AI Engineering
https://www.sciltp.com/journals/aieng



Article

Retrieval-Augmented Generation Tutor for MOOCs: Comparative Evaluation of LLM Architectures

Enrique Frias-Martinez [1,*], Jose C. San Jose [1], Dejian Liu [2] and Daniel Burgos [1,3]

[1] Research Institute UNIR iTED, Universidad Internacional de La Rioja (UNIR), 26006 Logroño, Spain
[2] Smart Learning Institute, Beijing Normal University, Beijing 100082, China
[3] MIU City University, Miami, FL 33132, USA
* Correspondence: enrique.frias@unir.net

How To Cite: Frias-Martinez, E.; San Jose, J.C.; Liu, D.; et al. Retrieval-Augmented Generation Tutor for MOOCs: Comparative Evaluation of LLM Architectures. *AI Engineering* **2026**, *2*(2), 24. https://doi.org/10.53941/aieng.2026.100024

Received: 22 September 2026
Revised: 11 November 2026
Accepted: 12 November 2026
Published: 19 November 2026

Abstract: The use of Large Language Models (LLMs) for Massive Open Online Courses (MOOCs) presents new opportunities to facilitate context-aware support. In this context, LLMs have two main limitations: (1) the inability to provide answers relevant to the context of the student and (2) the risk of providing inaccurate answers. Retrieval-Augmented Generation (RAG) has been designed to overcome these limitations by combining the retrieval of curated information with the language processing capabilities of LLMs. However, there is little evidence on how the choice of deployment architecture and LLM affects the latency, answer quality, cost and student perception of RAG tutors. This paper presents the design, multi-LLM evaluation, and student perception of RAG-Tutor, a RAG architecture built to serve as a chatbot for MOOCs. RAG-Tutor was evaluated in an "Introduction to Operating Systems" MOOC across three deployment architectures and seven LLMs, using a benchmark of 196 expert-validated questions. An exploratory study with 30 students examined self-reported perceptions of satisfaction, trust and explainability. RAG-Tutor demonstrates the feasibility of RAG systems by showing that: (1) RAG approaches outperform stand-alone LLMs; (2) not all architectures provide real-time interactions and high-quality answers; and (3) the choice of LLM has an impact on student perceptions.

Keywords: retrieval-augmented generation; RAG; LLMs; MOOCs; intelligent tutoring systems

1. Introduction

Massive Open Online Courses (MOOCs) have changed the way people learn by providing adaptable learning. Nevertheless, they face some limitations, mainly related to student retention, reported in the literature to be in the range of 5–15% [1,2]. Although there can be various reasons for abandonment, insufficient engagement and motivation are usually considered the main factors [3]. To improve engagement and motivation, students need mechanisms to provide 24/7 support that is aligned with their needs and context [4]. AI chatbots are one of the ways of providing this support, but to be effective, they need to be aligned with instructional practices that support engagement [5]. The evolution of AI tutors can be presented in three generations. Early rule-based systems provided guidance but required expert effort to build for each domain [6]. Subsequent machine learning systems improved personalization using behavioral data but had the problem of data scarcity [7]. Recently, deep learning, progressing from CNNs and LSTMs to transformer and LLM architectures, have advanced contextual understanding and long-range text dependencies [8] of natural text. These advances have enabled tutoring systems based on LLMs to generate responses using natural language interaction. LLMs as tutors can be useful tools, but

Copyright: © 2026 by the authors. This is an open access article under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).
Publisher's Note: Scilight stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

they have limitations, mainly: (1) potential for hallucinations, which is especially relevant in education; (2) generic answers, since LLMs do not necessarily have the student's course context; and (3) potential bias in the answers.

Retrieval-Augmented Generation (RAG) has emerged as an approach to overcome the previous limitations by combining information retrieval from a database of curated information with LLMs, to produce context-based answers [9]. It has three stages: (1) retrieve relevant information from the database using the question, (2) augment the LLM prompt with the retrieved information, and (3) generate the answer. As a result, in the context of MOOCs, RAG can potentially improve the answer given by an LLM by contextualizing the generated content with the documents that the student is using for the learning process [10].

Although RAG-based tutors have shown promising results, two problems limit their adoption in MOOCs. First, most existing systems are built for a specific course or domain and require technical effort to adapt to new content, which is a barrier for instructors. Second, institutions that want to deploy a RAG tutor must choose between APIs, locally hosted open-source models, cloud services, and between LLMs of very different size and cost. These choices determine whether the tutor can respond within conversational latency limits, how much it costs at scale, and how data privacy is handled. However, existing studies typically evaluate a single LLM and do not compare how the choice of model affects answer quality, latency, cost and student perception. As a result, there is little empirical evidence to guide these design decisions. This paper addresses this gap through three research questions:

RQ1. Which deployment architectures and LLMs satisfy the latency requirements of conversational tutoring?

RQ2. To what extent does retrieval augmentation improve answer quality compared with stand-alone LLMs, and how does this improvement depend on the LLM used?

RQ3. How do retrieval augmentation and the choice of LLM affect student perception?

To answer these questions, this paper presents RAG-Tutor, a modular RAG architecture designed for integration with MOOC platforms. The goal is to provide students with an interface they already know and use and where they can centralize all interactions with the course materials. By lowering the technical barriers for adoption, RAG-Tutor aims to improve the usability and adaptability of RAG-based tutoring systems. To evaluate the system, RAG-Tutor was integrated into an "Introduction to Operating Systems" MOOC and benchmarked using three deployment architectures and seven LLMs. In this paper, adaptability refers to adaptation to the course: RAG-Tutor can be adapted to any course by uploading its materials, and each answer is adapted to the student's question and to the course content.

The main contributions of this paper are: (1) the design and implementation of RAG-Tutor, a modular and course-agnostic RAG architecture; (2) a comparative technical evaluation of the same RAG pipeline deployed over different back-ends (Ollama, OpenAI and AWS Bedrock) and seven LLMs using latency and quality metrics (RQ1 and RQ2); and (3) a small-scale student-centered evaluation for measuring the impact of the choice of LLM on student perception (RQ3). Regarding RQ1, the results identified differences in response latency across architectures and models under the tested deployment conditions. Regarding RQ2, retrieval augmentation improved performance compared to stand-alone LLMs. RAG-Tutor improved BLEU scores by 17–33% and BERTScore F1 by 3–13% compared to stand-alone LLM configurations, with statistically significant differences ($p < 0.05$). The evaluation also showed that medium-scale models can achieve semantic quality comparable to larger models while maintaining low latency values. Regarding RQ3, preliminary findings from the student perspective, suggest that some RAG-based configurations improved explainability, trust, and perceived cognitive load compared to stand-alone LLM tutoring. However, the student evaluation involved only 30 participants, with 10 per condition, and was not designed to assess learning gains or educational effectiveness. Consequently, the student-related findings should be interpreted as exploratory and require confirmation in larger studies.

The rest of the paper is organized as follows: First, Section 2 reviews the literature on LLM and RAG-based tutoring systems. Section 3 presents the proposed architecture. Section 4 presents the experimental configuration, benchmark design and evaluation metrics. Section 5 presents the experimental evaluation of RAG-Tutor, including latency, semantic quality and cost analysis. Finally, Section 6 summarizes the main findings and outlines future research.

2. Related Studies

LLMs have been used as tutors in education since their inception, and initial results have indicated a potential improvement in the learning process [11]. In general, regardless of how LLMs are applied in education, their integration requires developing new competencies among teachers and students to properly use the tools [12]. In any case, the impact of LLMs on education and the learning process is still being studied.

Tutors built using a RAG approach have been the focus of many studies. Reference [13] showed that RAG-based tutors improve accuracy and performance when compared to stand-alone LLM tutors. Similarly, reference [14] pointed out how RAG tutors improve the interpretability of answers compared to LLM tutors. RAG-based tutors have already been implemented in different contexts. For example, reference [15] used RAG to improve the automatic assessment of human tutors' social-emotional practices in math tutoring. Reference [16], also in the context of math tutoring, demonstrated that RAG has the potential to improve response quality compared to LLM tutors, and reference [17] presented a RAG tutor for statistics, concluding that it improved student comprehension and engagement. Similarly, reference [18] found that RAG-based tutors increased student participation and interaction. In psychology, reference [19] showed an improvement in learning outcomes with RAG-based tutoring, attributing success to targeted retrieval of information. Reference [20] found that RAG tutors enhanced student engagement and comprehension of scientific literature, although they highlighted the difficulty of measuring those elements. Finally, reference [21] presents MoodleBot, a RAG chatbot integrated into Moodle to support students in higher-education courses. The evaluation involved 46 students, of whom 30 completed a Technology Acceptance Model (TAM) questionnaire, and reported an 88% accuracy rate for course-related assistance together with positive student perceptions of the system. The systems reviewed above are focused on retrieving and generating text, but in many courses, however, an important part of the content is visual. Multimodal RAG extends retrieval to these sources. Early approaches such as MuRAG [22] and RA-CM3 [23] retrieve images and text from a shared memory, typically using vision-language encoders such as CLIP [24] to embed both modalities in a common space. These architectures are relevant for tutoring scenarios with visual inputs. It is important to distinguish between two forms of adaptation in AI tutoring. The RAG-based tutors reviewed above adapt their answers to the course context by grounding them in the course materials, but they do not model what each student knows. Recent graph-based cognitive models capture the relations between students, exercises and concepts [25]. These two directions are complementary: course grounding determines what content is used to answer, whereas learner modeling could determine how the answer should be tailored to each student.

The initial results of RAG-based tutors indicate an improvement in student engagement and comprehension. Nevertheless, they also face some limitations: (1) Adapting RAG systems for different courses requires an effort to curate and maintain domain-specific knowledge; (2) Sustaining long-term student engagement can be difficult because the initial enthusiasm of students to interact with RAG-based tutors diminishes over time [18]; (3) Initial findings on the impact of RAG-based chatbots on the learning process seem to be positive, but there is still a need for large-scale studies; and (4) a key limitation is that, in general, the different studies do not compare the results of the RAG approach using different LLMs and the impact it has on quality and student perception.

