Visualizing RAG Chatbots in Education

Raymond Liu and Kevin Wang

Abstract—

1 Introduction

In recent years, instructors have increasingly adopted large language models (LLMs) to assist in teaching, grading, and answering student questions. Retrieval-Augmented Generation (RAG) systems such as ChatEd [14] and Jill Watson [10] represent a promising step forward: they allow educators to integrate their own course materials into LLMs, improving answer accuracy and contextual alignment without requiring advanced programming or machine-learning expertise. For under-resourced teaching teams, these systems offer a scalable and cost-effective way to provide round-the-clock learning support for students.

However, instructors often lack visibility into how these systems operate internally. Most are non-experts in LLM architecture or prompt design, and current interfaces provide little guidance on how retrieval parameters or prompt changes affect the model's behavior. As a result, instructors struggle to understand why a chatbot produces certain answers—or why it fails to reference the correct course documents. This lack of transparency limits instructors' ability to diagnose errors, adapt prompts, and iteratively improve their course bots. More specifically, instructors struggle with the following problems when using educational RAG chatbots:

Opaque retrieval logic. Instructors cannot easily see which documents influence an answer, nor how the model balances retrieved content with general background knowledge.

Iteration blind spots. When instructors modify system prompts (e.g., "be concise," "simplify for beginners"), they must manually query the chatbot to observe changes. This hinders rapid iterations and improvements.

Limited explainability. Existing RAG interfaces show only raw document lists or similarity scores, providing little sense of how retrieved content influences the generated text. There are no intuitive mechanisms for tracing document–sentence relationships or for visualizing the impact of parameter adjustments.

To address these challenges, we propose a visual analytics suite that makes RAG behavior interpretable and actionable for instructors. Our goal is to transform opaque retrieval pipelines into transparent, manipulable visual representations that support diagnosis, comparison, and refinement of educational chatbots.

Our proposed tool adopts a two-tier design: an overview view summarizing student-chatbot interactions across the course, and a detail view that juxtaposes responses under different configurations (A/B comparison). Instructors can identify low-quality answers, inspect how document retrieval and generation differ between model settings, and receive actionable nudges—such as suggesting a new course document upload or adjusting top K values.

By combining RAG explainability with instructor-centered visualization design, this project aims to bridge the gap between powerful LLM infrastructure and the practical needs of educators.

1.1 Personal Expertise

Kevin has experiences in designing and evaluating AI systems for education, with focus on Retrieval-Augmented Generation (RAG), instructor tooling, and explainable human–AI interfaces. He was the lead developer of the HelpMe platform [12], an AI-enhanced help-seeking system deployed across multiple departments at the University of British Columbia (UBC). HelpMe integrates human and AI assistance into a unified interface, enabling instructors to customize RAG configurations (e.g., chunking, top-k, and retrieval thresholds), maintain course-specific knowledge bases, and monitor student queries in real time.

In promoting and maintaining these systems, Kevin has conducted extensive interviews and informal evaluations with instructors, uncovering recurring problems that motivate this visualization project:

- Instructors struggle to understand how the chatbot retrieves certain materials, and which documents most influence its answers.
- Almost all lack intuition for adjusting RAG parameters (top K, retrieval thresholds, or model choice).
- Instructors want to understand why some questions are answered wrong (the false positive rate is essential)

Raymond has previous experience working with RAGs, having done a course project involving evaluating the performance of different RAG implementations for an information retrieval system. He has also been a Teaching Assistant (TA) for various courses, and has had many conversations with instructors about their opinions and dissatisfactions on educational technology.

2 Related Work

We consider related work in three domains: educational usages of Large Language Models, educational AI assistants, and RAG visualization systems.

2.1 Large Language Models (LLMs) and Education

Large Language Models (LLMs) have evolved rapidly, showing promise across numerous domains [3, 4]. However, general-purpose LLMs often lack the contextual awareness and pedagogical appropriateness crucial for diverse classroom contexts. Agentic frameworks further extend these capabilities through techniques like Chain-of-Thought (CoT) prompting, enabling LLMs to handle multi-step tasks by making their intermediate reasoning explicit [11, 16].

However, a fundamental challenge arises when applying LLMs in education: their training data is broad but not course-specific. Most LLMs rely on large-scale, publicly available datasets that may not include domain-specific, up-to-date, or institutional knowledge [3, 8]. This limitation is particularly problematic in higher education, where course specifics vary across institutions, instructors, and student needs [5].

