TABLE I
LAYER 1: STUDENT NEEDS PRINCIPLES FOR STRUCTURING LLM-GENERATED FEEDBACK IN NOVICE PROGRAMMING

ID	Prompt Design Principle	Pedagogical Basis
N1	Guide the prompt to generate constructive feedback focused on improvement rather than merely identifying errors.	[1], [2]
N2	Include instructions in the prompt that ensure feedback explains what the mistake is, why it occurred, and how to fix it.	[1], [3], [4]
N3	Direct the prompt to provide quick fix suggestions for minor, unintentional mistakes such as missing symbols or formatting errors.	[5], [6]
N4	Instruct the prompt to provide example-based feedback when student errors indicate a misunderstanding of syntax or structure.	[4], [5], [7]
N5	Design the prompt to generate feedback using clear and direct language that novice learners can easily understand and apply.	[1], [4], [7], [8]
N6	Instruct the prompt to provide hints instead of solutions, supporting student independence and critical thinking.	[3], [4], [9]
N7	Guide the prompt to include motivational feedback that acknowledges student effort and the achievement of subgoals.	[3], [10]
N8	Structure the prompt to deliver feedback progressively, guiding students step-by-step through the correction process.	[3], [4], [7]
N9	Design the prompt to generate feedback that includes metacognitive questions prompting students to reflect on their work.	[3], [11]
N10	Instruct the prompt to generate feedback that adjusts to the student's progress, acknowledges effort, and supports continued improvement.	[3], [4]
N11	Design the prompt to generate feedback segmented into distinct, specific parts to enhance clarity and support student comprehension.	[1], [4], [7], [8]
N12	Instruct the prompt to address common novice-level errors related to syntax, semantics, and conceptual misunderstandings.	[1], [12]

The prompt design principles presented in Layer 1, as shown in Table I, contribute to the development of three key components within the PPE-LLM framework: *Provide Student Background* **(Comp 3)**, *Structure Feedback Using Pedagogical Frameworks* **(Comp 6)**, and *Implement Additional Feedback Guidelines* **(Comp 7)**. They guide prompt engineering to ensure that the generated feedback is context-aware, pedagogically structured, and aligned with essential feedback qualities such as clarity, and tone.

TABLE II
LAYER 2: PEDAGOGICAL PRINCIPLES FOR STRUCTURING LLM-GENERATED FEEDBACK

ID	Pedagogical Principle	Pedagogical Basis
P1	Guide the prompt to include step-by-step worked examples that support conceptual understanding before students attempt independent problem-solving (e.g., pseudocode).	[8]
P3	Structure prompts to follow a Guidance-to-Independence approach: identify the issue, offer hints, and promote self-correction.	[4], [9]
P4	Prompt the model to encourage students to self-verify, reflect, and optimize their work to enhance self-regulation and understanding.	[11], [13]
P5	Ensure the prompt generates feedback that addresses task-level correctness, process-level strategies, and self-regulation for independent application.	[2]
P6	Guide the prompt to organize feedback into setting learning goals (Feed Up), evaluating current progress (Feed Back), and guiding the next steps for improvement (Feed Forward).	[2]
P7	Avoid prompts that generate personal evaluation; ensure feedback focuses on task improvement.	[4], [14]
P8	Instruct the model to extend feedback beyond simple verification by explaining what the issue is, how it occurs, and why it matters.	[4]
P9	Guide the prompt to segment feedback into clear, manageable parts to support student comprehension.	[4], [7], [8]
P10	Ensure the prompt elicits feedback that is specific, clear, and actionable.	[4]
P11	Avoid complex or abstract language in prompts; tailor explanations to match the learner's level.	[4], [7], [8]
P12	Design prompts that ensure objectivity and focus on performance outcomes, not personal traits.	[4], [10]
P13	Shift the focus in prompts from performance goals to learning goals to encourage growth-oriented thinking.	[4], [11]
P14	Encourage the model to highlight student strengths and suggest improvements without assigning grades.	[4], [14]
P15	Set the prompt to use a supportive and encouraging tone to maintain motivation and build confidence.	[4], [10]
P16	Guide the prompt to promote independent problem-solving through scaffolding rather than direct answers.	[4], [9]
P17	Limit feedback scope in the prompt by focusing on key errors and avoiding unnecessary detail.	[4], [7], [8]
P18	Tailor prompts to provide corrective guidance and foundational support for low-achieving learners.	[4]
P19	Adjust the prompt to challenge high-achieving learners through deeper questions and advanced hints rather than giving solutions directly.	[4], [10]

The prompt design principles presented in Layer 2, as shown in Table II, contribute to the development of four key components within the PPE-LLM framework: *Defining the Goal of Feedback* **(Comp 2)**, *Define Evaluation Criteria* **(Comp 5)**, *Structure Feedback Using Pedagogical Frameworks* **(Comp 6)**, and *Implement Additional Feedback Guidelines* **(Comp 7)**. They guide prompt engineering to ensure that the generated feedback is purpose-driven, tailored to the learner's level, structured according to pedagogical best practices, and delivered with supportive tone and clarity.

