OPEN ACCESS

Saudi Journal of Engineering and Technology

Abbreviated Key Title: Saudi J Eng Technol
ISSN 2415-6272 (Print) |ISSN 2415-6264 (Online)
Scholars Middle East Publishers, Dubai, United Arab Emirates
Journal homepage: https://saudijournals.com

Original Research Article

Evaluating Vibe Coding in Programming Education: Evidence from a Quasi-Experimental Study in Vocational IT Training

Ho Thi Thanh Nga[1]*, Nguyen Thi Phuong Thuy[1]

[1]Faculty of Information Technology, Ho Chi Minh City College of Economics, Ho Chi Minh City, Vietnam

DOI: https://doi.org/10.36348/sjet.2026.v11i07.002 | **Received:** 28.05.2026 | **Accepted:** 16.07.2026 | **Published:** 17.07.2026

***Corresponding author:** Ho Thi Thanh Nga
Faculty of Information Technology, Ho Chi Minh City College of Economics, Ho Chi Minh City, Vietnam

Abstract

Programming courses at the college level in Vietnam continue to suffer from high failure rates, early drop-out, and incomplete student projects. This paper presents Vibe Coding, a four-stage pedagogical model integrating AI coding assistants into Fundamentals of Programming and Basic Web Design. The model was evaluated using a hybrid quasi-experimental design combining a three-year historical trend (2023–2025, n = 735) with a concurrent control group in the 2025–2026 academic year (232 traditional vs. 150 Vibe Coding students). The Vibe-Coding group significantly outperformed the traditional group on the proportion scoring ≥7.0/10 (53.3% vs. 41.8%; z = 2.21, p = .027) and on course pass rate (83.3% vs. 67.2%; z = 3.48, p < .001), and also exceeded the three-year historical baseline (39.7%; z = 3.08, p = .002), while the traditional group did not differ from that baseline. However, this overall effect was driven almost entirely by Fundamentals of Programming (z = 3.13, p = .002); in Basic Web Design, Vibe Coding did not outperform traditional instruction on the merit measure (z = -0.24, p = .811), with only a non-significant favourable trend on pass rate (z = 1.66, p = .097). Class-level variation was substantial, including one traditional section that improved by 40 percentage points without the intervention. Despite this course-level heterogeneity, over 85% of surveyed students reported greater engagement and confidence in debugging. The findings indicate that Vibe Coding can meaningfully improve programming outcomes, but its benefits are course-dependent and require further validation through randomized, multi-institutional studies.

Keywords: Vibe Coding; Artificial Intelligence in Education; Programming Pedagogy; Prompt Engineering; Computer Science Education; Large Language Models; Vocational and Technical Education; Trends and Development Potential.

I. INTRODUCTION

Large language models (LLMs), including GitHub Copilot, ChatGPT, Claude, and Gemini, have transformed software development by improving coding productivity (Peng *et al.,* 2023). In early 2025, the term *vibe coding* was introduced to describe a conversational programming workflow in which developers use AI to generate, review, and refine code (Karpathy, 2025). Although widely adopted in industry, this approach has rarely been incorporated into programming education, particularly in vocational colleges in developing countries such as Viet Nam.

A. Background and Motivation

Between 2023 and 2025, instructors at the Faculty of Information Technology, Ho Chi Minh City College of Economics, observed persistent difficulties in the Fundamentals of Programming and Basic Web Design courses. Students struggled with C/C++ programming, algorithmic thinking, and responsive web development, resulting in failure rates of about 21% and many incomplete course projects. These challenges are consistent with previous studies showing that AI code-generation tools are reshaping introductory programming education (Becker *et al.,* 2023; Denny *et al.,* 2024a).

B. Problem Statement

Although students already use AI coding tools in their learning, the faculty has not established a formal framework for integrating them into programming courses. Many instructors remain concerned that excessive reliance on AI may weaken students' understanding of programming concepts. At the same time, traditional teaching methods no longer reflect AI-

assisted software development in industry. A structured instructional approach is therefore needed to help students use AI to support problem solving while maintaining conceptual understanding.

C. Proposed Solution

This paper proposes Vibe Coding, a four-stage instructional model that integrates AI assistants into programming courses through structured learning activities. Students progressively develop prompt-engineering, code evaluation, and debugging skills using a three-level task framework and a dedicated assessment rubric. The proposed model is consistent with recent studies advocating AI-assisted programming instruction that emphasizes active engagement with AI-generated code (Denny *et al.,* 2023, 2024b).

D. Contributions

This study makes four contributions. First, it introduces a structured Vibe-Coding model for programming education in Vietnamese vocational colleges. Second, it evaluates the model using both historical data and a concurrent control group, extending previous descriptive studies (Kazemitabaar *et al.,* 2023; Prather *et al.,* 2023). Third, it assesses prompt-engineering skills through a five-criterion grading rubric, addressing recent recommendations for AI-related assessment (Denny *et al.,* 2023). Finally, it discusses the model's applicability to similar educational contexts.

E. Paper Organisation

The remainder of this paper is organized as follows. Section II reviews related work. Section III presents the proposed methodology. Section IV reports and discusses the results. Section V discusses future directions, and Section VI concludes the paper.

II. RELATED WORK

A. AI Code Generation and Introductory Programming Education

Recent studies show that AI code-generation tools can outperform many CS1 and CS2 students on programming tasks, prompting calls to redesign assessment, strengthen academic integrity, and adapt computing curricula to the AI era (Becker *et al.,* 2023; Denny *et al.,* 2024a; Finnie-Ansley *et al.,* 2022, 2023).

