



Contents lists available at [Journal ELORA](#)

JRTI (Jurnal Riset Tindakan Indonesia)

ISSN: 2502-079X (Print) ISSN: 2503-1619 (Electronic)

Journal homepage: <https://jrti.eloracenter.org/jrti>



Structured prompt engineering as scaffolding in generative ai-supported project-based network programming learning: a systematic literature review and conceptual framework

Muhammad Rizki Ananda¹, Titi Sri Wahyuni^{2*}, Fahmi Rizal², Resmi Darni²

¹ Postgraduate Program in Vocational Technology Education, Universitas Negeri Padang, Indonesia

² Universitas Negeri Padang, Indonesia

Article Info

Article history:

Received Aug 20th, 2026

Revised Sept 16th, 2026

Accepted Oct 02nd, 2026

Keyword:

Prompt engineering,
Generative AI,
Project-based learning,
Network programming

ABSTRACT

This systematized literature review examines how prompt engineering in generative AI (GenAI) has been used to support Project-Based Learning (PjBL) and develops a conceptual framework for Network Programming courses. Following the PRISMA 2020 statement, records published from 2020 to 2026 were retrieved from IEEE Xplore, ScienceDirect, SpringerLink, Google Scholar, and Garuda/Sinta. Of 310 records identified, 242 remained after duplicate removal, 56 were assessed in full text, and 24 studies met the eligibility criteria. Screening decisions were cross-checked by a second reviewer (Cohen's $\kappa = [\dots]$), and study quality was appraised with the Mixed Methods Appraisal Tool (MMAT). Data were analyzed through a hybrid deductive-inductive thematic analysis following Braun and Clarke's six phases: the four research questions served as a priori categories, sub-themes were coded inductively until no new codes emerged, and coding disagreements were resolved by consensus. The results revealed that structured and scaffolded prompting is consistently associated with deeper engagement than unstructured prompting in programming classes, and that GenAI-supported PjBL preserves student autonomy and creativity when AI interaction is embedded in project stages and students must justify their technical decisions. Only one included study addressed computer networking directly. Reported challenges were excessive dependence on AI-produced solutions, shallow learning, and unclear assessment practices. Unlike existing PjBL-GenAI models, which treat AI as a generally available tool, the proposed three-layer framework (competency, scaffolding, and assessment) specifies when and how prompting is taught, constrained, and evaluated across network programming project milestones, and it yields testable propositions for future empirical research.



© 2026The Authors.

This is an open access article under the CC BY-NC-SA license
(<https://creativecommons.org/licenses/by-nc-sa/4.0>)

Corresponding Author:

Titi Sri Wahyuni
Universitas Negeri Padang
Email: titisriwahyuni@ft.unp.ac.id

Introduction

Generative AI (GenAI) can substantially support programming learning, but it also introduces pedagogical risks. Students who routinely delegate problem solving to AI may become dependent on AI-generated solutions and engage only in shallow learning (Amoozadeh et al., 2024; Prather et al., 2024), and AI code generators have changed what novices need to learn about programming (Becker et al., 2023; Prather et al., 2023). As such,

prompt engineering, defined as the activity of designing inputs for AI models so that their outputs are relevant, accurate, and pedagogically meaningful, becomes critical (Kasneci et al., 2023). The so-called "Prompt Problems" activity, for example, asks students to write natural language prompts that make the AI generate correct code instead of writing the code manually (Denny et al., 2024). Similarly, structured prompting frameworks guide novices through precise problem description, identification of edge cases, and debugging (Manley et al., 2024). Without such guidance, students in introductory programming courses often paste the whole problem description at once and accept the AI output as a solution without verification, instead of refining their prompts iteratively (Amoozadeh et al., 2024).

The subject of Network Programming is considered a fundamental part of Information and Communication Technology study programs because it requires both the ability to handle practical tasks and the mastery of intricate problem-solving skills. These range from setting up network devices and socket programming to simulating communication protocols. Network Programming also differs from generic programming in three respects. First, much of its content concerns invisible processes, such as protocol layering, packet flow, and connection states, which students cannot observe directly; consequently, computer networks courses rely heavily on simulation, emulation, and hands-on laboratories (Prvan & Ožegović, 2020). Second, correctness depends not only on the code but also on the configuration and behavior of the network environment, so errors frequently surface as timeouts, packet loss, or address conflicts rather than as compiler messages. Third, GenAI is less reliable in this domain than its fluent answers suggest: large language models made elementary IP-addressing errors on networking questions and offered plausible but incorrect explanations that can induce misconceptions (Anwar & Caesar, 2025), and they still struggle to translate high-level requirements into correct network configurations (Wang et al., 2024). Unstructured GenAI use is therefore riskier in Network Programming than in introductory programming. Since this subject heavily relies on practice and is problem-based in nature, Project-Based Learning (PjBL) is a suitable approach, as it develops competencies through the completion of authentic projects (Kokotsaki et al., 2016). Evidence from related areas of technical education supports this: a study on master's software engineering students who applied PjBL and generative AI found that structured project work supported cognitive development and applied skills although GenAI tools were commonly used (Balakayeva et al., 2026). In design education, an authentic PjBL framework promoted students' creative and analytical thinking when students were engaged in structured AI dialogue (Indriati et al., 2024). Thus, it is a deliberately designed AI interaction, not the mere provision of AI, that makes project-based learning more productive. Educators should therefore embed AI assistants into lessons and emphasize prompt formulation and verification skills instead of instinctively restricting AI use (Bull & Kharrafa, 2024).

