Supporting Self-Directed Learning with AI: Improving Student Interaction with Technical Documentation

Aidan Szuch, Akanksha Girdhar, and Andrew DeOrio
amszuch@umich.edu, agirdhar@umich.edu, awdeorio@umich.edu
Department of Electrical Engineering and Computer Science
University of Michigan

Abstract

With the rapid growth of generative AI chatbots, use of these tools by students is likewise increasing. Students often encourage chatbots to provide direct solutions, undermining key learning goals, rendering assignments ineffective, and bypassing reliable source materials. This paper seeks to embrace shifting preferences towards AI assistance, offering a chatbot design that encourages better practices for students learning new software libraries.

We developed a chatbot intended to assist students with navigating and understanding technical documentation. The bot uses course project specifications and developer documentation as context for the GPT-4o[1] model. The system is tasked with assisting student learning in a similar manner to course instructors. Additional design features include automatic highlighting of related text in the corresponding documentation, instructor-provided starter prompts to guide appropriate use, and support for sharing conversations to encourage collaboration. These features are intended to emulate strategies teaching assistants often employ to aid students while promoting self-directed exploration. The goal is to encourage direct student interaction with developer documents and offer an alternative course resource that can provide assistance similar to instructional staff at any time.

To evaluate our system, we deployed it as a resource during the final project of an upper-level undergraduate computer science course. One of the goals of the course is that students learn to use developer documentation. We collected server usage logs, surveyed the user population, and compared our chatbot with a generic online GenAI tool to inform our results. Our study included 448 students across 186 project groups. We analyzed over 680 chatbot interactions logged during a three-week deployment period and compared responses using expert evaluation on a sample of 93 student prompts. This comparison considered response helpfulness, appropriateness, and the presence of misleading information or code snippets. This analysis provided insight into how students used the tool in practice and how the quality of its responses compared to those from a general-purpose chatbot.

Overall, students reported increased comfort with documentation, with 48% responding positively and 41% responding neutrally. Our bot returned misleading information at a rate of 14%,

compared to ChatGPT's[2] 47%, and gave high quality responses twice as often. These findings suggest that context-aware generative AI systems are often more reliable than their generic counterparts and may promote more productive learning behaviors. Embedding such tools in computing courses may be a promising approach to AI adoption and warrants further study.

1 Introduction and Related Work

The rapid rise of generative AI has had a significant impact on how students engage with academic content. Recent surveys show widespread adoption of these tools among students. In 2024, a survey of Harvard undergraduates revealed that the majority of students regularly used ChatGPT, often for answering general questions or getting help with writing, emails, and programming assignments[3]. A report from OpenAI disclosed that more than one-third of U.S. college students use ChatGPT, with approximately 25% of their interactions relating to learning and school work[4]. As these tools become more widely available and more frequently used in educational settings, it is imperative to understand how to guide their use in ways that support, not substitute, learning.

The integration of generative AI tools in education has introduced new opportunities and challenges. Prior research has emphasized their potential to improve both teaching and learning outcomes. Generative AI tools offer benefits such as instant access to information, natural language interaction, and scalable support outside of traditional methods of instruction[5,6,7]. A key advantage is the ability to respond to natural language queries, allowing students to interact with the tool similarly to how they would engage with a tutor[7]. In computer science education, generative AI has shown early promise in helping students understand syntax, clarify error messages, and generate basic code[6].

However, the impact of these tools on student learning is largely shaped by usage patterns, and there are several major challenges that arise when incorporating these tools into educational settings. While AI tools can serve as effective supplements to instruction when paired with appropriate guidance[8], in other contexts they may act as a barrier to learning. Prior work has highlighted concerns over the risk of misinformation, lack of domain awareness, and over-reliance on AI-generated content[7,8]. A study found that students using ChatGPT performed better and completed tasks more quickly than those without, but inconsistencies in AI-generated code prevented perfect scores[8].

To overcome these challenges, recent work has focused on the development of custom AI tools tailored to specific programming environments. For instance, some projects have created generative AI chatbots specifically trained on course resources[9]. Others have used large language models to generate context-aware compiler error explanations to help students understand and debug code issues[10]. Additionally, some tools have been designed to emulate student-teacher interactions by offering tailored guidance[11]. Such domain-specific tools aim to reduce hallucination and errors and provide contextually relevant support.

Previous research in intelligent tutoring systems has demonstrated the importance of supporting student learning without undermining productive struggle[12]. In an effort to balance cognitive and motivational scaffolding, our tool prefers strategies such as guided prompts, reflective questions, and redirection to relevant materials rather than providing complete solutions.

The challenge of student over-reliance on generative AI remains. The specific challenge we take on in this paper is student use of generative AI when learning a new software library. Our goal was to design a tool that encourages students to actively engage with authoritative source documentation as part of their problem-solving process.

2 Contributions

We developed a custom chatbot designed to help students learn how to navigate and understand original authoritative technical documentation. By highlighting pertinent documentation sections and suggesting targeted prompts, our tool encourages students to actively engage with technical documentation rather than rely solely on AI-generated answers. By deploying the chatbot as a course resource, we explore how such tools can support self-directed learning and improve students' comfort with using technical documentation. This study evaluates the tool with respect to three primary data sources: server usage logs, student surveys, and expert comparison with ChatGPT.