B. Usability, Interaction Patterns, and Prompt Engineering with AI Coding Assistants

Another line of research examines how novice programmers interact with AI coding assistants. Prather et al. conducted an observational study of novice programmers using GitHub Copilot on CS1 assignments and identified distinct interaction patterns, including behaviours they termed "shepherding" and "drifting," underscoring that novices do not use AI assistants the same way experienced developers do (Prather *et al.,* 2023). Kazemitabaar et al. conducted a controlled experiment with 69 novice learners (ages 10–17) and found that access to an AI code generator did not harm learning outcomes and could improve retention for learners with prior block-based programming experience, provided the tool was embedded within a structured learning environment rather than used unsupervised (Kazemitabaar *et al.,* 2023). Denny et al. proposed "Conversing with Copilot," an exercise format requiring students to solve problems purely through natural-language prompts, an early operationalisation of prompt engineering as a gradable skill (Denny *et al.,* 2023); a follow-up design, "Prompt Problems," extended this idea into a reusable exercise type for the generative-AI era (Denny *et al.,* 2024b). More recently, Prather et al. described a "widening gap" between students who use generative AI critically and those who become passively dependent on it, reinforcing the need for pedagogical structures — such as graded prompt-engineering rubrics — that actively cultivate critical use (Prather *et al.,* 2024).

C. Curriculum and Assessment Design for AI-Integrated Programming Education

A smaller body of work addresses how AI code-generation capability itself should inform curriculum and benchmarking decisions. Xu et al. conducted a systematic evaluation of large language models of code, comparing proprietary and open-source models across programming languages and documenting substantial variation in code-generation reliability (Xu *et al.,* 2022) — a finding with direct implications for how much unsupervised trust an instructional model should place in AI-generated output. At the level of national and sectoral policy, Vietnam's Ministry of Education and Training has articulated a strategy for digital transformation in education that explicitly encourages the integration of AI-based tools into vocational and higher-education curricula, providing a policy context for institution-level initiatives such as the one reported here (Ministry of Education and Training, Vietnam, 2024).

D. Research Gap

Taken together, prior studies establish that AI code generators can match or exceed novice performance on standard exercises (Finnie-Ansley *et al.,* 2022, 2023); that novices interact with these tools in ways distinct from experienced developers (Prather *et al.,* 2023); that unsupervised use carries a risk of over-reliance (Kazemitabaar *et al.,* 2023; Prather *et al.,* 2024); and that prompt engineering can be operationalised as a gradable exercise format (Denny *et al.,* 2023, 2024b). However, most of this literature originates from single-course interventions in CS1 contexts at research universities in higher-income countries, and few studies report a multi-course, semester-long deployment that combines a documented historical trend with a concurrent, same-instructor control comparison embedded across an entire vocational-college curriculum. This study addresses that gap by evaluating a structured, four-stage AI-integration model across two core programming courses at a Vietnamese vocational college, combining three years of archival outcome records with a concurrent controlled

comparison in the final year, using a transparent statistical comparison of outcomes.

III. MATERIALS AND METHODS

A. Research Design and Setting

The study employed a hybrid quasi-experimental research design, combining a three-year historical trend analysis with a comparison across parallel class sections in the final year. The Vibe-Coding model was evaluated in the 2025–2026 academic year at the Faculty of Information Technology, covering two courses: Fundamentals of Programming (Semester 1, August–December 2025) and Basic Web Design (Semester 2, January–May 2026). For each course, the 2025–2026 class sections were divided into those that adopted the Vibe Coding model and those that continued conventional (non-AI-integrated) instruction under the same syllabus: in Fundamentals of Programming, three sections (010100096302, 010100096306, 010100096307; n = 98) used Vibe Coding while four sections (010100096301, 010100096303, 010100096304, 010100096305; n = 148) remained conventional; in Basic Web Design, two sections (010100100002, 010100100003; n = 52) used Vibe Coding while three sections (010100100001, 010100147701, 010100147702; n = 84) remained conventional (combined n = 232 conventional, 150 Vibe Coding). Assignment of sections followed normal class-scheduling and staffing procedures rather than random allocation at the individual level, so this remains a non-randomised, matched-section comparison rather than a randomised controlled trial (Section III.F). To provide additional context for interpreting this comparison, outcomes for both 2025–2026 groups were also compared against archival gradebook records from the same two courses, taught under the same curriculum in the two preceding academic years: 2023–2024 (n = 329) and 2024–2025 (n = 406), yielding a pooled historical baseline of 735 students. Comparing the 2025–2026 groups against both a concurrent conventional group and this historical baseline was intended to address two of the principal threats to internal validity identified in an earlier draft of this study: the absence of a same-year comparison group, and the possibility that year-specific factors could account for observed changes. It is important to note explicitly that allocation to Vibe Coding or conventional instruction occurred at the level of entire class sections rather than individual students, following normal scheduling and staffing practice rather than randomisation; this section-level, non-random allocation is a material threat to internal validity, since pre-existing differences between sections could contribute to observed differences independent of the intervention (Section III.F).

B. The Vibe-Coding Instructional Model

The intervention was implemented over a 16-week semester through four sequential stages (Figure 1).

