Human Interaction and Emerging Technologies (IHIET-FS 2026), Vol. 235, 2026, 100–109
https://doi.org/10.54941/ahfe1007265

Multimedia Web Content Generation Using Large Language Models With Chain-of-Thought Reasoning Strategy

Chao-Hsi Cheng[1], Shyi-Chyi Cheng[1], and Hsun-Yu Lan[2]

[1]Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung 202301, Taiwan

[2]Department of Aquaculture, National Taiwan Ocean University, Keelung 202301, Taiwan

ABSTRACT

Recent advances in Large Language Models (LLMs) and Chain-of-Thought (CoT) reasoning have improved the quality of AI-generated multimedia content and code. However, the reliability and stability of generated outputs still depend heavily on prompt design, model capability, and application context. This study examines a human–AI collaborative framework for multimedia web content generation and programming education. By integrating CoT-guided prompting with full-stack web development processes, the framework supports frontend interface construction, backend logic implementation, and database connectivity within a unified workflow. Two prompting approaches are compared: conventional zero-shot generation and CoT reasoning. The generated outputs are evaluated using quantitative indicators, including memory usage, execution time, and test coverage, together with user-based assessments of visual aesthetics and creative expression. The results show that CoT prompting improves the logical consistency and structural completeness of AI-generated code when compared with conventional zero-shot generation. Comparative benchmarking also indicates that ChatGPT and Gemini produce more stable results than Grok in tasks involving complex algorithmic reasoning. From an educational perspective, the framework further demonstrates the potential of generative AI to support learners without strong programming backgrounds through structured prompting, intermediate feedback, and iterative refinement. These findings suggest that CoT-guided human–AI collaboration can provide practical support for multimedia web development and programming education, while also revealing the current limitations of generative AI in resource-constrained and logic-intensive tasks.

Keywords: Generative AI, Chain-of-thought reasoning, Multimedia web development, Programming education, Human–AI collaboration

INTRODUCTION

Recent advances in generative artificial intelligence have significantly enhanced automated content generation capabilities. Large Language Models (LLMs) built upon the Transformer architecture have demonstrated strong performance in natural language processing and code generation tasks (Vaswani et al., 2017). In parallel, diffusion-based generative models have enabled high-fidelity multimedia synthesis through iterative denoising processes in latent feature spaces (Rombach et al., 2022). Multimodal

Received March 16, 2026; Revised April 3, 2026; Accepted April 22, 2026; Available online June 22, 2026

alignment techniques such as CLIP further enable AI systems to associate textual descriptions with visual representations, supporting the generation of diverse multimedia content from natural language instructions (Radford et al., 2021). These developments have expanded the potential of generative systems in areas such as interactive media, software development, and digital content creation.

As the scale of language models increases, researchers have begun to examine their reasoning capabilities and reliability. Large language models may exhibit emergent abilities when model parameters and training data reach sufficient scale (Wei et al., 2022). This observation has motivated the development of prompt-engineering strategies designed to guide model reasoning during generation. Chain-of-Thought (CoT) prompting, for example, encourages models to generate intermediate reasoning steps before producing final outputs, which can improve logical consistency in complex tasks. Nevertheless, generative models may still produce outputs that appear plausible but contain logical errors due to their probabilistic generation mechanisms. Such limitations have been discussed in studies describing large language models as "stochastic parrots," highlighting concerns regarding reliability and interpretability in large-scale AI systems (Bender et al., 2021).

Generative AI technologies have also begun to influence programming education and software development practices. Conversational AI systems can assist learners by generating example code, suggesting debugging strategies, and explaining programming logic (Kazerouni et al., 2023). Recent studies further indicate that large language models can support programming tasks and interactive learning environments by providing automated code generation and feedback mechanisms (Zheng et al., 2023). From an instructional perspective, such systems may reduce barriers for novice programmers and support scaffolded learning processes. According to Cognitive Load Theory, effective instructional design must balance the complexity of learning materials with the cognitive capacity of learners (Sweller, 1988). AI-assisted programming systems therefore have the potential to provide adaptive guidance and structured explanations that facilitate the understanding of complex programming concepts (Chen et al., 2020; Becker et al., 2023).