TABLE III
LAYER 3: BEST PROMPTING PRACTICES PRINCIPLES FOR FOR STRUCTURING LLM-GENERATED FEEDBACK

ID	Best Prompting Practices	References
B1	Guide the prompt to adapt feedback based on the student's achievement level to ensure relevance and appropriate challenge.	[15], [16]
B2	Structure the prompt to organize feedback in tiers, offering graduated support to scaffold learning effectively.	[16]
B3	Instruct the model explicitly to avoid giving direct answers by including terms like "hint" within the prompt.	[15]–[17]
B4	Avoid prompting for compliments, as it leads to overly lengthy and irrelevant responses.	[17]
B5	Use prompt terms such as "student" and "hint" to elicit a friendly tone and promote clarity in explanation.	[17]
B6	Avoid over-constraining the prompt with strict word or sentence limits, as this can reduce flexibility and feedback quality.	[17]
B7	Emphasize a clear and concise problem description in the prompt rather than including model solutions or code answers.	[16], [17]
B8	Incorporate information about students' prior knowledge in the prompt to help the model align feedback with the student's level.	[18]
B9	Instruct the model to avoid vague praise (e.g., "Good job" or "You're on the right track"), especially for simple tasks.	[18]
B10	Provide clear instructions in the prompt to prevent full answers, exact solutions, or executable code snippets in the feedback.	[18]
B11	Instruct the model to generate feedback that clearly separates what has been achieved from what needs improvement.	[18]
B12	Refine and adjust prompts continuously to enhance relevance and clarity.	[16]
B13	Ensure the prompt or system includes a clear statement that the feedback is AI-generated, in line with ethical disclosure practices to build student trust and promote responsible AI use.	[19]

The prompt design principles presented in Layer 3, as shown in Table III, contribute to the development of four key components within the PPE-LLM framework: *Provide Student Background* ***(Comp 3)***, *Provide Essential Inputs* ***(Comp 4)***, *Implement Additional Feedback Guidelines* ***(Comp 7)***, and *Disclose the Use of AI in Feedback* ***(Comp 10)***. These principles guide prompt engineering to ensure that feedback is tailored to the learner's prior knowledge and progress, grounded in relevant and pedagogically appropriate feedback, delivered in a clear, supportive, and instructional manner and transparently acknowledges the role of AI in the feedback process.

TABLE IV
LAYER 4: TECHNICAL PROMPT DESIGN GUIDELINES

ID	Technical Prompt Design Guidelines	References
T1	Clearly state your objective to ensure the response matches your needs.	[20]
T2	Specify the target audience and desired style or tone.	[20]–[22]
T3	Provide context and reference specific sources.	[20]
T4	Add external sources dynamically to provide relevant information.	[20], [21]
T5	Break complex tasks into simpler subtasks for better accuracy.	[20], [21]
T6	Define the output format (e.g., table, list, or paragraph) for clear structure.	[20]
T7	Begin the prompt with clear instructions and use separators such as `###` or `"""` to distinguish instructions from the context.	[20]
T8	Use precise language and avoid ambiguity.	[20], [21]
T9	Avoid vague or overly detailed descriptions.	[20], [21]
T10	Clearly defining the role of the LLM (e.g., tutor or instructor) to shape the tone and instructional style of the feedback.	[23]
T11	Verify responses with self-checks to mitigate hallucinations and improve accuracy.	[24]

The prompt design principles presented in Layer 4, as shown in Table IV, contribute the development of five key components within the PPE-LLM framework: *Determining the Role of LLMs* ***(Comp 1)***, *Provide Essential Inputs* ***(Comp 4)***, *Implement Additional Feedback Guidelines* ***(Comp 7)***, *Include the Verification and Validation Process* ***(Comp 8)***, and *Define the Feedback Output Format* ***(Comp 9)***. They guide prompt engineering to ensure that the LLM understands its instructional role, receives the right input, generates feedback in an appropriate tone and format, and verifies its outputs to reduce hallucinations and improve accuracy.

References

[1] A. Luxton-Reilly, B. A. Becker, Y. Cao, R. McDermott, C. Mirolo, A. Mühling, A. Petersen, K. Sanders, Simon, and J. Whalley, “Developing assessments to determine mastery of programming fundamentals,” in *Proceedings of the 2017 ITiCSE Conference on Working Group Reports*, ser. ITiCSE-WGR ’17. New York, NY, USA: Association for Computing Machinery, 2018, p. 47–69. [Online]. Available: https://doi.org/10.1145/3174781.3174784

[2] J. Hattie and H. Timperley, “The power of feedback,” *Review of Educational Research*, vol. 77, no. 1, pp. 81–112, 2007.