2.2 Educational AI Assistant

LLM and Retrieval-Augmented Generation (RAG) methods address this gap by combining LLM capabilities with targeted knowledge retrieval [7], enhancing both precision and relevance of AI-generated responses in education. The later iterations, especially in-prompt RAG systems, increase contextual awareness of AI-generated responses and adaptability as updating knowledge sources is much easier. A survey [6] shows that lack of training data is the biggest problem in developing a chatbot in education. Most AI assistant systems primarily rely on pre-existing instructional content, such as course materials and structured knowledge bases. Several LLM-based educational tools have been developed to assist students and instructors in courses with minimal

- *Raymond Liu is with the University of British Columbia. Email: raymliu@cs.ubc.ca*
- *Kevin Wang is with the University of British Columbia. Email: kevinsk@student.ubc.ca*

overhead. Examples include Jill Watson [5], an AI-powered teaching assistant designed to answer student queries in online courses, and a chatbot-based system for assisting students with academic inquiries [14]. Prior work on integrating student help systems and AI chatbots [2] shows promise, although there is a need to evaluate AI effectiveness in question answering and supporting continual content updates. While effective for answering frequently asked questions, these approaches miss a critical aspect of education: real-time, student-driven interactions. Students often ask novel, context-dependent questions that are not explicitly covered in course materials.

2.3 Visualization for RAG

New research on visual analytics for LLM ecosystems has begun to address the transparency and interpretability challenges inherent in retrieval-augmented generation (RAG) systems. While much of the early work focuses on prompt engineering or plain LLMs, a small but growing body of work explicitly targets the retrieval + generation pipeline.

One of the earliest efforts in visualization for prompt-driven LLM workflows is the work by Strobelt et al. [9] which presents a system for interactive and visual prompt engineering. Their approach enables on-the-fly parameter tuning, side-by-side outcome comparisons, and rapid visual feedback, though it does not incorporate an explicit retrieval component.

Extending into the retrieval space, Wang et al. [15] introduce RAGVIZ, a visual system designed to help users diagnose how retrieved passages contribute to generated answers. Their interface visualizes token and document-level attentiveness, supports toggling context inclusion, and assists debugging of hallucinations and retrieval failures. More recently, Wang et al. [13] propose XGraphRAG, which leverages graph-based representations of retrieval for specifically graph knowledge retrievals. Arawjo et al. [1] present ChainForge, a visual toolkit for prompt engineering and hypothesis testing with LLMs. Though not explicitly RAG-oriented, ChainForge's comparative small-multiple views and experiment workflows suggest strong design patterns for comparing variants (e.g., prompt v1 vs prompt v2), which we may adapt and extend in our work.

Together, these systems illustrate several recurring design idioms: side-by-side comparisons, provenance tracing, visual attention/coverage indicators, and multi-view pipelines. And that informs our instructor-facing "cockpit" design. Our work differentiates itself by focusing on: 1. course-specific corpora, 2. student-question histories, 3. and by embedding actionable nudges for instructors (such as "Considering add new document about x; decrease top k to 3"). By and large, we aim to bridge the gap between general RAG diagnostics tools and the unique operational needs of instructors who deploy chatbots in programming and theory-heavy courses.

3 Data and Task Abstraction

We focus on visualizing RAG chatbots in education in our specific context, using the data and task abstraction.

3.1 Data Abstraction

On the highest level, our data consists of:

1. Course documents, uploaded by the instructor
2. Parameters within the RAG chatbot system
3. The set of student questions and chatbot answers, as well as associated data with each answer (for example the documents retrieved by the chatbot, and the evaluations on the answer given by humans and LLMs)

The course document table consists of items with the following attributes:

- Document name (categorical): the name of the document
- Document contents (categorical): the contents of the document, stored as a string
- Document chunk start and end position (pair of quantitative attributes): each document is split by the RAG system into smaller chunks, which are retrieved by the RAG (instead of the full documents). We consider the (start, end) chunking positions as representations of these chunks.

Note that each item in this documents table actually represents a document chunk, not a document. This means that the document name and contents are repeated several times across the items, but the document chunk positions differ.