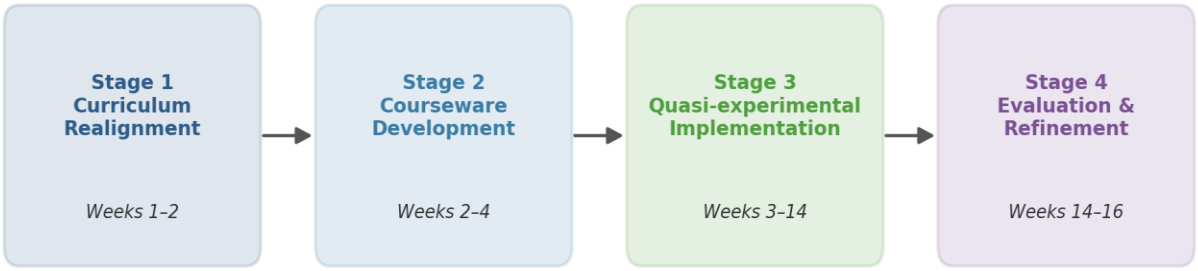

Figure 1: The Four-Stage Vibe Coding Instructional Model

- **Stage 1 – Curriculum Realignment (Weeks 1–2):** Course syllabi and lesson plans were revised to integrate AI-literacy outcomes and guidance on AI-assisted coding, debugging, and optimization.
- **Stage 2 – Courseware Development (Weeks 2–4):** AI usage guides and a three-level task hierarchy were developed, progressing from AI-generated code analysis, to hybrid AI-assisted coding, and finally independent coding with AI used only for review and optimization. A 10-point rubric based on five criteria was also created to assess prompt-engineering skills (Table 1).
- **Stage 3 – Implementation (Weeks 3–14):** The Vibe-Coding model was implemented in the intervention sections (n = 150), while the remaining sections (n = 232) continued with conventional instruction under the same syllabus. During weekly supervised lab sessions, students practiced prompt engineering, evaluated and refined AI-generated code, and documented their AI-assisted workflows.
- **Stage 4 – Evaluation and Refinement (Weeks 14–16):** The intervention was evaluated using academic performance, course completion rates, student surveys, instructor observations, and expert review. Results were compared with those of the concurrent control sections and historical cohorts.

C. Instruments and Tools

Students used GitHub Copilot, ChatGPT, and Claude in Visual Studio Code for C++ and web development. Evidence collected included revised syllabi and lesson plans, AI-supported practice exercises, representative student projects, grade records, and student questionnaires.

D. Assessment Rubric

Each course used a 10-point prompt-engineering rubric (Table 1) covering prompt quality, code analysis, refinement, task completion, and presentation. Integrating these criteria into course assessment encouraged students to demonstrate effective use of AI rather than simply copying generated code, extending previous AI-assisted learning activities (Denny *et al.,* 2023, 2024b).

Table 1: Five-criterion rubric for assessing prompt-engineering and AI-collaboration skill (10-point scale)

Assessment criterion	Points
Clarity and contextual precision of the prompt	2
Analysis and comprehension of AI-generated code	2
Revision and optimisation of the generated solution	2
Correctness / compliance with task requirements	2
Presentation and reporting of the finished product	2
Total	**10**

E. Statistical Analysis

Student achievement data were obtained from institutional gradebooks for the 2025–2026 academic year (Vibe Coding: n = 150; conventional: n = 232) and compared with historical data from 2023–2024 (n = 329) and 2024–2025 (n = 406). Complete datasets enabled direct two-proportion z-tests on the proportions of students scoring ≥7.0 and passing (≥5.0). Three comparisons were conducted: (1) Vibe Coding versus the concurrent control, (2) the concurrent control versus the historical baseline, and (3) Vibe Coding versus the historical baseline. Course- and class-specific results are presented in Tables 3 and 4.

IV. RESULTS AND DISCUSSION

A. Academic Performance

Academic outcomes are summarized in Table 1 and Figure 2, with course-specific results in Tables 3 and 4. During 2023–2025, all classes followed conventional instruction, yielding a historical benchmark of 39.7% of students scoring ≥7.0/10 (n = 735) and a pass rate (≥5.0) of 68.4%, with no evidence of an upward performance trend.

In 2025–2026, the conventional sections (n = 232) achieved a ≥7.0/10 rate of 41.8%, with no significant difference from the historical benchmark (z = 0.56, p = 0.573). The Vibe-Coding sections (n = 150), however, achieved 53.3%, significantly exceeding both the concurrent conventional sections (+11.5 percentage points; z = 2.21, p = 0.027) and the historical benchmark (+13.6 percentage points; z = 3.08, p = 0.002).

Pass rates showed the same trend: 68.4% in the historical cohorts, 67.2% in the conventional sections, and 83.3% in the Vibe-Coding sections (+16.1 percentage points; z = 3.48, p < 0.001). Because the conventional sections performed similarly to the historical cohorts, the observed improvement is more likely associated with the Vibe-Coding intervention than with differences between student cohorts. Course-level results, however, varied between the two subjects.

Table 2: Academic outcomes across three consecutive academic years, combining Fundamentals of Programming and Basic Web Design. The 2025–2026 cohort is split into conventional (n = 232) and Vibe Coding (n = 150) groups; group sizes are unequal because they reflect the actual number of class sections that adopted each mode of instruction rather than a balanced allocation. "n/a" in the "Total Students" column for the three test-statistic rows indicates that a sample size is not applicable to a row reporting a test statistic rather than a headcount.