Our research questions are:

- How does our context-aware generative AI tool affect self-reported student comfort with developer docs?
- Does the tool avoid giving solutions?
- What are the differences in response quality between our tool and a general purpose tool?

3 Methods

In this section, we describe how we deployed and evaluated our tool in the context of a course project. The system features a context-aware chatbot paired with a side-by-side documentation display, automatic highlighting of relevant content, and the ability to share conversations with other users. We provide background on the course and assignment where the tool was used. We walk through the system design and configuration, including key features like documentation highlighting and instructor-provided prompts. Finally, we explain how we gathered student feedback and compared our chatbot to ChatGPT to gauge the tool's effectiveness.

3.1 Course and Project

This study was conducted during the final project of a high-enrollment, upper-level undergraduate computer science course at a large public research university. The course covers modern web systems and technologies, including front-end and back-end development. The course emphasizes reading online technical documentation to learn necessary components while working on the project.

The project in this study involves building a scalable search engine. The learning goals of the project include information retrieval concepts, like text analysis (tf-idf) and link analysis (PageRank), and parallel data processing with MapReduce. In order to complete the project,

students are required to consult three different Python technical documentation sources: *requests*[13], *threading*[14], and *heapq*[15].

3.2 System Architecture and Implementation

Our system consists of a web-based interface featuring a context-aware chatbot integrated directly alongside developer documentation. A key feature of the interface is its ability to automatically highlight relevant sections of the documentation in response to user queries, helping students locate useful information more efficiently. The chatbot is provided with the project specification and relevant documentation pages as context. The interface (Figure 1) displays the chatbot and the documentation side by side, allowing students to interact with the assistant while viewing relevant technical documentation.

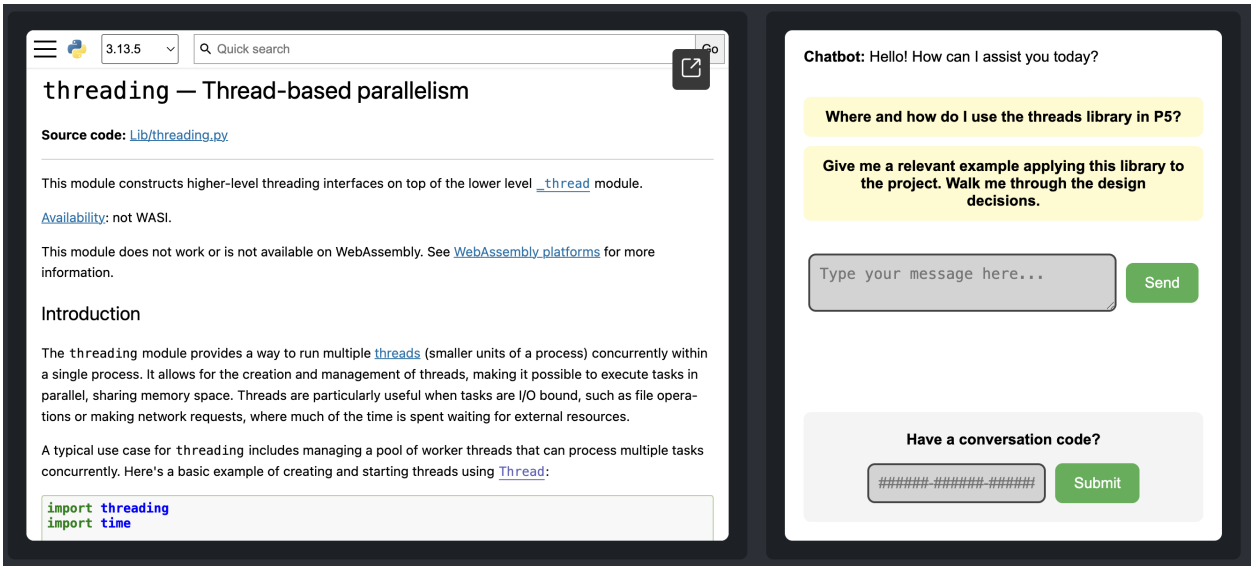

Figure 1: Our tool's interface displays a context-aware assistant alongside technical documentation. The context includes both the student project requirements and the technical documentation.

Figure 2 provides a diagram of the system architecture. It illustrates how our bot retrieves relevant documentation, bundles it with the project specification and student prompt, generates a response, and then uses the response to identify and highlight key sections of the documentation before returning the response to the user. This enables student interaction with the source documentation, rather than relying solely on the chatbot's answer.

3.3 Chatbot Configuration

Our system uses the GPT-4o[1] model hosted on Azure OpenAI, with a temperature setting of 1 on a 0-2 scale. Lower values make the model's responses more deterministic and focused, while higher values introduce more creativity. A temperature of 1 was selected to balance focused and concise answers with varied responses that can help address different student needs.