The intersection of these two aspects creates both opportunities and difficulties. GenAI guided by well-designed prompts might help students in network programming projects, for example, by locating configuration problems, explaining complex protocol concepts, or generating virtual troubleshooting scenarios. However, research on prompt engineering in education has mostly dealt with general higher education (Lee & Palmer, 2025) or with programming education in general (Qian, 2025; Elnaffar et al., 2026). Recent systematic reviews of LLMs in computing education likewise report evidence predominantly from introductory programming (Raihan et al., 2025; Elnaffar et al., 2026), and studies of GenAI-supported PjBL in technical fields have addressed software engineering and design (Balakayeva et al., 2026; Indriati et al., 2024) rather than network programming. Hence, there is an empirical gap between the ideal condition, in which GenAI use in Network Programming projects is guided by explicit, domain-appropriate prompting practices, and the actual condition, in which no validated model for such guidance exists. A systematic literature review is therefore needed to determine which prompting patterns are documented, how GenAI and PjBL have been integrated, and how these findings can be translated into a model suitable for a Network Programming course.

This study adopts a systematized literature review design (Grant & Booth, 2009) reported according to PRISMA 2020 (Page et al., 2021), followed by conceptual framework development. It is guided by four research questions: (1) What prompt-engineering utilization patterns in generative AI have been documented in the context of technical/vocational education? (2) What models, strategies, or frameworks have been used for integrating generative AI into PjBL? (3) To what extent have these been implemented in network programming or similar computing courses? (4) What challenges and success factors have been reported when generative AI was introduced in PjBL? Accordingly, the objectives of this study are (a) to identify and categorize prompt-engineering utilization patterns; (b) to compare GenAI-PjBL integration models; (c) to determine the extent of their implementation in networking and related computing courses; (d) to synthesize reported challenges and success factors; and (e) to derive a three-layer conceptual model with testable propositions for prompt-engineering-supported PjBL in Network Programming. By connecting three areas that have mostly been studied separately, namely prompt engineering, GenAI, and PjBL, this paper makes both a theoretical and a practical contribution.

Method

This paper utilized PRISMA 2020 guidelines (Page et al., 2021) and adopted a systematized literature review design, that is, a review that follows systematic procedures but is conducted with limited resources (Grant & Booth, 2009). These guidelines focus on making each stage of the research open to external scrutiny and replicable, from the initial identification of sources through successive phases of eligibility assessment to the final decision to include a work in the analysis. The methodology also incorporated the framework for systematic literature reviews in software engineering laid out by Kitchenham and Charters (2007), a widely respected method in domains such as software development and e-learning because it ensures that the review is thorough and based on logical principles rather than subjective impressions.

Literature searching was carried out in five online repositories: IEEE Xplore, Elsevier ScienceDirect, SpringerLink, Google Scholar, and Garuda/Sinta for nationally indexed articles (minimum Sinta 2). The searches were run between [...] and [...], and the final search was performed on [...]. Because the authors had no institutional subscription to Scopus, two mitigation steps were taken: (a) the same search strings were run in Google Scholar, which indexes most Scopus-listed venues, and the first [...] results of each query, sorted by relevance, were screened; and (b) the indexing status of every candidate article was verified title by title on scopus.com (Sources) or sinta.kemdikbud.go.id. The residual risk of missed records is discussed in the Limitations subsection.

The search strategy combined Boolean keywords representing the three core domains of this paper, namely prompt engineering, generative AI, and PjBL. Search string S1 was ("prompt engineering" OR "prompt design") AND ("generative AI" OR "large language model" OR "ChatGPT") AND ("project-based learning" OR "PjBL") AND ("education" OR "learning" OR "instruction"). Search string S2 replaced the PjBL clause with networking-domain terms to retrieve literature relevant to RQ3 even when PjBL was not mentioned explicitly: ("prompt engineering" OR "prompt design") AND ("generative AI" OR "large language model" OR "ChatGPT") AND ("network programming" OR "computer networking" OR "socket programming") AND ("education" OR "learning" OR "instruction"). For Garuda/Sinta, the Indonesian equivalent was used: [...]. Strings were applied to title, abstract, and keyword fields where the database allowed it, and to full text in Google Scholar, with filters for publication years 2020–2026 and language (English or Indonesian).

In this work, studies were considered if they were published anytime between 2020 and 2026; were peer-reviewed journal articles or conference proceedings indexed in reputable databases (Scopus, Web of Science, or Sinta S2 at minimum for national sources); were written in English or Indonesian; dealt with prompt engineering, generative AI, and/or PjBL in the field of education, operationalized as reporting an intervention, tool, framework, or empirical analysis in which at least one of these is a central component of teaching or learning in computing or technical education; and were available for full text reading. On the other hand, studies were discarded if they were purely opinion-related, editorial in nature, were non-peer-reviewed sources, or had no clear methodology; were the same article across multiple databases; or could not be accessed in full text.

In accordance with the PRISMA 2020 statement (Page et al., 2021), the selection process was conducted. We initially identified 310 potential records through database searches covering IEEE Xplore ($n = 42$), ScienceDirect ($n = 65$), SpringerLink ($n = 58$), Google Scholar ($n = 110$), and Garuda/Sinta ($n = 35$). Duplicates were removed and only 242 unique title-abstract pairs were identified for screening, resulting finally in the removal of 186 non-relevant records. The authors then carried out a detailed full-text review of the 56 retrieved papers. It turned out that the majority of the papers were excluded because: they were hard to obtain ($n = 6$), the methodologies were not clearly presented ($n = 10$), or they failed to mention programming/PjBL integration ($n = 16$). Only 24 papers were left that satisfied all criteria and were used for thematic analysis as shown in Figure 1.