Academic Year / Group	Total Students	Scoring ≥7.0, n (%)	Passing ≥5.0, n (%)	Failing <5.0, n (%)
2023–2024 (conventional, both courses)	329	126 (38.3%)	232 (70.5%)	97 (29.5%)
2024–2025 (conventional, both courses)	406	166 (40.9%)	271 (66.7%)	135 (33.3%)
Pooled historical baseline (2023–2025)	735	292 (39.7%)	503 (68.4%)	232 (31.6%)
2025–2026 — Control (conventional, both courses)	232	97 (41.8%)	156 (67.2%)	76 (32.8%)
2025–2026 — Vibe Coding (both courses)	150	80 (53.3%)	125 (83.3%)	25 (16.7%)
Concurrent z-test: Vibe Coding vs. Control (2025–2026)	n/a	+11.5 pp; z = 2.21, p = 0.027	+16.1 pp; z = 3.48, p < 0.001	−16.1 pp; z = 3.48, p < 0.001
Historical z-test: Control (2025–26) vs. pooled baseline	n/a	+2.1 pp; z = 0.56, p = 0.573 (not significant)	−1.2 pp; z = −0.34, p = 0.734 (not significant)	+1.2 pp; z = −0.34, p = 0.734 (not significant)
Historical z-test: Vibe Coding (2025–26) vs. pooled baseline	n/a	+13.6 pp; z = 3.08, p = 0.002	+14.9 pp; z = 3.66, p < 0.001	−14.9 pp; z = 3.66, p <0 .001

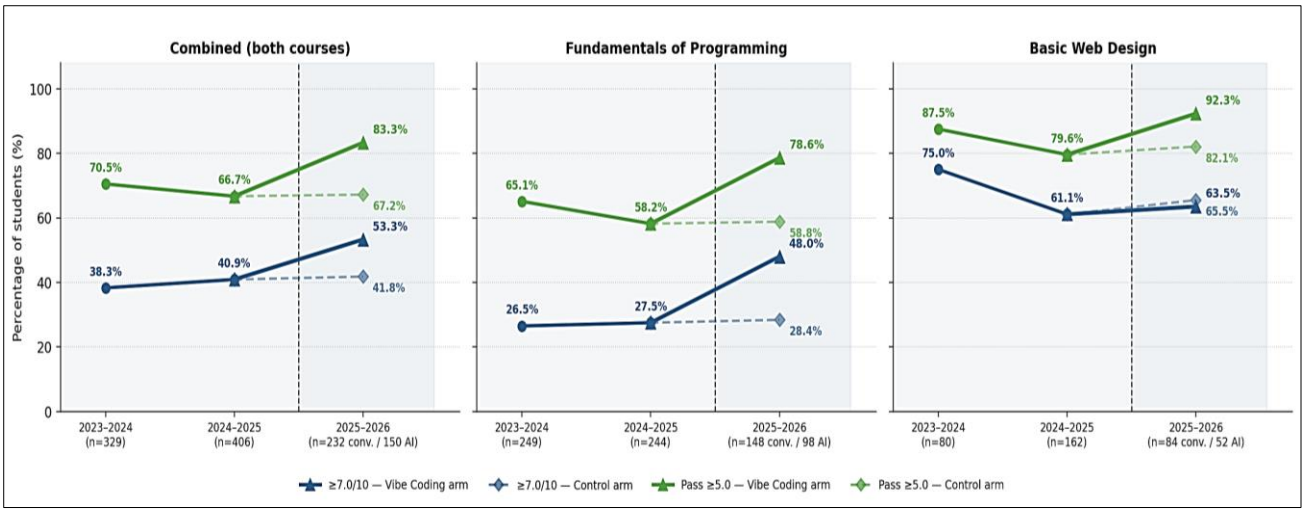

Figure 2: Trend in academic outcomes across three consecutive academic years, forking into conventional and Vibe-Coding groups in 2025–2026

Tables 3 and 4 show that the combined result above is driven almost entirely by one course, and that the two courses in fact point in different directions on the merit-level (≥7.0/10) measure. In Fundamentals of Programming (Table 4), three of the seven 2025–2026 class sections (010100096302, 010100096306, and 010100096307) adopted Vibe Coding (n = 98 combined), while the remaining four sections (010100096301, 010100096303, 010100096304, and 010100096305; n = 148 combined) continued under conventional instruction. The Vibe-Coding classes achieved a markedly higher ≥7.0/10 rate than the concurrent conventional classes (47.96% vs. 28.38%; z = 3.13, p = 0.002) and a higher pass rate (78.57% vs. 58.78%; z = 3.22, p = 0.001); the conventional classes, in turn, did not differ significantly from the two-year historical baseline for this course (28.38% vs. 26.98%; z = 0.34, p = 0.737). In Basic Web Design (Table 3), the two Vibe Coding sections (010100100002 and 010100100003; n = 52 combined) did not outperform the three conventional sections (010100100001, 010100147701, and 010100147702; n = 84 combined) on the ≥7.0/10 measure — if anything, the Vibe Coding group scored numerically slightly lower (63.46% vs. 65.48%; z = −0.24, p = 0.811, not statistically significant, direction reversed relative to expectation). On the pass rate, the Vibe-Coding group scored numerically higher (92.31% vs. 82.14%) but the difference remained short of conventional statistical significance (z = 1.66, p = 0.097). Neither the Vibe Coding nor the conventional 2025–2026 sections in Basic Web Design differed significantly from that course's own historical baseline on the ≥7.0/10 measure (65.7%; z = −0.31, p = 0.758 and z = −0.04, p = 0.970, respectively). The combined result reported above is therefore almost entirely attributable to Fundamentals of Programming; for Basic Web Design specifically, the data provide no statistically significant evidence of a Vibe-Coding effect on the merit-level measure, and only a non-significant trend on the pass rate. This course-level heterogeneity is discussed further in Section IV.H.