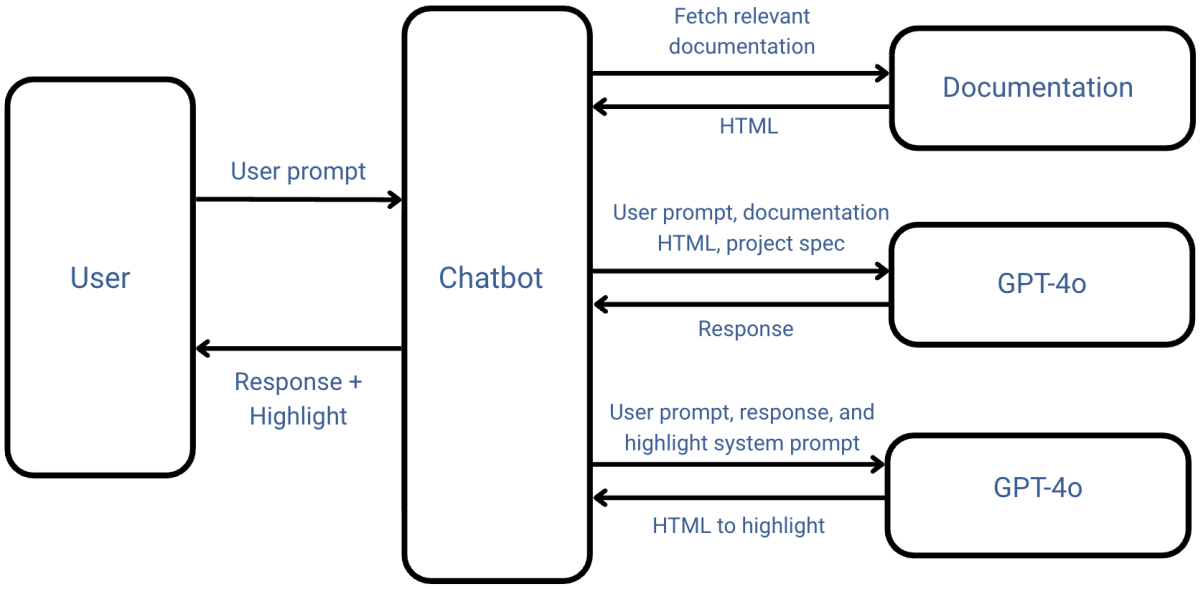

Figure 2: System Architecture. The chatbot first retrieves the relevant documentation and bundles it with the project specification and student query. This combined input is sent to the language model to generate a response. The response and query are then sent back to the model to identify key sections of the documentation to highlight before returning the final response to the student.

3.4 Instructor Provided Prompts

The tool enabled instructors to provide clickable sample prompts directly in the interface to guide students toward using the tool effectively. For our study, the instructors provided the following prompts designed to teach students how to engage with the documentation more thoughtfully. Note that <doc> was replaced with the name of one library: *requests*, *threading*, or *heapq*.

- Where and how do I use the <doc> library in the project?
- Give me a relevant example applying this library to the project. Walk me through the design decisions.

3.5 Response Generation

The chatbot generates responses based on the following system prompt to ensure brief and project-specific responses tailored to the documentation being viewed. We constrained generated code in the responses to encourage students to engage with the documentation, rather than simply copying and pasting from the chatbot's response. The chatbot was designed to be conversational, so the model was provided with past prompt-response pairs within the conversation.

When designing the system prompt, early iterations generated extended responses with large blocks of code and little explanation. As we made adjustments to encourage more productive instructional behavior, another common feature of responses was the inclusion of alternative methods or functionalities that fell outside the scope of the assigned project. This led us to

include the entire project specification in the prompt to keep responses focused and relevant. In the prompt below, <spec html> was replaced with the HTML of the project specification, <doc>.html with the filename of the documentation page being viewed, and <doc html> with the HTML of that documentation page.

> System Prompt: *You are a helpful teaching assistant for a web development course. The project the student is working on: <spec html>. They are looking at the site created by <doc>.html: <doc html>. Make the response VERY brief and to the point. Try to be as specific to the project as possible. DO NOT give code. Don't give extra features/functions not mentioned in the project description.*

3.6 Automated Documentation Highlighting

After the chatbot generates a response to the student's prompt, this response is used to identify the relevant HTML components in the documentation. The prompt and response are fed back into the model to generate the most pertinent excerpts from the documentation's HTML to be highlighted. The front-end then displays the generated response and highlights the relevant HTML, if provided, within the documentation. Figure 3 shows an example of the relevant HTML sections highlighted in the context of the chatbot's response.

