

An AI Agent Workflow for Generating Contextual Cybersecurity Hints

Hsiao-An Wang
Computer Science
Northeastern University
Boston, MA, USA
hs.wang@northeastern.edu
0009-0008-6967-3904

Joshua Goldberg
Computer Science
Northeastern University
Boston, MA, USA
goldberg.josh@northeastern.edu
0009-0002-8078-6041

Audrey Fruean
Computer Science
Northeastern University
Boston, MA, USA
fruean.a@northeastern.edu
0009-0005-3007-7660

Zixuan Zou
Computer Science
Northeastern University
Boston, MA, USA
zou.zix@northeastern.edu
0009-0006-2940-1352

Ruoyu Zhao
Computer Science
Northeastern University
Boston, MA, USA
zhao.ruoyu@northeastern.edu
0009-0000-9357-5759

Sakib Miazi
Computer Science
Northeastern University
Boston, MA, USA
m.miazi@northeastern.edu
0000-0002-9778-4678

Jens Mache
Computer Science,
Lewis & Clark College
Portland, OR, USA
jmache@lclark.edu
0000-0003-2927-7436

Ishan Abraham
Computer Science,
Lewis & Clark College
Portland, OR, USA
ishanabraham@lclark.edu
0009-0000-9508-7708

Taylor Wolff
Computer Science
Evergreen State College
Olympia, WA, USA
Taylor.Wolff@evergreen.edu
0009-0003-9567-5085

Jack Cook
Computer Science
Evergreen State College
Olympia, WA, USA
cookjackc@gmail.com
0009-0009-2452-2878

Richard Weiss
Computer Science
Evergreen State College
Olympia, WA, USA
weissr@evergreen.edu
0009-0007-6610-5197

Abstract—Large Language Models (LLMs) have been used to aide student learning across a multitude of domains such as computer science, data science, and mathematics. While these chatbots show promise, they may not perform well when quality student data is unavailable.

Hint generation for cybersecurity can be feasible with the technology we have today if we make the right simplifications. First, we need human-in-the-loop systems because the training data used to train general LLMs may not cover cybersecurity well. Second, using modular agents allows us to access dynamic data that are outside the training dataset and add specificity to the hints. In this paper, we leverage n8n, an agent deployment service, to establish the connections between our agents and Discord, the messaging system that our classroom uses to offer students a streamlined learning experience when working on interactive cybersecurity exercises. The system has undergone initial pilot testing, Hints provided were sufficient to propel students forward and make progress on their assigned work, in most cases.

Keywords—Cybersecurity, Cybersecurity Education, Generative AI, Retrieval Augmented Generation

I. INTRODUCTION

In this work, we present a system for automatically generating hints to the students without presenting direct answers. Our goal is to make interactive cybersecurity exercises in the classroom more accessible to students while remaining manageable for the instructor. Our aim is to address three key issues. The first issue arises when students use a broadly trained LLM, the hints or answers generated could rely on material not covered in the course. The second issue is that training an LLM typically requires large amounts of data. Lastly, the most critical issue we aim to address is that students often receive explicit, step-by-step solutions directly from the broadly trained LLM, which short-circuits their learning efforts. Ideally, the hints should guide students towards the intended solution without explicitly providing the answer. The cybersecurity exercises are designed to encourage students to try different methodical approaches to solve the problems after they spend time critically evaluating the problems at hand.

The solution we used for these problems is human-in-the-loop (HITL). Instead of directly interacting with an AI agent, the student interacts with the instructor, who in turn interacts with the AI agent. The instructor receives the hint and can accept, modify, or reject it. This provides guardrails that avoid the above problems.

Our implementation uses Pinecone vector store [1] and Retrieval-Augmented Generation (RAG) [2] to provide student information and the context in which the student is working. To script the integration of these tools, we used a workflow automation platform (n8n) [3]. To test our models, we used a publicly available cyber range framework. We chose an introductory exercise that introduces the command line interface.

II. BACKGROUND

We have noticed that students tend to rely too much on generative AI tools. Sun *et al.* [4] reported that the use of generative AI tools based on LLMs can negatively impact students' programming practices. However, a recent systematic review of 28 studies involving 4,597 K-12 students found that intelligent systems, such as generative AI, generally have positive effects on learning [5]. PyTutor, a ChatGPT-based system for Python programming, showed particular effectiveness for students with weak foundations through structured hint system [6]. Liu *et al.* found that their tailored chatbot helped students learn to write code, avoiding the use of a generic LLM [7]. Kazemitabaar *et al.* reported that providing hints in pseudocode rather than actual code improved learning outcomes [8]. It is becoming increasingly evident that the higher the conceptual level of the hints, the more students must engage in active learning. We believe that instructors intuitively know this.