Table 3: Basic Web Design: class-level academic outcomes by class identifier and academic year, 2023–2024 to 2025–2026. "Coursework" is the continuous-assessment average (Đ. TBTK); "Final" is the final-examination score; "Overall" is the weighted final course score used to determine ≥7.0/10 and pass/fail status

Class — Academic Year	Group	N	Coursework	Final	Overall	≥7.0	Fail
010100100001 — 2023–2024	No AI	40	7.62	5.86	6.02	60.0%	22.5%
010100100002 — 2023–2024	No AI	40	7.78	8.25	8.07	90.0%	2.5%
010100100001 — 2024–2025	No AI	30	6.67	6.72	6.66	70.0%	16.7%
010100100002 — 2024–2025	No AI	29	7.38	6.14	6.20	58.6%	24.1%
010100100003 — 2024–2025	No AI	30	6.76	6.88	6.83	70.0%	13.3%
010100100004 — 2024–2025	No AI	30	6.84	6.81	6.76	70.0%	20.0%
010100147701 — 2024–2025	No AI	24	5.71	4.88	5.21	33.3%	20.8%
010100147703 — 2024–2025	No AI	19	6.17	5.34	5.42	57.9%	31.6%
010100100001 — 2025–2026	No AI	32	6.76	6.25	6.28	56.2%	15.6%
010100100002 — 2025–2026	AI	29	7.56	6.95	7.10	62.1%	6.9%
010100147702 — 2025–2026	No AI	22	5.81	6.27	5.97	68.2%	22.7%
010100100003 — 2025–2026	AI	23	7.52	6.71	6.88	65.2%	8.7%
010100147701 — 2025–2026	No AI	30	7.42	7.19	7.03	73.3%	16.7%

Table 4: Fundamentals of Programming: class-level academic outcomes by class identifier and academic year, 2023–2024 to 2025–2026. Column definitions as in Table 3

Class — Academic Year	Group	N	Coursework	Final	Overall	≥7.0	Fail
010100096301 — 2023–2024	No AI	42	8.28	4.92	6.13	31.0%	21.4%
010100096302 — 2023–2024	No AI	41	6.46	3.59	4.33	14.6%	46.3%
010100096303 — 2023–2024	No AI	41	7.39	4.71	5.63	39.0%	24.4%
010100096304 — 2023–2024	No AI	42	6.45	3.28	4.32	9.5%	54.8%
010100096305 — 2023–2024	No AI	42	6.24	4.03	4.71	31.0%	45.2%
010100096306 — 2023–2024	No AI	41	7.82	5.11	6.03	34.1%	17.1%
010100096301 — 2024–2025	No AI	43	7.75	4.84	5.76	37.2%	32.6%
010100096302 — 2024–2025	No AI	42	7.79	4.96	5.90	42.9%	28.6%
010100096303 — 2024–2025	No AI	41	7.85	4.02	5.05	34.1%	34.1%
010100096304 — 2024–2025	No AI	40	6.14	3.24	3.75	10.0%	55.0%
010100096305 — 2024–2025	No AI	41	6.69	4.34	5.06	26.8%	36.6%
010100096306 — 2024–2025	No AI	37	6.68	2.17	2.85	10.8%	67.6%
010100096301 — 2025–2026	No AI	38	6.44	4.11	4.75	21.1%	42.1%
010100096303 — 2025–2026	No AI	38	7.69	4.81	5.73	36.8%	31.6%
010100096304 — 2025–2026	No AI	38	5.76	3.31	3.86	15.8%	55.3%
010100096307 — 2025–2026	AI	39	7.11	4.17	5.05	30.8%	25.6%
010100096302 — 2025–2026	AI	33	8.32	5.76	6.59	63.6%	15.2%
010100096305 — 2025–2026	No AI	34	7.17	4.57	5.41	41.2%	35.3%
010100096306 — 2025–2026	AI	26	7.58	5.23	5.95	53.8%	23.1%

B. Statistical Significance of Observed Differences

The primary comparison — Vibe Coding versus conventional instruction within 2025–2026, combining both courses — corresponds to a two-proportion z-statistic of 2.21 ($p = 0.027$, two-tailed) for the ≥7.0/10 measure (80/150, 53.3% vs. 97/232, 41.8%) and $z = 3.48$ ($p < 0.001$, two-tailed) for the course pass rate (125/150, 83.3% vs. 156/232, 67.2%). Both are statistically significant at conventional thresholds, though the ≥7.0/10 result is more modest than the pass-rate result and should not be overstated. As an additional validity check, the conventional group was also compared with the pooled 2023–2025 historical baseline: the two proportions differ by 2.1 percentage points (41.8% vs. 39.7%), a difference well within the range expected from ordinary variation between cohorts and not statistically significant ($z = 0.56$, $p = 0.573$). By contrast, the Vibe-Coding group differed from the same historical baseline by 13.6 percentage points (53.3% vs. 39.7%), a difference unlikely to arise by chance ($z = 3.08$, $p = 0.002$). This aggregate, combined-course pattern is consistent with a genuine effect of Vibe Coding. However, decomposing this pattern by course (Section IV.A) shows that it is driven almost entirely by Fundamentals of Programming ($z = 3.13$, $p = 0.002$ for ≥7.0/10, versus the concurrent conventional group), while the corresponding comparison in Basic Web Design is not statistically significant and numerically points in the opposite direction ($z = -0.24$, $p = 0.811$). Readers should therefore treat the combined-course result as describing Fundamentals of Programming much more than Basic Web Design, and should not interpret it as evidence of a uniform effect across both courses. This remains a non-randomised, matched-section comparison rather than a randomised controlled trial — class sections, not individual students, were assigned to the Vibe Coding or conventional condition according to normal scheduling and staffing practice — and potential selection bias at the class level cannot be fully excluded (Section III.F).

