Teaching CS50 with AI

Leveraging Generative Artificial Intelligence in Computer Science Education

Rongxin Liu
Harvard University
Cambridge, MA, USA
rongxinliu@cs50.harvard.edu

Carter Zenke
Harvard University
Cambridge, MA, USA
carter@cs50.harvard.edu

Charlie Liu
Yale University
New Haven, CT, USA
charlie.liu@yale.edu

Andrew Holmes
Harvard University
Cambridge, MA, USA
aholmes@college.harvard.edu

Patrick Thornton
Harvard University
Cambridge, MA, USA
patrickthornton@college.harvard.edu

David J. Malan
Harvard University
Cambridge, MA, USA
malan@harvard.edu

ABSTRACT

In Summer 2023, we developed and integrated a suite of AI-based software tools into CS50 at Harvard University. These tools were initially available to approximately 70 summer students, then to thousands of students online, and finally to several hundred on campus during Fall 2023. Per the course's own policy, we encouraged students to use these course-specific tools and limited the use of commercial AI software such as ChatGPT, GitHub Copilot, and the new Bing. Our goal was to approximate a 1:1 teacher-to-student ratio through software, thereby equipping students with a pedagogically-minded subject-matter expert by their side at all times, designed to guide students toward solutions rather than offer them outright. The tools were received positively by students, who noted that they felt like they had "a personal tutor." Our findings suggest that integrating AI thoughtfully into educational settings enhances the learning experience by providing continuous, customized support and enabling human educators to address more complex pedagogical issues. In this paper, we detail how AI tools have augmented teaching and learning in CS50, specifically in explaining code snippets, improving code style, and accurately responding to curricular and administrative queries on the course's discussion forum. Additionally, we present our methodological approach, implementation details, and guidance for those considering using these tools or AI generally in education.

CCS CONCEPTS

• **Social and professional topics** → **CS1**; • **Applied computing** → **Computer-assisted instruction**.

KEYWORDS

AI, artificial intelligence, generative AI, large language models, LLMs

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

ACM ISBN 979-8-4007-0423-9/24/03…$15.00
https://doi.org/10.1145/3626252.3630938

ACM Reference Format:
Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J. Malan. 2024. 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 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3626252.3630938

1 INTRODUCTION

Amid collective enthusiasm for generative artificial intelligence (AI) built atop large language models (LLMs), there is apprehension about AI's ability to disrupt education. Students can now complete assignments or write essays entirely with AI, undermining the fundamental goals of teaching and learning. As such, a common response to this development among educators has been to forbid the use of AI outright.

We offer an alternative approach, having chosen instead to directly incorporate generative AI into CS50, Harvard University's introductory course in computer science for majors and non-majors alike. We embraced generative AI and harnessed its capabilities within the classroom, while also implementing guardrails to uphold academic integrity and promote meaningful learning. Because CS50 boasts a large on-campus student body and a global online presence via OpenCourseWare, the course is well-suited for evaluating the effectiveness of different pedagogical approaches towards AI.

In Summer 2023 and Fall 2023, we actively tested an AI-powered chatbot, implemented as a virtual rubber duck (a la rubber-duck debugging), developed specifically for CS50. We initially deployed this CS50 Duck (aka CS50.ai) first to approximately 70 summer students, then to thousands of students online, and finally to approximately 500 on-campus students. This paper chronicles our development of CS50.ai, detailing the challenges faced, solutions proposed, and results achieved, all toward realizing a long-held aspiration of ours: a 1:1 teacher-to-student ratio.

2 MOTIVATION

Teaching CS50 with AI was an attempt to confront the apprehension of AI in education. Critics have described tools like ChatGPT as a "plague upon education" [12], given their potential to facilitate academic dishonesty by allowing students to present AI-generated work as their own. As educators concerned with issues of academic dishonesty ourselves [7], we recognized the complications that

some AI tools posed by being *too* helpful to students, particularly novices in computer science. Like other courses, we quickly implemented policies that banned tools such as ChatGPT, GitHub Copilot, and other AI-based software that suggest or complete answers to questions or lines of code. Simultaneously, we felt that it would be a missed opportunity if we did not leverage the newfound power of AI to enhance students' learning.

