AI-Generated Slides: Are They Good? Can Students Tell?

Juho Leinonen*
Aalto University
Espoo, Finland
juho.2.leinonen@aalto.fi

Lisa Zhang*
University of Toronto Mississauga
Mississauga, Ontario, Canada
lczhang@cs.toronto.edu

Arto Hellas
Aalto University
Espoo, Finland
arto.hellas@aalto.fi

Abstract

As generative AI (GenAI) tools become easily accessible, there is promise in using such tools to support instructors. To that end, this paper examines using GenAI to help generate slides from instructor authored course notes, emphasizing instructor and student perceptions. We examine an end-to-end education tool (NotebookLM), two general-purpose LLMs (Claude, M365 Copilot), and two coding assistants (Cursor, Claude Code). We first analyze whether GenAI generated slides are "good" via narrative assessment by educators. We choose the best slides to use (with some modification) in a real course setting, and compare the student perception of human vs. AI generated slides. We find that coding assistant tools produce slides that were most accurate, complete, and pedagogically sound. Additionally, students rate GenAI slides to be of similar quality as instructor-created slides, and cannot reliably identify which slides are AI-generated. Additionally, we find a negative correlation between a high quality rating and a high "AI-generated" rating, suggesting students associate poor quality with the source of the slides being AI. These findings highlight promising opportunities for integrating GenAI into instructional design workflows and call for further research on how educators can best harness such tools responsibly and effectively.

CCS Concepts

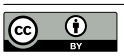
• **Computing methodologies** → **Artificial intelligence**; • **Social and professional topics** → **Computing education**.

Keywords

Computer Science Education; Computing Education; Generative AI; GenAI; LLMs; Large Language Models; Educational Materials; Automatic Generation of Educational Materials; Web Software Development

ACM Reference Format:
Juho Leinonen, Lisa Zhang, and Arto Hellas. 2026. AI-Generated Slides: Are They Good? Can Students Tell?. In *Proceedings of Western Canada Conference on Computing Education 2026 (WCCCE 2026).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

*Equal contribution. Author order is alphabetical, which coincidentally favours the first author.



WCCCE 2026, Vancouver, Canada

ACM ISBN 978-1-4503-XXXX-X/2026/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

As large language models (LLMs) and AI-powered educational tools rapidly advance, LLM and Generative Artificial Intelligence (GenAI) tools have been shown to create educational content [31]. They are perceived to increase productivity on both programming tasks [20] and writing tasks [17]. However, even within software engineering, there is variation in users' perception [3], with GenAI tools sometimes generating incorrect or hallucinated outputs. Computing instructors thus face both opportunities and uncertainties in integrating GenAI into their pedagogical practice. Student perceptions of AI-generated materials introduce further complications [13].

This paper presents a case study using GenAI tools in assisting instructors in generating slides from instructor-authored online textbook chapters, in the context of an upper-year undergraduate web programming course. We are interested in two questions:

(1) Are they good? Here, "they" refers holistically to the GenAI tools themselves, the process of using them, and the generated slide outputs. We explore and compare various types of GenAI tools: end-to-end education tools that students may already use (e.g., NotebookLM), general purpose LLMs that are accessible (e.g., Claude, M365 Copilot), and coding assistants that have been shown to be helpful for programming tasks (e.g., Cursor, Claude Code). We report on the instructor experience and their narrative assessment of factual accuracy, completeness, and pedagogical soundness. These factors are similar to work conducted at a similar time frame [33].

(2) Can students tell? We select the best slides generated across tools and, with some modifications, evaluate their quality in a real-world classroom; we then report student perception of quality and their ability to identify AI-generated content. Specifically, after matching the visual styling of GenAI-generated slides to the instructor's preferred format and selectively deleting content, we use these slides in actual course lectures alongside instructor-created slides. Students are informed that lectures may contain GenAI-generated content but are not told which segments or how many. After each lecture segment, students rate the slide quality and guess whether the slides were AI or human generated.

We choose tools that are accessible (free, <USD$30 per month, or available via institutional licenses). We prioritize tools that have not been widely discussed in the computing education literature [21, 22], to introduce new tools to the community. Additionally, slides are a common form of content used to support lectures that is time-consuming to create. Slides are also challenging for GenAI tools as its creation involves coordination across text, code, and diagrams.

As the authors are computing educators and researchers with minimal (<2 months) prior experience using these specific GenAI tools to build educational resources, we offer an authentic perspective on the instructor on-boarding experience. Additionally, one researcher serves as the instructor for the course examined,

providing domain expertise for evaluating content quality and pedagogical appropriateness. The student evaluation is conducted with upper-year undergraduate students enrolled in the actual course, providing authentic feedback from learners.

2 Related Work

Recent surveys have shown that LLMs are used to assist students, teachers, and for adaptive learning [31]. The potentials of multimodal models—models that combine text, image, and audio—are also being explored for STEM domains (e.g., in Xing et al. [34]).