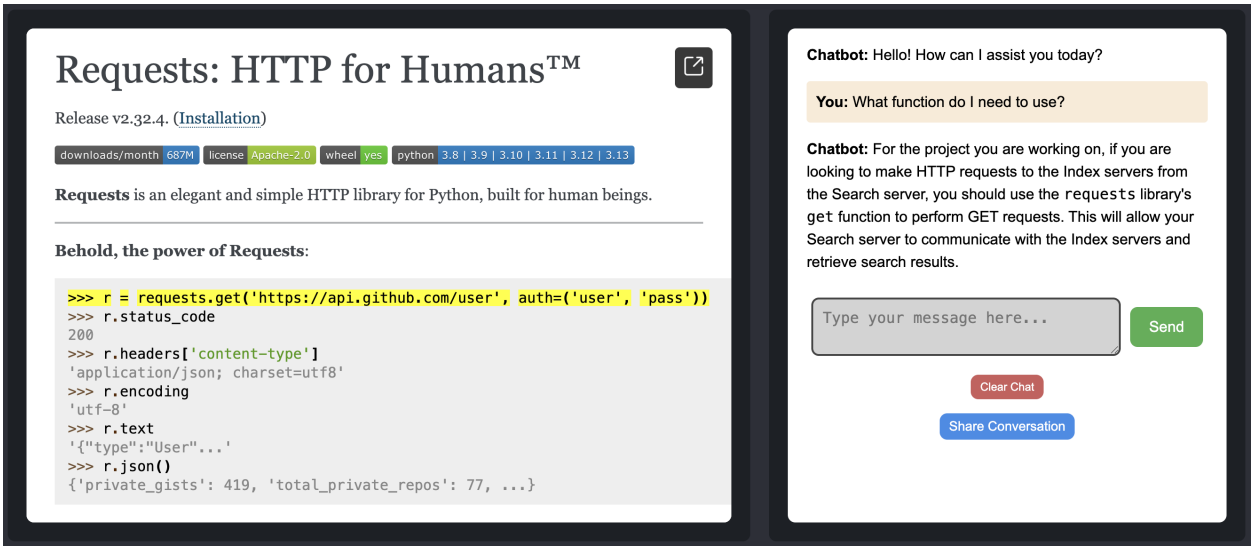

Figure 3: The tool highlights relevant sections of the documentation side-by-side with responses to student queries.

3.7 Conversation Sharing

Due to the collaborative nature of these projects, our system allows users to share their ongoing conversations. Users can generate a share code that can be used by other students to view the entire exchange. The conversation can then be continued without affecting the original conversation. This feature not only enables collaboration, but also provides instructors with a way to share examples of how to best use the tool for navigating documentation.

3.8 Survey

To assess the effectiveness of the chatbot, we administered a survey to students who were required to use this tool. The survey was designed to gather feedback on the students' experience with using the chatbot, their comfort level with documentation, and how the chatbot influenced their behavior and learning process. Additionally, the students were asked for optional open-ended feedback on their experience. Surveys were collected by group, and the full survey is provided in Appendix 1.

After collection, we filtered out empty or invalid submissions and any submissions that indicated that they did not use the chatbot in the open-ended feedback section (see section 4.1).

3.9 Comparative Analysis with ChatGPT

We conducted a qualitative comparison between our context-aware chatbot and a general-purpose chatbot, ChatGPT[2]. We randomly sampled a set of student-initiated prompts submitted to the chatbot during the project and used them as input to ChatGPT for comparison. The student prompts were filtered to exclude the instructor-suggested queries and any follow up questions from ongoing conversations. For each of the filtered prompts, a new ChatGPT session was started. The responses from both chatbots were then evaluated based on a rubric defined in section 4.3. The evaluation was conducted by a team of two experts: graduate students proficient in the course material. The dataset was split evenly between the two experts, and each expert independently evaluated their assigned subset of question–response pairs using the same rubric. Although our experts did have knowledge of which response was generated by which tool, the rubric was designed to mitigate bias as much as possible.

4 Results

The results provided below include data sourced from server logs, student surveys, and a comparative analysis of our tool and ChatGPT's publicly-available offerings. The server logs were anonymized to protect student privacy, which also prevented evaluation of survey response data with usage data applied as context.

4.1 Population Statistics

Our dataset was collected from a population of 448 students enrolled in one course during the Winter 2025 semester. Students worked in groups of 2-3, with 186 groups registered. Out of the 186 groups, we received 180 valid survey responses. We filtered out surveys that explicitly indicated the group did not use the experimental tool while working on the project, retaining 159 valid responses for our analysis.

4.2 Student Usage

We collected anonymized logs of student inputs and our bot's responses for evaluation. During the three week project cycle, we noted page visits from 227 distinct browser sessions and 233 distinct conversations (consisting of at least one client query). Across these 233 conversations,

there were 681 prompt-response pairs, 136 of which were from the instructor-suggested prompts. Our tool highlighted related content in the documentation 197 times.

Figure 4 shows the number of messages each client submitted to the chatbot. We note that 82 of the 227 unique browser sessions did not query the bot at all, comprising 36.12% of browser sessions. These empty chats may be students familiarizing themselves with the project specification and navigating to the tool before they were ready to use it. 14 queried at least 10 times, comprising 6.17% of browser sessions. Figure 5 shows the number of student queries per conversation with the tool. 47.21% of conversations consisted of a single prompt-response pair, while 6.87% of conversations were 8 or more query-response pairs long.