C. Learning Outcomes: Coursework, Final Examination, and Overall Scores

Table 5 decomposes the overall course score into its two constituent components — the continuous-assessment (coursework) average and the final-examination score — for the historical baseline and the two 2025–2026 groups, by course and combined. In Fundamentals of Programming, mean coursework scores were similar across the historical baseline (7.14), the 2025–2026 conventional group (6.75), and the Vibe-Coding group (7.64), but the mean final-examination score diverged more clearly: 4.12 historically, 4.19 for the conventional group, and 4.99 for the Vibe-Coding group, tracking the gain in the overall score (4.99 historically, 4.92 for the conventional group, vs. 5.81 for Vibe Coding). This pattern suggests that the Vibe-Coding intervention in this course was associated with improved performance specifically on the final examination, rather than merely with higher continuous-assessment grades. In Basic Web Design, the pattern was different from Fundamentals of Programming in an informative way: mean coursework was somewhat higher in the Vibe Coding group (7.54) than the historical baseline (6.99) or the conventional group (6.75), and the mean final-examination score was likewise higher for Vibe Coding (6.84) than for the conventional group (6.59) or the historical baseline (6.50); however, because the conventional group's own final-examination score (6.59) was already close to, or slightly above, the

historical baseline, and because the ≥7.0/10 merit-level comparison in Section IV.A showed no significant difference for this course, the higher mean scores for the Vibe Coding group in Basic Web Design should be interpreted as a modest, non-significant numerical trend rather than a confirmed effect.

Table 5: Mean coursework (continuous-assessment), final-examination, and overall scores (10-point scale), by course and group

Course	Group	N	Mean Coursework	Mean Final Exam	Mean Overall
Basic Web Design	Historical (23–25)	242	6.99	6.50	6.52
	2025–26 Control	84	6.75	6.59	6.47
	2025–26 AI	52	7.54	6.84	7.00
Fundamentals of Programming	Historical (23–25)	493	7.14	4.12	4.99
	2025–26 Control	148	6.75	4.19	4.92
	2025–26 AI	98	7.64	4.99	5.81
Combined (both courses)	Historical (23–25)	735	7.09	4.91	5.49
	2025–26 Control	232	6.75	5.06	5.48
	2025–26 AI	150	7.61	5.63	6.22

D. Pass–Fail Analysis

Considering the three outcome thresholds together — the merit threshold (≥7.0/10), the pass threshold (≥5.0/10), and the corresponding failure rate (<5.0/10) — the clearest and most statistically consistent gains associated with Vibe Coding were observed in Fundamentals of Programming: the merit rate rose from 28.4% to 48.0% and the failure rate fell from 41.2% to 21.4%, while the conventional sections of the same course remained close to their historical failure rate of 38.3% (41.2% observed). In Basic Web Design, the failure rate fell numerically from 17.9% (conventional, 2025–2026) to 7.7% (Vibe Coding), a pattern that, unlike the ≥7.0/10 measure, is at least directionally favourable and corresponds to the marginal, non-significant pass-rate result reported in Section IV.B ($z = 1.66$, $p = 0.097$); this should be read as a trend worth investigating further rather than as a confirmed effect, given the absence of significance and the null result on the merit-level measure for the same course. Combining both courses, the overall failure rate fell from a pooled historical 31.6% to 16.7% in the Vibe-Coding group, while the concurrent conventional group's failure rate (32.8%) remained close to the historical figure — a pattern that, once again, is attributable primarily to Fundamentals of Programming rather than to a uniform effect across both courses.

E. Engagement, Motivation, and Self-Efficacy

Student surveys and instructor observations also indicated higher engagement. More than 85% of students reported greater confidence in debugging and increased interest in programming, while instructors observed improved programming skills and more complete student projects. These findings are consistent with previous studies on structured AI-assisted learning (Kazemitabaar *et al.,* 2023). Because survey data were aggregated across both courses, future work should investigate course-specific differences.

F. Instructor Workload and Instructional Focus

Instructors reported, in informal qualitative feedback, that offloading manual syntax explanation to AI-assisted practice allowed them to devote more classroom time to higher-order instruction — algorithmic reasoning, analytical thinking, and system design — rather than routine syntax instruction. This shift would be consistent with the broader argument in the literature that AI code generation displaces low-level syntax instruction while leaving higher-order design and evaluation skills as the primary pedagogical target (Becker *et al.,* 2023; Denny *et al.,* 2024a). However, this claim is based entirely on unstructured instructor observations rather than any quantitative measurement of time use (e.g., logged hours spent on lesson preparation, grading, or one-on-one support), and no such data were collected during this pilot. This finding should therefore be treated as a preliminary, qualitative impression rather than an established result, and is reported here for completeness rather than as evidence of an instructor-workload effect; quantitative time-use measurement is identified as a priority for future work in Section IV.I.

G. Novel Contributions of the Model

Compared with previous practice at the institution, the Vibe-Coding model has four key features. First, it formalizes Vibe Coding into a structured semester-long instructional model with implementation guidelines and assessment tools, extending AI-assisted activities proposed in earlier studies (Denny *et al.,* 2023, 2024b). Second, it applies a unified AI-assisted framework across two core programming courses. Third, the model treats AI as a learning tool while requiring students to demonstrate understanding of AI-generated code, addressing concerns about AI over-reliance (Kazemitabaar *et al.,* 2023; Prather *et al.,* 2024). Finally, it incorporates prompt-engineering skills into the grading framework through a dedicated assessment criterion.