In computing education, early work showed that code-aware models like Codex could solve most introductory programming problems [5], prompting questions about assessment validity and curricular goals. More recently, Denny et al. [4] explored using natural language prompts to generate CS1 solutions, while Savelka et al. [25] documented GPT-4's improved performance on higher education programming assessments, underscoring rapid capability growth. Recent work found that instructors are adapting their teaching based on the widespread availability of generative AI tools, such as changing assessment practices and learning objectives [22].

Beyond student-facing applications, emerging work explores instructor support for content creation. Sarsa et al. [24] demonstrated that LLMs could generate programming exercises and explanations, though with limitations in diversity and pedagogical appropriateness. More recently, Logacheva et al. [14] found that newer LLMs perform better at generating thematically contextualized exercises, even though the contextualization was often shallow. Jordan et al. [10] found that LLMs can create suitable exercises in multiple languages, although exercises generated in low-resource languages such as Tamil had lower quality. Villegas Molina et al. [29] found that LLMs can be used to create culturally relevant textbooks for CS1. Taken together, these works show the potential and interest in leveraging LLMs in computing education.

Slide generation, in particular, is under-explored. Early work on automatic slide generation focused on extracting content from scientific papers [26], while more recent approaches leveraged LLMs. Xie et al. [33] present SlideBot, a multi-agent framework that generates educational slides using specialized agents for retrieval, summarization, figure generation, and LaTeX formatting. Other recent systems include PASS [1] for automated slide and speech generation, AutoPresent [6] for designing structured visuals from scratch, and SlideCraft [23] as a context-aware slide generation agent. These works primarily focused on system development and technical evaluation, rather than instructor and student experience.

Research on student perceptions of LLM-generated educational resources is limited. Prior work suggests that disclosure of AI assistance affects perception, e.g. in writing [13]. However, Leinonen et al. [12] found LLM-generated code explanations rated comparably to student-written ones. Similarly, Henderson et al. [9] compared student perceptions of generative AI versus teacher feedback, finding differences in perceived usefulness and trustworthiness. For slides specifically, Georgiev and Tinsley [7] examined student perceptions of instructor-created AI-assisted PowerPoint slides, finding that while students had favorable opinions of instructor AI usage, they expressed substantial concerns about slide design including structure, consistency between text and images, and typography.

3 Textbook to Slides in a Web Software Course: Are They (GenAI Tools) Good?

This section explores whether and how GenAI tools could help transform chapters from an online web programming textbook into lecture slides. This task represents a common need: instructors often maintain reading materials and generate corresponding slides.

We used course materials from a Web Software Development (WSD) course at Aalto University in Finland: a 5 ECTS[1] course providing an introduction to contemporary web software development. The course provides a "Full-Stack" view, teaching students to develop client-side and server-side applications backed by a database. Early parts of the course focus on client-side development with Svelte and SvelteKit, followed by designing server-side APIs that interact with a database, which are then later integrated to the client-side functionality. The course instructor maintains an online interactive textbook with intertwined examples and exercises.

We assessed various GenAI tools by producing slides for four topics covered in the first two weeks: basic HTML and Svelte, sharing states, CRUD (Create, Read, Update, and Delete), and the Fetch API. The instructor for the course chose these topics to represent a variety of content types—some heavy on code examples, others on diagrams or conceptual explanations. For each topic, our goal was to produce slides for a 10-15 minute lecture segment.

We explored five AI tools representing different approaches to content generation: NotebookLM, M365 Copilot, Claude (Sonnet 4.5), Cursor, and Claude Code, These tools were chosen for their accessibility to computing educators and represent different points on the spectrum from automated to instructor-in-the-loop workflows.

3.1 Methods

Two researchers, both computing educators, collaboratively explored each tool in pair-coding sessions, iteratively developing and refining prompts. To mimic the style of existing lecture materials, we collected instructor-generated lecture slides from earlier course iterations. We aimed to follow realistic instructor workflows: crafting straightforward prompts without requiring prompt engineering expertise, then refining based on what the tools produced. After initial collaborative exploration, one researcher continued building the remaining materials for all lecture topics.

For most tools, we used the following prompt, developed iteratively through initial trials:

> "Please generate a set of slides from <file>. The slides are intended to be used in a 15 minute lecture segment in a 2-hour web programming lecture in a university in a web-software development course. Please include some peer-instruction exercises at the conclusion of the segment."

We had initially included styling rules, but found that this degraded content quality, so we separated styling as a post-processing step for Section 4. We used default settings for all tools. For all tools, we used a separate session for each of the four chapters tested.

We provide narrative assessment of the quality of the generated materials by both the instructor and two non-instructor researchers. We consider three factors, similar to Xie et al. [33]. **Factual accuracy** assessed whether the content was correct by referencing

[1] European Credit Transfer System, 1 ECTS corresponds roughly to 27 hours of work.