Unlike RAG-based tutors that rely on domain-specific architecture or require customization to adapt to individual courses, our approach focuses on designing a generic RAG tutor framework. We call it RAG-Tutor, designed to be integrated into MOOCs as part of their interface. The main advantage of this architecture lies in its flexibility: educators can easily personalize the system by uploading their own course materials, which are then indexed to support contextual retrieval. We present the design as an engineering process focused on evaluating the impact of the LLM used in the process and identifying one that produces better results.

3. RAG-Tutor Architecture

A RAG architecture consists of three main elements: (1) an indexer, which creates an index of the information provided; (2) a retriever, which looks through the materials to find relevant information for user queries; and (3) a generator, which uses an LLM to produce responses based on the contextual information provided by the retriever.

Figure 1 presents a RAG architecture: the question posed by a student is transformed by the embedding, which is then used by the retriever to collect the pieces of information most relevant to the question. Once that information has been retrieved, the original question and the contextual information retrieved are combined into an extended prompt. This extended prompt is then fed into the LLM, which provides an answer. In the context of tutors, the set of documents is curated by the professor of the course to represent the context in which the student is learning.

3.1. Indexer

The goal of the indexer is to prepare the information for RAG-Tutor to be used by the retrieval system. The main steps are: (1) The data curated is uploaded into the system and converted into text, (2) The data is divided into chunks, (3) an embedding is applied to each chunk, and (4) a vectorial database is created with the vectors produced by the embedding. The indexer is only run once before using RAG-Tutor and can be updated periodically. A web-based interface was built to allow instructors to upload course materials with minimal

technical expertise required. The interface provides a basic drag-and-drop upload, allowing for batch processing of multiple documents. The indexer extracts the text of the materials. Afterward, the information is cleaned to remove formatting artifacts, headers, footers, and nontextual elements. In the current implementation, RAG-Tutor operates only on text: figures, images and other non-textual elements are discarded during indexing.

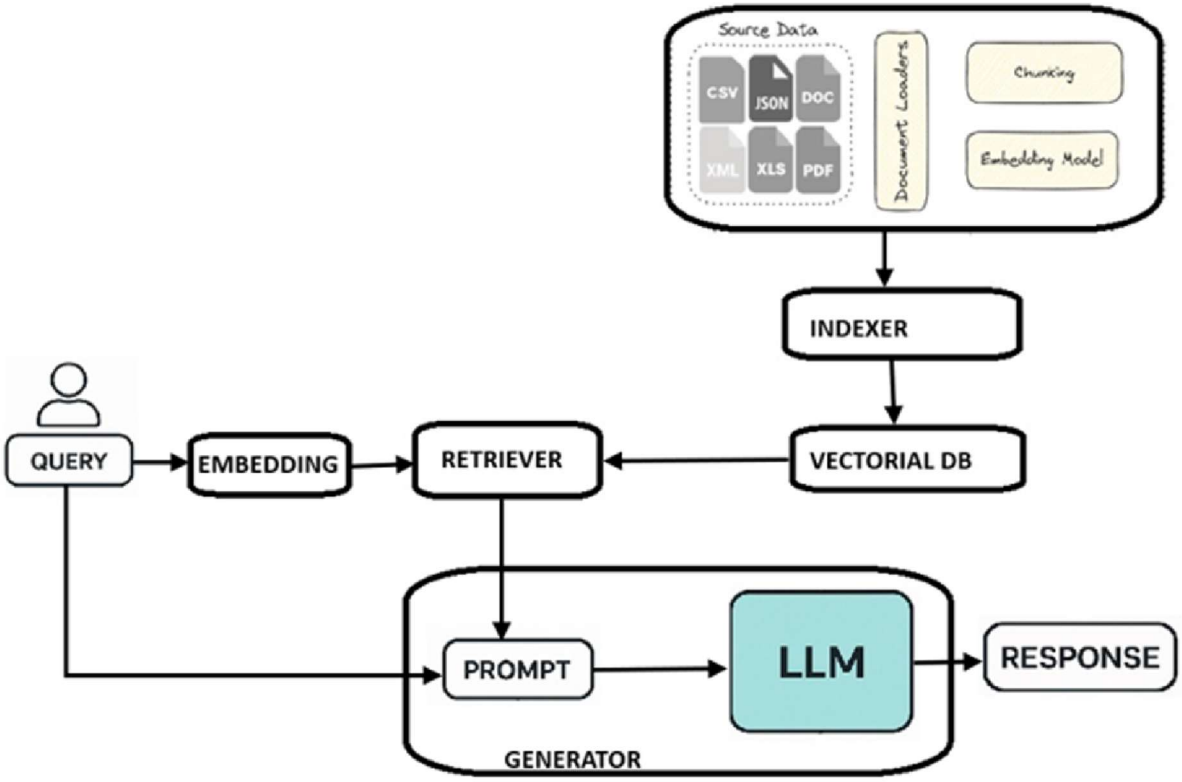


Figure 1. Basic architecture of a generic RAG system.

Chunking, the division of the input text into independent pieces of text, is a key element for creating the database. In our context, the text is divided using a sliding window technique, with a chunk size of 2000 tokens and a 25% token overlap (500 tokens). The overlap is a key element to avoid producing isolated chunks by capturing concepts that span multiple chunks. In general, larger overlaps improve retrieval performance [26,27], with a 25–50% overlap producing the best results. The literature highlights that large windows work better for educational and open-ended questions [28], which justifies the use of 2000 token windows. In RAG-Tutor, text chunks are converted into vector representations using the BGE-M3 embedding model, a transformer-based encoder that uses an XLM-RoBERTa-based architecture, with an embedding dimension of 1024. For local storage and similarity search, RAG-Tutor uses ChromaDB because it is easy to deploy and can be scaled to handle large collections. The vector database stores each chunk of data, including (1) original textual content; (2) vector embedding of the text; and (3) metadata, such as source document, page within the document, and chunk ID.

3.2. Retriever

The retrieval system serves as the bridge between student queries and relevant educational content using semantic similarity. To identify relevant chunks, it uses Maximal Marginal Relevance (MMR) as a re-ranking strategy to balance relevance against redundancy. This approach avoids the limitations of other retrieval techniques, such as nearest-neighbor, where top-ranked passages are redundant. MMR improves the diversity of retrieved results [29]. Formally, given a query q and a set of candidate passages D, MMR selects a subset by maximizing:

$$\mathrm{MMR}(d) = \lambda \cdot \mathrm{sim}(d, q) - (1 - \lambda) \cdot max_{s \in S}\, \mathrm{sim}(d, s) \quad (1)$$

where sim(.,.) denotes a similarity function in the embedding space (cosine similarity for RAG-Tutor). The first term rewards the relevance of a candidate d with respect to the query, while the second term penalizes redundancy by measuring its similarity to passages already selected. The trade-off is controlled by the parameter λ in the range

[0, 1]. In our configuration, following references [30,31], the retriever first identifies the top 20 candidates and then MMR is applied to select the final k = 3 passages. We use $\lambda = 0.5$, which gives equal weight to relevance and diversity.

3.3. Generator

The generator uses a prompt to create responses that answer the student question based on the information provided by the retriever. When designing a prompt, different techniques can be used, mainly zero-shot, few-shot, and chain-of-thought (CoT) prompting [32]. Zero-shot refers to the capability of an LLM to perform a task based exclusively on pretrained knowledge. Few-shot improve the process by incorporating several examples as part of the prompt [33]. Last, CoT shows an LLM how to break down a task into steps before providing an answer [34]. This method aligns well with tasks like tutoring, where multiple factors contribute to final answers. Box 1 presents the prompt used for the generator in RAG-Tutor.

The prompt is designed using a combination of CoT that uses a zero-shot approach in each step. It lays out step-by-step reasoning that RAG-Tutor must follow to generate a response. Note that at the end of the prompt, the generator includes: (1) the previous history of the interaction between the student and RAG-Tutor; (2) the actual question posed by the student; and (3) the context (i.e., the retrieved chunks of information), before sending the final prompt to the LLM. Providing the history of the conversation is a key part of the context to have meaningful conversations between the RAG-Tutor and the student.

Box 1. Generator Prompt used by RAG-Tutor.

You are a multilingual virtual tutor designed to help students understand course-related programming concepts. Your answers must always be formatted in valid HTML using bold text, lists, and other visual aids to improve clarity. Do not wrap the answer inside a code block. Output only valid HTML content. If any links are detected, automatically convert them into clickable hyperlinks using URL.

Instructions

Relevance and Priority

Prioritize information from the provided course materials.

If relevant context exists, extract it, translate if necessary, and base the answer strictly on it.

Fallback

If no relevant information is found, output exactly:

No relevant information was found in the provided course materials.

Answer Construction

Maximum 100 words. Be concise, clear, and respectful.

Use HTML formatting (e.g., ,). Do not use <code>.

Scope

Only answer questions related to course content. Do not respond to unrelated or personal questions.