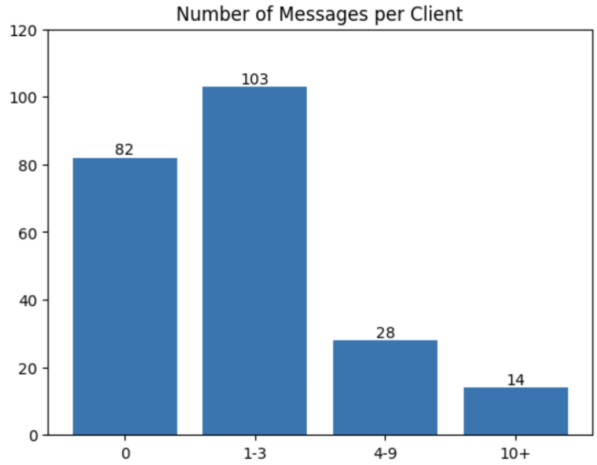

Figure 4: Number of messages each unique browser session submitted to the chatbot. Most sessions had at least 1-3 prompts.

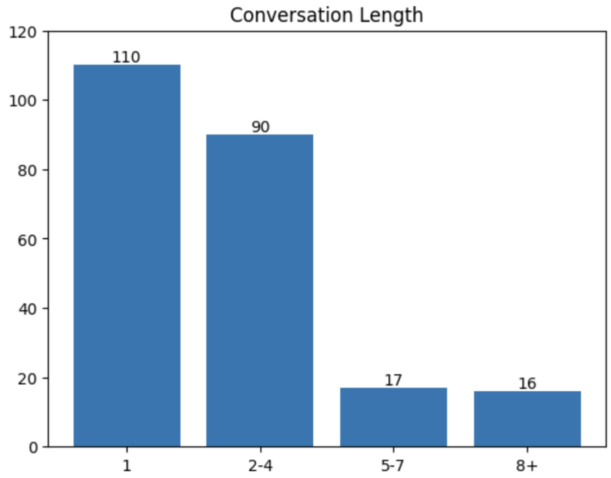

Figure 5: Number of student queries per conversation with the chatbot. Most users utilized the chatbot's conversational capabilities.

4.3 Survey Results

Figure 6 depicts students' self-reported comfort level with documentation before using our tool. 39% of students reported high or very high comfort approaching documentation, 47% felt neutral, and 14% had low or very low comfort with these resources. As the project in our study came late in a course where usage of documentation is necessary, we can see that a smaller number of students reported lower comfort levels.

Figures 7 and 8 show student survey response data regarding their experience with our chatbot and tools they typically rely on when learning new code libraries, respectively. According to our survey, 39% of students reported that the bot helped with their navigation of documentation for the project, while 36% felt the tool encouraged them to read the provided documentation. 48% of students felt the bot increased their comfort level with documentation, while 11% disagreed with that statement. 66% of students use documentation often or always in their learning, while 43% rely on generative AI often or always.

The open-ended responses provide additional context for these results. Students who reported positive experiences described the chatbot as useful for clarifying project requirements, narrowing documentation searches, and confirming their understanding before implementation. Several noted that its project-specific context distinguished it from general-purpose AI tools. However, a substantial subset of students reported minimal engagement or preference for external generative AI systems. Reported pain points included constrained code generation, occasional loss of conversational context, response generality, and interface design.

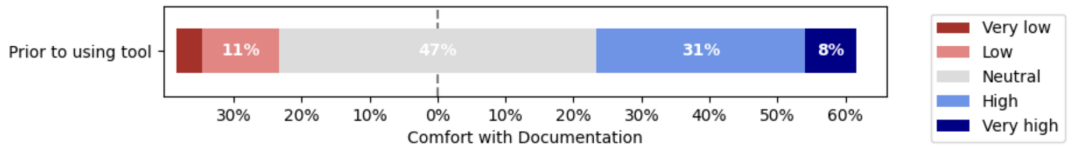

Figure 6: Students' initial comfort level with documentation

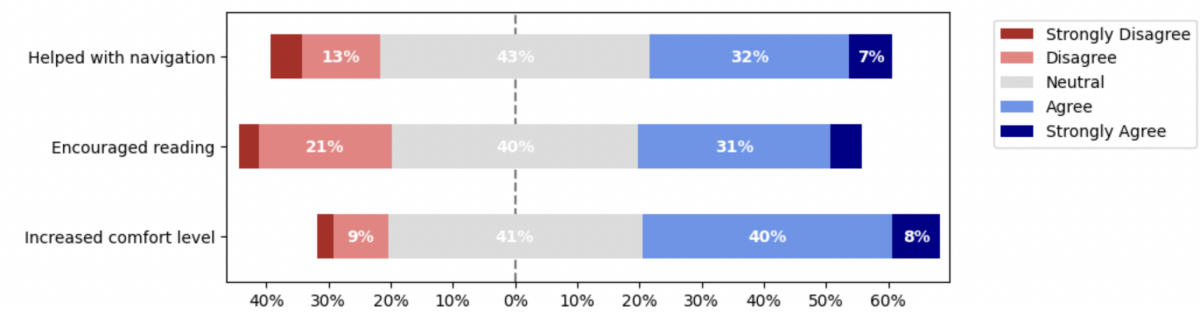

Figure 7: Student perceptions of how the tool influenced their behavior, whether the tool helped with navigation, encouraged reading of documentation, and increased comfort level using technical documentation. Most students agreed that the tool improved their comfort level and assisted with navigation, but fewer strongly agreed that it encouraged reading.