Mosqueira-Rey *et al.* found that Human-in-the-loop (HITL) helps to enable system-controlled active learning, interactive machine learning, and machine teaching (human-expert controlled), which has significant potential to improve learning outcomes [9]. Ninaus *et al.* found that educational AI systems can be conceptualized as closed-loop systems involving data recording, pattern detection, and adaptivity, with humans playing critical decision-making roles throughout [10]. Our approach uses human-in-the-loop guardrails to provide reliable, relevant, and accurate hints while avoiding the creation of specialized systems in our beginner-level cybersecurity courses. Our framework uses specific context information about the learning assignments. We assume the system is not perfect at providing hints, and that is where the instructor serves as the safeguard. Besides, HITL preserves the consistency of the knowledge path between a student and an instructor. Retrieval-Augmented Generation (RAG) has emerged as a promising solution to address the limitations of large language models in educational applications. RAG overcomes the main barrier for adoption of LLM-based chatbots in education - hallucinations - by making it relatively easy to implement chatbots that serve specific purposes and address various educational needs. [11] RAG enhances LLMs by retrieving relevant information from external knowledge bases and incorporating it into the generation process, improving factual accuracy and enabling dynamic knowledge updates, making LLMs particularly suitable for educational applications [2]. Hence, we decided to use RAG as a more reliable way of hint generation.

III. DESIGN PHILOSOPHY

One of the main challenges in today's classrooms is preventing students from short-circuiting their learning by using Generative AI-driven chatbots to complete assignments without thinking critically about the problem.

We introduce a hint-generation workflow that allows instructors to customize the context to ensure hints do not rely on content not covered in class. Additionally, this hint-generation system, powered by Retrieval Augmented Generation, can simulate student questions to aid instructors with low enrollment who wish to train an SLM rather than an LLM. The hint-generation workflow can also be used to reinforce knowledge retention of the concepts and methodologies introduced in the classroom offering. We intend for this workflow to be applicable to any cybersecurity exercises of interest. However, this will require the instructor to provide some context for the agent. This will be discussed in the System Design section.

IV. SYSTEM DESIGN

In the following section, we will introduce the independent components of the workflow used to create the agentic AI hint generation system. We will elaborate in detail on all the elements that were involved in the creation of this system in the corresponding subsections that follow.

Before describing each technical component, it is useful to give a brief overview of how the system is organized. The overall design relies on three separate workflows: one that loads and prepares assignment materials for later retrieval, one that generates hints by drawing on that stored context, and a third that answers general cybersecurity questions when a student's message is not tied to a specific assignment. Although they operate independently, these workflows work together to ensure that the system can respond appropriately to both context-dependent and broader questions that students may ask.

A. n8n Workflow

To create a simple and streamlined process, n8n [3] workflows were created and connected to the Discord messaging platform. Workflows consist of a chain of nodes, with each node performing a specific action. For this system, there are two main workflows: one that inserts data into a Pinecone vector store (Workflow A) and another that uses RAG with ChatGPT to generate hints for the student (Workflow B). Figure 1 shows the exact logic of each workflow. Workflow A is executed manually and searches a connected Google Drive for relevant content (marked in a special folder). The contents of the folder are then downloaded and loaded into the Pinecone vector store with an OpenAI embedding (text-embedding-3-small), a data loader to insert the data, and a recursive text splitter to break the content into chunks.