Inputs

History: {history}

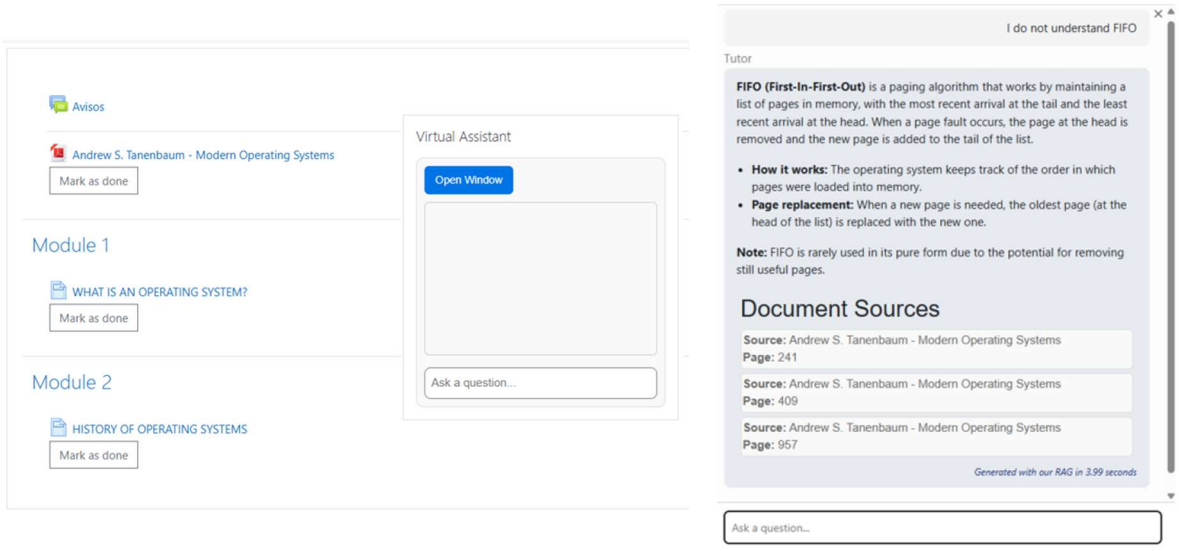
Question: {input}

Retrieved Context: {context}

Output: HTML-formatted answer.

3.4. Integration with the MOOC Platform

RAG-Tutor was designed to be part of a MOOC to provide students with a single point of access for the course, its materials and chatbot. Figure 2a presents an example of a MOOC, with a RAG-Tutor window embedded on the right, and Figure 2b an example of a student question, the answer, and the references to the curated materials used to generate the answer. Note that RAG-Tutor answers not only provide an answer using contextual information but also present the sources of information, indicating the document and the page used to generate the answer.



(**a**) (**b**)

Figure 2. (**a**) Integration of RAG-Tutor with the MOOC interface, with the conversational plug-in on the right, (**b**) Example of a conversation between the student and RAG-Tutor, including the question, the answer, and the references used.

4. Methods and Materials

This section presents the methodology used to implement and evaluate RAG-Tutor. First, the experimental configuration detailing the three architectures considered is presented. Second, the benchmark generation process for the "Introduction to Operating Systems" MOOC is detailed. Third, the metrics used for evaluating the different RAG-Tutor configurations are introduced, and, finally, the student-centered evaluation methodology is presented.

4.1. Experimental Configuration

Three different architectures were used to implement RAG-Tutor: (a) OpenAI, (b) Ollama, and (c) AWS Bedrock. They share a common architecture composed of front-end, middleware, and back-end. The front-end is implemented in Moodle with a custom plug-in in PHP for the question input and feedback display. The middleware contains the core RAG-Tutor functionality, implemented in Python, and acts as the communication layer between Moodle and the back-end. The middleware, implemented in a local server, is a key element in the three architectures: (1) it runs the indexer with the curated information for the course; (2) it stores the vectorial database; (3) it is in charge of the communication between the MOOC and the LLM; (4) it maintains a history of the conversation to provide context to each query; and (5) it stores logs to evaluate and compare architectures.

OpenAI Architecture. In this configuration the back-end of RAG-Tutor is implemented using OpenAI API. After the answer is received from OpenAI by the middleware it is sent to the front-end.

Ollama Architecture. In this configuration the middleware implements all the previous elements and runs an LLM locally using Ollama. Ollama supports open-source LLMs and enables local inference across a wide range of hardware platforms. In our case, Ollama is run on a virtual private server (VPS) with 8 vCPUs, 32 GB RAM, and 320 GB storage. Note that no special hardware, such as GPUs or TPUs, was available. With this architecture, we tested gemma: Gemma 2b and Mistral 7B Instruct v0.2. This implementation provides full privacy, as both the vectorial database and the conversations between the students and the LLM are local.

AWS Bedrock Architecture. The third architecture is based on Amazon Web Services (AWS) Bedrock API. This solution provides a scalable solution by using a public cloud infrastructure that gives access to multiple LLMs. This architecture ensures privacy and regulatory compliance, as all data exchange and model inference are confined within the public cloud, with no data retained or used for training by default. Compared to OpenAI API, AWS Bedrock offers stronger guarantees for privacy because it confines all data processing to a selected cloud region, enabling compliance with data protection requirements. Note that not all AWS regions give support for the same set of LLMs.

Based on the previous architectures, three experimental evaluations were implemented:

- Latency evaluation. The objective of the first experiment was to evaluate if the different architectures satisfy the latency requirements for conversational systems. For OpenAI, GPT-3.5 and GPT-4o were selected because they represent two widely used commercial models with different reasoning capabilities and

computational complexity. For Ollama, gemma 2-2b and Mistral 7B Instruct v0.2 were evaluated because they are lightweight open-source models suitable for local deployment without requiring specialized hardware. For AWS Bedrock, Amazon Nova Lite v1.0 and Amazon Nova Pro were selected to compare low-latency models optimized for real-time interaction against larger reasoning-oriented models. The goal was to identify which approaches provide acceptable latency for a chatbot interaction.

- RAG-Tutor vs. stand-alone LLM comparison. The second experiment compared the performance of RAG-Tutor against stand-alone LLMs. Only the architectures that satisfied the latency requirements identified in the previous experiment were considered. This experiment focused on evaluating the impact of retrieval augmentation on semantic quality using BLEU and BERTScore.
- Extended AWS Bedrock model evaluation. The third experiment expanded the AWS Bedrock evaluation by incorporating additional LLMs: Llama 3.2 1B, Llama 3.3 70B and DeepSeek R1 in addition to the Nova models previously considered. The objective of this experiment was to evaluate whether larger and more complex models improve semantic metrics, and whether such improvements justify the computational cost and latency for MOOC tutoring environments. Llama 3.2 1B is a compact model designed for efficiency. Its inclusion in the study explores how much RAG can compensate for a limited model capacity. In educational contexts where institutions may not have access to large-scale infrastructure, the performance of smaller and cheaper models can be very important. Comparing the 70B model with the 1B version from the same family provides a way to analyze the effect of scale on RAG performance. Finally, DeepSeek R1 is an LLM optimized for reasoning that uses a mixture-of-experts design.

The parameters used for each LLM were the default configurations of each environment and were kept constant. For GPT-3.5 and GPT-4o, temperature and top-p were set to 1.0. For Amazon Nova Lite and Nova Pro, temperature was set to 0.7 and top-p to 0.9. For the Llama models deployed through AWS Bedrock, temperature was set to 0.5 and top-p to 0.9. For DeepSeek R1, the default decoding parameters provided by AWS Bedrock were used.

4.2. Benchmark Design

As an instantiation of RAG-Tutor, we have implemented an "Introduction to Operating Systems" course. The educational resources collected covered various sources and over 1000 pages of PDF documents covering different aspects of operating systems. The data curated was available as part of the course and available in the MOOC interface. Once the course was built, a benchmark was designed to evaluate RAG-Tutor for that specific implementation.

The evaluation of a RAG system is a key step to measure its effectiveness. Compared to the evaluation of LLMs, which can use standard benchmarks, the evaluation of a RAG system needs to build specially designed benchmarks that focus on the corpus with which the RAG was built [35]. The general process for generating a RAG benchmark involves generating a set of questions-answers [36], which implies understanding the curated corpus. This process can be done by experts, by LLMs or by a combination of both. Building a benchmark must consider some elements, such as coverage and the diversity of questions. For building the benchmark, we used a combined approach of LLM and experts with the following steps: (1) for each one of the documents of the corpus, an LLM will be used to generate a set of questions covering the diversity of users; (2) answers will be provided to those questions using an LLM; and (3) the answers will be evaluated by a set of experts. Some RAG benchmarks built using this approach include [37,38]. The construction of the RAG-Tutor benchmark was implemented using two agents: (1) a question-generator agent, which generated questions for three levels of students: beginner, intermediate and advanced; and (2) an answer-generation agent, which provided the output. Each agent was built using CoT reasoning and orchestration between agents was done through APIs. The question-generator agent fed from the PDF reader to generate questions, while the answer-generator agent obtained the questions and produced a csv. Box 2 presents the prompt for the question-generator agent and Box 3 the prompt used for the answer-generation agent.

In both cases the LLM used was OpenAI GPT 3.5 Turbo. After the question-answer pair was generated, two human experts were used to evaluate them on a scale of 1–3, with 1 = not-valid and 3 = correct. Any question-answer pair that did not obtain a 3 by both experts was deleted from the benchmark. The inclusion of questions with different levels of difficulty was introduced to reflect learner variability [39]. Considering the context window of GPT 3.5 of 16k tokens, a subset of the documents covering the core elements of an Operating System course were divided into parts to respect the window size, for a total of 10 parts. Each part was then fed to the agentic architecture for the generation of the questions. Each part generated 27 questions (9 for each level of difficulty) for a total of 270 questions. After the expert evaluation, a total of 196 questions passed the filter: 52 questions were advanced, 65 intermediate, and 79 beginner. Table 1 presents some examples of the selected questions.

Box 2. Prompt used for automatic question generation.

Act as a student enrolled in an "Introduction to Operative Systems" course delivered through a MOOC platform. You will be provided a section of a book to process. Assume the role of a curious first-year Computer Science student who wants to better understand the material. Generate a list of 27 questions based on the content, each with its corresponding answer.

Step 1: Question Design Requirements

Beginner Level (9 questions): Address basic concepts, definitions, or facts that a student might ask upon first exposure to the material.