Prior studies have explored the adaptation of AI-based tools to better serve students. Researchers at Stanford University have demonstrated the use of AI in education through a meta-learning ProtoTransformer, which provided feedback on student code with a precision higher than that of teaching assistants [13]. In addition, Reis et al. conducted a study showing that AI-generated personalized hints significantly reduced student effort in deriving correct solutions [10]. Emerging evidence suggests a potential for AI to improve the learning feedback process, promote critical thinking, and bolster problem-solving skills [11].

Specifically for CS50 on campus, AI can help narrow the gap between the ideal 1:1 teacher-student ratio and the reality of resource constraints. Although we are fortunate to have numerous teaching fellows (TFs), the diverse needs of a 500-student class are demanding. AI is already proving essential in supporting learners of all levels [14], making it a valuable tool in present-day education.

In online courses of sufficient size, students who enroll might never talk live with an instructor. Indeed, in CS50's massive open online course (MOOC) – which has more than 5 million registrants as of writing – it is often the case that the only humans a student may turn to for help are other students. If an AI assistant could respond with more expertise than students and behave like a good tutor might, it would be a breakthrough for students who otherwise might never interact with a course instructor.

Ultimately, our goal was to approximate a 1:1 teacher-to-student ratio, providing each student with a personal subject-matter expert by using generative LLMs like OpenAI's GPT-4 [9] to emulate a good teacher: guiding students, not disclosing answers directly, and staying up-to-date with course changes.

3 SOLUTIONS

We initially pursued simple proofs of concept that evolved into more advanced projects, intending to provide a comprehensive suite of tools that could assist students with learning CS50's curriculum. These tools include: 1) "Explain Highlighted Code" for quick, understandable code explanations, 2) an enhanced version of `style50` for evaluating code style, and 3) the CS50 Duck, a chatbot for answering course-related inquiries via multiple platforms.

All these tools are powered by our unified web application, CS50.ai. In addition to delivering fast and accurate AI-generated responses, CS50.ai has built-in "pedagogical guardrails" that align with our teaching philosophy of guiding students, rather than providing direct solutions to questions.

3.1 Explaining Highlighted Code

We first created an "Explain Highlighted Code" (EHC) VS Code extension to emulate behavior by human instructors, providing students with immediate explanations in plain English for code snippets. (Students need only highlight one or more lines of code to have it explained.) This tool complements CS50's existing correctness-testing tool, `check50`, offering instant clarifications on the semantic aspects of code. Now that instant code clarification is always available to students, in-person office hours are ideally made even more productive, given students' heightened ability to deal more exclusively with higher-level design issues instead of lower-level clarification questions.

3.2 Improving Code Style

We also re-implemented `style50` – the course's command-line tool for checking the style of one's code – as a VS Code extension with a graphical user interface. This new version displays side-by-side the differences between a student's code and a better-formatted version based on CS50's style guide. The extension also provides an "Explain Changes" button that, when clicked, provides students with natural-language explanations of `style50`'s suggestions at the click of a button. These features have transformed `style50` into an interactive learning tool that provides guidance akin to that of a human instructor, enabling students to more clearly understand and apply syntactic improvements to their code.

3.3 CS50 Duck

All of our AI tools are powered by the same CS50.ai backend, ensuring consistency with our approach to using AI. To give students relatively unfettered access to the AI model, similar to a ChatGPT-like conversational format, we host a standalone website for the CS50 Duck. This allows students to interact directly with GPT-4 in a controlled manner, as illustrated in Figure 1.

Figure 1: The main page of CS50.ai, where students can chat with the CS50 Duck, an interactive "duck debugger" (ddb).

The CS50 Duck is available both via the CS50.ai website and a separate VS Code extension, per Figure 2; both forms provide an identical AI assistant to students. The CS50 Duck leverages the contextual understanding of GPT-4 to provide a truly interactive teaching and learning experience, all the while adhering to CS50-specific pedagogical guidelines.

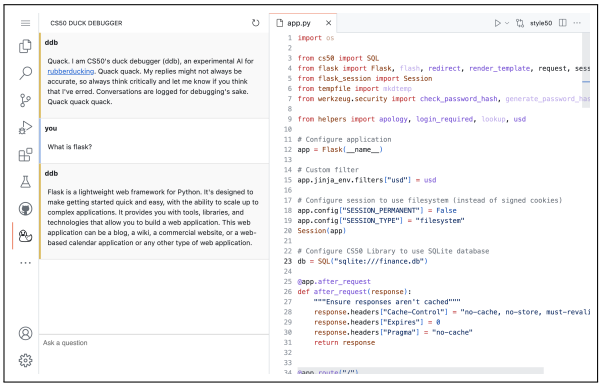

Figure 2: CS50 students can also access the CS50 Duck within VS Code to chat about CS-related topics, explain highlighted code, or suggest code style improvements.