Despite these advances, generating complete multimedia web applications remains challenging. Multimedia web systems typically require the integration of several technical components, including frontend interface design, backend program logic, and database connectivity. When conventional LLM-based approaches attempt to directly generate full application code, the outputs may contain fragmented structures, inconsistent variable usage, or incomplete functional integration. Moreover, when code is generated without explicit reasoning processes, learners may find it difficult to understand how system architecture and program logic are constructed, limiting the pedagogical value of AI-assisted programming tools.

This study therefore investigates the use of Chain-of-Thought reasoning within a human–AI collaborative framework for multimedia web content generation and programming education. The proposed approach integrates CoT-guided reasoning with full-stack web development processes, including frontend interface construction, backend logic implementation using Flask, and database connectivity through MySQL. Through structured prompting and interaction strategies, the system guides the model to decompose complex

programming tasks into sequential reasoning steps before generating executable code.

Based on this framework, the research addresses two questions. First, how do different large language models—ChatGPT, Gemini, and Grok—differ in their ability to generate logically consistent and structurally complete Python code for multimedia web system integration? Second, does the use of Chain-of-Thought reasoning as an instructional strategy improve the logical integrity and creative quality of student outcomes? By examining these questions, this study evaluates the feasibility of structured human–AI collaboration for supporting multimedia web development and programming education.

METHODOLOGY

This study proposes a human–AI collaborative framework for multimedia web content generation and programming support. The framework integrates large language models with structured prompting strategies to assist the development of multimedia web applications. Within this system, generative AI tools support the creation of web interfaces, backend program logic, and database connectivity, allowing users to iteratively construct functional multimedia web pages while maintaining control over the development process.

The framework is designed to support the generation of full-stack web applications. It incorporates multiple development stages, including frontend interface construction, backend logic implementation, and database interaction. During the generation process, the AI system receives structured prompts describing functional requirements and progressively produces corresponding program components. Human users can review intermediate outputs, refine prompts, and adjust system behavior to ensure that the generated results remain consistent with the intended application design.

The overall architecture of the proposed framework is shown in Figure 1. The system integrates generative AI models with human–computer interaction mechanisms, enabling collaborative development of multimedia web systems. Through iterative prompting and user feedback, AI-generated outputs can be progressively refined, allowing the framework to gradually construct complete multimedia web applications.

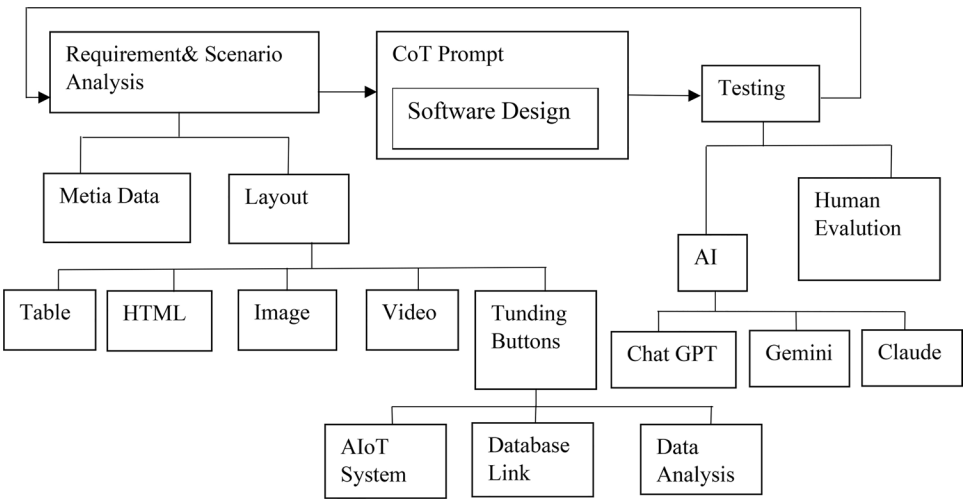

Figure 1: Human–AI collaborative framework for multimedia web content generation.