GenAI Tool	Accuracy	Completeness	Pedagogical Soundness
NotebookLM	Generally Accurate	Very Incomplete	Very Problematic
M365 Copilot	Generally Accurate	Incomplete	Problematic
Claude	Mostly Accurate	Somewhat incomplete	Mostly acceptable
Cursor	Accurate	Complete	Strong
Claude Code	Accurate	Complete	Strong

Table 1: Evaluation of GenAI tools for generating slides from textbook materials in a Web Programming course.

source materials, fact-checking examples, verifying conceptual explanations, and identifying hallucinations or misleading information. **Completeness** assessed whether important concepts were omitted, whether examples were sufficient, and whether materials could stand alone or required substantial supplementation. **Pedagogical soundness** assessed whether generated materials aligned with established principles of computing education, including cognitive load management [15, 28] (presenting information in digestible units, avoiding overwhelming detail), dual coding and visual support [2, 15] (using diagrams and illustrations to reinforce rather than duplicate text), and scaffolding and conceptual progression [27, 32] (building from concrete examples toward abstraction without abrupt jumps).

The evaluation was conducted qualitatively and collaboratively, prioritizing narrative insights for practitioners over inter-rater reliability. Initially, two researchers collaboratively evaluated several generated resources, including at least two sets of slides per GenAI tool. The researchers identified specific issues related to accuracy, completeness, and pedagogical soundness. Through this process, we found that clarity, emphasis, and use of examples were important elements of pedagogical soundness. Then, another researcher, who is the instructor of the web programming course, reviewed the notes, evaluated the remaining slides, and summarized the findings.

3.2 Results

Table 1 summarizes our assessment of the generated resources. The remaining subsections provide narrative details. Overall, the programming-based tools, Cursor and Claude Code, produced the most compelling slides and are used in Section 4.

3.2.1 NotebookLM. NotebookLM is a free education-focused tool that can create multimodal learning resources [30], particularly podcasts [19]. At the time that this research was conducted, NotebookLM provided functionalities to generate "Audio Overview", "Video Overview", quizzes, mind maps, and custom reports based on uploaded source materials; it did not have an out-of-the-box method for generating slides. However, as NotebookLM is advertised as a general-purpose educational tool, our students likely encounter it, so we were compelled to evaluate it. We evaluated the slides and visual materials used in the "Video Overview" using the "deep dive" setting, generally taking ~20 minutes to produce.

The visuals were **generally accurate**, though still with occasional errors like considering hyperlink tags like <a> to be "interactive". Content was **very incomplete** and superficial, even though it was visually appealing. For example, the heading tag <h1> is discussed, but other heading levels were not. The slides were also **very problematic** pedagogically: the most obvious feature of the "Video Overview" itself was the long-winded introduction and motivation for the topic, also described elsewhere [8]. The language was often unclear and non-specific. While there were many analogies, not all of them were of high quality (e.g., of HTML as a blueprint, using a house blueprint in the title slides). The visuals did not contain enough code examples, and when examples were present, they generally did not support comprehension. Figures were sometimes present, but generally showed high-level processes using non-technical language.

We experimented with generating textual descriptions of the slides via the chat interface and with the custom report generation tool. The descriptions were comparable to other tools; however this option does not save much time for instructors.

3.2.2 M365 Copilot. M365 Copilot is a general-purpose GenAI assistant integrated into Microsoft 365 applications, and is available through an institutional license at one of the author's institutions. For M365 Copilot, we began with the prompt above, attaching a file containing the textbook markdown source, and appending to the prompt text "Please generate either a PDF file or a PowerPoint file". The tool was able to produce a M365 PowerPoint presentation in SharePoint directly, which we then evaluated.

The generated content was **generally accurate**. However, some content was somewhat confusing as it missed key information: for example referencing textbook content (e.g., specific code example) that was not included. Thus, the slides were **incomplete**: there were relatively few concrete examples, despite the textbook following a pattern of first introducing a concept, then code examples, and then exercises. Much of the content seemed very surface level. The slide design exhibited **problematic pedagogical soundness** issues: many slides had irrelevant large images that were not related to the content or helpful for understanding the concepts presented (e.g., several large images of birds). Slides also included too much text. In addition to the poor balance between image size and text, there were some minor rendering issues (e.g., related to HTML tags, and inconsistent font size).

3.2.3 Claude. Claude is a general-purpose LLM created by Anthropic. For this paper, we used the Pro version of Claude at a subscription rate of USD$24 per month. We used the same prompt above, attaching a file containing the textbook markdown source, and appending to the prompt text "Please generate either a PDF file or a PowerPoint file". The Chain-of-Thought output showed that internally, Claude built the slides using html2pptx. Claude iterated to ensure that the produced output fit inside the screen, and therefore took a significant amount of time (slower than Copilot). The tool produced a downloadable PowerPoint file.

Claude's output was **mostly accurate** overall, although there were some issues caused by HTML rendering. For example, a slide on key HTML elements had bulleted text explanations where each HTML tag was missing (e.g. " – Declares document type"), likely due to issues rendering the characters < and >. Similarly, another

slide showed that both unordered and ordered lists would result in bulleted item lists, which is incorrect. In terms of **completeness**, Claude also did not provide as many examples (e.g., of the DOM's tree-like data structure). The instructor would need to fill in considerable content to make the slides classroom ready. The **pedagogical soundness** was somewhat mixed but mostly acceptable; although the slides followed the materials in general, some slides jumped directly from high-level descriptions to benefits, missing concrete examples that one would typically show in a classroom. On the positive side, Claude also identified content to highlight in slides, and the textual content was mostly well balanced.

