International Journal of
Innovation In Electronics and Computer Information Technology
Peer Reviewed, Referred & Indexed Journal
ISSN: 3143-4231 www.ijiecit.com Original Research Paper

ARTIFICIAL INTELLIGENCE-BASED PERSONALIZED LEARNING AND INTELLIGENT TUTORING SYSTEMS FOR PROGRAMMING EDUCATION: A REVIEW

S. Nivedha ,Hershini N ,Kalaiarasi K, Durga Syamala K, Kavitha R
Department of Electronics and Communication Engineering
Achariya College of Engineering Technology (An Autonomous Institution)
Approved by AICTE and affiliated to Pondicherry University

ABSTRACT

Writing programs calls for logical thinking, regular practice and useful feedback, since together they develop computational thinking and problem-solving skill. Yet many novices find concepts hard to absorb, cannot easily trace their own mistakes and falter as exercises grow more demanding, while standard classroom teaching rarely adjusts support to what each student already knows or how fast they progress. Artificial Intelligence (AI) is being adopted to narrow this gap through personalised learning environments and intelligent tutoring systems. The present review examines AI-supported personalised programming education, covering intelligent tutors, AI coding assistants, automated code assessment, adaptive learning and individualised feedback. It describes how machine learning, natural language processing and large language models assist students by spotting mistakes, clarifying concepts, generating context-aware hints and suggesting fitting tasks. The influence of gamification, interactive coding environments and gradually rising difficulty on motivation and steady practice is also discussed. These advantages are balanced against drawbacks: the dependability of generated feedback, excessive reliance on AI support, data privacy and academic integrity. By comparing existing approaches and the gaps between them, the review presents a consolidated view of how personalised and tutoring systems can aid programming education and indicates future work on trustworthy, learner-centred systems that pair automated support with independent problem solving.

Keywords—Artificial Intelligence, Personalized Learning, Intelligent Tutoring Systems, Programming Education, Adaptive Learning, Large Language Models, Automated Code Assessment, AI-Powered Feedback, Gamification.

I. INTRODUCTION

A. Background and Motivation

Programming has become central to engineering, science and business curricula, but the introductory course is often where learners get stuck. Beginners have to juggle syntax, invisible program state, breaking tasks into parts and debugging simultaneously, and each of these relies on prompt, precise feedback [1], [2]. When classes are large, that feedback comes slowly, enthusiasm drops, and those who most need support frequently quit [3], [4].

Two lines of technology respond to this problem. The first is AI, including tutoring systems, learner models and large language models (LLMs) that interpret code and answer in everyday language [1], [5], [6]; the second is gamification, which uses points, badges, levels and leaderboards to keep students practising [7], [8]. Each has built up its own body of evidence, yet they are seldom examined together, and the few systems that merge them depend on small or brief trials [9], [10]. This survey stems from our own project, a coding-practice platform featuring levels, test cases and an AI mentor that offers hints without giving away solutions, and it analyses 47 studies, one of which is the base paper [9].

B. Problem Statement and Scope

Traditional teaching hands identical exercises to students with differing backgrounds, and compiler errors or deducted marks seldom reveal the misunderstanding behind a mistake [11], [1]. Current tools each address one component: test-case assessment [8], [12], gamified rewards [13], [3], adaptive difficulty [9], [14] and LLM assistants [5], [6]. Very few environments bring together graded challenges, personal hints and game-based motivation while still prompting learners to think for themselves, and this is the gap the review targets.

International Journal of
Innovation In Electronics and Computer Information Technology
Peer Reviewed, Referred & Indexed Journal
ISSN: 3143-4231 www.ijiecit.com Original Research Paper

We consider programming education at school and undergraduate levels, covering AI personalisation, tutoring, LLM support, automated assessment, gamification and the handful of integrated systems. A study was retained when it presented a specific system, an experiment with learners or a structured review; where the evidence is limited, we state this openly.

C. Contributions and Organisation

This work contributes (1) a classification of 47 studies into seven groups; (2) a side-by-side comparison of techniques, evaluation scale and results; (3) an overview of recurring risks, namely unreliable AI output, over-reliance, privacy, unequal access and weak evaluation; and (4) a platform design, depicted as a flowchart in Section VIII, which puts these lessons into practice.