Specifically, the CoT prompting strategy in this study is implemented through a structured, multi-stage instructional design. Instead of requesting a monolithic code output, the system provides prompts that guide the model through a "think-before-code" workflow. This process includes logical decomposition of functional requirements, architectural planning of the frontend–backend interface, and iterative debugging of database connectivity scripts. By embedding explicit reasoning instructions, such as analyzing data flow before generating Flask routes, the framework encourages the model to maintain logical consistency throughout the development process.

The objective of this study is to examine the pedagogical effectiveness of generative AI in programming and multimedia design under different human–AI interaction strategies. In particular, the research investigates whether structured prompting methods can improve the logical consistency and structural reliability of AI-generated code. Two prompting approaches are therefore examined: conventional zero-shot generation, which serves as the baseline condition, and Chain-of-Thought (CoT) reasoning, which introduces explicit intermediate reasoning steps during the code generation process.

To evaluate the quality of the generated outputs, the study adopts several quantitative assessment criteria. Code quality is analyzed using indicators such as memory usage (space complexity), execution time (time complexity), and test coverage, which collectively reflect the computational efficiency and robustness of the generated programs. In addition to these system-level metrics, the multimedia outputs produced by the models are assessed through user-based evaluations focusing on visual aesthetics and creative expression.

The study further explores whether generative AI can assist learners without formal programming backgrounds. By providing structured guidance and iterative feedback during the development process, the system is intended to function as a pedagogical partner within hybrid hardware–software learning environments. For experimental comparison, three representative large language models are evaluated: ChatGPT (GPT-5 series), Gemini (version 3), and Grok (version 4.1). These models operate within a Human-in-the-Loop (HITL) interaction framework that allows users to iteratively refine prompts and review intermediate outputs. Retrieval-augmented context and self-correction mechanisms are incorporated to improve the reliability of generated code throughout the development process.

The experimental environment is standardized on a Raspberry Pi 4B running a 64-bit operating system in order to examine the interoperability between AI-generated software components and resource-constrained hardware platforms. Within this configuration, the effectiveness of the proposed framework depends primarily on prompt specificity and the iterative refinement enabled by the HITL interaction process.

The multimedia learning platform applies the CoT prompting framework to coordinate frontend presentation, application logic, and data management within a unified development structure. By guiding the reasoning process step by step, the model is able to maintain logical coherence while generating different components of the multimedia web system.

Figure 2 presents an example of a multimedia web page generated under the CoT prompting strategy. The result shows how structured reasoning prompts

guide the model to produce coherent webpage structures and functional elements that correspond to the specified instructional requirements. To support full-stack functionality, the generated web interface is integrated with backend processing and persistent data storage. Figure 3 illustrates the interaction between the multimedia user interface and the backend database system. In this architecture, HTML and CSS define the interactive frontend environment, while Python-based logic manages application states such as learner progress and task evaluation. Backend communication is implemented through the Flask microframework, enabling asynchronous data exchange between the user interface and server-side processes. MySQL is employed for persistent storage of learner performance data, allowing long-term tracking and analysis of learning outcomes.