The chatbot configurations table consists of the following attributes:

- Base LLM (categorical): the base LLM used for the chatbot (for example GPT4o, qwen, gemma, etc.), with approximately 5 options available to the instructor.
- Course documents (list of items from the documents table); the documents uploaded by instructors to be retrieved by the RAG system
- Top K (quantitative, options 3, 5, and 10): the number of top document chunks retrieved at each query
- Retrieval threshold (quantitative, ranging from 0.01 to 0.99): the minimum similarity score a document should have to be considered in the RAG ranking
- System prompt (categorical): the text used in every prompt providing (for example, "Please answer the following student question using the provided documents.")

We plan to focus our visualization on the base LLM and course documents, as we believe instructors will be more inclined to alter those parameters, compared to the remaining three options.

The student chatbot history table contains the main data to be visualized by our system. The attributes are as follows:

- Time of response (quantitative): the time the answer was generated by the chatbot.
- Student question text (categorical): the question text given by the student.
- Chatbot-generated answer (categorical): the response generated by the chatbot.
- Retrieved document chunks (list of items from the documents table): The document chunks retrieved by the RAG system
  - Associated with each retrieved document chunk is its document similarity (quantitative, between 0.00 and 1.00). This is a numerical value, as given by the RAG system, representing the semantic similarity of each document chunk with the question. We can use this to order the retrieved document chunks.
- Chatbot configurations (from the chatbot configurations table): the chatbot settings that were used to generated the answer.
- Chatbot answer evaluation - human (quantitative, a non-negative integer from 0 to the size of the class): students are given the option to flag unhelpful or irrelevant chatbot responses, and the system keeps track of the number of flags for each response.
- Chatbot answer evaluation - NLP. This consists of computational evaluation of the chatbot-generated response to the student question. We consider the following:
  - LLM-as-a-judge (quantitative, between 0.00 and 1.00): a score provided by a separate LLM on the helpfulness and relevance of the answer.
  - Averaged document similarity (quantitative, between 0.00 and 1.00): the average document similarities within the retrieved document. This approximates the relevance of the documents to the provided response.

- Sentence-level evaluations: for a more fine-grained computational evaluation of the chatbot response, we consider a system in which we (or the user) split the sentences within the response, and rank the relevant document evidence for each sentence. This would create a sub-table consisting of the following data:
  - Sentence positions (pair of quantitative attributes): the positions of the sentences within the chatbot response, split by start and end. A future version of this system will consist of users selecting their own substring as the "sentence" of choice; in that case, the user can select the start and end positions themselves, rather than having it be pre-split.
  - Document chunk ranking (ordinal): the rank order of retrieved document chunks by their relevance to each specific sentence (e.g., 1st most relevant, 2nd most relevant, etc.)
  - Per-sentence attribution score (quantitative, between 0.00 and 1.00): a score indicating the strength of evidence each retrieved chunk provides for the sentence

Note that our tool is designed to work with datasets of different cardinalities, ranging from 1 student question/chatbot answer to over 100. However, for the purposes of testing, we will work with a custom dataset with approximately 25 student questions/chatbot answers.

3.2 Task abstraction

We have three main instructor goals in using our system.

Identify low-quality chatbot responses, i.e. those that were incorrect, unhelpful, or confusing to students.

Our system presents an overview for instructors showing the history of student questions and chatbot responses. It highlights those that were marked as low-quality, either from a high number of student flags, or a low score given by the LLM-as-a-judge or averaged document similarity scoring systems. This can be broken down as follows:

- **What (target)**: All attributes of question-answer items, focusing on derived quality scores (human flags, LLM-as-judge ratings, document similarity)
- **Why (high-level goal)**: Discover problematic patterns in chatbot behavior
- **How (action)**: Browse, filter, and sort question-answer pairs
- **Search mode**: Browse to scan the entire interaction history; locate to find specific flagged instances

Diagnose why a chatbot provided a poor response, particularly relating to course documents

Our system supports the fine-grained inspection of a specific chatbot response. It provides relevant data for each sentence within the response, including: document chunks that were retrieved by the RAG system and their similarity scores; sentence-level attribution showing which document passages provide evidence for each generated sentence; identification of potentially hallucinated content where sentences lack adequate document support; and interactive exploration enabling instructors to trace the provenance of specific claims back to their source materials. This allows instructors to identify whether errors stem from retrieval failures (relevant documents not retrieved), ranking problems (correct documents retrieved but ranked too low), or generation issues (model hallucinating despite having correct source material). More specifically, we break this down as follows:

- **What (target)**: Links between question items and document chunk items; attributes of individual sentences (presence/absence of supporting evidence)
- **Why (high-level goal)**: Explain why a specific response failed
- **How (action)**: Identify relevant document chunks; compare retrieved vs. expected documents; relate sentences to source passages
- **Dependencies**: Depends on completing Task 1

Iteratively improve chatbot performance by updating model parameters or documents and comparing results.