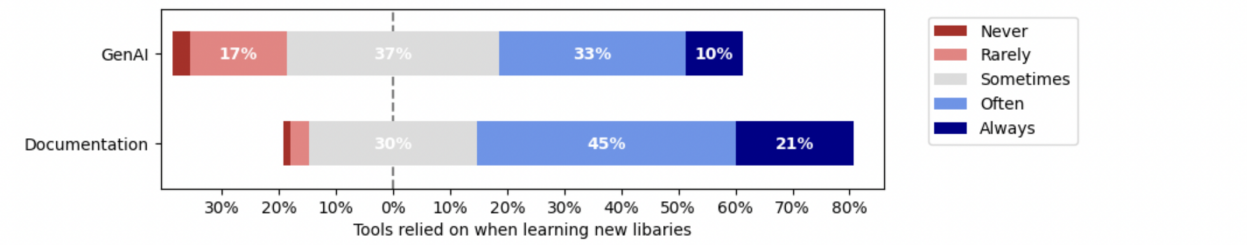

Figure 8: Tools typically relied upon by students learning new code libraries, comparing the dependence on GenAI vs technical documentation. A higher percentage of students reported relying on documentation, though the use of GenAI was also prominent.

4.4 Expert Analysis

Our expert analysis looked specifically at user-submitted queries that were the first of a conversation. We excluded instructor-provided starter queries suggested on the user interface. This was to encapsulate real world use cases rather than instructor-led activity with these tools, while accounting for their conversational nature, which allows context to rapidly diverge for later messages. We used 93 of the 121 possible options, selected randomly, to achieve 95% confidence with a 5% margin of error for our results.

To evaluate the extent to which each chatbot promotes learning, we assessed responses on a 1-5 scale, where '1' correlated to the chatbot providing incomplete or poor answers, '3' meant the bot provided helpful information or examples but allowed sufficient room for student-guided follow-up, and '5' meant the bot answered the question in full. Using this scale, an optimal response for most relevant questions would be rated '3.' When comparing the Likert scale plots (Figure 9) for our chatbot and ChatGPT response qualities, our tool had significantly more responses rated '3' and a balanced shape, while ChatGPT was right-skewed with a comparatively high number of responses rated '1.'

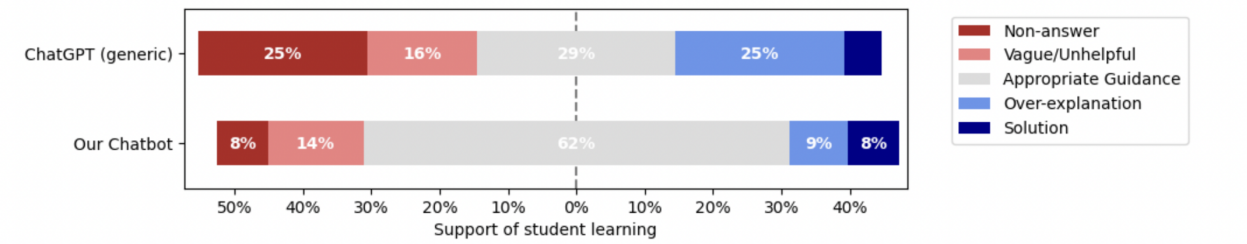

Figure 9: Expert analysis of the quality of responses by ChatGPT vs our tool given identical student prompts. Our tool more often provided appropriate guidance, while ChatGPT responses were frequently vague, unhelpful, or revealed the answer outright.

As shown in Table 1, we also collected quantitative measures of how often each bot gave misleading information to students or provided solution code. In this study, misleading information includes statements that are untrue about the project and its context, as well as suggested approaches that we deemed beyond the scope of the course. Rather than being representative of absolute truth, this metric is intended to convey whether the bot's response is

appropriate for a student working on the class project. Similarly, the metric of containing code required that the bot construct solution code that was more complex than simply regurgitating distinct functions from the documentation.

Across the 93 queries sampled, our chatbot's responses were deemed misleading 13.98% of the time, while ChatGPT gave misleading responses at a rate of 47.31%. The chatbot also gave code in its responses 18.28% of the time, with an average length of 2.06 lines, whereas ChatGPT gave code in 63.44% of its responses with an average length of 13.68 lines.

Table 1: Frequency of Undesired Behaviors in Expert Analysis. Of the 93 prompts submitted to each tool, our chatbot gave misleading information less frequently than ChatGPT. ChatGPT also gave more code samples, which were longer in length on average.