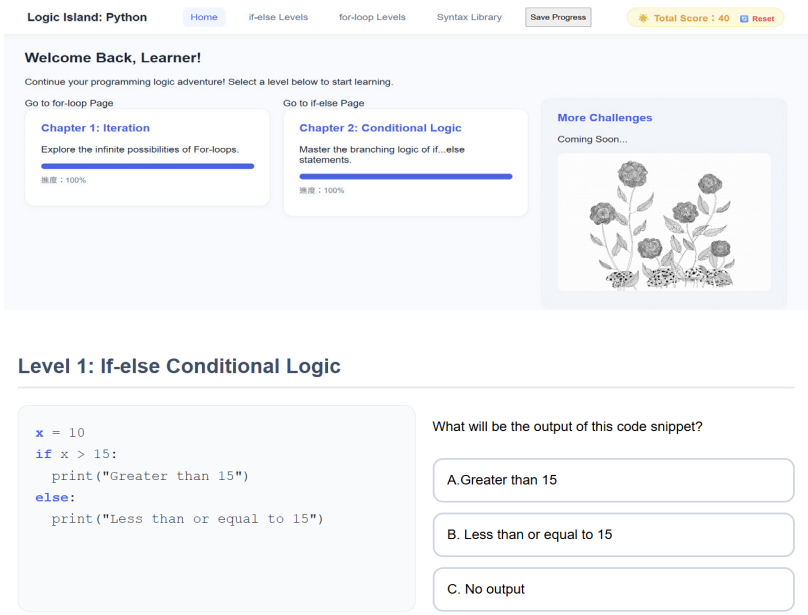

Figure 2: Multimedia web page generation using Chain-of-Thought reasoning.

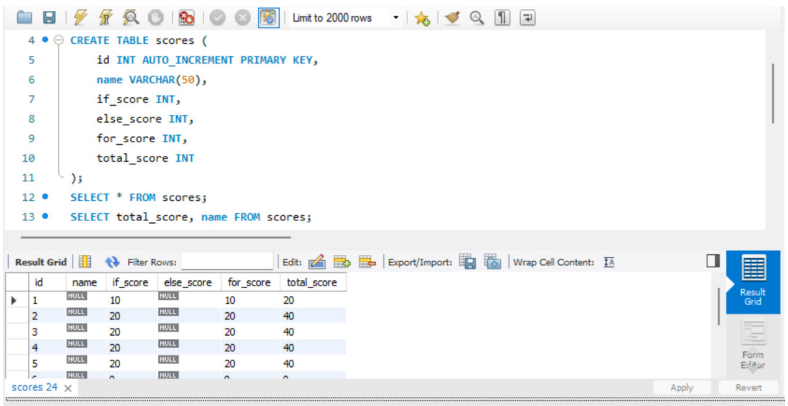

Figure 3: Integration of multimedia web interface with backend database.

Through this layered design, the platform integrates interface presentation, program logic, and database management into a coherent multimedia learning environment, allowing generative AI to support both web-based system construction and programming instruction. For comparison, the outputs generated without CoT guidance are shown in Figure 4, where structural inconsistency and incomplete functional integration can be observed more clearly. A corresponding comparison between zero-shot generation and CoT prompting is summarized in Table 1, which further highlights the advantages of structured reasoning in improving requirement alignment, variable consistency, and code annotation quality.

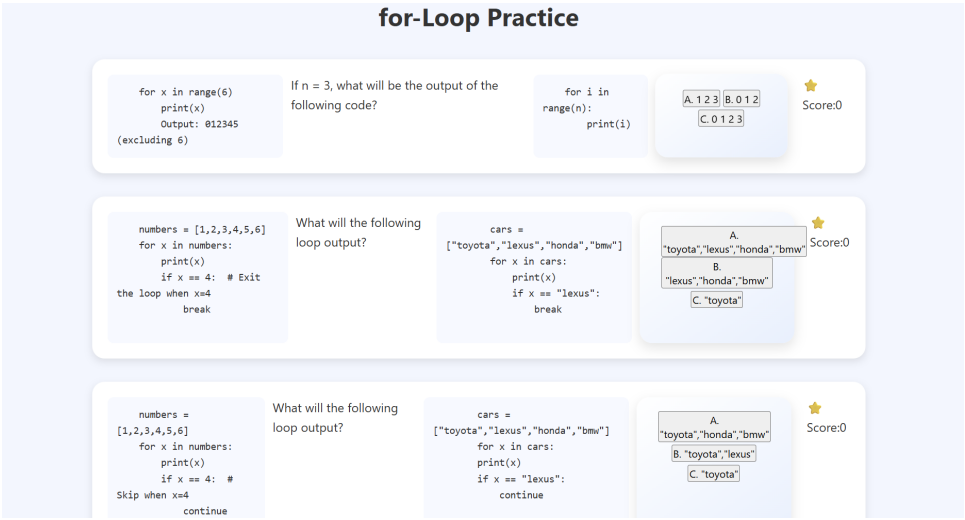

Figure 4: Web generation results without CoT guidance, showing structural inconsistency and incomplete functional integration.