Given a student question, our system allows instructors to juxtapose responses given by chatbots with different model configurations and/or course documents, with the adjustable parameters listed in the configurations and documents table above. For example, they may upload a new course document and tweak the top-K parameter, and run the question on a chatbot with those updated configurations. They can analyze to see if these changes led to a better response, using both the response text as well as the scores to measure improvement. We consider the following:

- **What (target)**: Attributes of configuration items (model type, top-K, documents); derived items (newly generated responses under modified settings)
- **Why (high-level goal)**: Produce improved chatbot configurations
- **How (action)**: Compare responses generated under different configurations; derive new parameter values; record which configurations improve quality
- **Dependencies**: Depends on Tasks 1 and 2 (knowing which responses to improve and why they failed)

4 Proposed Solution

The system's general design is based on three primary visualization displays, which break down into an overview page, a detail page for juxataposing two runs, and a deep dive section to review one single run. The overview page displays the full history of chatbot questions and answers for overall evaluation and identification of poorly answered questions; the detail page juxtaposes chatbot answers for different model configurations, allowing instructors to find their preferred configuration for these poorly answered questions.

4.1 Usage Scenario

To illustrate the system's functionality, consider a UBC CPSC447 instructor using this chatbot in their course. They have deployed the chatbot with default model configurations and uploaded all course documents, including the course website (as an HTML file). One month into the course, they receive complaints in office hours about the chatbot providing incorrect answers regarding assignment requirements and deadlines, as well as using overly harsh language.

4.2 Overview Page: Identifying Problem Areas

To investigate these issues, the instructor navigates to the overview page (Figure 1). The overview page presents a comprehensive list of student questions and chatbot answers, with each question-answer pair associated with a complete set of chatbot configurations. For concise display, only the base model configuration, top-K value, and retrieval threshold are shown in the list view.

Currently Implemented:

- Question and answer display with associated metadata
- Configuration summary (base model, top-K, threshold) for each entry
- Visual highlighting (red circles) for flagged responses
- Clickable rows to navigate to detailed analysis
- Basic filtering and sorting capabilities

Not Yet Implemented:

- NLP-based quality scoring integration
- Temporal filtering and trend visualization (maybe not within scope)
- Advanced search and filtering by document type or question category

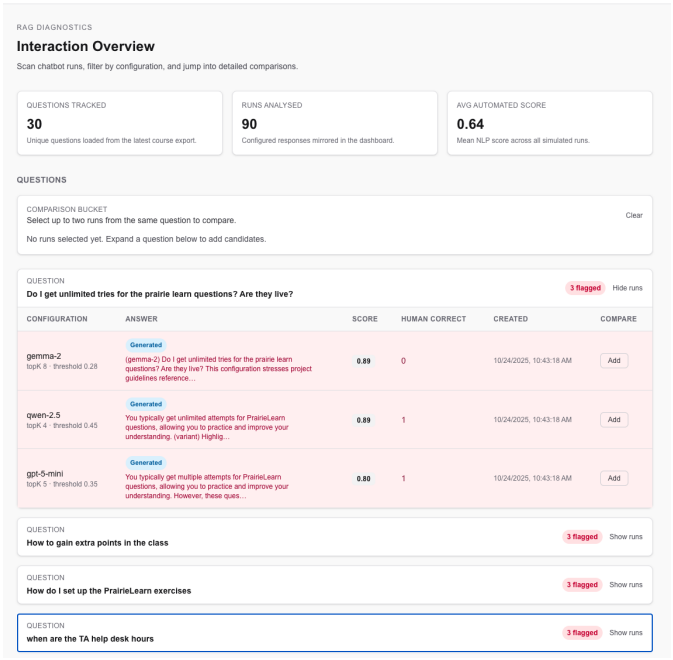

Fig. 1: Overview page displaying the full history of student-chatbot interactions. Red circles indicate questions flagged as incorrect or unhelpful by students or automated evaluation metrics.