The paper is organised as follows. Section II describes the method, Section III the background, Sections IV–VI the literature on AI, gamification and integrated approaches, Section VII the comparison and gaps, Section VIII our platform, Section IX the challenges, Section X future directions and Section XI the conclusion. For clarity, personalised learning denotes instruction fitted to an individual, adaptive learning denotes automatic adjustment based on learner data, and a hint denotes feedback that nudges toward a solution without stating it.

II. REVIEW METHODOLOGY

A. Source Collection and Selection

This review draws on 47 papers, the base paper among them, spanning a 2006 Bayesian tutor to articles from 2025–2026, together with our first-review presentation. Every paper was read completely, and its year, technique, setting, participants, outcome measures, findings and limitations were noted. Existing reviews, for example one covering 76 studies on AI in programming education [2] and another covering 125 on AI-based personalisation [15], served to cross-check patterns instead of acting as new evidence.

The set of papers was gathered for a single project and does not result from an exhaustive systematic search, in contrast to several of the reviews included [16], [11]. Because designs vary from single-group tests to meta-analyses, figures cannot be pooled, so each result is reported within its own context. A tendency to publish positive findings may further make reported benefits appear larger than they are.

B. Classification Framework

Each study was placed in one main category (Fig. 1): gamified systems and evidence (11), LLM feedback and assistants (9), automated assessment (7), reviews (7), tutoring and adaptive learning (6), student use of generative AI (4) and integrated systems (3). Evidence on gamification is well established, while work on LLM feedback is new and largely dates from 2024–2025. Integrated systems form the smallest group, consistent with reports that AI tutoring and gamification are typically investigated separately [10], [9]; this imbalance is the root of the gap discussed in Section VII.

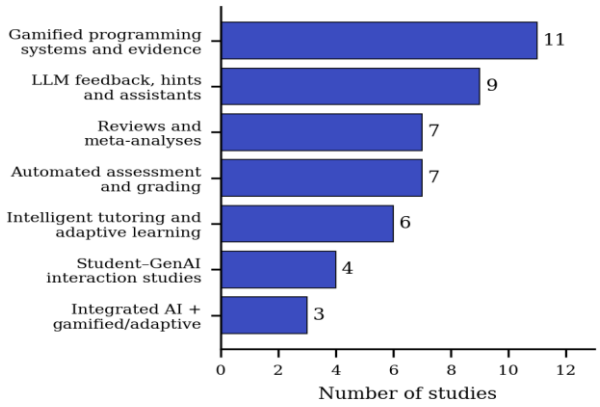

Fig. 1. Distribution of the 47 surveyed studies by primary focus.

III. FOUNDATIONS

A. Difficulties in Learning to Program

Programming is difficult because students need to master syntax, program state, algorithm planning and debugging together. The base study assesses this using Polya's four stages (understand the problem, devise a solution, write code, review) [9], although novices frequently cannot identify which stage went wrong. Feedback is critical: a systematic mapping showed that its form and assessment differ widely and that it clearly influences novice progress [11]. Café offers feedback on both process and product to large first-year cohorts [17], and a chatbot-based platform supplied formative feedback to more than ten thousand school pupils [18].

Motivation forms the second barrier: failing repeatedly without encouragement leads to disengagement, particularly in big introductory classes [3], [4]. Learner background also plays a role, as

International Journal of
Innovation In Electronics and Computer Information Technology
Peer Reviewed, Referred & Indexed Journal
ISSN: 3143-4231 www.ijiecit.com Original Research Paper

gamification did not automatically increase self-reported motivation among students from rote-learning backgrounds, even though their assignment results got better [19].

B. Personalised and Adaptive Learning

Personalised learning tailors teaching to each person. A review of 125 studies (2015–2025) sorted AI personalisation into content recommendation, tutoring and support, and feedback and assessment, reporting benefits as well as challenges around data quality, scalability and ethics [15]. Adaptive learning is the automated side of this: the base study rests on Vygotsky's Zone of Proximal Development and uses levels to pass, hints that can be reopened and additional questions after a failed test, giving a transparent rule-based design [9].