3.2.4 Cursor. Cursor is a coding assistant and AI-powered code editor built on VS Code. For this paper, we used the Pro version of Cursor (1.7) at a subscription rate of USD$20/month. Since Cursor is primarily an integrated development environment (IDE) intended to produce code, we asked Cursor to produce source materials that could be built into a set of slides. We appended to the prompt in Section 3 the following instruction: "Use Quarto to generate the slides. Include all makefiles, etc., so that the slides can be built. Please put the results in the <folder>". Quarto was chosen since it is frequently used by computing educators, because one of the authors was familiar with it, and because another author's name rhymed with it. In addition to the slides, Cursor generated presenter notes for instructors without prompting.

These slides were generally **accurate** and **complete**. During the evaluation, one of the questions that the evaluators raised was whether the slides were *too* complete; however, instructors could easily discard content. There were some minor inaccuracies that could prompt misconceptions, e.g., where the JavaScript and Svelte code was shown side-by-side but were not equivalent. The content was, **pedagogically, fairly strong**. Many examples were used, e.g., providing multiple examples for CRUD. The slides leveraged parallelism, e.g., ordered vs unordered lists were rendered side by side. Diagrams were used, most of which were from the source materials. There were some terms (like "RESTful API") that were used before they were introduced, though generally new terms were bolded and emphasized. The slides also had good progression and the content was mostly presented in a step-by-step fashion, where items would appear on slides one by one, which was the style that the instructor preferred and used. At the same time, some extra styling would be required: some slides did not fit inside the frame, and the font size in code examples was often too small.

3.2.5 Claude Code. Claude Code is a command line tool for agentic coding, available with the Claude Pro subscription. The interaction affordances were similar to Cursor, and Claude Code (version 2.0.29) also decomposed the prompt into multiple sub-tasks. The same prompts and methods were used as Cursor.

Claude Code's output was mostly **accurate**, with only minor issues such as a slide captioned "text paragraphs" also featuring headings without extra information that would guide the reader. In terms of completeness, Claude Code's slides were **mostly complete**, with some examples of missing relevant information. In some instances, similar to Cursor, the output was "too complete", meaning that the instructor would need to decide what content to keep. Regarding **pedagogical soundness**, similar to Cursor, Claude Code's slides had good logical progression, with theory intertwined with code examples. However, some slides contained content in 'iframe'-elements that required scrolling, and code examples were missing syntax highlighting.

4 Student Perception of GenAI Slides: Can Students Tell?

Based on the results in Section 3, we were confident that with minor modifications, the materials generated by Cursor and Claude Code can be used without negatively impacting student experience. All researchers agreed that Cursor and Claude Code generated slides were superior to the remaining tools, but disagreed on which was best; the web development course instructor preferred Cursor.

We thus evaluated the student perceptions of these materials in an instance of the Web Software Development course. This course included five 2-hour lectures. The lectures were organized using a peer instruction style; they were divided into segments with slides that focus on specific core areas. During each segment, the instructor complemented the slides with live-coding, and provided additional insights and examples. The instructor has taught web software development-related courses for over a decade, and has prior web development experience from the software industry. At the end of each segment, there were multiple-choice questions and possibly class discussions, depending on student performance in the multiple-choice questions. Active attendance (e.g., actively responding to the multiple-choice questions) in lectures provided a small amount of extra points (up to 5% of overall grade). A total of 88 students enrolled in the course in the fall of 2025, and 51 attended the first lecture.

4.1 Methods

We evaluated a combination of GenAI and human-generated slides. For weeks 1 and 2, we used the 4 slide segments generated by Cursor from Section 3. For weeks 3-5, we generated 6 additional slide segments using Claude Code. Slides for the remaining segments were created by the instructor without GenAI.

To ensure blinding, we matched the GenAI slides to the instructor's visual style, fixing formatting issues through iteration. Some of this work was done directly in Cursor, using the following prompt:

> "I would like to update <path_to_slides> to 'look and feel' like the instructor's slides. Here are some example images showing the rendered example slides. The instructor likes to use Arial font for the main text. The instructor also likes to use red text with 'Coming Soon' font, slightly slanted, to explain things in the code or bullets. There are transitions associated with these red text segments (they do not appear all at once). Any code fragments uses Consolas font. The styling used is consistent with those used in VSCode by default, but with the white background. I would like you to do this update by creating a new file called <new_path>. Please keep the content as similar as possible to the current version, though I understand that some content will need to change for the 'red text' to make sense."