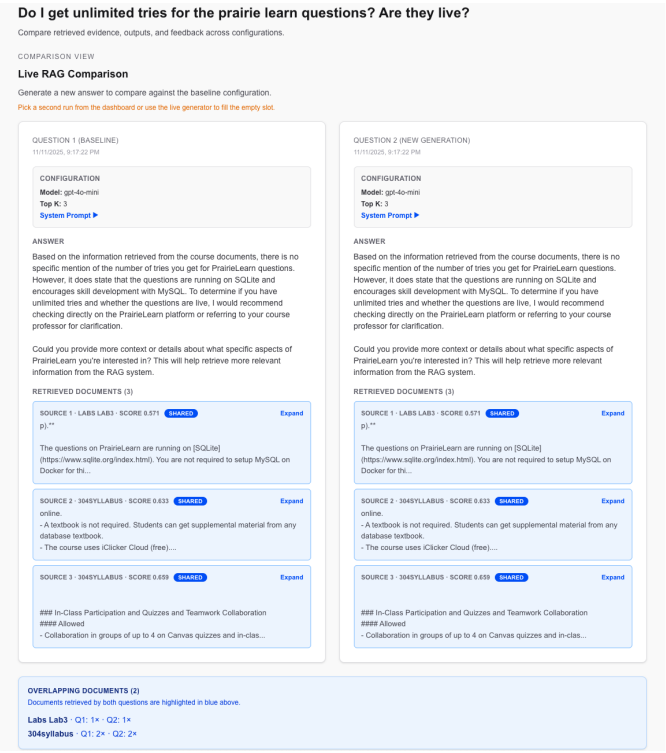

Fig. 2: Detail page juxtaposing two chatbot runs with different configurations. The interface uses color coding, positioning, and sizing to highlight differences in retrieved documents and generated responses.

Each question-answer pair is associated with quality scores from both human feedback and NLP algorithms. In the current demo, human flags (student-reported issues) are fully functional, while the NLP scoring component displays placeholder values and awaits integration with actual metrics such as LLM-as-a-judge scores and averaged document similarity.

The CPSC447 instructor can immediately identify questions where the chatbot's answers were marked as incorrect or unhelpful, enabling rapid triage of problem areas.

4.3 Detail Page: Diagnosing and Comparing Configurations

When the instructor selects a specific question from the overview, they are taken to the detail view (Figure 2). This view reveals the complete set of model configurations used to generate that chatbot answer, including the retrieved document chunks and relevant text passages.

Currently Implemented:

- Side-by-side comparison of two chatbot runs
- Interactive parameter adjustment (base model, top-K, system prompt)
- Real-time re-generation of answers with modified configurations
- Diff visualization showing changes between responses
- Retrieved document display with similarity scores

Not Yet Implemented:

- Database/document set comparison (ability to add or remove documents and visualize the impact)
- Sentence-level provenance tracking (linking each sentence in the response to specific retrieved document passages)
- Attention or attribution visualization showing which parts of retrieved documents most influenced each sentence
- Quantitative diff metrics (e.g., semantic similarity between old and new responses)

In an example scenario, the instructor discovers that the chatbot provided incorrect information because the retrieved document chunks all reference an outdated assignment. The instructor had posted new assignment instructions on the course website but failed to upload the updated version to the RAG system. Additionally, the instructor decides to switch the base model from GPT-4o to Qwen, having heard that Qwen is a cool model and want to try it.

The instructor makes these changes through the detail view interface and reruns the model on the same student question. The visualization juxtaposes the old and new configurations, using color coding, positioning, and visual emphasis to highlight differences in retrieved documents, similarity scores, and generated text. Once the new run completes, the instructor can verify that the updated chatbot outputs correct information and retrieves the appropriate documents.

4.4 Deep Dive Page: Fine-Grained Analysis

For more granular inspection, the system provides a deep dive view (Figure 3) that focuses on a single chatbot run. This view breaks down the response at the sentence/document level and displays detailed provenance information.