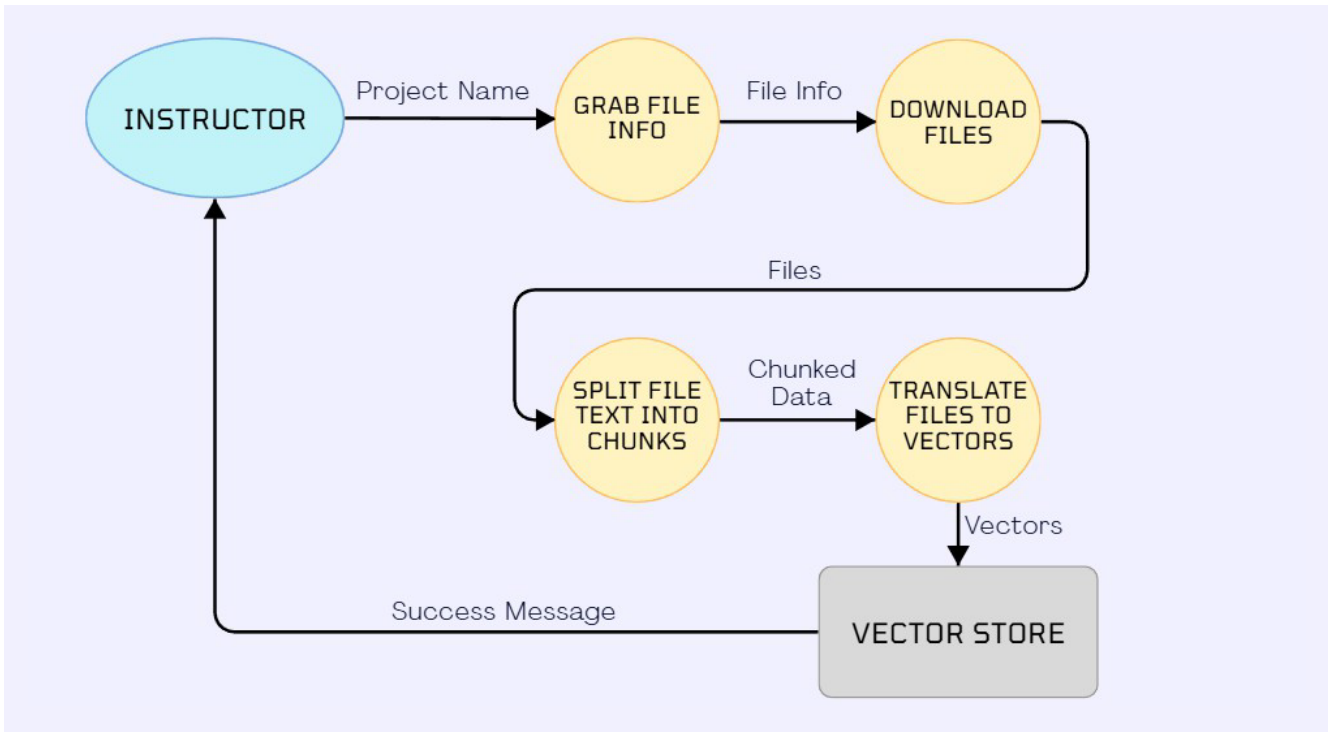


Fig. 1. Process flow diagram used to store, retrieve, and process contextual data.