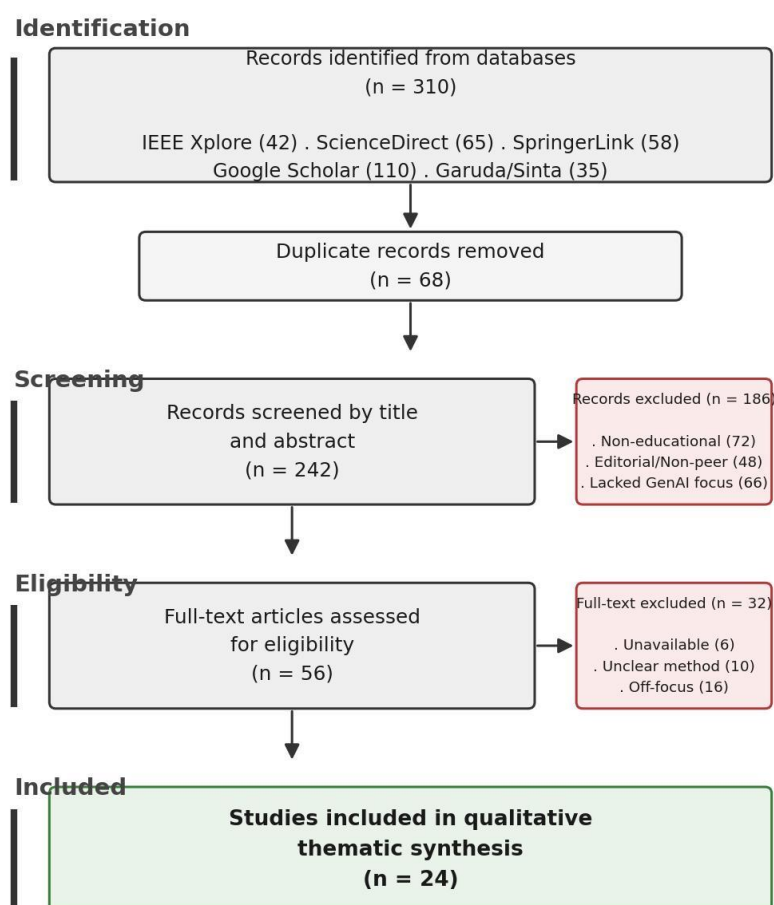


Figure 1. PRISMA 2020 flow diagram of the literature identification, screening, eligibility, and inclusion process.

Title/abstract screening and full-text eligibility assessment were conducted by the first author (MRA). To check the reliability of these decisions, the second author (TSW) independently screened a random sample of [...] records ([...] % of 242) and assessed all 56 full texts. Inter-rater agreement was Cohen's κ = [...] for title/abstract screening and κ = [...] for full-text eligibility (Cohen, 1960), interpreted following McHugh (2012). Disagreements were resolved through discussion, with the third author (RD) acting as arbiter. The methodological quality of the 24 included studies was appraised with the Mixed Methods Appraisal Tool (MMAT) version 2018 (Hong et al., 2018), which covers qualitative, quantitative, and mixed-methods designs; reviews and conceptual papers were appraised narratively for the transparency of their methods. MMAT results were not used to exclude studies but to weigh the strength of evidence in the synthesis; [...] of the [...] empirical studies met at least four of the five MMAT criteria. The review protocol was not prospectively registered because PROSPERO accepts only health-related reviews; the search log, screening decisions, and appraisal sheet are available from the corresponding author.

Data collected from each included article were compiled into a spreadsheet containing the following information for each study: author(s), publication year, country of origin, research method, targeted educational domain, main findings, and their relevance to each research question (RQ1–RQ4). The data were analyzed through a hybrid deductive–inductive thematic analysis following the six phases of Braun and Clarke (2006): familiarization, initial coding, theme searching, theme review, theme definition, and reporting. The four research questions served as a priori (deductive) categories, namely prompting patterns, integration models, domain application, and challenges/success factors, while sub-themes within each category were coded inductively. The first author coded all 24 studies; the second and third authors reviewed the codebook and a subset of coded extracts, and disagreements were resolved by consensus. Coding was considered saturated when the last [...] studies coded yielded no new sub-themes. As a result, the analysis was combined with the narrative answers to each research question and served as the basis for the proposed conceptual model.

Results and Discussions

In the systematic review, 24 studies published from 2020 to 2026 were included. They originated from North America, Europe, Central Asia, East and Southeast Asia, and Oceania; no nationally indexed (Garuda/Sinta) study met all eligibility criteria. The included studies comprised peer-reviewed journal articles ($n = 14$) and conference proceedings ($n = 10$). By design, they were systematic reviews ($n = 4$), experimental or quasi-experimental studies ($n = 5$), empirical case, mixed-methods, or evaluation studies ($n = 9$), design, development, or action research studies ($n = 4$), and conceptual or position papers ($n = 2$). Twelve studies were situated in introductory or general programming courses, whereas only one addressed computer networking directly. These studies deal with generative AI, prompt engineering frameworks, that is, ways of formulating prompts to interact with AI and to optimize its output, and the dynamics of project-based learning as a major component of technology education.

To give the reader a holistic view of the literature, the main features of the 24 primary papers, i.e., the authors, publication year, region, research method, subject area, important outcomes, and related research questions (RQ1–RQ4), were systematically compiled in a matrix presented in Table 1.

Table 1. Synthesis Matrix of Included Studies (N = 24)