More sophisticated adaptation draws on behavioural and psychological data. One human-centred system tied motivation and interest to improved results among 100 participants [14], while PredictCS applied machine learning to analytics from three courses to identify students at risk, sending them adaptive feedback and giving instructors a dashboard [20]. Simple rules are easier to explain, but predictive models individualise more precisely at the cost of more data; a new platform should therefore begin with transparent rules.

C. Intelligent Tutoring Systems

Intelligent tutoring systems maintain a model of the learner's knowledge and use it to pick the next task or hint. One review observed that most programming tutors target introductory university courses and each handle only a few concepts [1]. Early designs relied on Bayesian networks (BITS) [21] and fuzzy logic, the latter reporting favourable satisfaction, recommendation accuracy and engagement among Greek informatics students [22]; a further system displayed code as road signs in augmented reality [23]. Writing hints by hand restricted coverage, and LLMs can now produce explanations on request, yet a tutor still requires a learner model and a pedagogical policy, which a language model on its own does not provide.

IV. AI TECHNIQUES IN PROGRAMMING EDUCATION

A. Learner Modelling and Predictive Analytics

Learner models are built from interaction traces such as compile attempts, error categories, time spent and test outcomes. PredictCS trained classifiers on three courses to detect probable failures early and deliver tailored feedback [20]. A different study gave SHAP-selected learning features to GPT-4 to compose mid-semester reports, which improved self-regulation and performance, although some students questioned the advice and the authors underlined the need for human oversight [24]. Even basic signals, such as repeated failure on one task, can trigger adaptation, but every data source carries privacy and consent obligations (Section IX).

B. Large Language Models as Programming Assistants

CodeAid responds to conceptual questions, writes pseudo-code with explanations and marks faulty code with suggested fixes, while never supplying complete code. In a twelve-week classroom deployment with around 700 students and about 8,000 uses, the authors derived four design considerations: exploit the distinctive strengths of AI, simplify queries yet keep students thinking, avoid giving direct answers, and provide transparency and control [5]. Other systems adjust exercises and learning paths, with a 20-student experiment reporting quicker learning [25], or combine a chatbot with personalised quizzes [26].

How students behave is just as important. Analysis of ChatGPT logs uncovered five interaction profiles ranging from AI-dependent to AI-independent, and training in prompting moved students between them [27]; a ten-week study of 32 undergraduates connected heavy dependence on AI for code generation and debugging with lower grades [28]. Students judged generative AI useful but found it hard to fix its faulty code [29], and a mind-map chatbot approach was devised to stop younger learners from copying answers [30]. Assistants should thus scaffold reasoning, and students need instruction in using them well.

C. Feedback and Hint Generation

The quality of feedback is the chief worry. A GPT-4 application tested with 51 students handled most errors correctly but also offered wrong suggestions and made up problems [6]. A benchmark of 45 solutions across four models rated 63 percent of hints as accurate and complete and 37 percent as flawed, for example citing wrong lines or imagined issues [31]. For concurrency errors, ChatGPT and Bard achieved

International Journal of
Innovation In Electronics and Computer Information Technology
Peer Reviewed, Referred & Indexed Journal
ISSN: 3143-4231 www.ijiecit.com Original Research Paper

roughly fifty percent accuracy compared with teachers [32], so confidence in AI feedback should decrease as errors become harder.

Limiting the model's freedom is beneficial. One similarity-based system used sentence embeddings to compare student code with a reference solution [33], and a knowledge-graph framework anchored generation in the learner's history; across 4,956 submissions, GenAI feedback outperformed adaptive-only feedback and a hybrid performed best [34]. A copilot framework allowed educators to vet comments before students saw them [35]. For a hint-giving mentor, test-case evidence of what failed is the obvious anchor.

D. Automated Code Assessment and Grading

Test-case assessment underpins most practice platforms: it scales well but says nothing about code quality. Static analysis adds structural checks, as in a customisable C/C++ library [36] and a model that rates originality, accuracy, structure and syntax through AST analysis and plagiarism detection [12]. Café evaluates process via loop invariants [17], while the chatbot platform relied on pattern matching and finite state automata [18].