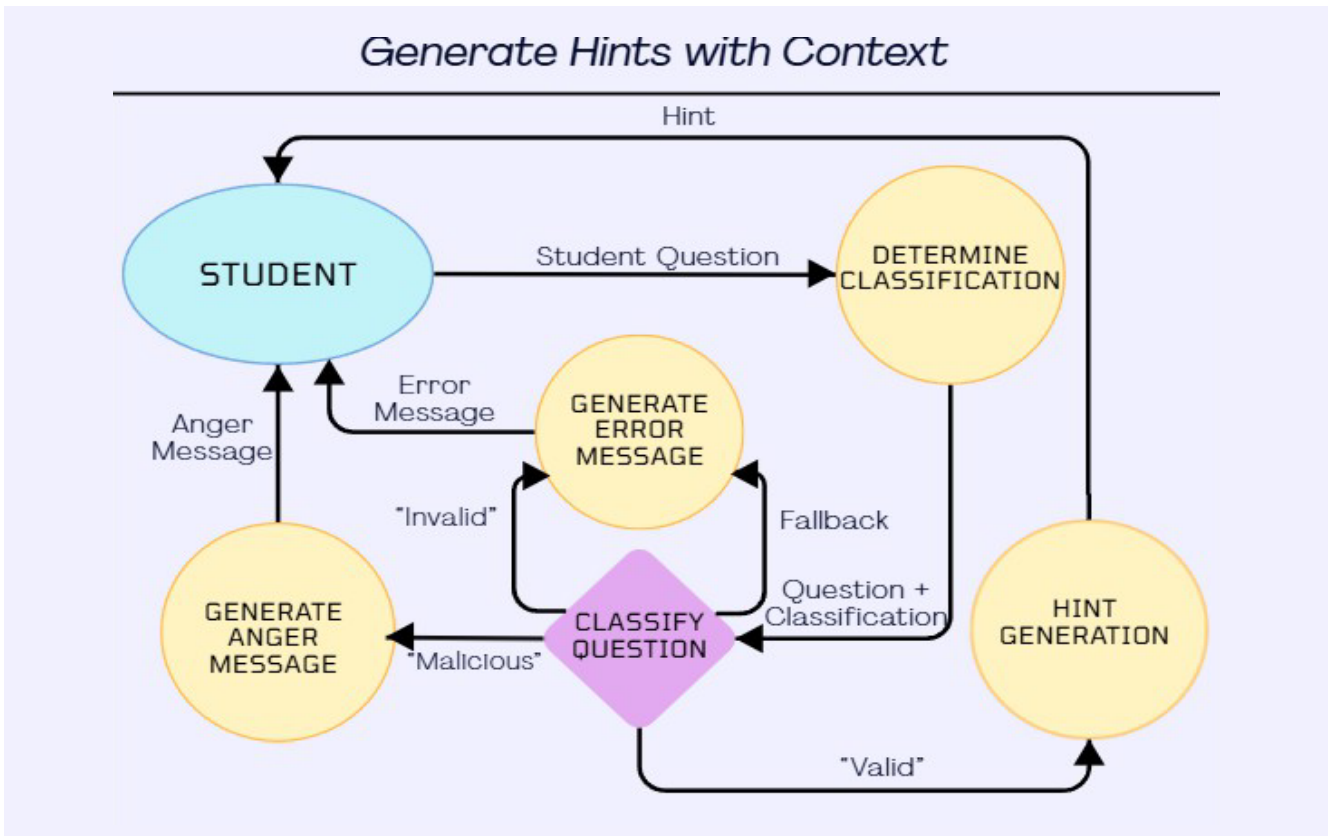


Fig. 2. The Hint Generation Process determines the context of questions

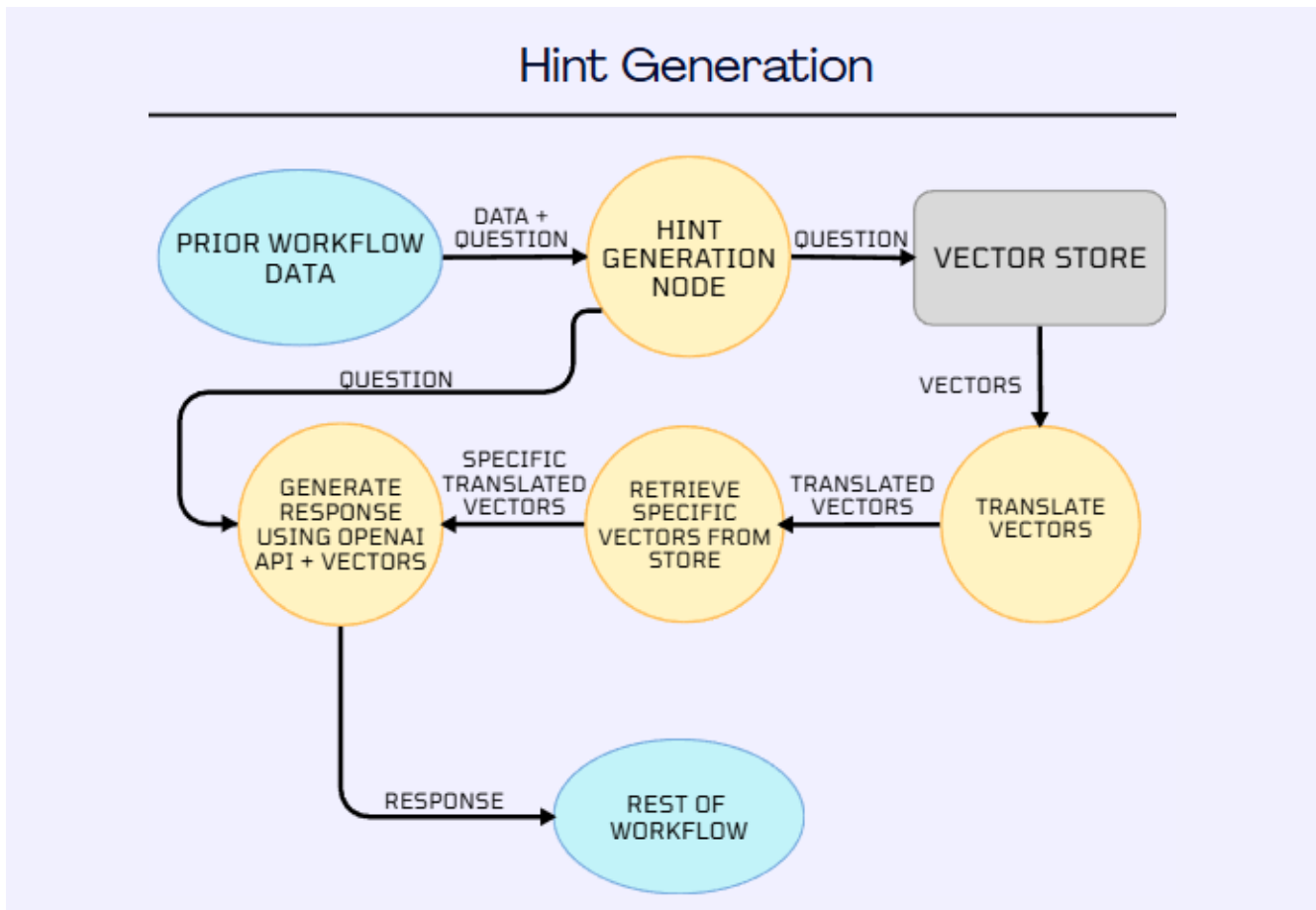


Fig. 3. The sub-process flow diagram demonstrates hint generation

Workflow B, as shown in Figure 2 and 3, executes when a POST request containing a student's question is received. The question is then fed into a question-and-answer chain, which uses RAG alongside an OpenAI chat model to generate a hint for the given question. The chain is connected to the same vector store that Workflow A inserts into, providing the necessary context for the chain to generate a relevant hint. Finally, the generated hint is sent back to the instructor (HITL) or the student via a Discord message. If the message fails to send (for instance, if the message is too long), the recipient is notified of the error via a separate Discord message.

B. Workflow Flexibility in n8n

The key benefit of using n8n in this system is the ease with which we can modify, expand, and reference workflows, which is especially valuable in educational settings where system requirements can evolve rapidly. Since both workflows A and B are composed of configurable nodes, we can easily adjust or extend the system with minimal effort. For instance, if we needed to use a different LLM or embedding model, we would only need to change the corresponding node, with minimal to no changes in the rest of the workflow. Considering this modularity in a classroom context, n8n enables much quicker

adaptation to changing environments. Instructors can easily update prompt formats or adjust vector parameters by editing node configurations in the n8n workflow editor. In our workflows, tools such as Google Sheets or messaging platforms can be easily integrated by inserting new nodes at points in the workflow as needed. n8n also provides built-in tools for debugging and testing workflows, including execution logs and the ability to run workflows step by step to work through each node individually. This makes it incredibly convenient for developers or instructors to rapidly modify and adjust the workflow's design without disrupting the system's operation. The specifics of each vital node in the workflow are explained in more detail below.