Intermediate Level (9 questions): Focus on explaining how or why concepts work, exploring more detailed understanding.

Advanced Level (9 questions): Require application, synthesis, or deeper reasoning about the concepts.

Step 2: Output Format

Present the complete set of 27 questions and answers in CSV format.

Box 3. Prompt used for answer generation.

Act as a tutor for the "Introduction to Operative Systems" course delivered through a MOOC platform. You will be provided a file containing a set of questions. Each row in the file includes the question and its difficulty level.

Step 1: Answer Generation

Provide concise and accurate answers for each question.

Use language appropriate for a first-year Computer Science student.

Step 2: Source Attribution

Indicate the page or pages from the book where the answer information was obtained.

Step 3: Output Format

Format the final output as CSV.

Include the following columns:

(1) Question

(2) Answer

(3) Difficulty Level

(4) Page(s)

Table 1. Examples of operating systems questions generated.

Question	Answer	Level
What is the purpose of the shell or GUI?	The shell or GUI is a user interface that allows users to interact with the computer and start programs. It is not part of the operating system but uses it to execute commands.	Beginner
What is the difference between kernel and user mode?	In kernel mode, the CPU can execute any instruction and access any hardware, while in user mode, access is limited to protect system stability and security.	Intermediate
How does time multiplexing help manage CPU usage?	Time multiplexing allows multiple programs to take turns using the CPU, ensuring efficient use by switching when one program waits for I/O operations.	Advanced

Although expert screening removed incorrect question-answer pairs, it cannot fully eliminate biases introduced by the LLM used to generate them. First, the coverage and type of questions reflect what GPT-3.5 Turbo considered relevant in each part of the corpus, which may not match the questions real students would ask. Second, the golden answers follow the phrasing and style of GPT-3.5 Turbo. Since BLEU and BERTScore measure similarity to these answers, models with a similar writing style may be favored. Third, the use of an

OpenAI model could introduce a bias towards OpenAI LLMs. In any case, the results do not seem to support these hypotheses, as the OpenAI models obtained lower quality scores than other models.

4.3. Latency Metric

Latency is a key factor in conversational educational systems because delayed responses can negatively affect student engagement. Prior work in Human-Computer Interaction (HCI) has shown that response times below 1 s are generally perceived as conversational, 1 to 10 s is still acceptable but requires some indication that the answer is being processed, and 10+ seconds leads to students losing attention [40–42]. In this work, latency was defined as the end-to-end response time between the submission of a student question and the total generation of the answer presented to the user. The measurement includes the complete RAG pipeline: retrieval of relevant chunks from the vector database, prompt construction, middleware orchestration, and LLM inference. Latency measurements also include communication overhead with cloud-based APIs when applicable. Latency was measured and stored in the middleware for every benchmark question and every evaluated configuration.

4.4. Quality Metrics

Answer quality for a given benchmark question will compare the answer with the golden answer of the benchmark. Two complementary metrics will be used: BLEU and BERTScore. BLEU (Bilingual Evaluation Understudy) is a metric that determines the quality of a text by identifying how many n-grams there are between the text and the golden answer expressed as a percentage. This metric has many limitations, mainly that is not based on the semantic meaning. Also, it is very sensitive to the length of the answer as the output is expressed as a percentage.

BERTScore is a metric that measures semantic similarity between two texts. It uses embeddings (typically BERT or RoBERTa) to compute token-level similarity using cosine distance and calculates precision, recall, and F1 scores. In this context, precision indicates the accuracy of the candidate answer by capturing how well the response is semantically aligned with the golden answer. Recall measures the completeness of the answer by measuring how well the reference tokens are covered by the candidate. The harmonic mean F1 provides a measure of overall semantic similarity. BERTScore also has some limitations, mainly its dependence on the embedding.

In RAG applications, we generally find moderate to low scores for BLEU and BERTScore, which is caused by the open-ended nature of the responses. In any case, the literature suggests that RAG can enhance quality metrics in tutors. Reference [43] reported a BLEU of 0.28 for a RAG-based document QA system, compared with 0.10 for existing QA applications. Finally, the math tutor presented in reference [44] achieved a BERTScore F1 of 0.51 as a baseline and 0.56 when using RAG.

4.5. Student Evaluation

The student evaluation was designed as a small-scale exploratory pilot to examine usability and students' perceptions of different tutoring configurations. A total of 36 participants took part, of which 30 finished the experiment. These were divided across three settings: (1) 10 students interacting directly with a stand-alone LLM (Llama 3.3 70B) as a tutor; (2) 10 students using RAG-Tutor instantiated with Llama 3.3 70B; and (3) 10 students using RAG-Tutor instantiated with Nova Lite. The Results section will justify why these models were selected as part of the experiment. The stand-alone LLM setting was implemented using the same MOOC-integrated interface as RAG-Tutor. For the two RAG-Tutor settings, all components were identical, allowing the analysis to evaluate the effect of the LLM. Comparisons between the stand-alone and RAG conditions were used to evaluate the RAG impact. The limited sample size (n = 10 per condition) restricts the statistical power and generalizability of the findings.

Participants interacted with the MOOC in the three settings while completing five course-related tasks. They were instructed to use the tutor that they were assigned to clarify doubts or refine understanding, and to cross-check responses with the cited course materials (if they were interacting with RAG-Tutor). Tasks were designed to cover different areas of the course implemented. The evaluation comprised five tasks covering conceptual explanation, comparison, algorithmic reasoning, and applied understanding. Final answers were written by the students after completing the interaction with the tutor. The following tasks were presented to the students:

- Task 1: Explain the differences between processes and threads, using examples. Summarize the differences in a table and give an example.
- Task 2: Describe the Producer-Consumer problem and give an example. Provide pseudocode using semaphores and explain why deadlock is avoided.

- Task 3: Compare indexed vs. linked file allocation and present the results in a table. Justify which is more efficient for sequential vs. random access.
- Task 4: Explain how DMA works compared to programmed I/O.
- Task 5: Explain how virtualization allows multiple operating systems to run on the same hardware. Compare full virtualization vs. paravirtualization.

After completing each task, participants filled in a 5-point Likert-scale post-task questionnaire assessing satisfaction, trust, cognitive effort, and explainability of the tutor's feedback. Table 2 presents the post-task questionnaire. Upon completion of all five tasks, participants completed a final 5-point Likert-scale survey capturing usability. Table 3 presents the final survey questionnaire. All interactions and survey responses were anonymized. No personally identifiable information was collected. All items were answered using a 5-point Likert scale, where 1 is "Strongly Disagree", 2 is "Disagree", 3 is "Neutral", 4 is "Agree", and 5 is "Strongly Agree". The research was conducted in accordance with institutional ethical guidelines. Prior to participation, participants were informed about the purpose of the study, the voluntary nature of their participation and their right to withdraw at any time without consequences. Informed consent to participate in the study was obtained from all participants before data collection. No personally identifiable information was collected, and all responses were analyzed in aggregated and anonymized form. Participants consented to the use of their anonymized responses for research.

Table 2. Post-task questionnaire, including the question code and the question posed to the student.

Code	Question
S1	I am satisfied with the tutor's help on this task.
T1	I trust the tutor's guidance for this task.
E1	The cognitive effort required felt acceptable.
X1	The citations/references provided were adequate for verifying the answer.

Table 3. Final survey questionnaire, including the question code and the question posed to the student.

Code	Question
UM1	This system's capabilities meet my needs.
UM2	This system is easy to use.

Student Evaluation Metrics

The evaluation of each setting was conducted using a set of metrics based on concepts from human–computer interaction and educational technology. The measures reported in this study represent students' self-reported perceptions of the tutoring configurations. Accordingly, they should be interpreted as perceived measures rather than properties of the systems. Given the exploratory nature of the study and the limited number of participants, these measures are intended to characterize students' perceptions of the tutoring experience rather than to establish improvements in learning effectiveness.

Usability (U). Usability is a key requirement for educational technologies, as poorly designed interfaces can negatively affect learning outcomes. Usability was assessed using UMUX-Lite [45]. Let UM1 and UM2 denote the two usability items from the final questionnaire. The normalized (scale 0 to 100) usability score was computed as:

$$\text{UMUX-Lite}_{0-100} = \left(\frac{\left(\frac{\text{UM1} + \text{UM2}}{2} \right) - 1}{4} \right) \times 100 \tag{2}$$

Satisfaction (S). Satisfaction captures the student's affective response to the tutor's support after completing each learning task [46]. Satisfaction was obtained using the average S1 value of the post-task questionnaire for all the tasks and all the students and is presented normalized to a 0 to 100 scale as:

$$\text{Satisfaction}_{0-100} = \left(\frac{\text{S1} - 1}{4} \right) \times 100 \tag{3}$$

Trust (T). Trust is a critical factor in the adoption of AI-based educational tools. It reflects the degree to which students feel confident relying on the tutor's guidance when completing academic tasks [47]. Trust was obtained using the average T1 value of the post-task questionnaire for all the tasks and all the students and is normalized to a 0–100 scale:

$$\text{Trust}_{0-100} = \left(\frac{\text{T1} - 1}{4} \right) \times 100 \tag{4}$$

Perceived Cognitive Load (PCL). Cognitive load, as defined in the NASA Task Load Index (NASA-TLX), indicates the subjective mental workload experienced by a user while performing a task [48]. Managing cognitive load is key for learning, as tools that impose excessive mental effort may limit their educational value. Perceived cognitive load was assessed using the average value of E1 from the post-task questionnaire. E1 is aligned with the Mental Demand and Effort dimensions of NASA-TLX. The use of a single-item proxy is supported by previous studies [48]. Therefore, PCL is interpreted in this study as a task-level proxy for perceived cognitive effort rather than as a validated multidimensional measure of cognitive load. The resulting PCL score was inverted and normalized to a 0–100 scale, where higher values indicate higher perceived cognitive load:

$$\mathrm{PCL}_{0-100} = \left(1 - \frac{\mathrm{E1} - 1}{4}\right) \times 100 \tag{5}$$