We then followed up with "Could you render slides_styled into a PPTX file?" Styling was still challenging for the tool. We note that the GenAI styled presentations were inconsistent and contained issues. For example, the instructor used red overlay text

Table 2: Overall quality and human vs. AI guesses by segment on a 7-point Likert scale. For quality, 1 = very bad, 7 = excellent, for human vs. AI, 1 = definitely human, 7 = definitely AI, and for source AI-C = Cursor, AI-CC = Claude Code. Only students who reported attending the lecture are included. Also Spearman correlation between overall quality and human vs. AI guess.

Lecture Segment				Overall quality				Human vs. AI guesses				Correlation	
	Topic	Source	n	Mean	Median	SD	Range	Mean	Median	SD	Range	ρ	p
1-1	Practicalities	Human	22	5.5	6.0	1.2	[3, 7]	3.5	3.0	1.7	[1, 7]	0.42	0.05
1-2	HTML	AI-C	18	5.5	5.5	1.1	[3, 7]	3.6	3.5	1.5	[1, 7]	-0.48	0.03
1-3	JS and Components	Human	16	5.1	5.0	0.8	[4, 7]	3.5	4.0	1.6	[1, 6]	-0.39	0.12
1-4	Pages and Components	Human	16	5.0	5.0	1.3	[2, 7]	3.8	4.0	1.3	[1, 6]	-0.46	0.06
1-5	Reactivity	Human	20	4.4	4.0	1.5	[2, 7]	4.0	4.0	2.0	[1, 7]	-0.41	0.07
1-6	Shared State	AI-C	12	4.7	5.0	0.8	[4, 6]	4.7	4.0	1.5	[2, 7]	0.49	0.09
2-1	HTTP	Human	28	5.3	5.5	0.9	[3, 7]	3.0	3.0	1.5	[1, 6]	-0.39	0.04
2-2	Hono	Human	28	5.3	5.5	1.3	[3, 7]	3.6	4.0	1.4	[1, 6]	-0.10	0.62
2-3	Building an API	Human	28	5.1	5.0	0.9	[3, 7]	3.7	4.0	1.2	[2, 6]	-0.07	0.72
2-4	CRUD	AI-C	27	4.8	5.0	1.0	[2, 7]	4.4	4.0	1.6	[2, 7]	-0.50	< 0.01
2-5	Fetch API	AI-C	26	4.8	5.0	0.9	[3, 7]	4.0	4.0	1.5	[1, 7]	-0.31	0.12
3-1	Auth x2, Storing Passwords	Human	21	5.0	5.0	0.9	[4, 7]	3.3	3.0	1.7	[1, 7]	-0.35	0.12
3-2	Tracking Users	Human	22	5.0	5.0	1.1	[3, 7]	3.4	4.0	1.3	[1, 6]	-0.51	0.02
3-3	Client-side User Management	Human	22	4.9	5.0	1.2	[3, 7]	3.4	4.0	1.4	[1, 5]	-0.40	0.07
3-5	Web Security Essentials	AI-CC	22	5.3	5.0	0.9	[4, 7]	3.1	3.0	1.4	[1, 6]	-0.63	< 0.01
4-1	Cascading Style Sheets	AI-CC	23	5.0	5.0	0.9	[3, 7]	3.5	4.0	1.3	[1, 6]	-0.13	0.57
4-2	Styling with Tailwind CSS and Skeleton	Human	21	4.9	5.0	1.0	[4, 7]	3.8	4.0	1.4	[1, 7]	-0.37	0.10
4-3	Responsive Web Design	AI-CC	22	5.0	5.0	1.0	[4, 7]	3.8	4.0	1.4	[1, 7]	-0.37	0.10
4-4	Web Accessibility and WCAG	AI-CC	22	5.0	5.0	0.9	[3, 7]	3.6	4.0	1.6	[1, 6]	-0.43	0.06
5-1	Unit Testing etc	AI-CC	21	5.0	5.0	1.0	[3, 7]	3.5	3.5	1.5	[1, 6]	-0.24	0.30
5-2	End-to-End Testing	AI-CC	21	6.0	6.0	1.1	[4, 7]	3.5	3.0	2.2	[1, 7]	-0.26	0.32
5-3	Evolution of Web Development	Human	16	5.2	6.0	1.1	[4, 7]	3.5	3.0	2.2	[1, 7]	-0.26	0.32
Overall (AI-C / Cursor)		AI-C	83	5.0	5.0	1.0	[2, 7]	4.1	4.0	1.6	[1, 7]	-0.32	< 0.01
Overall (AI-CC / Claude Code)		AI-CC	116	5.1	5.0	0.9	[3, 7]	3.5	4.0	1.5	[1, 7]	-0.32	< 0.01
Overall (AI)		AI	199	5.1	5.0	0.9	[2, 7]	3.8	4.0	1.5	[1, 7]	-0.33	< 0.01
Overall (human)		Human	260	5.1	5.0	1.1	[2, 7]	3.5	4.0	1.5	[1, 7]	-0.24	< 0.01
Overall (all)		-	459	5.1	5.0	1.0	[2, 7]	3.6	4.0	1.5	[1, 7]	-0.28	< 0.01

in a font resembling handwriting to annotate key ideas and provide commentary, essentially drawing on the signaling principle to help learners understand the content [16]. The signaling, including styling, font, and transitions, were difficult for GenAI tools to emulate. The instructor fixed the style before using the materials.