Author & Year	Country / Context	Methodology	Educational Domain	Key Findings & Synthesis	Focus (RQ)
Denny et al. (2024)	International	Empirical / Case Study	CS1 / Intro Programming	Introduced "Prompt Problems" shifting focus from code writing to natural-language problem articulation.	RQ1
Manley et al. (2024)	USA	Mixed Methods	CS1 & CS2 Courses	Structured prompting frameworks scaffold novice students through edge-case identification and debugging.	RQ1
Amoozadeh et al. (2024)	International	Empirical Case Study	Introductory Programming	Unstructured prompting leads students to accept AI solutions without verification or iterative refinement.	RQ1, RQ4
Elnaffar et al. (2026)	International	Systematic Review (58 studies)	Computing Education	GenAI used in 98.2% of studies; prompt engineering identified as an essential prerequisite skill.	RQ1, RQ4
Qian (2025)	International	Systematic Review	General & Technical Ed.	Synthesized prompt engineering frameworks across educational settings and scaffolding levels.	RQ1
Lee & Palmer (2025)	International	Systematic Review	Higher Education	Explored curriculum integration of prompt engineering in broad university education contexts.	RQ1
Balakayeva et al. (2026)	Kazakhstan	Quasi-Experimental / PjBL	Software Engineering (Master's)	GenAI-supported PjBL preserves cognitive engagement when students must justify technical decisions.	RQ2, RQ4
Indriati et al. (2024)	Malaysia	Action Research / PjBL	Design Education	Authentic PjBL scaffolded via structured AI dialogue enhances creative and critical thinking.	RQ2
Bull & Kharrufa (2024)	UK / International	Conceptual Framework	Software Development	Advocates purposeful integration of GenAI assistants into dev curricula rather than restrictive policies.	RQ2
Akcapinar & Sidan (2024)	International	Quantitative / Survey	Programming Education	Highlighted risks of academic integrity and direct solution copying in chatbot-assisted learning.	RQ4
Wei et al. (2022)	USA	Benchmark / Empirical	AI & Large Language Models	Demonstrated Chain-of-Thought (CoT) prompting for step-by-step reasoning in technical problem-solving.	RQ1
Zhang et al. (2025)	China	Quasi-Experimental	Programming (High School)	ChatGPT-based dialogic programming reduced off-task behavior and increased higher-order cognitive engagement and self-	RQ1, RQ4

Author & Year	Country / Context	Methodology	Educational Domain	Key Findings & Synthesis	Focus (RQ)
Becker et al. (2023)	International	Position Paper	Introductory Programming	efficacy compared with pair programming. Mapped opportunities and threats (over-reliance, academic integrity) of AI code generation for programming education.	RQ4
Raihan et al. (2025)	USA	Systematic Review	Computer Science Education	Synthesized how LLMs are used in CS education, the tasks they support, and the challenges reported.	RQ1, RQ4
Denny, Kumar & Giacaman (2023)	New Zealand / India	Empirical	CS1 / Intro Programming	Copilot solved about half of CS1 problems at first attempt and 60% of the rest after natural-language prompt reformulation.	RQ1
Sun et al. (2024)	China	Quasi-Experimental	Python Programming (Undergraduate)	Prompt-based learning produced deeper iterative coding/debugging and better interaction quality than unprompted ChatGPT use.	RQ1, RQ4
Anwar & Caesar (2025)	USA	Empirical Evaluation	Computer Networking	LLMs answered about 88–89% of 503 networking questions correctly but made elementary IP-addressing errors with plausible explanations.	RQ3, RQ4
Kazemitabaar et al. (2023)	Canada / USA	Controlled Experiment	Introductory Programming	AI code generator access improved code authoring without reducing manual code-modification performance.	RQ1, RQ4
Saeliang & Chatwattana (2025)	Thailand	Research & Development	Vocational Programming	Developed a PjBL model via GenAI to promote programming skills; experts rated its suitability at the highest level.	RQ2, RQ3
Dai et al. (2025)	China (Hong Kong)	Applied Longitudinal Study	Undergraduate PjBL	An online PjBL platform with GenAI dialogue changed the problem-solving behavior of 21 groups over one academic year.	RQ2, RQ4
Zheng et al. (2024)	China (Hong Kong)	Co-Design Study	Project-Based Learning	Students envisioned AI support across PjBL stages while preserving their own agency over projects.	RQ2
Kosar et al. (2024)	Slovenia / USA	Controlled Experiment	Object-Oriented Programming	With redesigned assignments, code defenses, and paper exams, ChatGPT use did not significantly change performance (N = 182).	RQ2, RQ4
Prather et al. (2024)	USA	Mixed Methods	Novice Programming	GenAI helped metacognitively strong novices but compounded difficulties and illusions of competence for struggling ones.	RQ4
Liffiton et al. (2023)	USA / New Zealand	Tool Deployment / Case Study	Programming Classes	The CodeHelp LLM assistant with guardrails provided help without revealing solution code.	RQ1, RQ2

Patterns of prompt-engineering utilization (RQ1). Thirteen studies addressed prompting practices, and three patterns were identified. The first pattern treats prompting as a learning task: the "Prompt Problems" exercise asks students to write natural-language prompts that make the AI generate correct code (Denny et al., 2024), and reformulating problem descriptions in natural language enabled Copilot to solve 60% of the CS1 problems it initially failed (Denny et al., 2023). The second pattern is structured or guided prompting, including frameworks that lead novices through problem description, edge-case identification, and debugging (Manley et al., 2024); prompt-based learning with structured outputs and delimiters, which produced deeper iterative coding and debugging than unprompted ChatGPT use (Sun et al., 2024); guardrailed assistants that give help without revealing solution code (Liffiton et al., 2023); human-computer dialogic programming activities (Zhang et al., 2025); and chain-of-thought prompting for step-by-step reasoning (Wei et al., 2022). The third pattern is unstructured prompting, in which students pasted the whole problem description and accepted AI-generated code without verification (Amoozadeh et al., 2024). Across the 58 studies reviewed by Elnaffar et al. (2026), prompt engineering was described as an emerging instructional objective, mainly in first-year courses, and the

authors recommended teaching prompting before or alongside programming concepts. None of the thirteen studies examined prompting in network programming tasks.