LLMs push this towards rubric-based grading. Across 315 Python assignments, ChatGPT-4 matched instructors most closely (Pearson 0.91, mean deviation 0.68 points), and a hybrid evaluation was recommended [37]; BeGrading reported discrepancies of roughly 19 and 15 percent [38]; and a correction strategy used with 118 web-programming students individualised statements and highlighted possible plagiarism, earning over ninety percent approval [39]. The most robust design treats test cases as ground truth and keeps language models for explaining why code failed.

V. GAMIFICATION IN PROGRAMMING EDUCATION

A. Game Elements and Motivational Foundations

Gamification brings elements such as points, badges, levels, missions and leaderboards into non-game settings to increase participation and persistence. The base study anchors it in Self-Determination Theory (autonomy, competence, relatedness) and insists that mechanics be linked to learning goals and supported by sound feedback and interface design [9]. Designs also vary by learner: Gamify-IT matches mechanics to HEXAD player types [3], and CodeTraining lets students turn off game components [40]. The effect on intrinsic motivation depends on how familiar learners are with the activity [4], and results may not carry over between cultures [19]. A scoping review of 113 articles found no common method for game design [41].

B. Gamified Programming Platforms

The 2TSW system pairs automated testing and grading with experience points, medals and a leaderboard [8]. FGPE supplies open-source tools that let teachers run gamified courses [13], [42], CodeTraining blends problems, test cases and game elements [40], and GOP1 relies on a model-driven language with a code generator [43]. DolananCoding, a serious game in C++, earned a satisfaction rating of 4.3 out of five from 276 students [44]; Gamify-IT showed higher initial and lasting motivation across two cohorts [3]; and TechTeach recorded 87 percent positive quiz responses [45]. These platforms depend on test cases for immediate feedback but largely offer no tailored guidance after a failure, which is what motivates AI mentors.

C. Empirical Evidence on Effectiveness

A meta-analysis of controlled studies reported a positive effect of about 0.55 (medium) once an outlier was removed, with indications of publication bias [7]; a review of 101 studies concluded that personalised gamification may improve motivation, engagement and results depending on design and context [16]. Single studies give mixed pictures: students with a rote-learning background did better on assignments but not on other measures [19], a flipped, cooperative, gamified course maintained achievement while developing employability skills [46], and the motivational effect varied with familiarity [4]. In sum, gamification supports participation and performance when built into practice with instant feedback, yet it is an auxiliary ingredient rather than a teaching method, and many studies are brief, small or uncontrolled.

VI. INTEGRATING ADAPTIVITY, AI AND GAMIFICATION

A. Adaptive Gamified Learning: The NgodingSeru Base Study

NgodingSeru.com is an adaptive and gamified platform for programming in vocational high schools and is the closest relative of our system [9]. Its levels

International Journal of
Innovation In Electronics and Computer Information Technology
Peer Reviewed, Referred & Indexed Journal
ISSN: 3143-4231 www.ijiecit.com Original Research Paper

progress from variables to modular programming, each including materials, a forum, missions and a competency test; a failed test leads to highlighted errors, explanations and further questions, and the game layer consists of role-play, scores, badges and a leaderboard.

In a mixed-methods study, 96 students from three schools used the platform together with a cooperative learning model. Average scores climbed from 61.7 to 74.2 (about 20.2 percent), the overall Cohen's d was 0.90, and effects at individual schools spanned 0.71 to 1.30. Six students who were interviewed found the platform engaging, appreciated the hints and help from peers, and requested more visual badges and shorter materials.

The authors acknowledge the lack of a control group, confounding with the cooperative model, strict output matching that rejected correct programs, and unexplained variation between schools. We treat these weaknesses as design goals: flexible output checking, briefer explanations, richer rewards and a better hint mechanism, which our AI mentor provides.

