trends, benefits, and challenges. *Multimodal Technologies and Interaction, 9*(8), 84. [CrossRef]

[3] Lo, C. K., Hew, K. F., & Jong, M. S. Y. (2024). The influence of ChatGPT on student engagement: A systematic review and future research agenda. *Computers & Education, 219*, 105100. [CrossRef]

[4] Qadir, J. (2023, May). Engineering education in the era of ChatGPT: Promise and pitfalls of generative AI for education. In *2023 IEEE global engineering education conference (EDUCON)* (pp. 1-9). IEEE. [CrossRef]

[5] Raihan, N., Siddiq, M. L., Santos, J. C., & Zampieri, M. (2025, February). Large language models in computer science education: A systematic literature review. In *Proceedings of the 56th ACM technical symposium on computer science education V. 1* (pp. 938-944). [CrossRef]

[6] Kim, M., Kim, J., Knotts, T. L., & Albers, N. D. (2025). AI for academic success: Investigating the role of usability, enjoyment, and responsiveness in ChatGPT adoption. *Education and Information Technologies, 30*(10), 14393-14414. [CrossRef]

[7] Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., & Reeves, B. N. (2024, March). Prompt Problems: A new programming exercise for the generative AI era. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (pp. 296-302). [CrossRef]

[8] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems, 33*, 9459-9474.

[9] Swacha, J., & Gracel, M. (2025). Retrieval-augmented generation (RAG) chatbots for education: A survey of applications. *Applied Sciences, 15*(8), 4234. [CrossRef]

[10] Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE transactions on big data, 7*(3), 535-547. [CrossRef]

[11] Pan, J. J., Wang, J., & Li, G. (2024). Survey of vector database management systems. *The VLDB Journal, 33*(5), 1591-1615. [CrossRef]

[12] Reimers, N., & Gurevych, I. (2019, November). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)* (pp. 3982-3992).

[CrossRef]

[13] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., ... & Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems, 36*, 68539-68551.

[14] Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., ... & Wen, J. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science, 18*(6), 186345. [CrossRef]

[15] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., ... & Clark, P. (2023). Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems, 36*, 46534-46594.

[16] Phung, T. N. (2026). Using explainable AI to diagnose institutional inequality in student dropout across ethnic and regional groups in Vietnam. *AI & SOCIETY*, 1-18. [CrossRef]

[17] Phung, T. N., Do, D. C., Nguyen, T. T., Nguyen, V. S., Nguyen, T. V., & Le, D. N. (2026). An integrated framework for outcome based education and AI supported blended learning in curriculum redesign and intelligent training management. *Discover Computing, 29*(1), 196. [CrossRef]

Tran Huu Van is an undergraduate student of the Class CNTT 1 K24 in the Faculty of Information Technology, Hai Phong University, Vietnam. His research interests include educational artificial intelligence, retrieval-augmented generation, and intelligent tutoring support. He contributed to the system design, experimentation, and manuscript preparation. (Email: trhuvan@gmail.com)

Nguyen Phu Vinh is an undergraduate student of the Class CNTT 3 K24 in the Faculty of Information Technology, Hai Phong University, Vietnam. His interests include local LLM deployment, system integration, and practical AI engineering. He contributed to the prototype implementation, testing, and performance analysis. (Email: nguyenphuvinh3112@gmail.com)

Vu Thi Tuyet Ngan is an undergraduate student of the Class CNTT 3 K24 in the Faculty of Information Technology, Hai Phong University. Her interests include natural language processing, software analysis, and academic support systems. She contributed to requirement analysis, user workflow design, and evaluation planning. (Email: nganvuthi1203@gmail.com)

Luong Ha Phuong is an undergraduate student of the Class CNTT 3 K24 in the Faculty of Information Technology, Hai Phong University, Vietnam. Her research interests include educational technology, document processing, and human-centered AI applications. She contributed to document preparation, prototype analysis, and data organization. (Email: luonghaphuong2312@gmail.com)

Dac-Nhuong Le has an M.Sc. and PhD in computer science from Vietnam National University, Vietnam in 2009 and 2015, respectively. He is an Associate Professor of Computer Science, and the Dean of the Faculty of Information Technology at Hai Phong University, Vietnam. He has a total academic teaching experience of 20+ years with many publications in reputed international conferences, journals, and online book chapter contributions (Indexed by SCI, SCIE, SSCI, Scopus, ACM, DBLP). His area of research includes Soft computing, Network communication, security and vulnerability, network performance analysis and simulation, cloud computing, IoT, and Image processing in biomedical. His core work is in network security, soft computing, IoT, and image processing in biomedical. Recently, he has been on the technique program committee, the technique reviews, and the track chair for international conferences under the Springer-ASIC/LNAI Series. He serves on the editorial board of international journals and has authored/edited 30+ computer science books by Springer, Wiley, CRC Press, IET, DeGuy, etc. (Email: nhuongld@dhhp.edu.vn)