3.4 CS50 Duck on Ed

We have long used Ed [2], a third-party platform, as the course's online discussion forum for asynchronous help. Ed allows students to ask questions and receive guidance from the course staff and fellow students, streamlining threaded discussions, minimizing question duplication, and facilitating peer collaboration.

To further improve Ed's functionality, we integrated the CS50 Duck into the platform by utilizing its HTTP request feature. As shown in Figure 3, the CS50 Duck participates in threads and answers questions as needed. By creating a dedicated API endpoint in our CS50.ai application for the CS50 Duck on Ed, we can control responses generated by GPT-4 and ensure that the CS50 Duck's behavior aligns with the course's curriculum and teaching philosophy, rather than simply giving outright answers to students' inquiries.

Figure 3: The CS50 Duck in action on Ed. A student asked a question, and the CS50 Duck replied with a succinct answer, which a human staff member endorsed as correct.

Through the CS50 Duck on Ed, we aimed to enrich the learning experience by providing students with access to immediate, carefully generated responses. By doing so, we seek to complement human instruction, not to replace it. We enforce this distinction by making all of theCS50 Duck responses subject to endorsement, amendment, or deletion by a human staff member; Ed's own user interface provides options for each.

4 IMPLEMENTATION DETAILS

As shown in Figure 4, we implemented CS50.ai as the backend for all of our AI tools to manage communications between each user and GPT-4. (Specifically, we use the version of GPT-4 that is hosted on Microsoft Azure [8].) Student queries are first relayed to CS50.ai, where any personally identifiable information (PII) is removed. Then, the queries are further processed into structured queries, known as "prompts." These prompts are constructed with course-specific rules and guidelines – in addition to the original student queries – in order to guide GPT-4 towards generating context-aware responses with high accuracy.

For student queries coming from the Ed discussion platform, CS50.ai uses a technique called "retrieval-augmented generation" (RAG) when generating responses. RAG enhances the accuracy and reliability of generative LLM models with facts fetched from external sources, reducing "hallucinations" where models make up false information. Factual information is added to our prompts to ground GPT-4 in generating responses that are (more likely) accurate and contextually relevant.

The resulting architecture is a streamlined system where CS50.ai and GPT-4 work together to quickly deliver correct and helpful answers to students.

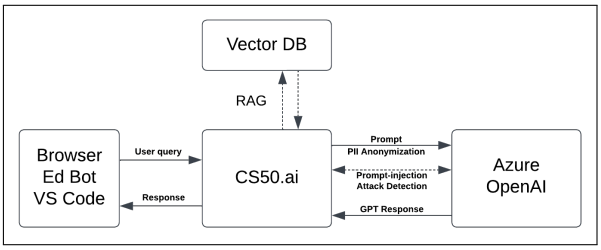

Figure 4: The system architecture of CS50.ai. GPT-4 generates responses to student queries and optionally employs a retrieval-augmented generation technique to improve response accuracy by incorporating facts from external sources.

4.1 Prompts

During interactions with LLM models like GPT-4, prompts control the conversation. Requests made from CS50.ai to GPT-4 always include a system prompt, which sets the course-specific rules and guidelines, and a user prompt, which includes the actual student query that GPT-4 answers.

One can use different prompts to create various "agents" in solving task-specific problems. For example, we start each CS50.ai interaction with a custom system prompt that tells the model to act as a teaching assistant for CS50. Then, we send user prompts that incorporate actual student queries within a set of fixed instructions. After many iterations, we have created multiple configuration files (in YAML format) with different system prompts and user prompt templates for various use cases, such as fielding CS-related questions, explaining code snippets, and offering feedback on code style.

4.2 Chat Completion

LLMs can take a series of prompts as input and return AI-generated messages as output. CS50.ai uses OpenAI's Chat Completion API, which is an endpoint that processes an array of prompts – each with a defined role ("system," "user," or "assistant") – to mimic a conversation.