Table 1 shows that prompt design has a direct influence on the quality of generated outputs. Under standard prompting, vague or single-layer instructions often failed to support multi-step tasks, particularly those involving cross-platform data retrieval, interface–logic coordination, or complex event handling. In contrast, the introduction of CoT reasoning enabled these tasks to be decomposed into more manageable subtasks, resulting in better requirement alignment, more consistent variable usage, and improved code annotation quality. The comparison also suggests that structured reasoning can reduce manual correction effort and improve the accuracy of frontend-to-backend functional mapping.

To further evaluate the proposed CoT-driven framework, a series of quantitative and qualitative assessments was conducted. The experiments were executed in a standardized x86-64 workstation environment to establish a performance baseline. Each test was repeated five times, and the average values were recorded. The evaluation focused on three aspects: logic coverage, which reflects the ability of the generated code to handle edge cases and exception conditions; execution efficiency, measured in terms of runtime

and memory consumption; and readability and pedagogical value, assessed through the comment-to-code ratio and the semantic clarity of variables.

The benchmarking results for ChatGPT, Gemini, and Grok across different programming tasks are presented in Table 2. Overall, ChatGPT and Gemini showed relatively stable execution times across most tasks, whereas Grok exhibited substantial latency in more complex algorithmic problems, particularly in linear search and binary search tasks. These results indicate that model performance varies considerably depending on task structure and computational complexity.

Table 1: Comparison of zero-shot generation and Chain-of-Thought prompting in multimedia web code generation.

Evaluation Metrics	Zero-Shot	Chain-of-Thought
Structural Integrity & Requirement Alignment	Generated partial HTML with functional misalignment to requirements.	Generated complete multimedia web pages, including HTML, CSS, Flask transmission logic, and backend database integration.
Error Rate	Inconsistent variable naming.	Consistent variable naming.
Annotation Rate	Code with modular/segmented annotations	Code featuring modularized/segmented annotations.

Table 2: Comparison of time spent by various AI models on different programming topics (Unit: Seconds).

	Chat GPT GPT-4	Gemini Gemini 3	Grok Grok 4.1	Manual Creation
if else	0.053299	0.05107	0.051838	0.049712
OOP	0.5199	0.53929	0.55738	0.51854
Web crawlers	0.05016	0.058332	0.083993	0.055443
Linear Search	0.000114	0.000122	22.6712	0.000161
Binary Search	0.000104	0.000108	18.46326	0.000213
Binary Search: left-closed, right-open	0.000111	0.000092	9.386846	0.000228
Bubble Sort	0.000229	0.002801	0.000349	0.000206

In addition to runtime performance, the robustness of generated code was examined through logic coverage. As shown in Figure 5, ChatGPT and Gemini generally achieved higher coverage across algorithmic tasks, while Grok showed weaker performance in cases involving deeper nesting and more complex logical conditions. This pattern suggests that CoT-guided generation is more effective when supported by models with stronger reasoning consistency, particularly in programming tasks that require structured control flow and reliable exception handling.

Taken together, these results indicate that the proposed framework not only improves the organization of code generation through structured

prompting, but also reveals clear differences in performance among current large language models when applied to multimedia web development and programming education.