Models of generative AI–PjBL integration (RQ2). Eight studies described integration models. Master's students in software engineering maintained meaningful cognitive engagement in GenAI-supported projects when they had to justify technical decisions and deliver functional systems (Balakayeva et al., 2026). In design education, authentic PjBL supported by a structured AI dialogue framework was associated with increased creativity and critical thinking (Indriati et al., 2024). Saeliang and Chatwattana (2025) developed a PjBL model via GenAI for vocational programming students, which experts rated at the highest level of suitability. Dai et al. (2025) embedded a GenAI dialogue tool in an online PjBL platform and observed changes in the problem-solving behavior of 21 student groups over one academic year. In co-design sessions, students envisioned AI support across PjBL stages while retaining ownership of their projects (Zheng et al., 2024). At the course level, Bull and Kharrufa (2024) argued for purposeful integration of AI assistants rather than restriction, Kosar et al. (2024) permitted ChatGPT use while adding code defenses and paper-based examinations, and Liffiton et al. (2023) constrained AI help through guardrails. The models differ mainly in where AI interaction is placed: open access throughout the project, stage-specific structured dialogue, or constrained assistance.

Application to network/computing-related courses (RQ3). Evidence for RQ3 was scarce. Only one included study addressed computer networking directly: Anwar and Caesar (2025) found that GPT-3.5, GPT-4, and Claude 3 answered about 88–89% of 503 networking questions correctly but still made elementary errors, particularly in IP addressing, accompanied by plausible explanations that could create misconceptions. One further study targeted vocational programming, a course profile close to Network Programming (Saeliang & Chatwattana, 2025). The remaining empirical studies were conducted in introductory or general programming, software engineering, or design courses. No included study implemented GenAI-supported PjBL with structured prompting in a Network Programming course.

Challenges and success factors (RQ4). Thirteen studies reported challenges or success factors. The most frequently reported challenge was over-reliance on AI-generated solutions, noted in 65.52% of the 58 studies reviewed by Elnaffar et al. (2026) and observed in a CS1 case study in which students accepted AI solutions with only superficial checking (Amoozadeh et al., 2024). GenAI benefited novices with strong metacognitive skills but compounded the difficulties of struggling students and created illusions of competence (Prather et al., 2024). Direct copying of AI-generated content raised academic integrity concerns (Akçapınar & Sidan, 2024; Becker et al., 2023), and domain-specific inaccuracies were documented for networking content (Anwar & Caesar, 2025). Reported success factors were explicit instruction in prompting skills (Denny et al., 2024; Manley et al., 2024; Sun et al., 2024), the requirement to justify AI-assisted decisions in project deliverables (Balakayeva et al., 2026), structured rather than open AI dialogue (Indriati et al., 2024), assessment redesign through code defenses and supervised examinations (Kosar et al., 2024), and dialogic activities that increased cognitive engagement and self-efficacy (Zhang et al., 2025). Access to an AI code generator improved novices' code-authoring performance without reducing their manual code-modification performance (Kazemitabaar et al., 2023).

Interpretation against comparator studies. The results indicate that the pedagogical design surrounding GenAI, rather than its mere availability, determines whether it promotes or impedes learning. This interpretation is consistent with comparator studies outside the review sample and across the included studies. Becker et al. (2023) and Prather et al. (2023) anticipated that AI code generators would shift instruction from code writing toward problem specification and code evaluation, which is exactly what the prompting-as-learning-task pattern operationalizes (Denny et al., 2023, 2024). The positive effects of structured prompting (Manley et al., 2024; Sun et al., 2024; Zhang et al., 2025) and of guardrailed assistance (Liffiton et al., 2023) contrast with the unverified acceptance observed under unstructured use (Amoozadeh et al., 2024) and with the widening gap between strong and struggling novices (Prather et al., 2024). Kazemitabaar et al. (2023) and Kosar et al. (2024) show that GenAI does not necessarily harm learning outcomes, but both relied on carefully designed tasks or assessment, which supports the accountability principle in Balakayeva et al. (2026). The general opportunities and risks described by Kasneci et al. (2023), Raihan et al. (2025), and Elnaffar et al. (2026) are confirmed, while the PjBL literature (Kokotsaki et al., 2016) explains why projects need teacher scaffolding to remain productive.

Theoretical tensions across studies. Read analytically, Table 1 reveals four tensions rather than a single consensus. The first is productivity versus learning: AI support raises immediate task performance (Kazemitabaar et al., 2023; Denny et al., 2023) but can mask weak understanding (Prather et al., 2024; Amoozadeh et al., 2024). The second is openness versus constraint: some models give continuous open access to GenAI throughout the project (Dai et al., 2025), whereas others restrict it to structured dialogue or guardrails (Indriati et al., 2024; Liffiton et al., 2023); students themselves want support that preserves their agency (Zheng et al., 2024). The third is product versus process assessment: grading only the final artifact cannot reveal how AI was used, which is why code defenses (Kosar et al., 2024) and justification of decisions (Balakayeva et al.,

2026) were introduced. The fourth is domain transfer: findings from introductory programming assume that AI answers are mostly correct, whereas networking evidence shows plausible but wrong explanations and configuration errors (Anwar & Caesar, 2025; Wang et al., 2024). Existing PjBL–GenAI models (Saeliang & Chatwattana, 2025; Dai et al., 2025) address the second tension only partially and do not address the fourth. Because of these tensions, prompt engineering, scaffolding, and GenAI-supported PjBL must be combined: prompting skills without project scaffolding leave the openness problem unresolved, scaffolding without explicit prompting skills leaves students unable to verify AI output, and neither works without process-oriented assessment.