Additionally, when adding the AI-generated slides to the slide deck of the lectures, the instructor was allowed to delete content, e.g., entire slides, examples, or other content (e.g., the "learning objectives" slides). We believe deletion is a reasonable, efficient, and realistic change that instructors would make when using GenAI tools. These adjustments required little effort, and making them helped the instructor familiarize themselves with the slides.

Students were informed that lectures might contain GenAI-generated content but not told which segments or how many. After each segment in the lectures, students completed a brief survey rating the segment quality on a 7-point Likert scale[2] and guessing whether it was GenAI-generated[3]. Students were instructed to focus their evaluation only on the slides, not other content of the segment. Participating in the survey was voluntary and did not contribute towards the points that students received from being active in the lectures[4]. The study had ethics approval from the university's ethics committee.

We note two limitations to this approach: First, individual students may have contributed different numbers of ratings[5]. Second, despite students being told to focus on the slides in their evaluation, the peer instruction ratings might reflect not only the quality of the slides but also other aspects of the instructional segment, such as live coding or verbal explanations.

We calculated the mean, median, standard deviation and range of students' responses to the two Likert-scale questions per segment and overall. We examined if there were any differences between AI- and human-made slides using a Mann-Whitney U test (since the data was ordinal). In addition, we analyzed the correlation between the

[2] *Overall, on a scale from 1 = Very bad to 7 = Excellent, how would you rate the slides in the previous segment?*

[3] *On a scale from 1 = Definitely Human to 7 = Definitely AI, what's your best guess: were the slides in the previous segment AI-made or human-made?*

[4] In the first lecture, the software used for collecting the ratings failed to store a part of the student responses, while all student responses from the second lecture onwards were stored.

[5] This violates the independence assumption of the Mann-Whitney U test, and some students could influence results more than others.

two questions to see if students' perceptions of the slides correlates with their guess of whether the slides were AI-generated.

4.2 Results

The means, medians, ranges, and correlations of the analysis can be seen in Table 2. We found that the quality of the AI- and human-made slides was similar: the median for both was 5.0. This is supported by the Mann-Whitney U test results ($U = 25600$, $p = 0.84$), which indicate we failed to find evidence that the slides were of different quality. While the median for the guesses for both AI and human slides was 4.0, students seemed not to be able to guess the source (AI vs. human) of the slides based on the Mann-Whitney U test results ($U = 28171.5$, $p = 0.1$). Essentially, this suggests that **the quality of the AI slides is similar to that of human slides, and students cannot reliably distinguish the source of the slides.**

We also see a weak, negative correlation between perceived quality and guess on whether the slides were produced by AI or a human (-0.28, $p < 0.01$). This suggests that, even though the actual quality of the slides between the AI and human was similar, **students associate poor quality with AI as the source**. This mirrors findings that humans rate writing quality as being lower when its source is revealed to be AI [13].

Looking at the biggest mismatches in guesses (i.e., human slides most rated as AI and vice versa), we found some potential explanations for students' ratings. Slide set 1-5, which was human-created, had the highest average score overall for AI being the source among the human-created slides. This was likely due to a rendering issue where the code on the slide did not display correctly. Similarly, slide set 3-5 had the lowest average score for being AI among AI slides, and second lowest overall even though it was created by Claude Code. A likely explanation is that the slide set included the XKCD comic "Exploits of a Mom"[6] about "little Bobby Tables", which probably caused students to think the slides were by a human.

5 Discussion and Recommendations

We were positively surprised by the quality of the materials produced by Cursor and Claude Code. While edits were required, the instructor of the web programming course found lecturing with the slides similar to using one's own slides or a colleague's slides.

We purposefully included different types of GenAI tools but found that coding assistants worked best, even for tasks that are not traditional "programming" tasks. We suggest that **computing instructors think of themselves as programmers first**. Similar to programmers working in the industry, instructors can work iteratively with tools, prompting and re-prompting, highlighting issues, etc. [11]. This includes building slides in a tool like Quarto to generate HTML, rather than PowerPoint, so that coding assistants can more easily work with textual representations of the materials. Thinking of oneself as a programmer also means leveraging existing documentation and tooling meant for software engineers: for example, we did not write an `AGENTS.md`, provide custom configuration to the coding assistants, or use parallel agents. We suspect that more advanced use could provide additional time-saving. Coding assistants require programming knowledge, and are more accessible to computing instructors than those in other fields.

[6] https://xkcd.com/327/

Rather than expecting a perfect first-draft, **willingness to iterate was key in using GenAI tools.** The drafts were imperfect, and the GenAI tools were particularly poor at styling the content appropriately. The drafts also still contained inaccuracies despite recent improvements in the underlying models [18]. Moreover, it is possible for the generated material to duplicate copyrighted material. However, in our spot checking, we did not find content that was verbatim copied from another source. All the content we explicitly provided to the models was originally created by the instructor who participated in this study. We encourage instructors to check their institution's guidelines on GenAI use.