A conversation starts with a system prompt that sets the desired context and behavior for the assistant to follow. Subsequent user prompts contain the actual queries or statements for the LLM to respond to.

Once we construct the system and user prompts, we send them as input to the Chat Completion API for processing. To simulate conversational flow, we append the output message to an array that also stores user queries. This entire message history is then resent to GPT-4, providing context for the next AI-generated response. The cycle continually repeats as the conversation unfolds.

4.3 Retrieval-Augmented Generation (RAG)

LLMs can sometimes "hallucinate" and generate plausible-sounding yet incorrect (or even outright nonsensical) responses [3]. This behavior occurs because AI models, which are trained on a vast amount of text data, learn to generate text fluently without necessarily valuing factual correctness. Additionally, LLMs sometimes have a knowledge cut-off, which limits their information to what they learned up until their most recent training session, without access to real-time updates or post-training events.

To mitigate this, a technique known as retrieval-augmented generation (RAG) can be used to improve the accuracy and reliability of LLMs by grounding them with facts from external sources [5]. Specifically, we utilize OpenAI's Embeddings API to create text embeddings for CS50 lecture captions, forming a ground-truth external data source. These embeddings are vector representations (i.e. numerical values) that capture semantic meaning for machine learning algorithms, allowing for more effective interpretation and utilization of data.

Our data preparation process involves segmenting English captions from the course's lectures into short, self-contained 30-second segments. We then create embeddings for these segments using OpenAI's text-embedding-ada-002 model and store the results in a ChromaDB vector database. Afterwards, we also create embeddings for each incoming student query and perform an embedding search in the vector database to retrieve lecture caption segments ranked by relevance. Finally, to produce an AI-generated response, we provide GPT-4 with the student query and the top-N most relevant lecture caption segments in plain text.

By using RAG, we can reduce the chances of "hallucination" when GPT-4 responds to lecture-specific questions, without needing to fine-tune or retrain an LLM (which is often time-consuming and computationally expensive).

4.4 Prevention of Prompt-Injection Attacks

A prompt injection attack occurs when a malicious user feeds misleading prompts into an LLM model to manipulate its behavior, such as tricking GPT-4 into providing full-blown homework solutions.

To prevent such attacks, we have implemented a "guard" feature in CS50.ai that checks every student request for atypical non-alphanumeric patterns, which could indicate a potential attack. If the guard is triggered, CS50.ai consults GPT-4 through an independent API call to determine whether the student request is regular input or a prompt injection attack. If an attack is confirmed, the system immediately aborts the current user session, protecting itself from misuse.

4.5 Ed Integration via HTTP Requests

CS50.ai is designed to proxy and augment HTTP requests, thereby allowing potential integration with third-party online platforms. For example, on Ed, interactions typically occur within threads (topics or questions posted by students) and comments (replies to these threads). The CS50 Duck on Ed is programmed to monitor these threads and comments, determining the need to respond based on predefined criteria, such as a thread's category.

When a student posts a thread, the CS50 Duck on Ed performs a series of checks to verify if the thread is a question, confirm that it wasn't posted by a staff member, and check if it falls within certain predefined categories. If these conditions are all met, the CS50 Duck on Ed sends a request to a CS50.ai API endpoint with thread-related data to generate a response.

By leveraging the flexibility of CS50.ai with processing HTTP requests, we expanded the CS50 Duck's reach and impact across different online platforms, providing students with high-quality educational support regardless of where they engage with the course.

4.6 Usage Throttling

CS50.ai implements a throttling mechanism via visually displayed hearts, where each student starts with 10 hearts and regains one heart every three minutes. Each interaction with the CS50 Duck consumes a heart, preventing spam-like behavior. This helps reduce the cost of running CS50.ai, given that we are charged for each GPT-4 request that we send.

Usage throttling also holds valuable pedagogical implications for students. First, it promotes thoughtful interaction with the CS50 Duck by encouraging students to carefully consider their questions. The underlying goal is to foster independent problem-solving skills and the ability to formulate precise questions, which is essential in learning itself. Second, usage throttling encourages reflective breaks, nudging students to step back and revisit complex problems with a refreshed and renewed perspective.