B. AI-Mediated Feedback in Gamified Environments

A quasi-experimental study of vocational students compared gamified platforms with and without AI feedback; the AI version produced significantly better performance and interaction-related motivation, isolating the contribution of AI [10]. A generative-AI system that produced questions and graded answers alongside leaderboards was found easy to use by 56.67 percent of students, and 53.33 percent said the leaderboard heightened competitiveness [47], though these are perceptions from a small survey. Adaptive and generative feedback have also been merged without game elements [34], [14], [30].

Integrated systems therefore echo the separate literatures: personalised hints raise performance, and game mechanics keep students engaged. The evidence is limited and short-term, and the central open question is whether hints develop skill or merely help students finish faster.

VII. COMPARATIVE ANALYSIS

A. Cross-Study Comparison and Research Gaps

Table I lists representative studies. Work on assessment and LLM grading is the most quantitative, gamification studies differ more in scale, and studies combining AI with games are few and short-term.

Five gaps emerge. (1) Integration: few systems combine graded difficulty, AI hints and full gamification. (2) Pedagogical control: LLM studies examine correctness more than whether hints actually teach or lessen dependence. (3) Quality of evidence: many experiments involve one group, a few weeks or fewer than fifty learners, and effects differ by site (0.71 to 1.30 in the base study). (4) Language coverage: most studies employ Python, C or C++, while HTML, CSS and JavaScript are less well documented. (5) Ethics and governance: evidence on bias, transparency, privacy and data governance is scarce [2].

Our proposal answers these by joining test-case-grounded checking, an AI mentor that holds back complete solutions and optional competition, with a comparison-group evaluation planned.

TABLE I
SUMMARY OF REPRESENTATIVE STUDIES BY TECHNIQUE, EVALUATION SCALE AND REPORTED OUTCOME

Ref.	Year	Technique / system	Evaluation scale	Main reported outcome
[9]	2025	Adaptive, gamified e-learning platform with missions, hints and levels	96 students, 3 vocational schools; single group	Mean score 61.7 → 74.2; Cohen's d = 0.90
[10]	2026	AI-mediated feedback added to a gamified platform	Vocational students; AI vs. non-AI gamified groups	Higher performance and interaction-related motivation
[47]	2025	Generative AI question creation and grading with leaderboard	Small student survey	Positive perceptions; 56.67% found it easy to use

International Journal of
Innovation In Electronics and Computer Information Technology
Peer Reviewed, Referred & Indexed Journal
ISSN: 3143-4231 www.ijiecit.com Original Research Paper

Ref.	Year	Technique / system	Evaluation scale	Main reported outcome
[3]	2026	Gamify-IT game with HEXAD-aligned mechanics	About 100 students; two cohorts	Higher initial and sustained motivation
[7]	2022	Meta-analysis of gamification in programming	Pooled controlled studies	Positive effect; SMD about 0.55 after sensitivity analysis
[5]	2024	CodeAid: LLM assistant that withholds code solutions	About 700 students, 12 weeks, about 8,000 usages	Four design guidelines for pedagogical AI assistants
[6]	2024	GPT-4 feedback web application	51 students, one semester	Mostly effective; some incorrect or invented hints
[31]	2025	Four LLMs generating hints for student code	45 student solutions	63% accurate and complete; 37% flawed
[32]	2025	ChatGPT and Bard on concurrency exercises	Student exercises vs. teacher judgement	Accuracy about 50%
[37]	2025	LLM rubric grading (GPT-4, Claude 3, Gemini 1.5 Pro)	100 of 315 Python assignments	r = 0.91; mean deviation 0.68 points (ChatGPT-4)
[34]	2026	LLM with retrieval, knowledge graph and learner history	4,956 code submissions	Hybrid adaptive–GenAI mode performed best
[14]	2025	Human-factors adaptive learning system	100 participants; pre/post	Motivation and interest linked to outcomes
[2]	2026	Systematic review of AI in programming education	76 studies	Value in adaptive paths and timely feedback; risks noted