We also believe that **GenAI can be effectively leveraged without hurting student experience when the instructor remains in the loop**. In Section 4, we found no statistically significant difference between students' rating of GenAI vs. instructor slides, and their ratings of the qualities were the same (mean rating of 5.1). The statistically significant negative correlation between quality and guess suggests that the AI would have been rated even higher in quality if it was more similar to instructor slides in style. This, again, emphasizes iteration: with more editing of just the style (which we believe is not time-consuming), perceived quality might be higher.

We emphasize that our results are a snapshot in time in late 2025, and models and tools improve constantly. For example, during the writing of the paper, Cursor 2.0 was released. Shortly after, NotebookLM added support for slide deck generation. Our experience is still useful to share, to document GenAI capabilities at a snapshot in time, while presenting timely insights for practitioners.

We also emphasize that the study was run at a single institution and course, so the results might not generalize beyond this context. For example, it could be possible that coding tools worked better because the context was programming related. We also used Claude Code and Cursor on different weeks of the course, which could have affected the results, and we did not measure effects on learning.

6 Conclusion

We described a case study of using generative AI tools to support the creation of lecture slides based on existing course e-book materials. Our goal was to report our holistic initial experiences, and not to conduct a research study. We envision future work that evaluates these ideas more rigorously. We found that using generative AI tools could save significant instructor time, with coding assistant tools being particularly powerful. We also found no statistically significant differences in student ratings of the quality between AI- and human-generated slides, and students seemed not to be able to guess which slides were created by AI. These findings highlight promising opportunities for integrating generative AI into instructional design workflows and call for further research on how educators can best harness such tools responsibly and effectively.

Acknowledgments

This work was supported by Research Council of Finland grants #356114 and #367787. Generative AI was used to assist with the writing and polishing of the text in this paper. All GenAI-generated or polished text was reviewed by the (human) authors and the human authors take full responsibility for the text.

References

[1] Tushar Aggarwal and Aarohi Bhand. 2025. PASS: Presentation Automation for Slide Generation and Speech. *arXiv preprint arXiv:2501.06497* (2025).

[2] James M. Clark and Allan Paivio. 1991. Dual coding theory and education. *Educational Psychology Review* 3, 3 (1991), 149–210. doi:10.1007/BF01320076

[3] Arghavan Moradi Dakhel, Vahid Majdinasab, Amin Nikanjam, Foutse Khomh, Michel C. Desmarais, and Zhen Ming (Jack) Jiang. 2023. Github Copilot AI pair programmer: Asset or liability? *J. of Systems and Software* 203 (2023), 111734.

[4] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with Copilot: Exploring prompt engineering for solving CS1 problems using natural language. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. ACM, 1136–1142.

[5] James Finnie-Ansley, Paul Denny, Andrew Luxton-Reilly, Eddie Antonio Santos, James Prather, and Brett A. Becker. 2023. My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. In *Proceedings of the 25th Australasian Computing Education Conference*. ACM, 97–104.

[6] Jiaxin Ge, Zora Zhiruo Wang, Xuhui Zhou, Yi-Hao Peng, Sanjay Subramanian, Qinyue Tan, Maarten Sap, Alane Suhr, Daniel Fried, Graham Neubig, and Trevor Darrell. 2025. AutoPresent: Designing Structured Visuals from Scratch. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[7] Svetoslav Georgiev and Joseph Tinsley. 2024. Exploring Student Acceptance and Perceptions of AI-Assisted PowerPoint Creation. *African Journal of Inter/Multidisciplinary Studies* 6, 1 (2024), 1–13.

[8] Quan Connie Gu, Daniel Hickey, and Kimiko Ryokai. 2025. When AI Tells Their Story: Researchers' Reactions to AI-Generated Podcasts as a Tool for Communicating Research. In *Extended Abstracts of the CHI Conf. on Human Factors in Computing Systems (CHI EA '25)*. ACM.

[9] Michael Henderson, Margaret Bearman, Jennifer Chung, Tim Fawns, Simon Buckingham Shum, Kelly E Matthews, and Jimena de Mello Heredia. 2025. Comparing Generative AI and teacher feedback: student perceptions of usefulness and trustworthiness. *Assessment & Evaluation in Higher Education* (2025), 1–16.

[10] Mollie Jordan, Kevin Ly, and Adalbert Gerald Soosai Raj. 2024. Need a Programming Exercise Generated in Your Native Language? ChatGPT's Got Your Back: Automatic Generation of Non-English Programming Exercises Using OpenAI GPT-3.5. In *Proc. of the 55th ACM Technical Symposium on Computer Science Education V. 1*. Association for Computing Machinery.

[11] Ranim Khojah, Mazen Mohamad, Philipp Leitner, and Francisco Gomes de Oliveira Neto. 2024. Beyond Code Generation: An Observational Study of ChatGPT Usage in Software Engineering Practice. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 1819–1840.