[3] J. Jeuring, H. Keuning, S. Marwan, D. Bouvier, C. Izu, N. Kiesler, T. Lehtinen, D. Lohr, A. Peterson, and S. Sarsa, “Towards giving timely formative feedback and hints to novice programmers,” in *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education*, ser. ITiCSE-WGR ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 95–115. [Online]. Available: https://doi.org/10.1145/3571785.3574124

[4] V. J. Shute, “Focus on formative feedback,” *Review of educational research*, vol. 78, no. 1, pp. 153–189, 2008.

[5] U. Z. Ahmed, N. Srivastava, R. Sindhgatta, and A. Karkare, “Characterizing the pedagogical benefits of adaptive feedback for compilation errors by novice programmers,” in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*, ser. ICSE-SEET ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 139–150. [Online]. Available: https://doi.org/10.1145/3377814.3381703

[6] S. Narciss, “Feedback strategies for interactive learning tasks,” in *Handbook of research on educational communications and technology*. Routledge, 2008, pp. 125–143.

[7] R. E. Mayer, “Applying the science of learning: evidence-based principles for the design of multimedia instruction.” *American psychologist*, vol. 63, no. 8, p. 760, 2008.

[8] J. Sweller, “Cognitive load during problem solving: Effects on learning,” *Cognitive science*, vol. 12, no. 2, pp. 257–285, 1988.

[9] D. Wood, J. S. Bruner, and G. Ross, “The role of tutoring in problem solving,” *Journal of child psychology and psychiatry*, vol. 17, no. 2, pp. 89–100, 1976.

[10] J. Hattie, *Visible learning for teachers: Maximizing impact on learning*. Routledge, 2012.

[11] D. J. Nicol and D. Macfarlane-Dick, “Formative assessment and self-regulated learning: A model and seven principles of good feedback practice,” *Studies in higher education*, vol. 31, no. 2, pp. 199–218, 2006.

[12] A. Robins, J. Rountree, and N. Rountree, “Learning and teaching programming: A review and discussion,” *Computer science education*, vol. 13, no. 2, pp. 137–172, 2003.

[13] D. H. Schunk and B. J. Zimmerman, *Self-regulated learning: From teaching to self-reflective practice*. Guilford Press, 1998.

[14] P. Black and D. Wiliam, “Assessment and classroom learning,” *Assessment in Education: principles, policy & practice*, vol. 5, no. 1, pp. 7–74, 1998.

[15] H. Nguyen, N. Stott, and V. Allan, “Comparing feedback from large language models and instructors: Teaching computer science at scale,” in *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, ser. L@S ’24. New York, NY, USA: Association for Computing Machinery, 2024, p. 335–339. [Online]. Available: https://doi.org/10.1145/3657604.3664660

[16] H. Nguyen and V. Allan, “Using gpt-4 to provide tiered, formative code feedback,” in *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, ser. SIGCSE 2024. New York, NY, USA: Association for Computing Machinery, 2024, p. 958–964. [Online]. Available: https://doi.org/10.1145/3626252.3630960

[17] L. Roest, H. Keuning, and J. Jeuring, “Next-step hint generation for introductory programming using large language models,” in *Proceedings of the 26th Australasian Computing Education Conference*, ser. ACE ’24. New York, NY, USA: Association for Computing Machinery, 2024, p. 144–153. [Online]. Available: https://doi.org/10.1145/3636243.3636259

[18] A. Hellas, J. Leinonen, S. Sarsa, C. Koutcheme, L. Kujanpää, and J. Sorva, “Exploring the responses of large language models to beginner programmers’ help requests,” in *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1*, ser. ICER ’23. New York, NY, USA: Association for Computing Machinery, 2023, p. 93–105. [Online]. Available: https://doi.org/10.1145/3568813.3600139

[19] G. E. Cacciamani, M. B. Eppler, C. Ganjavi, A. Pekan, B. Biedermann, G. S. Collins, and I. S. Gill, “Development of the chatgpt, generative artificial intelligence and natural large language models for accountable reporting and use (cangaru) guidelines,” *arXiv preprint arXiv:2307.08974*, 2023.

[20] OpenAI, “Prompt engineering,” 2025, accessed: 2025-01-04. [Online]. Available: https://platform.openai.com/docs/guides/prompt-engineering

[21] Google AI, “Prompting strategies for gemini api,” 2025, accessed: 2025-03-03. [Online]. Available: https://ai.google.dev/gemini-api/docs/prompting-strategies

[22] Google Cloud, “Prompt engineering: overview and guide,” 2025, accessed: 2025-01-03. [Online]. Available: https://cloud.google.com/discover/what-is-prompt-engineering

[23] OpenAI, “Teaching with ai,” 2023, accessed: 2025-04-14. [Online]. Available: https://openai.com/index/teaching-with-ai/

[24] F. Harrington, E. Rosenthal, and M. Swinburne, “Mitigating hallucinations in large language models with sliding generation and self-checks,” *Authorea Preprints*, 2024.