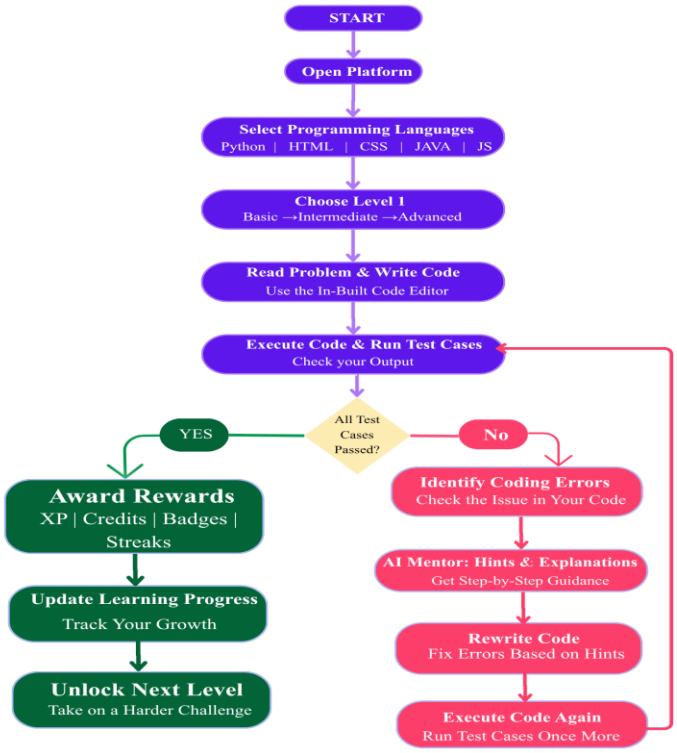

Fig. 3. Workflow of the proposed AI-driven gamified coding platform (learner journey with test-case branching and AI mentor loop).

Received: 10-10-2026 | Accepted: 19-11-2026 | Published: 25-11-2026

International Journal of
Innovation In Electronics and Computer Information Technology
Peer Reviewed, Referred & Indexed Journal
ISSN: 3143-4231 www.ijiecit.com Original Research Paper

VIII. PROPOSED AI-DRIVEN GAMIFIED CODING PLATFORM

A. Platform Workflow

Fig. 3 presents the workflow. The learner starts the platform, selects a language (Python, HTML, CSS, Java or JavaScript), picks a level from basic to advanced, reads the task and writes code in the built-in editor. Running the code applies it to test cases and displays the result.

A decision node verifies whether every test passed. If so, the platform grants experience points, credits, badges and streaks, updates the progress record and opens a harder level, following the lesson that rewards are most effective after clear proof of competence [8], [3]. If not, the system pinpoints the errors, and the AI mentor offers hints and step-by-step explanations; the learner revises the code and runs it again in an open-ended loop, echoing the adaptive support of the base study [9]. Rewards depend on passing tests rather than on time spent or number of attempts, which keeps the game layer fair.

B. AI Mentor Module

The mentor provides hints without disclosing the answer, in line with CodeAid [5] and the GPT-4 study [6]. We apply four hint stages: pinpoint the problem area, name the concept, show a small similar example, and only as an option give pseudo-code for the step. Since more than a third of LLM hints contained errors [31], [32], the mentor is grounded in the problem statement, the reference solution, failing tests with expected and actual output, and previous attempts [33], [34], and a check ensures that a hint neither reveals the solution nor conflicts with test results.

Explanations are tuned to the learner's level and history, and frequent use of the last hint level signals a knowledge gap, much as predictive systems flag at-risk students [20]. To foster independence, because heavy dependence on generated code accompanies weaker results [27], [28], the mentor requests an attempt before offering more help, restricts explicit hints and records patterns for teachers.

C. Gamification and Progress Tracking Module

The gamification module offers experience points and levels (competence), badges (milestones), streaks (regular practice) and credits (a currency of progress); students in the base study asked for trophy-like badges [9]. Leaderboards are optional, with streak goals and badge collections as alternatives [40], [3], and rewards taper off as confidence grows [4].

After every level, a profile stores completed levels, rewards, attempts, hint levels and time. It shows learners their progress, guides the selection of the next challenge and supplies an instructor dashboard [20]. As safeguards, points reward demonstrated skill, and lenient output comparison prevents correct programs from being rejected over formatting [9].