Toward a conceptual model. When one looks back over what has been discussed above and considers these matters from a Technological Pedagogical Content Knowledge (TPACK) perspective (Mishra & Koehler, 2006) and from scaffolding theory, which holds that learners can accomplish tasks beyond their independent capability when support is structured and gradually faded (Wood et al., 1976), this review proposes that prompt-engineering-supported PjBL in Network Programming will be most successful when an effective model combines three main components:

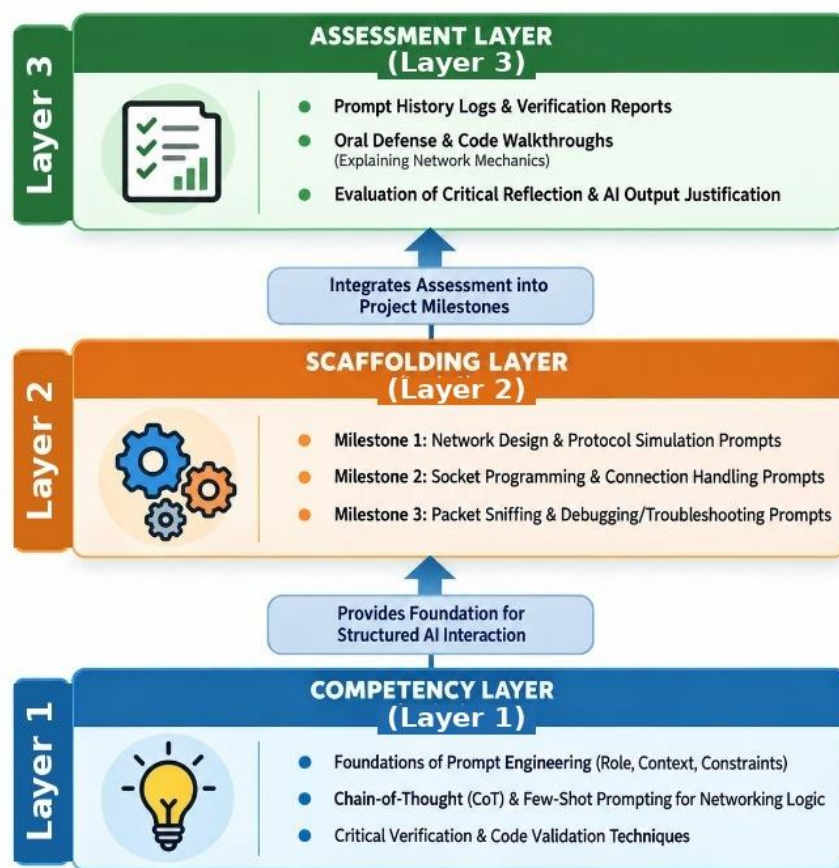


Figure 2. Three-layer conceptual model for prompt-engineering-supported Project-Based Learning (PjBL) in Network Programming courses.

(1) the *competency layer* aims at familiarizing students with core prompting strategies such as structured problem description and chain-of-thought-style reasoning prompts, which have been shown to aid large language models in performing step-by-step reasoning (Wei et al., 2022); these strategies are taught explicitly as part of the course content. The sequence principle states that prompting should be taught before or together with the technical content (Elnaffar et al., 2026); (2) the *layer of scaffolding* embeds such prompts in the different phases of network design and implementation to delimit AI assistance instead of leaving it free and limitless; stage-wise structured dialogue has been shown to enhance the depth of PjBL learning (Indriati et al., 2024); (3) the *assessment layer* requires students to explain and defend their prompting and technological choices together with the results of their projects; this aligns with the accountability principle, which is expected to sustain students' engagement when GenAI is available in PjBL (Balakayeva et al., 2026). Figure 2 above shows the proposed three-layer conceptual model, designed with the particular needs of the Network Programming course in mind:

The Competency Layer (Foundational Skills) involves explicitly teaching students basic-to-advanced prompting methods before project execution, including role assignment, context specification, constraint setting, and Chain-of-Thought (CoT) prompting (Wei et al., 2022). In network programming, students are taught to use GenAI not to generate fully written code blocks, but to explain protocol mechanisms, such as TCP/UDP handshakes, or to break down socket logic. They are also encouraged to cross-check AI-generated explanations of addressing and configuration against relevant documentation, as AI explanations may appear plausible while containing inaccuracies (Anwar & Caesar, 2025). Promoting prompt literacy from the beginning of the educational process aligns with the view that students should develop skills for interacting with AI before engaging in more complex technological tasks (Elnaffar et al., 2026).

The Scaffolding Layer (Project Milestone Integration) integrates prompt engineering directly into specific project milestones rather than treating it as an open-ended utility. During the design phase (Milestone 1), students use prompts to model network topology through hands-on engagement with the project. In the implementation phase (Milestone 2), prompts are limited to supporting socket communication and error-handling methods. During the troubleshooting phase (Milestone 3), students use debugging prompts iteratively to identify issues such as packet loss and logic errors. This support is gradually reduced across the project milestones in accordance with scaffolding theory (Wood et al., 1976). Restricting AI use to structured project phases can help manage students' cognitive load while reducing the risk of automated submissions (Indriati et al., 2024; Balakayeva et al., 2026; Liffiton et al., 2023).

The Assessment Layer (Accountability & Verification) is designed to minimize over-reliance on GenAI and promote genuine learning outcomes (Amoozadeh et al., 2024; Akçapınar & Sidan, 2024). Evaluation therefore extends beyond the successful execution of the final program. Students are required to maintain a Prompt Engineering Log documenting their prompt revisions, evaluations of AI-generated outputs, and subsequent code modifications. In addition, students participate in a short oral code walkthrough in which they explain and defend their solutions. This combination of process documentation and oral verification provides evidence of students' understanding and ensures that GenAI functions as a learning support rather than replacing students' own reasoning and programming skills (Kosar et al., 2024).

Theoretical contribution and testable propositions. The model advances theory beyond a pedagogical combination in two ways. First, it reframes prompt engineering from a technical skill into a form of scaffolding: prompts are the medium through which support is provided, constrained, and faded across project milestones. Second, it adds a domain-verification requirement that existing PjBL–GenAI models lack, because AI output in networking cannot be assumed to be correct. The model yields three falsifiable propositions for future research: (P1) students who receive explicit prompt-literacy instruction before project work will show higher debugging accuracy in network programming tasks than students given unguided GenAI access; (P2) milestone-bounded prompting will produce deeper conceptual understanding of protocols than unrestricted GenAI access, with the largest gains for students with lower prior knowledge; and (P3) assessment that includes prompt logs and code defenses will reduce uncritical acceptance of incorrect AI output compared with product-only assessment.