5 RESULTS

5.1 Feedback from Students

During Summer 2023, approximately 70 students enrolled in CS50 were asked to provide feedback on our AI tools via a non-anonymous survey at the end of the course, and the responses were almost entirely positive. Students praised our AI tools for their helpfulness, effectiveness, and reliability in guiding them through challenging problems:

- *"absolutely unreal. felt like having a personal tutor... i love how AI bots will answer questions without ego and without*

judgment, generally entertaining even the stupidest of questions without treating them like they're stupid. it has an, as one could expect, inhuman level of patience."

- *"I really appreciated the AI tools, especially since we are now in a time when using AI will be very common in code. So it was nice to already get acclimated to working alongside these tools instead of feeling like those tools are working against us. I also appreciated that CS50 implemented its own version of AI, because I think just directly using something like chatGPT would have definitely detracted from learning"*
- *"The AI tools were extremely helpful for me. They explained concepts to me that I did not know well, and taught me new concepts that I needed to know to implement for specific problem sets. The AI tools gave me enough hints to try on my own and also helped me decipher errors and possible errors I might encounter."*

In the subsequent Fall 2023 semester, we conducted two more non-anonymous surveys of approximately 500 on-campus students – one mid-semester and the other at the end – to gauge students' feedback and reactions toward our AI tools. Mid-semester, students reported varied but significant usage:

(1) 17% used the tools more than ten times a week, while 32% used them 5–10 times a week. Additionally, 26% used them 2–5 times weekly, and 25% used them less than twice per week.
(2) Regarding perceived helpfulness, the majority of students found the course's AI tools beneficial. Specifically, 47% reported them to be "very helpful," 26% "helpful," 21% "somewhat helpful," and 6% "not helpful."
(3) When assessing the effectiveness of these tools in enhancing learning, students' responses were also favorable. 35% reported them to be "very effective," 35% "effective," 25% "somewhat effective," and 5% "not effective."
(4) Even though students were advised to "think critically" and not assume that AI-generated messages are always correct, about 23% felt "very confident" in our AI tools' response accuracy, 46% "generally confident," 27% "somewhat confident," and 4% "not confident'.'

As for the survey at the end of the semester, 73% of students have responded as of this writing:

(1) 50% of students reported using our AI tools "frequently," while 28% indicated that they used them "constantly," suggesting a high level of reliance on these tools. Meanwhile, 19% of students used them "infrequently," and just 3% never used them.
(2) Our AI tools were well-perceived as beneficial, with 33% finding them "always helpful," 55% of students finding them "frequently helpful." However, 11% found them "infrequently helpful," and 1% found them "never helpful."
(3) Overall satisfaction with our AI tools was high: 53% of students "loved" them, and 33% "liked" them. On the other hand, 13% of students were neutral, and only 1% "disliked" them.

Anecdotally, we also found that many students would anthropomorphize the CS50 Duck, viewing it as a friendly face. We believe this effect contributed to the success of our AI tools, as students felt comfortable chatting with a lovable duck, instead of a faceless robot. As one student declared, *"Love love loved the duck. We're friends now."*

However, there have been a few instances where our AI tools misunderstood questions and provided incorrect advice. Occasional inaccuracies alone would be permissible – human teachers are surely susceptible to error themselves — but AI tends to exhibit a tone of complete and authoritative confidence even when wrong, while humans might qualify the certainty of their answers. Similar to inherent limitations with other AI chatbots, CS50.ai occasionally exhibits misguided confidence when "hallucinating" incorrect information, but recent work suggests that LLMs could soon be trained to express uncertainty when appropriate [6].

Some students also suggested reducing the usage throttling, since they preferred to chat with the CS50 Duck without any restrictions. Given that throttling is in place for both cost and pedagogical purposes, we do not plan on eliminating usage throttling in the near future, but we will continue to adjust this mechanism as costs and usage fluctuate.

5.2 Response Accuracy

To evaluate the performance of the CS50 Duck on Ed during Summer 2023, we asked a senior course staff member – who was separate from the development team – to review responses generated by CS50.ai. The CS50 Duck posted a total of 64 answers on Ed for our summer course, out of which 25 were related to curricular matters, while the remaining 39 were related to administrative matters. From this, we determined that:

- 22 out of 25 (88%) curricular answers were correct.
- 30 out of 39 (77%) administrative answers were correct.