C. Pinecone vector store

Pinecone [1] provides easy access to vector stores, enabling easy storage of relevant data in context. Storage is organized as indices. The current index for data storage uses cosine similarity to compare vectors. With this setup, vectors can be generated using OpenAI's text-embedding-3-small model (which produces embeddings with 1,536 dimensions) and easily inserted into the index for later access using RAG.

D. Google Drive

Workflow A is connected to a Google Drive account [12]. This account stores information and relevant data we would like to include in the Pinecone vector store. Within Google Drive, data is consolidated into a specific folder that contains various documents processed by Workflow A and inserted into the vector store. This integration keeps data organized and allows us to precisely control the information we want to include in the context provided to the AI, ensuring that the hint-generation system does not provide out-of-context solutions that typical, broadly trained LLMs do. Figure 4-5 shows a simulated question set generated based on prior experience hosting introductory Linux exercises in class for the student as an early test of the questions students may ask, to examine whether we can generate helpful hints using the developed system to support learning.

- What does the "pwd" command do, and how is it used to identify the current working directory?
- How do I use the "ls" command, and what options can I add (e.g., to show hidden files)?
- What is the purpose of the "man" command, and how can it guide me in understanding other commands
- How do I change directories using the "cd" command effectively?
- What is the correct syntax for renaming files using the "mv" command?
- How should I copy files using the "cp" command, and what are the differences between "mv" and "cp"?
- How do file permissions work, and what do the different modes (read, write, execute) mean?
- What is the syntax for changing file permissions using the "chmod" command?
- How can I troubleshoot common errors that occur when using these commands?
- How can I combine multiple commands in an exercise to perform complex file manipulation tasks?

Fig. 4. First set of questions simulated in the connected Google Sheet, managed by Workflow C. This set simulates a student who has little/no understanding of cybersecurity.

- What would be the best way to confirm that a file is formatted as an OpenSSH RSA public key?
- How do I combine "sort" and "uniq"?
- I'm having trouble finding/reading the "hidden-instructions.txt" file. Would I need to use commands like "grep" or "chmod" to do so?
- I'm a bit unsure which permission mode is most appropriate here. Is there a difference between using a command like "chmod u+r" or something like "chmod 644" to grant the needed permission?
- When I run `ls -a`, I just see hidden files, but not file sizes. Is there a way to also see the size of files alongside hidden files?
- I'm having trouble with the tasks. In the file-practice directory, I need to rename "file1.txt" to "renamed_file1.txt" and copy "file2.txt" to "copied_file2.txt", and then run the command "wrangle_flag" to get the output flag. I used "mv" and "cp", but when I run "wrangle_flag" I get an unexpected error. What should I do?
- Is there a way I can jump to the deepest subdirectory where the final file is located?

Fig. 5. Second set of questions simulated in the connected Google Sheet, managed by Workflow C. This set simulates a student who has a basic understanding of cybersecurity.

E. Discord Integration

Discord [13] is leveraged by using a custom-built chatbot. The bot accepts commands from a student. Commands are currently issued in the format of '!hint [scenario] [question]', where the scenario represents the name of the set of problems a student is answering, and the question is the specific question the student wants answered. Since the assignments have varying levels of challenge, the bot currently accepts level-specific questions. When a student uses the command to ask the bot a question, a POST request is sent to the workflow, which then executes to generate a hint. Currently, each hint piece takes approximately 30 seconds to generate. This latency is caused by three factors: (1) the time required for Pinecone to perform vector retrieval, (2) the sequential execution overhead of n8n's node, and (3) the LLM's generation time. Because the workflow executes synchronously in n8n, these components run serially, resulting in a minimum latency of 30 seconds per hint. Once the hint has been successfully generated, the bot will directly respond to the student who asked the question, ensuring they receive a response upon workflow completion.