Limitations. This review has several limitations. First, Scopus could not be accessed directly; although searches in Google Scholar and national databases and title-by-title verification of indexing partly compensated for this, some eligible studies may have been missed. Second, screening was led by a single reviewer, and the second reviewer double-checked only a sample of the title/abstract decisions, so some selection bias cannot be excluded. Third, the included studies were heterogeneous in design and quality, which precluded a meta-analysis, and evidence from conceptual and review papers carries less weight than empirical evidence. Fourth, only English and Indonesian publications from 2020 to 2026 were considered, and the small number of included studies ($n = 24$), with only one study in computer networking, limits the generalizability of the proposed model. The model therefore has the status of a theoretically grounded proposal that requires empirical validation.

Conclusions

This review aimed to identify how prompt engineering has been used in educational settings, what models of generative AI integration into Project-Based Learning (PjBL) exist, and how far they have been applied to network programming or similar courses. Based on 24 studies identified through PRISMA-guided searches in five databases, three conclusions can be drawn. First, structured and scaffolded prompting is consistently associated with deeper engagement and better verification behavior than unstructured prompting (Denny et al., 2024; Manley et al., 2024; Sun et al., 2024). Second, GenAI–PjBL integration works best when AI interaction is planned, bounded by project stages, and coupled with accountability for technical decisions (Balakayeva et al., 2026; Indriati et al., 2024; Kosar et al., 2024). Third, no study has combined prompt engineering, scaffolding, and GenAI-supported PjBL in computer-networking education, a domain where AI output is particularly prone

to plausible errors (Anwar & Caesar, 2025). The proposed three-layer model (competency, scaffolding, and assessment) contributes to theory by reframing prompt engineering as a scaffolding medium that is taught, bounded, and faded across project milestones and by adding a domain-verification requirement, and it contributes to practice by giving Network Programming lecturers a concrete design for milestone-based, AI-supported projects. Future research should validate the model empirically, for example through quasi-experimental studies in Network Programming courses that test propositions P1–P3 by comparing structured, milestone-bounded prompting with unguided GenAI access on debugging accuracy, protocol understanding, and the quality of students' verification of AI output. Design-based research involving lecturers and students could further refine the scaffolding and assessment layers for vocational and higher-education contexts in Indonesia.

Acknowledgments

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. The authors declare no conflict of interest. Muhammad Rizki Ananda contributed to the conceptualization, literature search, data extraction, formal analysis, visualization, and writing of the original draft. Titi Sri Wahyuni contributed to supervision, methodology validation, and writing—review and editing. Resmi Darni contributed to validation of the synthesis and writing—review and editing. Fahmi Rizal contributed to supervision, conceptual model review, and writing—review and editing. All authors have read and agreed to the published version of the manuscript. The data extraction matrix (Table 1), search log, screening decisions, and MMAT appraisal sheet supporting the findings of this review are available from the corresponding author upon reasonable request.

References

- Akçapınar, G., & Sidan, E. (2024). AI chatbots in programming education: Guiding success or encouraging plagiarism. *Discover Artificial Intelligence*, 4(1), 87. <https://doi.org/10.1007/s44163-024-00203-7>
- Amoozadeh, M., Nam, D., Prol, D., Alfageeh, A., Prather, J., Hilton, M., Srinivasa Ragavan, S., & Alipour, A. (2024). Student-AI interaction: A case study of CS1 students. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research* (pp. 1–13). ACM. <https://doi.org/10.1145/3699538.3699567>
- Anwar, M., & Caesar, M. (2025). Understanding misunderstandings: Evaluating LLMs on networking questions. *ACM SIGCOMM Computer Communication Review*, 54(4), 14–24. <https://doi.org/10.1145/3717512.3717515>
- Balakayeva, G., Zeinolla, S., Akzhalova, A., & Kalmenova, G. (2026). Project-based learning in the age of generative AI: Developing student autonomy and creative problem-solving in software engineering education. *International Journal of Learning, Teaching and Educational Research*, 25(6), 372–392. <https://ijlter.org/index.php/ijlter/article/view/17119>
- Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming is hard – or at least it used to be: Educational opportunities and challenges of AI code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 500–506). ACM. <https://doi.org/10.1145/3545945.3569759>
- Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), 77–101. <https://doi.org/10.1191/1478088706qp063oa>
- Bull, C., & Kharrufa, A. (2024). Generative artificial intelligence assistants in software development education: A vision for integrating generative artificial intelligence into educational practice, not instinctively defending against it. *IEEE Software*, 41(2), 52–59. <https://doi.org/10.1109/MS.2023.3300574>
- Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37–46. <https://doi.org/10.1177/001316446002000104>
- Dai, Y., Xiao, J.-Y., Huang, Y., Zhai, X., Wai, F.-C., & Zhang, M. (2025). How generative AI enables an online project-based learning platform: An applied study of learning behavior analysis in undergraduate students. *Applied Sciences*, 15(5), 2369. <https://doi.org/10.3390/app15052369>
- Denny, P., Kumar, V., & Giacaman, N. (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* (pp. 1136–1142). ACM. <https://doi.org/10.1145/3545945.3569823>
- Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., & Reeves, B. N. (2024). Prompt problems: A new programming exercise for the generative AI era. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education*, Vol. 1 (pp. 296–302). ACM. <https://doi.org/10.1145/3626252.3630909>