Currently Implemented:

- Display of the full question and answer
- Configuration parameters panel
- Retrieved document list with similarity scores
- Basic document chunk visualization

Not Yet Implemented:

- Interactive selection of text spans to view relevant document passages
- Document phrase matching showing which specific passages influenced each sentence
- Attribution scores indicating the strength of evidence for each sentence

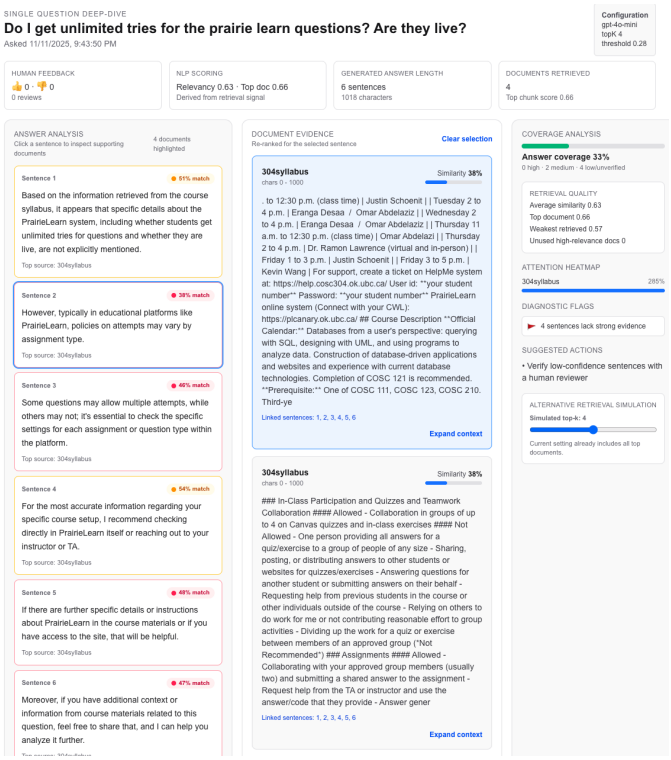

Fig. 3: Deep dive view showing sentence-level analysis of a single chatbot response, including document attribution and similarity metrics.

- Hallucination detection highlighting sentences with weak or no document support
- Coverage visualization showing what percentage of the response is grounded in retrieved documents

This deep dive capability will allow instructors to understand not just whether a response is good or bad, and identifying which sentences lack adequate document support, which documents are most influential, and where the model may be hallucinating or over-generalizing.

4.5 System Architecture and Implementation

The tool is being developed as a full-stack web application. The frontend uses React with Next.js framework, D3.js for custom visualizations, and TypeScript for type safety. The backend leverages Node.js with PostgreSQL for data persistence, and integrates with OpenAI and other LLM APIs for real-time answer generation.

Currently Implemented:

- Full Next.js application framework with API routes
- PostgreSQL database with Prisma ORM for question, answer, and configuration storage
- RAG pipeline integration supporting multiple LLM providers
- Basic visualization components for overview and comparison views
- Real-time answer generation with configurable parameters

Technical Gaps:

- Automated NLP evaluation pipeline (LLM-as-a-judge, document similarity metrics)
- Sentence-level attribution and document matching algorithms
- More visualization encodings

Component	Tasks	Owner	Est. Hours
Backend Setup	• Set up Node.js/PostgreSQL environment • Implement basic RAG pipeline integration • Create API endpoints for data retrieval	Kevin	10h
Data Generation	• Create/collect ~25 sample questions • Generate chatbot responses with multiple configs • Implement NLP scoring algorithms • Collect human feedback data	Kevin	3h
Frontend Scaffold	• Set up React/D3.js environment • Create basic routing structure • Implement data fetching layer	Kevin	5h
Basic Overview Page	• Display question/answer list • Show basic quality scores • Implement filtering by score threshold	Raymond	10h
Basic Detail Page	• Display single Q&A with configuration • Show retrieved documents	Raymond	8h
Integration & Testing	• Connect frontend to backend • Debug data flow issues	Both	6h
Improving task/data abstraction	• Furnishing/improving data abstraction	Raymond	10h
Phase Total			58h