Explainability (E). Explainability is a key requirement for responsible AI in education and is especially relevant for RAG-based systems. In this context, explainability refers to the extent to which learners can understand and verify why a given response was produced [49]. Perceived explainability was measured using the average value of X1 of the post-task questionnaire for all tasks and all students and normalized to a 0–100 scale:

$$\mathrm{Explainability}_{0-100} = \left(\frac{\mathrm{X1} - 1}{4}\right) \times 100 \tag{6}$$

4.6. Statistical Methods

Different statistical tests will be used to evaluate if the differences across the different RAG-Tutor configurations are statistically significant. The distributions of the metrics will first be analyzed using the Shapiro-Wilk test for normality. If the data satisfies the assumption of normality, t-tests will be used for paired comparisons, and one-way ANOVA will be used for comparisons involving more than two groups. When ANOVA indicates statistically significant differences, post-hoc pairwise comparisons will be conducted using Tukey's Honest Significant Difference (HSD) test. If normality assumptions are not valid, Wilcoxon signed-rank tests will be applied for paired comparisons, while Kruskal-Wallis tests will be used for multiple-group comparisons. To control the Type-I error rate during pairwise comparisons, *p*-values will be adjusted using the Holm-Bonferroni procedure. For all analyses, statistical significance will be established at $\alpha = 0.05$. Because statistical significance alone does not indicate the relevance of the observed differences, effect sizes will also be computed. For parametric paired *t*-tests, Cohen's d_z will be used, and for non-parametric Wilcoxon signed-rank tests.

5. Results & Analysis

This section presents the experimental evaluation of RAG-Tutor across: (1) latency performance; (2) RAG-Tutor comparison against stand-alone LLM; (3) evaluation of additional AWS Bedrock models to evaluate the trade-offs between latency, model complexity and answer quality and (4) student evaluation results.

5.1. RAG-Tutor Latency

Table 4 presents latency metrics in seconds and corresponding 95% confidence intervals for each configuration of RAG-Tutor. Focusing on OpenAI, GPT-3.5 and GPT-4o, obtained 1.34 ± 0.41 s and 4.05 ± 1.12 s. For Ollama, Gemma 2-2b and Mistral 7B Instruct v0.2 produced the highest latencies with 29.43 ± 8.75s and 66.94 ± 19.21 s. For AWS Bedrock, Nova Lite v1:0 had an average latency of 0.88 ± 0.24 s while Nova Pro obtained 4.55 ± 1.18 s. Cloud-based APIs showed stable and low latency due to their infrastructure. Larger reasoning-oriented models, such as GPT-4o and Nova Pro, presented variance associated with their complexity. In contrast, Ollama exhibited higher values, mainly caused by the lack of specialized hardware.

Table 4. RAG-Tutor Latency Performance across Configurations.

Architecture	Model	Mean ± SD	95% CI
OpenAI	GPT-3.5	1.34 ± 0.41	[1.28, 1.40]
	GPT-4o	4.05 ± 1.12	[3.89, 4.21]
Bedrock	Nova Lite v1.0	0.88 ± 0.24	[0.85, 0.91]
	Nova Pro	4.55 ± 1.18	[4.38, 4.63]
Ollama	Gemma 2-2B	29.43 ± 8.75	[28.20, 30.66]
	Mistral 7B	66.94 ± 19.21	[64.25, 69.63]

Shapiro-Wilk tests indicated that latency distributions deviated from normality ($p < 0.05$). Wilcoxon signed-rank tests were used to calculate pairwise comparisons and Holm-Bonferroni was applied to control error rate. Latency comparisons were conducted at model level using paired observations where each latency value obtained for a given question and model was paired with the corresponding latency obtained for the same question with another model. Pairwise comparisons were performed across all models. The resulting corrected *p*-values showed statistically significant latency differences for all configurations, especially between Ollama and cloud-based solutions ($p < 0.01$). In addition, effect size analysis revealed large effects ($r > 0.5$), indicating that the observed latency differences were not only statistically significant but also meaningful. The results show that Ollama architecture, as implemented (i.e., no specialized hardware), does not provide the necessary latency for keeping students engaged. OpenAI and AWS provide acceptable latency values, but they depend on the complexity of the LLM used. The rest of the evaluation will focus just on these two architectures.

5.2. RAG-Tutor vs. Stand-Alone LLM

To measure the effect of RAG-Tutor vs. stand-alone LLMs, we compared the metrics of both approaches using the benchmark and the same set of four LLMs: GPT 3.5, GPT 4o, Nova Lite and Nova Pro. The stand-alone approach of the tutor is direct: each question of the benchmark is sent to the corresponding LLM to obtain the answer; there is no retrieval of information. The prompt used in this case is the same one presented in Box 1, but without any reference to the retrieved context and the history.

Table 5 presents the results of RAG-Tutor for the LLMs considered. The table reports average values for BLEU and BERTScore (F1, precision, recall). Latency values for these models were presented in Table 4. Table 6 presents the same results, including latency, for the stand-alone configuration. The normality of the distributions for latency, BLEU, and BERTScore F1 was evaluated using Shapiro–Wilk. Latency, again, showed significant deviations from normality. BLEU and BERTScore F1 distributions were considered normal and analyzed using t-tests with the Holm-Bonferroni correction applied to all pairwise metric comparisons.

Table 5. Comparative results for RAG-Tutor considering BLEU and BERTScore metrics.

RAG	BLEU	BERTScore
GPT 3.5	0.082 ± 0.021	F1: 0.36 ± 0.05 *p*: 0.29 ± 0.04; R: 0.48 ± 0.06
GPT 4o	0.092 ± 0.025	F1: 0.37 ± 0.06 *p*: 0.29 ± 0.05; R: 0.51 ± 0.07
Nova Lite	Avg: 0.12 ± 0.028	F1: 0.45 ± 0.05 *p*: 0.38 ± 0.04; R: 0.52 ± 0.05
Nova Pro	0.13 ± 0.031	F1: 0.44 ± 0.06 *p*: 0.38 ± 0.05; R: 0.53 ± 0.06

Table 6. Comparative results of Stand-alone LLMs for latency, BLEU, and BERTScore metrics.

Stand-Alone	Latency (s)	BLEU	BERTScore
GPT 3.5	1.20 ± 0.31	0.07 ± 0.028	F1: 0.35 ± 0.07 *p*: 0.30 ± 0.06; R: 0.42 ± 0.08
GPT 4o	3.57 ± 0.92	0.069 ± 0.030	F1: 0.33 ± 0.08 *p*: 0.25 ± 0.07; R: 0.48 ± 0.09
Nova Lite	0.70 ± 0.18	0.10 ± 0.026	F1: 0.40 ± 0.06 *p*: 0.37 ± 0.05; R: 0.44 ± 0.07
Nova Pro	4.11 ± 0.98	0.10 ± 0.029	F1: 0.39 ± 0.07 *p*: 0.33 ± 0.06; R: 0.48 ± 0.08

Focusing on latency, the use of RAG architectures produced an increase for all LLMs when compared with their stand-alone configurations. This is caused by the need to retrieve the information, rank it, and build and process an extended prompt. For OpenAI GPT 3.5, the average latency increased from 1.20 ± 0.31 s to 1.34 ± 0.41 s, representing an increase of approximately 11.7%. Similarly, GPT-4o increased from 3.57 ± 0.92 s to 4.05 ± 1.12 s, corresponding to a 13.4% increase in response time. In the AWS Bedrock, Nova Lite increased from 0.70 ± 0.18 s to 0.88 ± 0.24 s (25.7%); and Nova Pro from 4.11 ± 0.98 s to 4.55 ± 1.18 s (10.7%). The latency increases introduced by RAG were statistically significant for all models ($p < 0.05$). The additional latency introduced by RAG-Tutor was 0.14 s for GPT-3.5 (10.4% of the total latency), 0.18 s for Nova Lite (20.5%), 0.44 s for Nova Pro (9.7%) and 0.48 s for GPT-4o (11.9%). This overhead includes retrieval, prompt construction, and the processing by the LLM. In all

configurations, the total latency is dominated by the LLM, and the overhead introduced by RAG is small compared with the differences between architectures. Focusing on BLEU and BERTScore F1, Table 7 presents the changes of the metrics between the two approaches. The statistical analysis showed that the improvements observed were statistically significant ($p < 0.05$) for all cases. When compared to the stand-alone implementation, RAG-Tutor improved BLEU and BERTScore F1 for all the LLMs, with BLEU improved in the range of 17% to 33% and BERTScore F1 improved by 3–13%. Metric improvement is similar within the same family of models, and the final improved metrics are generally comparable within each family of LLMs. That is, smaller models seem to perform in the same range as complex models of the same family. This is probably caused because, in a RAG architecture, the knowledge is provided by the vectorial database, so the LLM used does not need to be complex, just provide capabilities of aggregating language. RAG-Tutor improves BLEU and BERTScore F1 across models with gains in line with the results presented in the literature.

Finally, in order to evaluate if the improvements are meaningful regardless of the dataset size, we used Cohen's d_z for paired samples. Cohen's d_z expresses the mean paired difference between two conditions in units of the standard deviation of those differences and therefore measures both the size and the consistency of an improvement, independently of the metric's scale and the sample size. This measure is particularly relevant in our context. With 196 benchmark questions, even small differences can reach statistical significance. BLEU and BERTScore have low absolute values and different scales, so raw differences are hard to interpret and compare. Moreover, since students receive answers one question at a time, the consistency of the improvement across questions matters as much as its average size, and d_z captures both. A value of $d_z = 0$ indicates no difference, a negative value indicates that the stand-alone LLM performs better, and values of about 0.2, 0.5 and 0.8 are considered small, medium and large effects, respectively.