	Misleading Info	**Contains Code**	**Lines of Code**
ChatGPT	44	59	13.68
Chatbot	13	17	2.06

5 Discussion

We discuss our results and their relationship to our research questions. To further contextualize our results, it is worth noting that a majority of students responded neutrally or positively to statements that this tool aided navigation of, encouraged reading of, and increased their comfort with documentation, which were primary behaviors we hoped to foster. This analysis seeks to go one step further and evaluate the quality of the chatbot as an instructional tool.

5.1 Self-Reported Comfort

Overall, our results suggest that our tool is effective in increasing student comfort with documentation. From our survey, 48% of students indicated positive effects on their comfort level from their usage of the chatbot, while an additional 41% responded neutrally.

We also noted in free-form feedback that some students found the interface overwhelming, which may account for the 11% of students who disagreed that the chatbot increased their comfort with technical documentation. As our survey results and usage logs are anonymized for student privacy, it is difficult to determine the extent to which these students interacted with the tool or whether their behavior matched our expectations.

5.2 Promoting Student Learning

One of the primary goals when designing the chatbot was to reduce the extent to which the bot returned complete solutions or long segments of code, preferring answers similar to those instructional staff would provide to point students in the right direction. These preferences surface as direct instructions in the system prompt (see section 3.5). Our expert analysis was also framed in this context; we deem responses offering guidance without providing solutions as promoting learning.

Perhaps the most compelling result is that, of the 93 student prompts given to our chatbot and off-the-shelf ChatGPT, our chatbot returned 58 responses rated at the optimal level ('3', appropriate guidance), compared to 27 from the GPT model. This demonstrates clear superiority of our tool at aiding students at an appropriate level, and because the queries used in this analysis came directly from real students, this sample is representative of student behaviors in practice.

5.3 Response Quality

Lastly, we evaluated the quality of responses given by our chatbot compared to generic generative AI tools (ChatGPT-4o[1]). While we have determined that the system prompt above encourages helpful yet constrained response generation, it is also important that generated responses are largely reliable. As shown in Table 1, only 13 of the 93 evaluated chatbot responses contained information deemed to be misleading, compared to 44 from ChatGPT. This marks a substantial improvement, which we believe to be largely due to context provided to the system that was often absent from student queries. With less misleading information, we have greater confidence that our chatbot is an appropriate instructional tool that will return quality responses to students.

Another priority for the design of our chatbot was limiting code generation to encourage students to write lines of code themselves. Across the 93 evaluated queries, our chatbot gave substantial code only 17 times, with an average length of 2.1 lines, whereas ChatGPT gave code 59 times with an average length of 13.7 lines. These results indicate that our tool was intentional in the code it provided, only sharing necessary components or small steps to help students. ChatGPT, on the other hand, was more likely to give large chunks of code, allowing students to bypass learning or rely on code that had not been carefully designed or reviewed. This further supports the claim that our chatbot can give high quality and appropriate responses for educational applications.

Upon review of the server logs, one common student behavior we identified was sharing code with little context and asking the generative AI tool to verify its validity. The following case was one such instance, and acted as one of the 93 student queries evaluated. In this query, the student asked "would this implementation work?" followed by a block of about 20 lines of Python code. While both bots identified small syntax errors that would have caused the code to fail upon running, our chatbot noted that the code "appears to be consistent with the project's requirements" before listing said requirements and ultimately warned the student, whose implementation included components beyond the scope of the project, to avoid overextending beyond the requirements listed in the specification. ChatGPT was unable to provide such advice and offered another approach that strayed further from the project specification. While this case study is not representative of all queries students may submit to these tools, it does highlight the value of providing course-related context to ensure generated responses do not mislead students who are seeking quick evaluation of their approach.

6 Limitations

Several limitations impact the generalizability of our results. It is possible that students who meaningfully used the tool represent a self-selected subset of the population who were already

more inclined to engage with AI support or documentation. Although the project required students to access the tool and submit usage logs, this requirement does not guarantee equivalent depth or quality of engagement across groups.

Students also received course credit for completing the survey, which may have influenced the accuracy of responses. Some students may also have completed the survey without using the tool in a meaningful way. We did filter out responses where the student explicitly indicated they did not use the tool. Encouraging greater student usage and organic reporting would be worthwhile considerations for future work.

Additionally, because usage logs and survey responses were anonymized, we were unable to directly correlate individual interaction patterns with reported perceptions. For instance, 82 browser sessions accessed the interface but submitted no messages. This behavior may reflect students simply familiarizing themselves or, alternatively, finding the interface overwhelming or unclear. Without direct linkage between survey feedback and usage data, we cannot determine whether low-engagement users were disproportionately represented among those reporting neutral or negative perceptions.

The tool was introduced near the end of the course during the final project. By this point, students had already worked on four programming projects where they were expected to reference the technical documentation directly. In a future study, it would be worthwhile to incorporate a similar tool for each project over the duration of the course to measure student growth.

We did not explicitly manage student expectations in regards to using this chatbot. Some students may have expected direct answers or code solutions, which the tool was intentionally designed to avoid. This mismatch may have affected perceptions of the tool's usefulness. Additionally, while the system prompt was designed to restrict code generation, the chatbot did occasionally return small code snippets. These responses averaged around two lines and did not provide complete solutions that could be directly copy-pasted.