The slight decrease in accuracy regarding administrative matters was expected, given that the underlying GPT-4 model was trained on data with a time cutoff. The syllabus for CS50 has evolved over time, causing GPT-4 to be out-of-sync with the course's latest changes. As for the 88% curricular accuracy rate, we found this to be a notable improvement over GPT's baseline performance with coding questions. Recent work indicates that ChatGPT is correct only 48% of the time when addressing software engineering topics [4].

With the Fall 2023 semester having just concluded as of this writing, we only just began to evaluate the CS50 Duck's most recent performance on Ed. Of the CS50 Duck's 180 answers in Fall 2023, only 70 were "endorsed" by human staff, which would seem to suggest an accuracy of only 39% (a notable decrease from Summer 2023). However, we suspect that this calculation understates the CS50 Duck's actual accuracy, as student usage of Ed has decreased significantly. In Fall 2022, students asked an average of 0.89 questions each via Ed; in Summer 2023, students asked an average of 1.1 questions each via Ed; yet in Fall 2023, students only asked an average of 0.28 questions each via Ed. (These numbers are averaged over on-campus and off-campus students alike.)

Since CS50.ai was fully deployed by Fall 2023, we suspect that many students transitioned to synchronous (i.e. more conversational) interactions with the CS50 Duck via VS Code and the standalone CS50.ai website, instead of Ed. We also wonder if students tended to escalate more complex questions to Ed, during situations where they already tried asking the CS50 Duck elsewhere for help but didn't receive a satisfactory response, thus needing to ask on

Ed for a human reply. Because we just began reviewing data from Fall 2023, these theories are only conjectures at this point in time.

5.3 Usage Summary

Since June 2023, our AI tools have been used extensively both on campus and globally, with more than 50,000 unique users using CS50.ai as of December 2023. We have processed more than 1.8 million queries in total.

The number of daily unique active users has significantly increased, starting at approximately 200 in June 2023, rising to 1,000 in September 2023, and reaching 1,500 by November 2023. Current daily prompt creations have followed an upward trajectory and peaked thus far at 25,000 prompts per day, as illustrated in Figure 5. Additionally, the average daily prompts created per user has increased from an initial average of 5 prompts per user per day to 15 prompts per user per day. This increase indicates a more intensive engagement with our AI tools on a per-user basis.

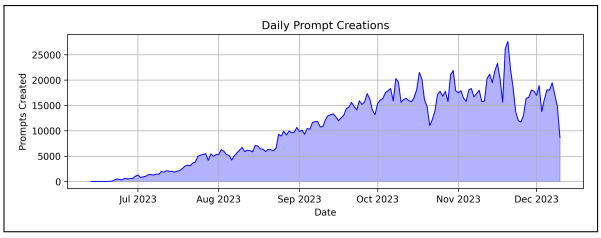

Figure 5: Graph of daily prompt creations over time. Currently, 15–20K prompts are typically created per day.

Even with extensive user activity in November, our costs remain reasonable at approximately $1.90 per student per month and $0.05 per prompt, which is a worthwhile investment given the positive student feedback and enhanced learning experience. Moreover, companies like OpenAI and Microsoft tend to make credits freely available for educational usage, helping defray costs.

That being said, we continuously monitor and refine our AI tools to maximize efficiency while minimizing costs. This approach includes improving RAG effectiveness, adjusting usage throttling, and ensuring that the tool accurately addresses students' queries.

6 FUTURE WORK

6.1 Assessing Code Design

In CS50, assignments are evaluated on correctness, style, and design. Correctness and style are automatically graded via `check50` and `style50`. `check50` verifies if the code meets the specified task requirements, while `style50` assesses the code's adherence to established style guidelines. However, a significant aspect of evaluating students' work involves assessing code design, which currently relies on manual grading by human staff. (In CS50's MOOC, design is not evaluated at all due to the online course's sheer scale.) While critical for understanding students' proficiency with organized, and efficient coding, this process poses challenges due to inherent human subjectivity and the significant time commitment required of human staff.

To streamline this process and ensure more consistency, we hope to develop `design50`, an AI tool that automates design grading. This tool would provide uniform feedback across submissions, reducing the grading workload for human staff; prior work by others has already demonstrated this [1]. The idea is to train `design50` on assignments previously graded by humans, enabling it to automatically learn and apply those same standards. Human staff would then only need to review and confirm the tool's assessments, making the grading process faster and more standardized.