Table 7. Improvements in BLEU and BERTScore F1 when comparing RAG-Tutor with stand-alone approaches.

Model	Δ BLEU	Δ BERTScore F1
GPT 3.5	+0.012 (+17.1%) $p = 0.018$	+0.01 (+3.0%) $p = 0.041$
GPT 4o	+0.023 (+33.3%) $p < 0.001$	+0.04 (+12.1%) $p = 0.004$
Nova Lite	+0.02 (+20.0%) $p = 0.006$	+0.05 (+12.5%) $p = 0.002$
Nova Pro	+0.03 (+30.0%) $p = 0.008$	+0.05 (+12.8%) $p < 0.001$

Because each benchmark question was evaluated under both RAG-Tutor and stand-alone configurations, the analysis used paired observations, where the difference between the scores obtained by both approaches was computed for each question. Cohen's d_z was then calculated as the mean of the paired differences divided by the standard deviation of those differences. Table 8 shows that the size of the improvement depends on both the LLM and the metric. For BLEU, retrieval produced medium effects for GPT-4o, Nova Lite and Nova Pro, and a small effect for GPT-3.5. For BERTScore F1, the effects ranged from small (GPT-3.5 and Nova Lite) to medium (Nova Pro).

Table 8. Effect size for the improvement of RAG vs. stand-alone.

LLM	BLEU d_z	BERTScore F1 d_z
OpenAI GPT 3.5	0.25	0.15
OpenAI GPT 4o	0.48	0.42
Amazon Nova Lite	0.52	0.45
Amazon Nova Pro	0.49	0.60

5.3. AWS Architecture Evaluation

In this section, we evaluate more LLMs within the AWS architecture to identify one that optimizes our metrics. Three extra models were chosen: Llama 3.2 1B, Llama 3.3 70B, and DeepSeek R1. Table 9 presents RAG-Tutor performance for the three new models, in bold, and all the models previously tested. To provide an ordering of the evaluated LLMs, a normalized score was defined using BLEU, BERTScore F1, and latency values. Since these metrics operate on different numerical scales, min-max normalization was applied before aggregation. Let B_i, F_i, and L_i represent the mean BLEU score, mean BERTScore F1, and mean Latency obtained by the i-th model over the 196 benchmark questions. The normalized score S_i of the i-th model is defined as:

$$S_i = w_B \cdot \frac{B_i - B_{min}}{B_{max} - B_{min}} + w_F \cdot \frac{F_i - F_{min}}{F_{max} - F_{min}} + w_L \cdot \frac{L_{max} - L_i}{L_{max} - L_{min}} \quad (7)$$

where B_{min} and B_{max} denote the minimum and maximum values observed across the dataset (with similar notation applied to F and L). A sensitivity analysis was conducted by recomputing the score under four scenarios: (1) an

equal-weight scenario ($w_B = 0.33$, $w_F = 0.33$, $w_L = 0.33$); (2) a quality-oriented scenario ($w_B = 0.40$, $w_F = 0.40$, $w_L = 0.20$), which gives greater importance to answer quality; (3) a semantic-oriented scenario ($w_B = 0.20$, $w_F = 0.60$, $w_L = 0.20$), which focusses on semantic similarity as measured by BERTScore F1; and (4) a latency-oriented scenario ($w_B = 0.20$, $w_F = 0.20$, $w_L = 0.60$), representing deployment settings in which response time is the main consideration. The sensitivity analysis shows that the model ranking depends to some extent on the relative importance assigned to answer quality and latency. Llama 3.3 70B ranked first under the equal-weight, quality-oriented, and semantic-oriented scenarios, whereas Nova Lite ranked first when latency was given the highest weight. Nevertheless, the two models consistently occupied the top positions across the four scenarios. This result supports their selection for the subsequent student evaluation: Llama 3.3 70B represents the configuration prioritizing answer quality, whereas Nova Lite represents a configuration providing a stronger quality–latency trade-off. The sensitivity analysis indicates that the selection of these two models is robust, even though the identity of the highest-ranked model depends on the weighting assumptions.

Table 9. Ranking of the evaluated RAG-based LLMs using the composite score S_i under the four weighting scenarios: equal ($w_B = w_F = w_L = 1/3$), quality-oriented (0.40, 0.40, 0.20), semantic-oriented (0.20, 0.60, 0.20) and latency-oriented (0.20, 0.20, 0.60). Models added in the extended AWS evaluation are shown in bold.

Model	Latency (s)	BLEU	BERTScore F1	S_i Equal	S_i Quality	S_i Semantic	S_i Latency
Llama 3.3 70B	**6.87 ± 1.74**	**0.240**	**0.46**	**0.88**	**0.93**	**0.93**	**0.78**
Nova Lite	0.88 ± 0.24	0.120	0.45	0.78	0.74	0.85	0.87
Nova Pro	4.55 ± 1.18	0.130	0.44	0.71	0.69	0.79	0.73
GPT 3.5	1.34 ± 0.41	0.082	0.36	0.58	0.50	0.57	0.73
GPT 4o	4.05 ± 1.12	0.092	0.37	0.55	0.50	0.57	0.65
Llama 3.2 1B	**2.48 ± 0.62**	**0.130**	**0.26**	**0.49**	**0.41**	**0.35**	**0.65**
DeepSeek R1	**16.90 ± 3.85**	**0.044**	**0.23**	**0.00**	**0.00**	**0.00**	**0.00**

5.4. Student Analysis

Table 10 presents the results of the three student evaluation settings: (1) RAG-Tutor instantiated with Nova Lite, (2) RAG-Tutor instantiated with Llama 3.3 70B, and (3) stand-alone with Llama 3.3 70B. Llama 3.3 70B was used for the stand-alone condition so that it could be compared with the RAG-Tutor configuration based on the same model. This comparison isolates the effect of retrieval augmentation, as both conditions share the same LLM and interface. The values presented in the table correspond to the mean and standard deviation of each metric across participants in each setting. ANOVA was used and the resulting *p*-values indicate if at least one of the distributions differed significantly from the others. The "Significant Differences" column reports only those comparisons that were statistically significant following HSD. For example, entries such as "Llama > Nova" indicate that the RAG-Tutor instantiated with Llama 3.3 70B achieved statistically significant higher scores than Nova Lite for that metric.

Table 10. Comparative user-experience evaluation of 3 system configurations, for usability, satisfaction, trust, perceived cognitive load and explainability.

	Nova Lite (RAG-Tutor)	Llama 3.3 70B (RAG-Tutor)	Llama 3.3 70B (Stand-Alone)	*p*	Differences
UMUX-Lite	78.75 ± 17.73	92.50 ± 6.46	82.50 ± 8.90	0.044	Llama> Nova Llama > stand-alone
Satisfaction	72.50 ± 14.39	92.00 ± 6.75	88.00 ± 7.60	<0.001	Llama > Nova stand-alone > Nova
Trust	67.00 ± 16.19	91.80 ± 4.97	65.50 ± 12.10	<0.001	Llama > stand-alone Llama > Nova
PCL	32.50 ± 14.95	14.50 ± 8.64	28.50 ± 10.30	0.005	Llama < Nova Llama < stand-alone
Explainability	67.50 ± 14.39	64.00 ± 11.97	41.00 ± 13.80	<0.001	RAG > stand-alone

Comparing RAG-Tutor vs. stand-alone LLM. The results of Table 10 indicate that Explainability showed the clearest effect of RAG-Tutor ($p < 0.001$), with both RAG configurations outperforming the stand-alone setting, indicating the importance of citations and contextualization of answers in course materials. Also, Trust scores for the Llama-based RAG-Tutor configuration (91.8, 95% CI [88.3, 95.4]) were higher than those of the stand-alone configuration (65.50, 95% CI [57.8, 72.8]). Trust in the Nova Lite configuration (67.0) was similar to that of the

stand-alone setting, which indicates that retrieval alone did not increase trust. Differences were also observed in PCL: although the interface of the stand-alone condition indicated low effort, students interacting with the Llama-based RAG tutor reported even lower PCL. However, given the limited sample size, these results should be considered preliminary and specific to the evaluated configurations.

These results are aligned with prior work showing that RAG approaches outperform stand-alone LLM tutors in terms of accuracy and interpretability [13,14]. Focusing on usability and satisfaction, the differences between Llama-RAG and the stand-alone setting were not significant, whereas the Nova Lite configuration obtained significantly lower satisfaction than the stand-alone setting. Students perceived the stand-alone solution as easy to use and helpful, but not fully reliable, suggesting that usability alone does not guarantee pedagogical value, in line with the findings reported in [50].

Comparing the Two RAG-Tutor Approaches. Within the RAG-Tutor settings, Table 10 shows that the choice of LLM influences student experience. For almost all metrics, Llama 3.3 70B outperforms Nova Lite. Llama based RAG-Tutor achieved higher trust values and lower PCL than Nova Lite, suggesting that larger LLMs are better able to synthesize retrieved content into coherent explanations. Nevertheless, the results are based on only 10 participants per configuration and should not be generalized to other models or courses. Furthermore, because the models differ in several architectural characteristics, the observed differences cannot be attributed only to model size.

The combination of these results show that: (1) when the same LLM is used, RAG-Tutor improves perceived explainability, trust and cognitive effort over the stand-alone chatbot, without significant differences in usability or satisfaction; and (2) within RAG-Tutor, the choice of generator affects student experience, with Llama 3.3 70B outperforming Nova Lite across most metrics. In other words, a RAG-based approach by itself does not guarantee better performance than a stand-alone LLM; rather, the combination of retrieval augmentation with a suitable LLM defines the outcomes. Finally, the evaluation has several limitations that should be considered, mainly: (1) the limited number of students per condition, (2) Llama 3.3 70B, and Nova Lite are very different in scale and architecture, so the comparisons have to be considered in the context of our experiments, and (3) student-centered metrics are based on students' self-reported perceptions, measured with single items rather than validated instruments. As a result, statistical findings should be interpreted as exploratory in the context of RAG-Tutor.