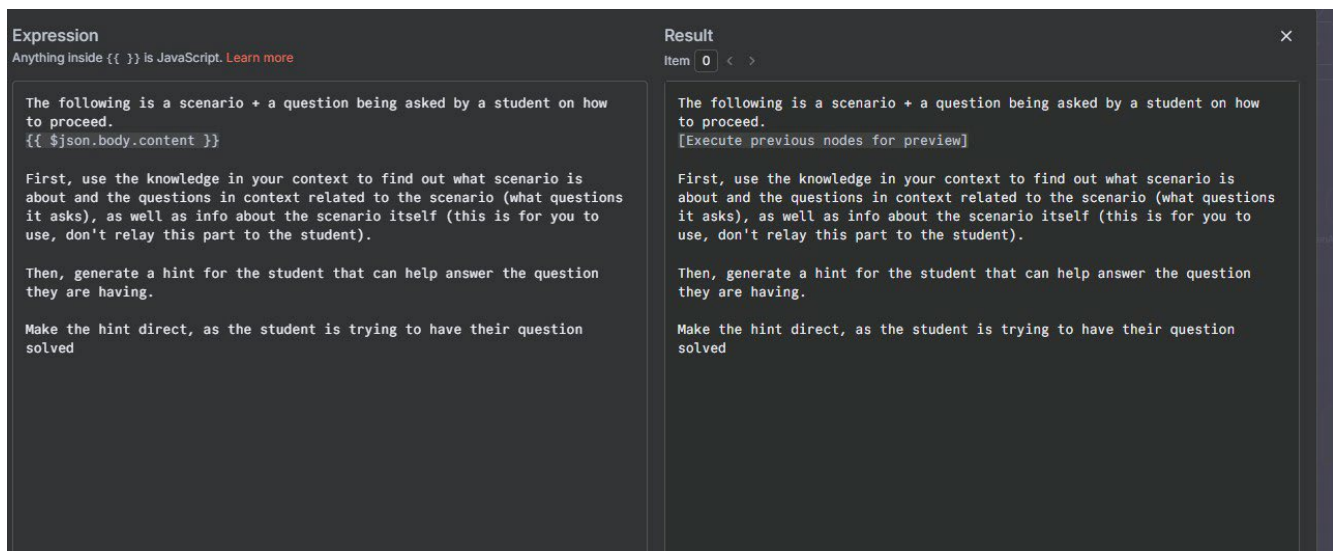


Fig. 6. Prompt Details for Model to Generate Context-Specific Hints

F. Question and Answer Chain

The Question and Answer Chain is the brain of the system, processing the student's question and generating an appropriate response. This is achieved by utilizing RAG, in conjunction with an OpenAI chat model. The system currently uses o3-mini (specifically o3-mini-2025-01-31) to help generate hints. RAG is utilized by accessing the previously defined Pinecone vector store. To guide the model when developing a hint, the prompt shown in Figure 6 is used; Scenario & Question is a placeholder that represents the data detailing the scenario the student is working on and the specific question the student asked. The said data is then provided to the chain by the data from the webhook. This prompt informs the AI model on how it should respond and reinforces that the model should only draw data from the context. It also ensures that the response from the model is appropriately concise, hints should be neither excessive in length nor unnecessarily complex. The chain completes by sending the response from the model further down the workflow to the Discord integration node. In addition to the main hint-generation chain, the system includes a separate workflow that handles questions that cannot be matched to any assignment context. Rather than using RAG, this workflow provides a short, general cybersecurity explanation suitable for an introductory course and sends the reply directly back to Discord. This ensures that students still receive help for broad or non-assignment questions, without the system relying on restricted assignment materials or producing overly specific hints.

V. QUESTION EVALUATION

The hint generation system will also evaluate the student's questions before taking any next steps. Once a question is received, it will be evaluated to determine whether the

question is on-topic or off-topic, and whether it should trigger an anger response from the system. There are four different paths through which Workflow B could go.

Conceptually, these paths correspond to four categories: a Valid Query Path for questions that are on-topic, an Off-Topic Query Path for unrelated questions, a Misuse Warning Path for inappropriate messages, and a Low-Confidence Handling Path when the system is unsure how to classify the input. These labels help to keep the workflow simple while still reflecting the purpose of each pathway.

Valid Query: If the question is on-topic, the system will proceed through the rest of the workflow and generate a proper response that fits the student's question based on the contents of the Pinecone vector store. **Off-Topic Query:** If the student's question is off-topic, the system will instruct the student to ask a different question. **Misuse Warning:** The student will receive a warning message, and the student's individual misuse level of the system will increase, which in severe cases can lead to the bot becoming unresponsive to the student as a measure to encourage students to be more thoughtful about their questions. **Fallback:** If the system cannot process the question properly, it will inform the student that their question is too unusual for the system to generate a valid response.

With these security measures, we ensure our hint-generation system remains functional and provides the most accurate responses to the questions asked. To mitigate the risk that the agent might make incorrect decisions and generate invalid responses, we chose a more lenient prompt. Specifically, if the agent is unsure whether the question should be elevated, it will always return valid. This reduces the likelihood that our system will overlook a student's question.

Example 1: Typical Use Case Example

This subsection demonstrates when a student misuses the initial syntax and makes adjustments after the hint-generation system informs them that the required context for the question is missing.

Student:	!hint project1 level8
JustinBot3000:	Sorry, I can't answer that. Could you please reformat/clarify what you're trying to ask? Thank you.
Student:	!hint project1 level8 I'm not really sure how to start on this, can I get some help?
JustinBot3000:	Perhaps try using the <code>ls</code> command, which lists directory contents. You could consider adding the <code>-l</code> modifier to display additional details about each item, along with the <code>-a</code> modifier to include hidden files—this might help you reveal a file that isn't shown with a standard listing.

Example 2: PGP and Asymmetric Key Encryption Interaction

In this subsection, we present the hint-generation system that provides students with help for their asymmetric-key encryption assignment.

Student:	!hint project2 both the <code>-refresh-keys</code> and the <code>-send-keys -keyserver</code> haven't worked. The refresh always says "not changed," and <code>-send-keys</code> shows "sending key..." but nothing is ever sent to the other person, or vice versa.
JustinBot3000:	Sorry, I can't answer that. Could you please reformat/clarify what you're trying to ask? Thank you.
Student:	!hint project2 in a previous message, you told me to use the commands <code>-refresh-keys</code> and <code>-send-keys -keyserver</code> to send and receive the signatures, but none of those commands work when run. How should I go about sending and receiving signatures if these don't work?
JustinBot3000:	Perhaps try using the <code>gpg</code> command, which manages your cryptographic keys. One option you might consider is the modifier <code>--refresh-keys</code> ; this instructs <code>gpg</code> to update your keys by checking with a keyserver. Similarly, you could experiment with the modifier <code>--send-keys</code> , which tells <code>gpg</code> to transmit your key to a keyserver. It might be worthwhile to ensure your internet connection is active and that the keyserver at <code>keyserver.ubuntu.com</code> is reachable. If problems continue, you might consider reviewing your GPG configuration or trying an alternative keyserver.