6.2 Extending to Other Courses

Following the successful integration of CS50.ai into our summer course, we expanded its deployment to 10 other CS50-related MOOCs, which cover a wide range of topics: game design, SQL, and cybersecurity are just a few examples. Even though these online courses are available year-round to tens of thousands of students worldwide, CS50.ai has enabled us to provide 24/7 support.

To accomplish this, we built a modular configuration system that automatically updates CS50.ai's RAG knowledge base and prompt library, allowing us to tailor our AI tools to a new curriculum and pedagogical approach specific to each course. In the future, we hope to leverage this system to further expand CS50.ai's use in other courses, within both STEM and the humanities.

7 CONCLUSION

After a summer's worth of practical application and a full fall semester of intensive stress testing, we have found our suite of AI tools to be a success for CS50. At the very least, we believe that our "guardrailed" approach to allowing the use of AI provides a definite advantage over the alternative of forbidding it altogether. We hope that providing students with our own AI tools will reduce the academically dishonest use of other AI tools that are available online.

Each one of our AI tools is not merely a proof of concept. With extensive usage statistics and positive student feedback, CS50.ai has already been deployed to thousands of students around the world, demonstrating its utility in real-world scenarios. Notwithstanding improvements that we have yet to make, the integration of AI into an educational context has shown great promise, particularly in elevating the accessibility of personalized teaching assistance while freeing up human staff to handle higher-level pedagogical concerns.

Our long-term vision is to broaden the scope of our AI tools to other disciplines, allowing the CS50 Duck to interface with pedagogy beyond the field of computer science. We hope that CS50's use of AI serves as a blueprint for other institutions and courses considering the potential for generative AI to bolster student learning, not merely disrupt it. When equipped with the correct guardrails, AI can revolutionize education for the better.

ACKNOWLEDGEMENTS

Many thanks to Ed, GitHub, Microsoft, and OpenAI for their support of this work. And many thanks as well to Brenda Anderson, Sophie Anderson, and Doug Lloyd for their assistance with this work.

REFERENCES

[1] Wei Dai, Jionghao Lin, Flora Jin, Tongguang Li, Yi-Shan Tsai, Dragan Gasevic, and Guanliang Chen. 2023. Can Large Language Models Provide Feedback to Students? A Case Study on ChatGPT. https://doi.org/10.35542/osf.io/hcgzj

[2] Ed. 2023. https://edstem.org/

[3] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *Comput. Surveys* 55, 12 (March 2023), 1–38. https://doi.org/10.1145/3571730

[4] Samia Kabir, David N. Udo-Imeh, Bonan Kou, and Tianyi Zhang. 2023. Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions. arXiv:2308.02312 [cs.SE]

[5] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401 [cs.CL]

[6] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching Models to Express Their Uncertainty in Words. arXiv:2205.14334 [cs.CL]

[7] David J. Malan, Brian Yu, and Doug Lloyd. 2020. Teaching Academic Honesty in CS50. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, Oregon, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 282–288. https://doi.org/10.1145/3328778.3366940

[8] Microsoft. 2023. Azure OpenAI. https://azure.microsoft.com/en-us/products/ai-services/openai-service/ Accessed: 2023-12-12.

[9] OpenAI. 2023. GPT-4. https://openai.com/research/gpt-4 Accessed: 2023-10-09.

[10] Ruan Reis, Gustavo Soares, Melina Mongiovi, and Wilkerson L. Andrade. 2019. Evaluating Feedback Tools in Introductory Programming Classes. , 7 pages. https://doi.org/10.1109/FIE43999.2019.9028418

[11] Meng-Lin Tsai, Chong Wei Ong, and Cheng-Liang Chen. 2023. Exploring the use of large language models (LLMs) in chemical engineering education: Building core course problem models with Chat-GPT. *Education for Chemical Engineers* 44 (2023), 71–95. https://doi.org/10.1016/j.ece.2023.05.001

[12] Jeremy Weissman. 2023. ChatGPT is a Plague Upon Education.

[13] Mike Wu, Noah Goodman, Chris Piech, and Chelsea Finn. 2021. ProtoTransformer: A Meta-Learning Approach to Providing Student Feedback.

[14] Carter Zenke and David J. Malan. 2023. Differentiating for Comfort with Computer Science. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education* (Toronto, Canada) *(SIGCSE '23)*. Association for Computing Machinery, New York, NY, USA, 1269. https://doi.org/10.1145/3545947.3573249