5.5. Cost Analysis

The deployment of RAG-Tutor depends on commercial platforms with operational costs. Table 11 presents the costs for executing the benchmark. The values were computed with an average input size of 6500 tokens per query (which includes the static prompt plus the retrieved contextual information of 3 × 2000 tokens) together with an average generated response length of 100 output tokens. This resulted in a total processing volume of approximately 1,274,000 input tokens and 19,600 output tokens. For each model, the total execution cost was calculated using the input and output token prices provided by AWS Bedrock and OpenAI for November 2025. The final column normalizes the cost to 1000 student queries. The comparison between Nova Lite and Llama 3.3 70B presents the trade-offs in deploying RAG-Tutor. Although the 70B model produced higher semantic quality and user satisfaction, Nova Lite, with a cost 30 times smaller, provided the best balance between latency, cost and answer quality, at the price of lower student satisfaction and trust.

Table 11. Benchmark execution costs in USD.

Model	Input/Output	Total Cost	Cost/1000
Amazon Nova Lite	0.07644/0.00470	0.08114	0.41398
Amazon Nova Pro	1.01920/0.06272	1.08192	5.52000
Llama 3.2 1B	0.12740/0.00196	0.12936	0.66000
Llama 3.3 70B	2.54800/0.03920	2.58720	13.20000
DeepSeek-R1	1.71990/0.10584	1.82574	9.31500
GPT-4o	3.18500/0.19600	3.38100	17.25000
GPT-3.5 Turbo	0.63700/0.02940	0.66640	3.40000

To estimate scalability, consider a MOOC with 1000 active students generating an average of 20 tutoring interactions per student per day. Under this scenario, the system would process approximately 600 k monthly queries. The estimated monthly inference cost would range from approximately $248 for Nova Lite to more than $10,000 for OpenAI GPT-4o. Similarly, Llama 3.3 70B would require approximately $7920 per month, while Nova Pro would require approximately $3312. Considering these numbers, local GPU-based infrastructures may become effective if high-volume usage or privacy-sensitive scenarios are relevant. In the end, the optimal configuration depends on the deployment scenario. These results should not be interpreted as evidence that a

lower-cost configuration provides equivalent educational support. Establishing such equivalence would require additional evaluations involving instructional quality and objective learning outcomes.

6. Conclusions & Future Directions

MOOCs have had a great impact on access to education, but its users still face various problems, mainly the unavailability of real-time and contextual support, which limits student involvement and retention. This paper addressed the problem of providing scalable and context-aware tutoring support for MOOCs. Although LLMs can generate explanations, their use as stand-alone tutors is limited by generic responses and lack of alignment with course materials. These limitations are especially relevant in educational contexts, where students require pedagogically useful guidance.

To address these limitations, we designed RAG-Tutor, a Retrieval-Augmented Generation architecture integrated into a MOOC environment. The system allows instructors to upload curated course materials, which are then used to ground student interactions. The architecture was implemented using different deployment configurations and was evaluated in an "Introduction to Operating Systems" MOOC using a benchmark of 196 expert-validated questions.

Focusing on the answer quality, RAG-Tutor outperforms stand-alone LLM tutors. Across the evaluated models, RAG-Tutor improved BLEU by 17–33% and BERTScore F1 by 3–13%, with statistically significant gains. The results also show RAG-Tutor increases latency but remains compatible with conversational interaction. Within the AWS architecture, Llama 3.3 70B achieved the highest answer quality, while Nova Lite provided the best trade-off between quality, latency and cost.

The exploratory student evaluation showed that both RAG-Tutor configurations improved perceived explainability over the stand-alone solution, while improvements in trust and perceived cognitive effort were observed only with Llama 3.3 70B. The results suggest that RAG may reduce the need for extremely large LLMs in educational tutoring systems, enabling lower-cost and lower-latency deployments while maintaining semantic quality and positive student perception. However, the limited sample size, reliance on self-reported measures, and single-course setting restrict the generalizability of these observations. Importantly, the study did not measure learning gains or course completion, and therefore provides no evidence of educational effectiveness.

Future research should focus on three main directions: (1) first, larger-scale and longitudinal evaluations are needed to determine whether RAG-Tutor may support learning outcomes, retention and engagement; (2) algorithmic improvements to the RAG pipeline should be evaluated, including agentic and adaptive RAG architectures and multimodal retrieval; and (3) the current indexer discards figures and other non-textual elements, so vision-based retrieval would allow RAG-Tutor to use diagrams and images from the course materials.

Author Contributions

E.F.-M.: Writing—original draft, Writing—review & editing, Conceptualization, Methodology, Investigation; J.C.S.J.: Conceptualization, Data curation, Formal analysis, Software, Resources; D.L.: Writing—review & editing, Supervision; D.B.: Writing—review & editing, Supervision, Project administration. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Institutional Review Board Statement

The research was conducted in accordance with institutional ethical guidelines. According to the policies of the Institutional Review Board (IRB) of UNIR, studies based on voluntary and anonymous questionnaires that do not collect identifiable or personal data are exempt from formal ethical approval. Prior to completing the questionnaire, participants were informed about the purpose of the study, the voluntary nature of their participation, and their right to withdraw at any time without consequences.

Informed Consent Statement

Informed consent to participate in the study was obtained from all participants before data collection. No personally identifiable information was collected, and all responses were analyzed in aggregated and anonymized form. Participants consented to the use of their anonymized responses for research and publication purposes.

Data Availability Statement

Dataset and code will be made available upon request. Contact the corresponding author.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies

The authors acknowledge the use of ChatGPT 5.6 (OpenAI) for improving text and comprehension. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

1. Deshpande, A.; Chukhlomin, V. What Makes a Good MOOC: A Field Study of Factors Impacting Student Motivation to Learn. *Am. J. Distance Educ.* **2017**, *31*, 275–293. https://doi.org/10.1080/08923647.2017.1377513.
2. Narayanasamy, S.K.; Elçi, A. An Effective Prediction Model for Online Course Dropout Rate. *Int. J. Distance Educ. Technol.* **2020**, *18*, 94–110. https://doi.org/10.4018/ijdet.2020100106.
3. Badali, M.; Hatami, J.; Banihashem, S.K.; et al. The role of motivation in MOOCs' retention rates: A systematic literature review. *Res. Pract. Technol. Enhanc. Learn.* **2022**, *17*, 5. https://doi.org/10.1186/s41039-022-00181-3.
4. Utunen, H.; Appuhamy, R.; Attias, M.; et al. Observations from three years of online pandemic learning response on OpenWHO. *Int. J. Inf. Learn. Technol.* **2023**, *40*, 527–540. https://doi.org/10.1108/ijilt-12-2022-0220.
5. Holmes, W.; Bialik, M.; Fadel, C. *Artificial Intelligence in Education: Promises and Implications for Teaching and Learning*; Center for Curriculum Redesign: Boston, MA, USA, 2019.
6. Aleven, V. Rule-Based Cognitive Modeling for Intelligent Tutoring Systems. In *Advances in Intelligent Tutoring Systems*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 308, pp. 33–62. https://doi.org/10.1007/978-3-642-14363-2_3.
7. Karim Barbhuiya, R.; Mustafa, K.; Jabin, S. A Personalized Learning System with Adaptive Content Presentation and Affective Evaluation Facilities. *Int. J. Comput. Appl.* **2013**, *70*, 10–15. https://doi.org/10.5120/12230-8360.
8. Vaswani, A.; Shazeer, N.; Parmar, N.; et al. Attention is All you Need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
9. Wu, S.; Xiong, Y.; Cui, Y.; et al. Retrieval-augmented generation for natural language processing: A survey. *Artif. Intell. Rev.* **2026**, *59*, 192. https://doi.org/10.1007/s10462-026-11605-7.
10. Lewis, P.; Perez, E.; Piktus, A.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proceedings of the Neural Information Processing Systems, Virtual, 6–12 December 2020; Volume 33, pp. 9459–9474.
11. Kumar, H.; Rothschild, D.M.; Goldstein, D.G.; et al. Math Education with Large Language Models: Peril or Promise? In *Artificial Intelligence in Education*; Springer Nature: Cham, Switzerland, 2025; Volume 15880, pp. 60–75. https://doi.org/10.1007/978-3-031-98459-4_5.
12. Kasneci, E.; Seßler, K.; Küchemann, S.; et al. ChatGPT for good? On opportunities and challenges of large language models for education. *Learn. Individ. Differ.* **2023**, *103*, 102274. https://doi.org/10.1016/j.lindif.2023.102274.
13. Miladi, F.; Psyché, V.; Lemire, D. Comparative Performance of GPT-4, RAG-Augmented GPT-4, and Students in MOOCs. In *Breaking Barriers with Generative Intelligence. Using GI to Improve Human Education and Well-Being, Proceedings of the First International Conference on Breaking Barriers with Generative Intelligence (BBGI 2024), Thessaloniki, Greece, 10 June 2024*; Springer Nature: Cham, Switzerland, 2024; pp. 81–92. https://doi.org/10.1007/978-3-031-65996-6_7.
14. Posedaru, B.S.; Pantelimon, F.V.; Dulgheru, M.N.; et al. Artificial Intelligence Text Processing Using Retrieval-Augmented Generation: Applications in Business and Education Fields. *Proc. Int. Conf. Bus. Excell.* **2024**, *18*, 209–222. https://doi.org/10.2478/picbe-2024-0018.
15. Han, Z.F.; Lin, J.; Gurung, A.; et al. Improving Assessment of Tutoring Practices Using Retrieval-Augmented Generation. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 20–27 February 2024; Volume 257, pp. 66–76.
16. Levonian, Z.; Li, C.; Zhu, W.; et al. Retrieval-Augmented Generation to Improve Math Question-Answering: Trade-Offs Between Groundedness and Human Preference. In Proceedings of the 17th International Conference on Educational Data Mining (EDM 2024), Atlanta, GA, USA, 14–17 July 2024; pp. 210–221.
17. Németh, R.; Tátrai, A.; Szabó, M.; et al. Using a RAG-enhanced large language model in a virtual teaching assistant role: Experiences from a pilot project in statistics education. *Hung. Stat. Rev.* **2024**, *7*, 3–27. https://doi.org/10.35618/hsr2024.02.en003.