Example 3: Project 4 Interaction

In this subsection, we present a case study of the application of the hint generation system to help students engage in password cracking.

Student:	!hint project4
JustinBot3000:	Sorry, I can't answer that. Could you please reformat/clarify what you're trying to ask? Thank you.
Student:	!hint project4 wordlists, permutation rules
JustinBot3000:	Sorry, I can't answer that. Could you please reformat/clarify what you're trying to ask? Thank you.
Student:	!hint project4 how do I use wordlists and custom permutation rules to help crack passwords with john the ripper?
JustinBot3000:	<p>To crack passwords with John the Ripper using wordlists and your own custom permutation rules, start by working with several dictionaries—ideally, smaller lists that capture the most common passwords. The idea is to mimic how people typically construct passwords (e.g., a word appended with digits or special characters). Here's what you can do:</p> <ul style="list-style-type: none"> • Begin with a basic dictionary attack using a common wordlist. • Use custom rules to extend these words by appending digits or special characters. For example, try modifying words by adding one digit first, then one special character, and later even combining both. • To see what rules John supports (and to plan your custom permutations), run the command <code>john--list=rules</code>. This will give you a list of available rule modifiers. • Build your approach step by step. Start small with simple modifications, and then increase the complexity gradually. This helps manage performance, as using very complex or exhaustive rule sets can significantly slow down your cracking speed. <p>As discussed in the lectures, think creatively about the common ways people create their passwords, and use what you learn in class to craft your custom rules accordingly. This iterative approach—using multiple dictionaries along with controlled, progressive rule modifications—should help you systematically crack more passwords.</p>

Example 4: Hint Provided With Cautionary Note

The hint-generation bot distinguishes itself by issuing a cautionary note when asked a question outside the available context. While the system will still attempt to provide an answer, it will include a cautionary note indicating that the hint you asked for may be out of scope, helping students distinguish whether the particular hint is needed.

Student1:	!hint project1 level9 are we supposed to transfer a file we made on the VM to a challenge server or use the preexisting README?
HintGenBot:	You need to transfer the text file you created on your VM to the SSH host (challenge server). There's no need to use the preexisting README.
Student2:	!hint project1 level9 what is the challenge server
HintGenBot:	A challenge server is a remote machine set up specifically to host cybersecurity exercises. It provides a controlled environment for you to practice techniques like identifying vulnerabilities and testing exploits without affecting real-world systems. (Note: The hint you asked for may be out of scope for the project you are working on. As such, the information may be incorrect, and you may not need it to finish the project.)

VI. PRELIMINARY RESULTS

During the initial deployment of the hint-generation system in the Fall semester, we collected usage data across approximately 150 students enrolled in an introductory cybersecurity course. Analysis of system-generated responses and student interactions revealed several encouraging trends regarding both classification accuracy and hint quality.

First, the system demonstrated strong performance in categorizing student queries. As shown in the usage analysis, a majority of student messages were classified as valid and relevant to the assignment context. Of all categorized questions, 56.5% were identified as positive/on-topic, while only 3.4% were marked negative and 9.5% as neutral. The system produced no response for roughly 30.6% of messages, corresponding largely to malformed or incomplete queries that lacked sufficient detail for contextual grounding. Despite this, students routinely revised and resubmitted their questions after receiving clarification prompts, suggesting that the system effectively encouraged reflective question-posing.

Across the 396 hints generated during the evaluation period, the majority (69.7%) were deemed valid and contextually appropriate for the associated assignment. The remaining hints fell into three smaller categories: 11.6% were out of scope, 10.9% required clarification from the student, and 7.8% were associated with an invalid or unrecognized project tag. These distributions indicate that, while the system

occasionally required corrective input, the overall workflow reliably produced meaningful guidance that supported students without revealing full solutions.

Qualitatively, classroom observations showed that students were willing to iterate on their questions to obtain more specific hints. In several cases, students refined vague or overly broad queries through multiple interactions with the system until a targeted hint was produced. This aligns with our design goal of promoting active engagement and metacognitive awareness rather than passive answer-seeking. Although formal student feedback was not collected during this pilot phase, chat logs show that the generated hints frequently helped students progress in their assignments and reduced instructor bottlenecks during high-traffic periods.

VII. DISCUSSION AND CONCLUSION

The hint-generation system has undergone initial pilot testing in one remote asynchronous course offering. We are currently experimenting with a new cohort of students enrolled in the in-person course offering. We have adapted the hint-generation system for class assignments, and preliminary user interaction and usage data indicate that it can answer most students' questions about the homework assignment without additional human intervention, provided the question is sufficiently detailed. During the trial run with remote asynchronous students, we were unable to collect user feedback on the hint-generation system due to limited use. However, since then, the system's adaptation over the past three weeks of the in-person cohort has shown promise, and the current findings are derived from the bot's chat history on Discord, where the classroom discussion forum was hosted, along with user surveys included as part of a midterm exam question. We examined the chat history and concluded that this hint-generation system has been and will continue to be helpful to students. Students attempted to rephrase their questions whenever the chatbot requested further clarification, and the hints provided were sufficient to help them progress on their assigned work in most cases, as shown in Examples 1-4.