H. Discussion: Why the Effect of Vibe Coding Was Uneven Across Courses and Classes

Tables 3 and 4 show that the impact of Vibe Coding varied across courses and classes. One conventional section also recorded a notable improvement. First, Fundamentals of Programming had a higher historical failure rate than Basic Web Design (38.3% vs. 17.8%), leaving greater room for

improvement. Second, gains differed across the Vibe Coding sections: the proportion of students scoring ≥7.0 increased from 10.8% to 53.8% in class 010100096306 and from 42.9% to 63.6% in class 010100096302, whereas no baseline was available for class 010100096307. Because class 010100096306 had the lowest prior-year examination mean (2.17; Table 4), part of its improvement may reflect regression toward the mean rather than the intervention alone. Third, and most tellingly for interpreting the Basic Web Design result, one conventional (non-Vibe-Coding) class in that course, 010100147701, improved sharply between 2024–2025 and 2025–2026 (33.3% to 73.3% on the merit-level measure) despite not adopting Vibe Coding — the single largest year-over-year jump observed anywhere in the dataset, occurring in a class that remained under conventional instruction. This demonstrates directly, within this study's own data, that large swings in class-level performance can and do occur for reasons unrelated to Vibe Coding, reinforcing the caution urged throughout this paper against attributing every observed improvement to the intervention. A similar, if smaller, pattern occurred in Fundamentals of Programming, where conventional class 010100096305 improved from 26.8% to 41.2% without adopting Vibe Coding. Fourth, the two Vibe Coding classes in Basic Web Design showed essentially flat or declining merit-level performance relative to their own prior years (010100100002: 58.6% in 2024–2025 to 62.1% in 2025–2026, itself down from 90.0% in 2023–2024; 010100100003: 70.0% in 2024–2025 to 65.2% in 2025–2026), which is consistent with the null combined-group result for this course and inconsistent with a straightforward beneficial effect of the intervention in these particular classes.

Taken together, these class-level patterns indicate that class-to-class variability unrelated to Vibe Coding is substantial in this dataset — evidenced most clearly by class 010100147701's large gain without the intervention — and that observed aggregate differences must be interpreted with this variability explicitly in mind rather than treated as uniformly attributable to the intervention. This pattern only partially matches the original rationale for Vibe Coding (Section I), which anticipated that structured, supervised AI assistance would help students across the performance spectrum by handling routine syntax and freeing attention for higher-order design. The results instead suggest that any realised benefit in this pilot was concentrated in Fundamentals of Programming and, within that course, more evident in classes with weaker prior-year performance, while no such benefit was statistically detectable in Basic Web Design. Plausible explanations include: (i) students in Fundamentals of Programming faced a steeper syntactic barrier (C/C++) that AI assistance could reduce more directly than the comparatively gentler HTML/CSS/JavaScript learning curve in Basic Web Design (Section I.A), leaving less headroom for AI to add measurable value in the latter; (ii) instructors may have implemented the model with varying fidelity across classes, an aspect not captured by the current data because implementation fidelity was not separately measured (Section IV.F, Section III.F); and (iii) some of the observed gains, particularly in classes with the weakest prior-year baselines, are consistent with regression toward the mean and cannot be attributed to Vibe Coding with confidence on the basis of a single post-intervention cohort — a possibility made concrete by the size of the swings observed even among classes that did not use the intervention.

These findings are broadly consistent with, and add class-level nuance to, prior work on AI-assisted programming education. Kazemitabaar *et al.,* found that AI code generators improved outcomes primarily for learners with weaker prior preparation, provided the tool was embedded in a structured environment (Kazemitabaar *et al.,* 2023) — a pattern that matches the concentration of gains in previously low-performing Fundamentals of Programming classes observed here. Prather et al.'s account of a "widening gap" between students who use generative AI critically and those who become passively dependent on it (Prather *et al.,* 2024) offers one plausible account for why the two Vibe Coding classes in Basic Web Design showed no net improvement; without individual-level prompt-engineering rubric data in the current dataset, however, this cannot be confirmed and remains a testable hypothesis for future work. Finally, the absence of a detectable effect in Basic Web Design — combined with the direct evidence, from class 010100147701, that large swings in performance can occur without any AI intervention — cautions strongly against the assumption, implicit in some prior single-course studies (Denny *et al.,* 2023, 2024b; Finnie-Ansley *et al.,* 2022, 2023), that AI-integrated interventions validated in one programming context will generalise uniformly to others. The present study's two-course design was specifically intended to test this assumption, and the discrepant, and in one course null, results suggest that intervention design, and expectations for effect size, may need to be tailored to the specific difficulty profile of each course rather than assumed to be uniform.

I. Limitations

This study has several limitations. First, although a concurrent conventional comparison group was added in 2025–2026 — addressing the principal limitation of an earlier draft of this study, which relied solely on historical comparison — assignment of class sections to Vibe Coding or conventional instruction followed normal scheduling and staffing decisions rather than random allocation at the individual or section level. This is a methodologically important limitation, not a minor caveat: section-level (rather than individual-level or randomised) allocation means that pre-existing differences between sections — in student composition, section-specific instructor practice, or scheduling — could account for part of the observed differences,

independent of Vibe Coding itself. This concern is not merely theoretical in the present data: as discussed in Section IV.H, one conventional class (010100147701) improved by 40 percentage points on the merit-level measure without adopting Vibe Coding, a swing of similar magnitude to the largest gains observed among the Vibe Coding classes, directly illustrating that section-level, non-random allocation can produce large apparent effects unrelated to the intervention. Second, the historical trend itself (2023–2024 to 2024–2025) spans only two prior years and six to eight class sections per course; the three-way comparison in Section IV.B mitigates but does not eliminate the risk that year-specific factors unrelated to the intervention contributed to the results. Third, the effect of Vibe Coding differed substantially, and in one case reversed direction, by course and by outcome measure: large and statistically significant in Fundamentals of Programming on both the merit-level and pass-rate measures; null (and numerically reversed) on the merit-level measure in Basic Web Design; and only a non-significant trend ($p = 0.097$) on the pass-rate measure in Basic Web Design. This heterogeneity means the combined, two-course result should not be read as evidence of a uniform effect, and should be understood as describing Fundamentals of Programming specifically; conclusions about Basic Web Design, and generalisation beyond Fundamentals of Programming, should be made with particular caution. Fourth, the study remains confined to two courses at a single institution, and generalisability to other institutions, disciplines, or student populations should be made cautiously. Fifth, outcome measures relied partly on instructor-graded assessments and self-report surveys, which are subject to social-desirability and observer bias; independent or blinded assessment was not feasible within the scope of this pilot. Sixth, because the specific versions of GitHub Copilot, ChatGPT, and Claude were not systematically logged, periodic updates to instructional materials are required to counter rapid AI model evolution, making explicit version control essential for future iterations. Seventh, because instructor workload data were not quantitatively measured, the claimed workload reduction (Section IV.F) must be treated as a preliminary impression based on informal observations. Eighth, implementation fidelity was not separately measured, presenting a plausible source of the observed between-class variation in outcomes. Consequently, future research priorities should focus on implementing random assignment to eliminate selection bias, directly measuring implementation fidelity and individual prompt-engineering scores, replicating the model multi-institutionally to test course-level heterogeneity, tracking long-term employability, and quantitatively measuring instructor workload to validate preliminary operational claims.