[12] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, Seth Bernstein, Joanne Kim, Andrew Tran, and Arto Hellas. 2023. Comparing Code Explanations Created by Students and Large Language Models. In *Proc. of the 2023 Conf. on Innovation and Technology in Computer Science Education V. 1*. ACM.

[13] Zhuoyan Li, Chen Liang, Jing Peng, and Ming Yin. 2024. How Does the Disclosure of AI Assistance Affect the Perceptions of Writing?. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 4849–4868.

[14] Evanfiya Logacheva, Arto Hellas, James Prather, Sami Sarsa, and Juho Leinonen. 2024. Evaluating Contextually Personalized Programming Exercises Created with Generative AI. In *Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1*. 95–113.

[15] Richard E. Mayer. 2005. Cognitive theory of multimedia learning. In *The Cambridge Handbook of Multimedia Learning*, Richard E. Mayer (Ed.). Cambridge University Press, New York, 31–48.

[16] Richard E. Mayer. 2017. Using multimedia for e-learning. *Journal of computer assisted learning* 33, 5 (2017), 403–423.

[17] Shakked Noy and Whitney Zhang. 2023. Experimental evidence on the productivity effects of generative artificial intelligence. *Science* 381, 6654 (2023), 187–192.

[18] OpenAI. 2025. *GPT-5 System Card.* https://openai.com/index/gpt-5-system-card/ Accessed on 2025-11-03.

[19] Vinay Patel. 2024. Fake Or Real? Audio Captures AI Podcast Hosts Realising 'We're Not Human... What Happens When They Turn Us Off?'. *International Business Times* (UK). https://www.ibtimes.co.uk/fake-real-audio-captures-ai-podcast-hosts-realising-were-not-human-what-happens-when-they-1728290 Accessed: 20 April 2026.

[20] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. *arXiv preprint arXiv:2302.06590* (2023).

[21] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Petersen, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proc. of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*. ACM.

[22] James Prather, Juho Leinonen, Natalie Kiesler, Jamie Gorson Benario, Sam Lau, Stephen MacNeil, Narges Norouzi, Simone Opel, Vee Pettit, Leo Porter, Brent N. Reeves, Jaromir Savelka, IV Smith, David H., Sven Strickroth, and Daniel Zingaro. 2025. Beyond the Hype: A Comprehensive Review of Current Trends in Generative AI Research, Teaching Practices, and Tools. In *2024 Working Group Reports on Innovation and Technology in Computer Science Education*. ACM, 300–338.

[23] Kunal Rao, Giuseppe Coviello, Murugan Sankaradas, Ciro Giuseppe De Vita, Gennaro Mellone, and Srimat Chakradhar. 2025. SlideCraft: Context-aware Slides Generation Agent. In *2025 IEEE Conference on Pervasive and Intelligent Computing (PICom)*. IEEE, 165–172.

[24] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*. ACM.

[25] Jaromir Savelka, Arav Agarwal, Marshall An, Christopher Bogart, and Majd Sakr. 2023. Thrilled by Your Progress! Large Language Models (GPT-4) No Longer Struggle to Pass Assessments in Higher Education Programming Courses. In *Proc. of the 2023 ACM Conf. on Int. Computing Education Research-Volume 1*. ACM.

[26] Athar Sefid, Jian Wu, Prasenjit Mitra, and C. Lee Giles. 2019. Automatic slide generation for scientific papers. In *Third International Workshop on Capturing Scientific Knowledge co-located with the 10th International Conference on Knowledge Capture (K-CAP 2019), SciKnow@ K-CAP 2019*.

[27] Juha Sorva. 2013. Notional machines and introductory programming education. *ACM Transactions on Computing Education (TOCE)* 13, 2 (2013), 1–31. doi:10.1145/2483710.2483713

[28] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive Science* 12, 2 (1988), 257–285. doi:10.1207/s15516709cog1202_4

[29] Ismael Villegas Molina, Audria Montalvo, Shera Zhong, Mollie Jordan, and Adalbert Gerald Soosai Raj. 2024. Generation and Evaluation of a Culturally-Relevant CS1 Textbook for Latines using Large Language Models. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*. 325–331.

[30] Biao Wang. 2024. NotebookLM now lets you listen to a conversation about your sources. *Google Blog. September* 11 (2024).

[31] Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S. Yu, and Qingsong Wen. 2025. Large Language Models for Education: A survey and outlook. *IEEE Signal Processing Magazine* 42, 6 (2025), 51–63.

[32] Leon E. Winslow. 1996. Programming pedagogy—a psychological overview. *ACM SIGCSE Bulletin* 28, 3 (1996), 17–22. doi:10.1145/234867.234872

[33] Eric Xie, Danielle Waterfield, Michael Kennedy, and Aidong Zhang. 2026. SlideBot: A Multi-Agent Framework for Generating Informative, Reliable, Multi-Modal Presentations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 40907–40915.

[34] Weicheng Xing, Tianqing Zhu, Jenny Wang, and Bo Liu. 2024. A Survey on MLLMs in Education: Application and Future Directions. *Future Internet* (2024).