- Elnaffar, S., Rashidi, F., & Abualkashik, A. Z. (2026). Teaching with AI: A systematic review of chatbots, generative tools, and tutoring systems in programming education. *International Journal of Learning, Teaching and Educational Research*, 25(1), 1–28. <https://doi.org/10.26803/ijlter.25.1.1>
- Grant, M. J., & Booth, A. (2009). A typology of reviews: An analysis of 14 review types and associated methodologies. *Health Information & Libraries Journal*, 26(2), 91–108. <https://doi.org/10.1111/j.1471-1842.2009.00848.x>
- Hong, Q. N., Fàbregues, S., Bartlett, G., Boardman, F., Cargo, M., Dagenais, P., Gagnon, M.-P., Griffiths, F., Nicolau, B., O'Cathain, A., Rousseau, M.-C., Vedel, I., & Pluye, P. (2018). The Mixed Methods Appraisal Tool (MMAT) version 2018 for information professionals and researchers. *Education for Information*, 34(4), 285–291. <https://doi.org/10.3233/EFI-180221>
- Indriati, L., Mai, N., & Yeen-Ju, H. T. (2024). Enhancing authentic assessment in large-class design education through authentic project-based learning. *International Journal of Learning, Teaching and Educational Research*, 23(9), 432–452. <https://doi.org/10.26803/ijlter.23.9.22>
- Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., ... Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, 102274. <https://doi.org/10.1016/j.lindif.2023.102274>
- Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). Studying the effect of AI code generators on supporting novice learners in introductory programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (pp. 1–23). ACM. <https://doi.org/10.1145/3544548.3580919>
- Kitchenham, B., & Charters, S. (2007). *Guidelines for performing systematic literature reviews in software engineering* (Version 2.3). University of Durham.
- Kokotsaki, D., Menzies, V., & Wiggins, A. (2016). Project-based learning: A review of the literature. *Improving Schools*, 19(3), 267–277. <https://doi.org/10.1177/1365480216659733>
- Kosar, T., Ostojić, D., Liu, Y. D., & Mernik, M. (2024). Computer science education in ChatGPT era: Experiences from an experiment in a programming course for novice programmers. *Mathematics*, 12(5), 629. <https://doi.org/10.3390/math12050629>
- Lee, A., & Palmer, M. (2025). Prompt engineering in higher education: A systematic review to help inform curricula. *International Journal of Educational Technology in Higher Education*, 22(1). <https://doi.org/10.1186/s41239-025-00503-7>
- Liffiton, M., Sheese, B. E., Savelka, J., & Denny, P. (2023). CodeHelp: Using large language models with guardrails for scalable support in programming classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research* (pp. 1–11). ACM. <https://doi.org/10.1145/3631802.3631830>
- Manley, E. D., Urness, T., Migunov, A., & Reza, M. A. (2024). Examining student use of AI in CS1 and CS2. *Journal of Computing Sciences in Colleges*, 39(6), 41–51.
- McHugh, M. L. (2012). Interrater reliability: The kappa statistic. *Biochemia Medica*, 22(3), 276–282. <https://doi.org/10.11613/BM.2012.031>
- Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record*, 108(6), 1017–1054. <https://doi.org/10.1111/j.1467-9620.2006.00684.x>
- Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., & Moher, D. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ*, 372, n71. <https://doi.org/10.1136/bmj.n71>
- Prather, J., Denny, P., Leinonen, J., Becker, B. A., Albluwi, I., Craig, M., Keuning, H., Kiesler, N., Kohn, T., Luxton-Reilly, A., MacNeil, S., Petersen, A., Pettit, R., Reeves, B. N., & Savelka, J. (2023). The robots are here: Navigating the generative AI revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education* (pp. 108–159). ACM. <https://doi.org/10.1145/3623762.3633499>
- Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., Kimmel, B., Wright, J., & Briggs, B. (2024). The widening gap: The benefits and harms of generative AI for novice programmers. In *Proceedings of the 2024 ACM Conference on International Computing Education Research – Volume 1* (pp. 469–486). ACM. <https://doi.org/10.1145/3632620.3671116>
- Prvan, M., & Ožegović, J. (2020). Methods in teaching computer networks: A literature review. *ACM Transactions on Computing Education*, 20(3), 1–35. <https://doi.org/10.1145/3394963>
- Qian, Y. (2025). Prompt engineering in education: A systematic review of approaches and educational applications. *Journal of Educational Computing Research*. <https://doi.org/10.1177/07356331251365189>

-
- Raihan, N., Siddiq, M. L., Santos, J. C. S., & Zampieri, M. (2025). Large language models in computer science education: A systematic literature review. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (pp. 938–944). ACM. <https://doi.org/10.1145/3641554.3701863>
- Saeliang, S., & Chatwattana, P. (2025). The project-based learning model via generative artificial intelligence to promote programming skills for vocational students. *International Education Studies*, 18(3), 1. <https://doi.org/10.5539/ies.v18n3p1>
- Sun, D., Boudouaia, A., Yang, J., & Xu, J. (2024). Investigating students' programming behaviors, interaction qualities and perceptions through prompt-based learning in ChatGPT. *Humanities and Social Sciences Communications*, 11, 1447. <https://doi.org/10.1057/s41599-024-03991-6>
- Wang, C., Scazzariello, M., Farshin, A., Ferlin, S., Kostić, D., & Chiesa, M. (2024). NetConfEval: Can LLMs facilitate network configuration? *Proceedings of the ACM on Networking*, 2(CoNEXT2), 1–25. <https://doi.org/10.1145/3656296>
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35, 24824–24837.
- Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17(2), 89–100. <https://doi.org/10.1111/j.1469-7610.1976.tb00381.x>
- Zhang, L., Jiang, Q., Xiong, W., & Zhao, W. (2025). Effects of ChatGPT-based human–computer dialogic interaction programming activities on student engagement. *Journal of Educational Computing Research*, 63(4), 988–1023. <https://doi.org/10.1177/07356331251333874>
- Zheng, C., Yuan, K., Guo, B., Hadi Mogavi, R., Peng, Z., Ma, S., & Ma, X. (2024). Charting the future of AI in project-based learning: A co-design exploration with students. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (pp. 1–19). ACM. <https://doi.org/10.1145/3613904.3642807>