In subsequent semesters, we will adapt this tool across all cybersecurity foundation course offerings and collect student feedback after its use to determine the system's effectiveness in learning reinforcement. In addition to homework assignments, we anticipate using this tool to assist student learning during impromptu in-class assignments. Given that the needs of the students may differ between working on interactive activities in class versus when doing homework assignments, we will be using an evaluation survey to determine the accuracy of the hint, the completeness and helpfulness of the hint it provides, the frequency in which it asks the user for clarifications among other measurements to determine the effectiveness and receptiveness of the tool in regular course offerings.

We also want to take a moment to discuss how the developed hint-generation system differs from students

prompting a broadly trained LLM. The primary difference lies in the context in which our hints are generated. Leveraging RAG, we can generate and derive actionable hints from assignment descriptions, rubrics, and solution manuals. In Example 4, we illustrate the difference between the answers the bot provides when students ask a question related to the homework assignment versus a poorly phrased question lacking in context. In our hint generation system, we added an explicit disclaimer that shows up when student's questions are not detailed enough or are lacking context which states: "(Note: The hint you asked for may be out of the scope for the project you are working on. As such, the information may be incorrect, and you may not need it to finish the project.)" This indicates that the underlying, broadly trained LLM is in use and may be providing answers to the question, though not necessarily in the correct context. Students should revise their question to ask something within the assignment's context, or proceed with caution if the response they receive includes the disclaimer message.

VIII. FUTURE WORK

This system is currently capable of supporting course assignments and in-class interactive hands-on activities related to cybersecurity exercises. We foresee this tool being adapted for use in other disciplines to enhance students' learning experience, without being overly reliant on large, broadly trained language models. This tool will also be a valuable resource for instructors, ensuring that students receive the necessary assistance even when instructors and teaching staff are unavailable. We also anticipate leveraging student usage data to continuously optimize the bot's output and to create additional features that allow it to answer general cybersecurity domain questions to some extent or offer a list of helpful resources to students who wish to learn more about particular subject domains. We believe this work has the potential to help instructors and students engage in meaningful learning activities, especially in the rapidly changing higher education landscape following the introduction of large language models. This work aims to leverage the accessibility of LLM APIs to help educators more effectively support students' learning throughout their academic journey.

REFERENCES

- [1] "Pinecone." [Online]. Available: <https://www.pinecone.io/>
- [2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.
- [3] "n8n." [Online]. Available: <https://n8n.io/>
- [4] D. Sun, A. Boudouaia, C. Zhu, and Y. Li, "Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? an empirical study," *International Journal of Educational Technology in Higher Education*, vol. 21, no. 1, p. 14, 2024.
- [5] A. Létourneau, M. Deslandes Martineau, P. Charland, J. A. Karran, J. Boasen, and P. M. Léger, "A systematic review of AI-driven intelligent tutoring systems (ITS) in K-12 education," *npj Science of Learning*, vol. 10, no. 1, p. 29, 2025.
- [6] A. C. Yang, J.-Y. Lin, C.-Y. Lin, and H. Ogata, "Enhancing python learning with PyTutor: Efficacy of a ChatGPT-based intelligent tutoring system in programming education," *Computers and Education: Artificial Intelligence*, vol. 7, p. 100309, 2024.
- [7] R. Liu, C. Zenke, C. Liu, A. Holmes, P. Thornton, and D. J. Malan, "Teaching CS50 with AI: leveraging generative artificial intelligence in computer science education," in *Proceedings of the 55th ACM technical symposium on computer science education V. 1*, 2024, pp. 750–756.
- [8] M. Kazemitabaar, R. Ye, X. Wang, A. Z. Henley, P. Denny, M. Craig, and T. Grossman, "Codeaid: Evaluating a classroom deployment of an LLM-based programming assistant that balances student and educator needs," in *Proceedings of the 2024 chi conference on human factors in computing systems*, 2024, pp. 1–20.
- [9] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascara'n, and Á. Fernández-Leal, "Human-in-the-loop machine learning: a state of the art," *Artificial Intelligence Review*, vol. 56, no. 4, pp. 3005–3054, 2023.
- [10] M. Ninaus and M. Sailer, "Closing the loop—the human role in artificial intelligence for education," *Frontiers in psychology*, vol. 13, p. 956798, 2022.
- [11] J. Swacha and M. Gracel, "Retrieval-augmented generation (RAG) chatbots for education: A survey of applications," *Applied Sciences*, vol. 15, no. 8, 2025. [Online]. Available: <https://www.mdpi.com/2076-3417/15/8/4234>
- [12] Google, "Google drive." [Online]. Available: <https://drive.google.com/>
- [13] J. Citron and S. Vishnevskiy, "Discord." [Online]. Available: <https://discord.com/company>