V. TRENDS AND DEVELOPMENT POTENTIAL

The findings support the integration of AI coding assistants into programming education by emphasizing students' ability to guide, evaluate, and refine AI-generated code (Denny *et al.,* 2024a; Peng *et al.,* 2023; Prather *et al.,* 2023). The proposed Vibe-Coding model offers a structured, assessment-based framework that aligns with Vietnam's digital transformation goals and can be adapted to other programming courses and computing programs (Denny *et al.,* 2024a; Ministry of Education and Training, Vietnam, 2024; Peng *et al.,* 2023).

VI. CONCLUSION

This study evaluated Vibe Coding, a four-stage instructional model that integrates AI coding assistants into programming education at a Vietnamese vocational college. Using a quasi-experimental design that combined a three-year historical dataset with a concurrent comparison between Vibe Coding and conventional class sections, the study found that students in the Vibe-Coding group achieved significantly higher rates of academic success overall. Compared with both the conventional group and the historical baseline, students exposed to Vibe Coding demonstrated higher proportions of passing grades and higher proportions of students achieving scores of 7.0 or above.

However, the results were not consistent across courses. The observed improvements were concentrated largely in Fundamentals of Programming, whereas no statistically significant improvement was detected in Basic Web Design. Furthermore, substantial variation was observed across individual class sections, including one conventional class that improved considerably without adopting the intervention. These findings suggest that the effectiveness of Vibe Coding is likely influenced by course characteristics, implementation conditions, and differences between student cohorts. Consequently, while the model shows promise as a structured approach for integrating AI into programming instruction, its effectiveness should not be assumed to generalize uniformly across learning contexts. Future research should employ randomized or matched-group designs, measure implementation fidelity and prompt-engineering performance more directly, and examine the model across additional courses and institutions.

ACKNOWLEDGEMENT

The authors thank the Head of the Faculty of Information Technology, participating instructors, and students at Ho Chi Minh City College of Economics for their cooperation and support.

Conflict of Interest: The authors declare no conflict of interest.

Funding: This research received no external funding; no state or institutional financial support was provided for the execution of this study.

REFERENCES

- Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The impact of AI on developer

productivity: Evidence from GitHub Copilot. arXiv:2302.06590. https://doi.org/10.48550/arXiv.2302.06590

- Karpathy, A. (2025). Vibe coding [Public commentary popularising the term, February 2025].
- Denny, P., Prather, J., Becker, B. A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B. N., Santos, E. A., & Sarsa, S. (2024a). Computing education in the era of generative AI. Communications of the ACM, 67(2), 56–67. https://doi.org/10.1145/3624720
- 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 (SIGCSE 2023), 500–506. https://doi.org/10.1145/3545945.3569759
- 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 (SIGCSE 2023), 1136–1142. https://doi.org/10.1145/3545945.3569823
- Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., & Reeves, B. N. (2024b). Prompt problems: A new programming exercise for the generative AI era. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V.1 (SIGCSE 2024), 296–302. https://doi.org/10.1145/3626252.3630909
- Prather, J., Reeves, B. N., Denny, P., Becker, B. A., Leinonen, J., Luxton-Reilly, A., Powell, G., Finnie-Ansley, J., & Santos, E. A. (2023). "It's weird that it knows what I want": Usability and interactions with Copilot for novice programmers. ACM Transactions on Computer-Human Interaction, 31(1), Article 4, 1–31. https://doi.org/10.1145/3617367
- Kazemitabaar, M., Chow, J., Ka To Ma, C., 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 (CHI '23), 1–23. https://doi.org/10.1145/3544548.3580919
- Finnie-Ansley, J., Denny, P., Becker, B. A., & Luxton-Reilly, A. (2022). The robots are coming: Exploring the implications of OpenAI Codex on introductory programming. In Proceedings of the 24th Australasian Computing Education Conference (ACE '22), 10–19. https://doi.org/10.1145/3511861.3511863
- Finnie-Ansley, J., Denny, P., Luxton-Reilly, A., Santos, E. A., Prather, J., & Becker, B. A. (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 (ACE '23), 97–104. https://doi.org/10.1145/3576123.3576134
- 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 V.1 (ICER '24), 469–486. https://doi.org/10.1145/3632620.3671116
- Xu, F. F., Alon, U., Neubig, G., & Hellendoorn, V. J. (2022). A systematic evaluation of large language models of code. In Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming (MAPS 2022), 1–10. https://doi.org/10.1145/3520312.3534862
- Ministry of Education and Training, Vietnam. (2024). National strategy for digital transformation in education and training to 2025, with orientations to 2030. Hanoi: MOET.