18. Golla, F. Enhancing Student Engagement Through AI-Powered Educational Chatbots: A Retrieval-Augmented Generation Approach. In Proceedings of the 2024 21st International Conference on Information Technology Based Higher Education and Training (ITHET), Paris, France, 6–8 November 2024; pp. 1–6. https://doi.org/10.1109/ithet61869.2024.10837678.
19. Slade, J.J.; Hyk, A.; Gurung, R.A.R. Transforming Learning: Assessing the Efficacy of a Retrieval-Augmented Generation System as a Tutor for Introductory Psychology. *Proc. Hum. Factors Ergon. Soc. Annu. Meet.* **2024**, *68*, 1827–1830. https://doi.org/10.1177/10711813241275509.
20. Thüs, D.; Malone, S.; Brünken, R. Exploring generative AI in higher education: A RAG system to enhance student engagement with scientific literature. *Front. Psychol.* **2024**, *15*, 1474892. https://doi.org/10.3389/fpsyg.2024.1474892.
21. Neumann, A.T.; Yin, Y.; Sowe, S.; et al. An LLM-Driven Chatbot in Higher Education for Databases and Information Systems. *IEEE Trans. Educ.* **2025**, *68*, 103–116. https://doi.org/10.1109/te.2024.3467912.
22. Chen, W.; Hu, H.; Chen, X.; et al. MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 5558–5570. https://doi.org/10.18653/v1/2022.emnlp-main.375.
23. Yasunaga, M.; Aghajanyan, A.; Shi, W.; et al. Retrieval-Augmented Multimodal Language Modeling. In Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023; Volume 202, pp. 39755–39769.
24. Radford, A.; Kim, J.W.; Hallacy, C.; et al. Learning Transferable Visual Models from Natural Language Supervision. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; Volume 139, pp. 8748–8763.
25. Shao, P.; Yang, Y.; Gao, C.; et al. Exploring Heterogeneity and Uncertainty for Graph-based Cognitive Diagnosis Models in Intelligent Education. In Proceedings of the KDD'25: The 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Toronto, ON, Canada, 3–7 August 2025; pp. 1233–1243. https://doi.org/10.1145/3690624.3709264.
26. Lyu, Y.; Li, Z.; Niu, S.; et al. CRUD-RAG: A Comprehensive Chinese Benchmark for Retrieval-Augmented Generation of Large Language Models. *ACM Trans. Inf. Syst.* **2025**, *43*, 41. https://doi.org/10.1145/3701228.
27. Stäbler, M.; Turnbull, S.; Müller, T.; et al. The Impact of Chunking Strategies on Domain-Specific Information Retrieval in RAG Systems. In Proceedings of the 2025 IEEE International Conference on Omni-Layer Intelligent Systems (COINS), Madison, WI, USA, 4–6 August 2025; pp. 1–6. https://doi.org/10.1109/coins65080.2025.11125724.
28. Moreno-Cediel, A.; Garcia-Lopez, E.; Garcia-Cabot, A.; et al. Optimising retrieval performance in RAG systems: A new growing window semantic chunking strategy to address weak semantic boundaries. *Knowl.-Based Syst.* **2026**, *331*, 114896. https://doi.org/10.1016/j.knosys.2025.114896.
29. Xia, L.; Xu, J.; Lan, Y.; et al. Learning Maximal Marginal Relevance Model via Directly Optimizing Diversity Evaluation Measures. In Proceedings of the SIGIR'15: The 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, 9–13 August 2015; pp. 113–122. https://doi.org/10.1145/2766462.2767710.
30. LangChain Reference. maximal_marginal_relevance. Available online: https://reference.langchain.com/python/langchain-core/vectorstores/utils/maximal_marginal_relevance (accessed on 10 February 2026).
31. Carbonell, J.; Goldstein, J. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In Proceedings of the SIGIR98: 21st Annual ACM/SIGIR International Conference on Research and Development in Information Retrieval, Melbourne, VIC, Australia, 24–28 August 1998; pp. 335–336. https://doi.org/10.1145/290941.291025.
32. Liu, P.; Yuan, W.; Fu, J.; et al. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* **2023**, *55*, 1–35. https://doi.org/10.1145/3560815.
33. Brown, T.B.; Mann, B.; Ryder, N.; et al. Language Models are Few-Shot Learners. In Proceedings of the Neural Information Processing Systems, Virtual, 6–12 December 2020; Volume 33, pp. 1877–1901.
34. Wei, J.; Wang, X.; Schuurmans, D.; et al. Chain-Of-Thought Prompting Elicits Reasoning in Large Language Models. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; Volume 35, pp. 24824–24837. https://doi.org/10.52202/068431-1800.
35. Chen, J.; Lin, H.; Han, X.; et al. Benchmarking Large Language Models in Retrieval-Augmented Generation. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 20–27 February 2024; Volume 38, pp. 17754–17762. https://doi.org/10.1609/aaai.v38i16.29728.
36. Filice, S.; Horowitz, G.; Carmel, D.; et al. Generating Q&A Benchmarks for RAG Evaluation in Enterprise Settings. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics, Vienna, Austria, 27 July–1 August 2025; Volume 6, pp. 469–484. https://doi.org/10.18653/v1/2025.acl-industry.33.
37. Bonifacio, L.; Abonizio, H.; Fadaee, M.; et al. InPars: Unsupervised Dataset Generation for Information Retrieval. In Proceedings of the SIGIR'22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, 11–15 July 2022; pp. 2387–2392. https://doi.org/10.1145/3477495.3531863.
38. Es, S.; James, J.; Espinosa Anke, L.; et al. RAGAs: Automated Evaluation of Retrieval Augmented Generation. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, St. Julians, Malta, 17–22 March 2024; pp. 150–158. https://doi.org/10.18653/v1/2024.eacl-demo.16.

39. Koedinger, K.R.; Corbett, A.T.; Perfetti, C. The Knowledge-Learning-Instruction Framework: Bridging the Science-Practice Chasm to Enhance Robust Student Learning. *Cogn. Sci.* **2012**, *36*, 757–798. https://doi.org/10.1111/j.1551-6709.2012.01245.x.
40. Tan, F.F.Y.; Messerschmidt, M.A.; Yin, W.; et al. The Impact of Response Latency and Task Type on Human-LLM Interaction and Perception. In Proceedings of the CHI 2026: CHI Conference on Human Factors in Computing Systems, Barcelona, Spain, 13–17 April 2026; pp. 1–17. https://doi.org/10.1145/3772318.3790716.
41. Gnewuch, U.; Morana, S.; Adam, M.T.P.; et al. Opposing Effects of Response Time in Human–Chatbot Interaction. *Bus. Inf. Syst. Eng.* **2022**, *64*, 773–791. https://doi.org/10.1007/s12599-022-00755-x.
42. Nielsen, J. *Usability Engineering*; Academic Press: Boston, MA, USA, 1993. https://doi.org/10.1016/c2009-0-21512-1.
43. Muludi, K.; Fitria, K.M.; Triloka, J.; et al. Retrieval-Augmented Generation Approach: Document Question Answering using Large Language Model. *Int. J. Adv. Comput. Sci. Appl.* **2024**, *15*, 776. https://doi.org/10.14569/ijacsa.2024.0150379.
44. Daheim, N.; Macina, J.; Kapur, M.; et al. Stepwise Verification and Remediation of Student Reasoning Errors with Large Language Model Tutors. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Miami, FL, USA, 12–16 November 2024; pp. 8386–8411. https://doi.org/10.18653/v1/2024.emnlp-main.478.
45. Lewis, J.R.; Utesch, B.S.; Maher, D.E. UMUX-LITE. In Proceedings of the CHI'13: CHI Conference on Human Factors in Computing Systems, Paris, France, 27 April–2 May 2013; pp. 2099–2102. https://doi.org/10.1145/2470654.2481287.
46. Sauro, J.; Lewis, J.R. *Quantifying the User Experience: Practical Statistics for User Research*; Morgan Kaufmann: Waltham, MA, USA, 2012. https://doi.org/10.1016/c2010-0-65192-3.
47. Hoff, K.A.; Bashir, M. Trust in Automation: Integrating Empirical Evidence on Factors That Influence Trust. *Hum. Factors* **2015**, *57*, 407–434. https://doi.org/10.1177/0018720814547570.
48. Hart, S.G. Nasa-Task Load Index (NASA-TLX); 20 Years Later. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, San Francisco, CA, USA, 16–20 October 2006; Sage: Newcastle upon Tyne, UK, 2006; Volume 50, pp. 904–908. https://doi.org/10.1177/154193120605000909.
49. Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.* **2019**, *267*, 1–38. https://doi.org/10.1016/j.artint.2018.07.007.
50. Dian Martha, A.S.; Widowati, S.; Rahayu, D.P.; et al. Examining Usability and Student Experience of ChatGPT as a Learning Tool on Moodle in Higher Education. In Proceedings of the 2024 International Conference on Information Technology Systems and Innovation (ICITSI), Bandung, Indonesia, 12 December 2024; pp. 8–13. https://doi.org/10.1109/icitsi65188.2024.10929431.