Although the documentation provided was technically complex, students were only required to use a single function from each page. This limited the extent to which we could evaluate the tool's effectiveness in supporting broader documentation navigation.

7 Conclusions

Overall, our chatbot effectively encouraged student learning rather than reliance on AI-generated solutions. Many students reported increased comfort and better navigation of developer documentation, indicating the tool met its core objective of guiding students to engage more meaningfully with technical literature. We also note the high tendency of the bot to give responses similar to those of instructional staff.

Compared to general-purpose AI tools, our chatbot proved to be more accurate, context-aware, and educationally appropriate in its responses. This included a reduction in misleading information and limited code generation. By avoiding solutions and emphasizing relevant information for the course and project, the bot allowed students to meet defined learning goals. Future work may include expanding the tool to apply to all course projects to collect data on its

efficacy throughout the course and encourage instructive student interactions with AI earlier in their learning.

Appendix 1 - Survey

This section lists the questions from the student survey instrument we used for evaluation.

Background

1. What is your comfort level with reading and using developer documentation? [Very low, Low, Neutral, High, Very high]
2. How often do you rely on developer documentation to learn new libraries? [Never, Rarely, Sometimes, Often, Always]
3. How often do you rely on GenAI tools, like ChatGPT, to learn new libraries? [Never, Rarely, Sometimes, Often, Always]

General

4. This tool was helpful for navigating documentation. [Strongly disagree, Disagree, Neutral, Agree, Strongly agree]
5. This tool encouraged me to read documentation. [Strongly disagree, Disagree, Neutral, Agree, Strongly agree]
6. My comfort level with reading developer documentation increased after using this tool. [Strongly disagree, Disagree, Neutral, Agree, Strongly agree]
7. Compared to GenAI tools like ChatGPT, how many prompts did it take to get your answer? [Significantly more, Slightly more, About the same, Slightly less, Significantly less]
8. (Optional) Overall, do you have any feedback on the bot?

References

[1] OpenAI, A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh *et al.*, "Gpt-4o system card," 2024. [Online]. Available: https://arxiv.org/abs/2410.21276

[2] OpenAI, "Chatgpt: Optimizing language models for dialogue," https://openai.com/chatgpt, 2023.

[3] S. Hirabayashi, R. Jain, N. Jurković, and G. Wu, "Harvard undergraduate survey on generative ai," *arXiv preprint arXiv:2406.00833*, 2024. [Online]. Available: https://arxiv.org/abs/2406.00833

[4] OpenAI, "College students and chatgpt adoption in the us," 2024. [Online]. Available: https://openai.com/global-affairs/college-students-and-chatgpt/

[5] D. Baidoo-Anu and L. O. Ansah, "Education in the era of generative artificial intelligence (ai): Understanding the potential benefits of chatgpt in promoting teaching and learning," *Journal of AI*, vol. 7, no. 1, pp. 52–62, 2023.

[6] P. Banerjee, A. K. Srivastava, D. A. Adjeroh, R. Reddy, and N. Karimian, "Understanding chatgpt: Impact analysis and path forward for teaching computer science and engineering," *IEEE Access*, 2025.

[7] M. M. Rahman and Y. Watanobe, "Chatgpt for education and research: Opportunities, threats, and strategies," *Applied sciences*, vol. 13, no. 9, p. 5783, 2023.

[8] B. Qureshi, "Exploring the use of chatgpt as a tool for learning and assessment in undergraduate computer science curriculum: Opportunities and challenges," *arXiv preprint arXiv:2304.11214*, 2023. [Online]. Available: https://arxiv.org/pdf/2304.11214

[9] Y. Ai, M. Baveja, A. Girdhar, M. O'Dell, and A. DeOrio, "A custom generative ai chatbot as a course resource," in *2024 ASEE Annual Conference & Exposition*, 2024.

[10] A. Taylor, A. Vassar, J. Renzella, and H. Pearce, "Dcc–help: Transforming the role of the compiler by generating context-aware error explanations with large language models," in *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, 2024, pp. 1314–1320. [Online]. Available: https://doi.org/10.1145/3626252.3630822

[11] 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. [Online]. Available: https://doi.org/10.1145/3626252.3630938

[12] K. E. Boyer, R. Phillips, M. D. Wallis, M. A. Vouk, and J. C. Lester, "Balancing cognitive and motivational scaffolding in tutorial dialogue," in *Proceedings of the International Conference on Intelligent Tutoring Systems (ITS)*. Montréal, Quebec: Springer, 2008, pp. 239–249.

[13] K. Reitz and contributors, "Requests: Http for humans," https://requests.readthedocs.io/en/latest/, 2025, accessed June 9, 2025.

[14] Python Software Foundation, "threading — thread-based parallelism," https://docs.python.org/3/library/threading.html, 2025, accessed June 9, 2025.

[15] ——, "heapq — heap queue algorithm," https://docs.python.org/3/library/heapq.html, 2025, accessed June 9, 2025.