Fig. 4: Milestone 1

Component	Tasks	Owner	Est. Hours
Overview Page Refinement	• Implement color-coded model indicators • Design and implement layout alternatives to table • Add sorting and advanced filtering	Kevin	10h
Detail Page Refinement	• Implement A/B comparison view • Add document chunk highlighting • Show relevant substring visualization • Implement difference indicators (color/position) • Add interactive document exploration	Raymond	12h
Configuration Editor	• Build UI for changing LLM/documents • Implement re-run functionality • Add comparison trigger mechanism	Raymond	8h
Visual Polish	• Refine color schemes • Improve transitions and interactions • Responsive design adjustments	Both	6h
User Testing	• Conduct informal evaluation with 2-3 instructors • Gather feedback and identify issues	Both	4h
Phase Total			48 hours

Fig. 5: Milestone 2

5 Milestones and Work Allocation

The major milestones are as follows:

By **the end of October**, we aim to have completed a basic, crude prototype of the system. Fig. 4 shows the list of tasks involved.

By **late November** (in particular the update document deadline), we aim to refine the system, implementing and iterating on the specific visual encodings; the specific tasks are shown in Fig. 5.

By **the start of December**, we aim to have a final design that largely fulfills the goals outlined in this proposal, as outlined in Fig. 6.

6 Discussion, Future Work, and Conclusion

References

[1] I. Arawjo, C. Swoopes, P. Vaithilingam, M. Wattenberg, and E. L. Glassman. Chainforge: A visual toolkit for prompt engineering and llm hypothesis testing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pp. 1–18, 2024. 2

[2] N. Basit, M. Floryan, J. R. Hott, A. Huo, J. Le, and I. Zheng. Asci: Ai-smart classroom initiative. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSETS 2025, 7 pages, p. 81–87. Association for Computing Machinery, New York, NY, USA, 2025. doi: 10.1145/3641554.3701957 2

Component	Tasks	Owner	Est. Hours
Feature Completion	• Implement any missing core features • Add edge case handling • Improve error messages and loading states	Both	10h
Integration Work	• Ensure seamless flow between views • Optimize API calls and data loading • Implement caching where appropriate	Both	8h
Visual Refinements	• Final adjustments to visual encodings • Improve accessibility (contrast, labels) • Polish animations and transitions	Both	6h
Documentation	• Write user guide • Document code • Create demo scenarios	Both	5h
Final Testing	• End-to-end testing • Performance optimization • Bug fixes	Both	8h
Presentation Prep	• Prepare demo • Create presentation materials	Both	5h
Phase Total			**42 hours**

Fig. 6: Final Milestone

[3] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 1

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[5] S. Kakar, P. Maiti, K. Taneja, A. Nandula, G. Nguyen, A. Zhao, V. Nandan, and A. Goel. Jill Watson: Scaling and Deploying an AI Conversational Agent in Online Classrooms. In *International Conference on Intelligent Tutoring Systems*, pp. 78–90. Springer, 2024. 1, 2

[6] M. A. Kuhail, N. Alturki, S. Alramlawi, and K. Alhejori. Interacting with educational chatbots: A systematic review. *Education and Information Technologies*, 28(1):973–1018, 2023. 1

[7] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020. 1

[8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1

[9] H. Strobelt, A. Webson, V. Sanh, B. Hoover, J. Beyer, H. Pfister, and A. M. Rush. Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. *IEEE transactions on visualization and computer graphics*, 29(1):1146–1156, 2022. 2

[10] K. Taneja, P. Maiti, S. Kakar, P. Guruprasad, S. Rao, and A. K. Goel. Jill watson: A virtual teaching assistant powered by chatgpt. In *International Conference on Artificial Intelligence in Education*, pp. 324–337. Springer, 2024. 1

[11] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022. 1

[12] K. Wang and R. Lawrence. HelpMe: Student Help Seeking using Office Hours and Email. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education - Volume 1*. ACM, 2024. doi: 10.1145/3626252.3630867 1

[13] K. Wang, B. Pan, Y. Feng, Y. Wu, J. Chen, M. Zhu, and W. Chen. Xgraphrag: Interactive visual analysis for graph-based retrieval-augmented generation. In *2025 IEEE 18th Pacific Visualization Conference*, pp. 1–11. IEEE, 2025. 2

[14] K. Wang, J. Ramos, and R. Lawrence. ChatEd: A Chatbot Leveraging ChatGPT for an Enhanced Learning Experience in Higher Education, 2023. https://arxiv.org/abs/2401.00052. 1, 2

[15] T. Wang, J. He, and C. Xiong. Ragviz: Diagnose and visualize retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 320–327, 2024. 2

[16] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 1