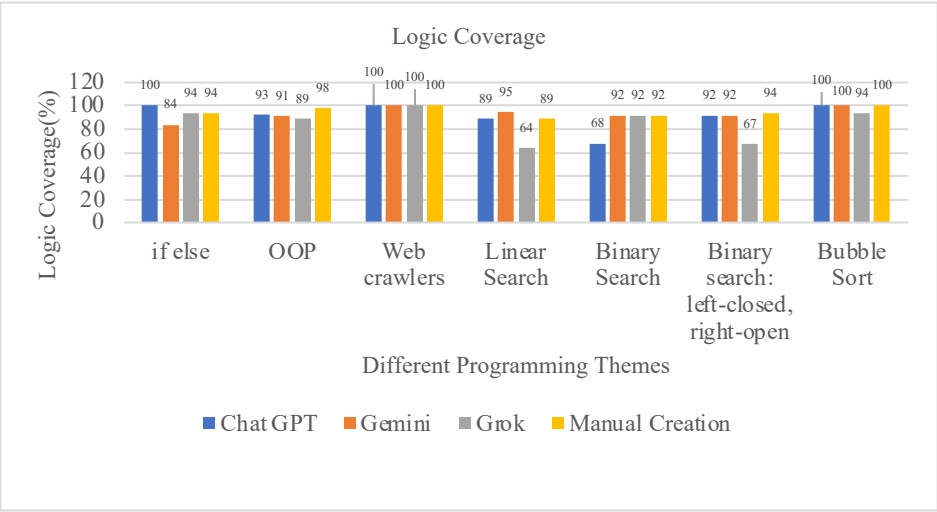

Figure 5: Logic coverage of different AI models across programming tasks.

CONCLUSION

This study investigated the use of Chain-of-Thought (CoT) reasoning within a human–AI collaborative framework for multimedia web content generation and programming education. By integrating structured prompting strategies with full-stack web development tasks, the proposed framework was designed to support frontend interface generation, backend logic implementation, and database connectivity within a unified development process.

The results indicate that CoT prompting improves the logical consistency and structural completeness of AI-generated code when compared with conventional zero-shot generation. In particular, structured reasoning was found to be effective in reducing requirement misalignment, improving frontend-to-backend function mapping, and lowering the amount of manual correction needed during development. These findings suggest that CoT can serve not only as a prompting technique for content generation, but also as a practical mechanism for organizing complex programming tasks in educational and multimedia application contexts.

The comparative evaluation of ChatGPT, Gemini, and Grok further showed that current large language models exhibit clear differences in programming performance. ChatGPT and Gemini produced more stable results in terms of execution time and logic coverage, whereas Grok showed weaker performance in more complex algorithmic tasks. These results confirm that model capability remains an important factor in determining

the effectiveness of CoT-guided code generation, especially in tasks involving structured control flow and multi-step reasoning.

From an educational perspective, the proposed framework demonstrates the potential of generative AI to support learners who do not have strong programming backgrounds. Through iterative prompting, intermediate feedback, and stepwise task decomposition, the system can provide a more interpretable and guided development process. In this sense, the study contributes not only a technical framework for multimedia web generation, but also a pedagogically oriented approach to human–AI collaborative learning.

At the same time, the study also reveals practical limitations. The performance of AI-generated code remains sensitive to prompt specificity, model capability, and hardware constraints. Future work will therefore focus on improving inference efficiency, refining prompt design strategies, and enhancing the stability of code generation in resource-constrained educational settings.

ACKNOWLEDGMENT

This work was supported in part by the National Science and Technology Council, Taiwan, under grant numbers NSTC 114-2221-E-019-022-MY3 and NSTC 114-2637-8-011-004-.

REFERENCES

Becker, B. A., Denny, P., Finnie-Ansley, J., Prather, J., Santos, E. A., and Wallingford, E. (2023). Programming is hard—or at least it used to be: Educational opportunities and challenges of AI code generators. In: Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE).

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In: Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT), pp. 610–623.

Chen, X., Xie, H., Zou, D., and Hwang, G. J. (2020). Application and theory gap studies of artificial intelligence in education. Computers and Education: Artificial Intelligence, 1, 100002.

Kazerouni, A. M., Chen, Z., Li, Y., and Worsley, M. (2023). Conversational AI for learning to code: An empirical study. In: Proceedings of the ACM Conference on International Computing Education Research (ICER).

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 8748–8763.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10684–10695.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. Cognitive Science, 12(2), 257–285.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998–6008.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., and Zhou, D. (2022). Emergent abilities of large language models. Transactions on Machine Learning Research.

Zheng, Z., Liu, X., Xie, H., and Zou, D. (2023). A survey on large language models for code generation. arXiv preprint, arXiv:2306.01234.