D. Implementation Roadmap and Evaluation Plan

At the time of the first review, requirement analysis and system design were finished, UI design and feature development were under way, and AI integration, support for Python, HTML, CSS and JavaScript, testing and validation, and final integration were scheduled. This order builds a functioning practice-and-reward loop first, so that the mentor can later be evaluated against a stable baseline.

We plan to compare three conditions: the platform with the AI mentor, the same platform with only test-case feedback, and a conventional practice environment [10]. Measurements include a pre/post problem-solving test with effect sizes, usage logs, brief questionnaires [10], [47], instructor review of hint quality [31], and a delayed follow-up to check whether gains last.

IX. CHALLENGES AND LIMITATIONS

A. Reliability, Over-Reliance and Academic Integrity

Reported challenges cover inaccurate AI output, over-reliance, doubts about academic integrity, unequal access and uneven ability to check generated solutions [2]. Hints were incorrect in a sizeable minority of cases [31], [6], [32], and beginners are the least able to notice flaws. Intensive LLM use was associated with lower grades [28], and students had trouble correcting faulty AI code [29]; remedies include restricting how explicit hints are, demanding an attempt first and teaching prompting and verification [27].

To protect integrity, platforms can individualise problem statements, flag questionable authorship using effort data [39] and examine code structure and plagiarism [12]. No mitigation is perfect, so teachers should review feedback [35], [37], and errors should be detectable by rerunning tests.

International Journal of
Innovation In Electronics and Computer Information Technology
Peer Reviewed, Referred & Indexed Journal
ISSN: 3143-4231 www.ijiecit.com Original Research Paper

B. Privacy, Fairness and Accessibility

Privacy: code, error histories and hint requests passed to external LLM services leave institutional control, and evidence on privacy-preserving design is scarce [2]; data should be minimised, logs anonymised and consent obtained. Fairness: adaptive systems can reproduce bias [15], and explanation methods such as SHAP can help [24]. Access: lightweight, offline-capable interfaces and hints in local languages assist resource-limited schools [9], and optional competition eases the pressure of leaderboards and streaks.

C. Evidence Quality and Evaluation Gaps

The evidence base is modest: several experiments involve 20 students [25], 31 submissions [33] and 45 solutions [31]; the studies in [14] and [9] had no comparison group; durations are short and novelty effects are possible. Outcome measures differ greatly [11], publication bias cannot be excluded [7], and LLM results go out of date as models change. The claims are thus promising rather than conclusive, and our evaluation plan relies on a comparison group, logged mentor output and a delayed post-test.

X. FUTURE RESEARCH DIRECTIONS

A. Hybrid, Explainable and Pedagogically Constrained AI

Hybrid designs surpass single components [34], [24], [37], so future work should test deterministic layers (test cases, static analysis, learner models) that constrain and verify generative ones. Open questions concern how detailed a hint should be, when hints turn counterproductive and how to recognise productive struggle; controlled studies should vary hint granularity and delay and measure retention and transfer. Showing why a hint is given, possibly in visual form [23], might also teach students to check AI advice [29], and shared benchmarks of annotated student solutions [31] would let systems be compared.

B. Long-Term, Controlled and Cross-Context Studies

The greatest need is for long-term, controlled and cross-context evidence: longitudinal studies with delayed tests [9], [2], randomised and factorial comparisons of AI feedback and game layers [10], large-scale deployments [18], and validation across schools, cultures and languages [9], [19], including markup languages such as HTML and CSS. Research should also look at how in-platform mentors interact with external assistants and how students and teachers judge whether hints can be trusted.

XI. CONCLUSION

This survey reviewed AI, personalised learning and gamification in programming education through 47 studies. Tutoring and adaptive systems tailor practice, automated assessment evaluates correctness consistently, LLMs explain errors, and gamification keeps practice going; the base study's increase from 61.7 to 74.2 (large effect) and a medium meta-analytic gamification effect point to measurable benefits. Limitations persist: LLM hints are frequently flawed, heavy reliance accompanies weaker outcomes, privacy, fairness and access receive little study, and many experiments are small or uncontrolled.

Our proposed platform allows learners to choose a language and level, write code and run tests; success earns XP, credits, badges, streaks and the next level, whereas failure triggers an AI mentor offering step-by-step hints without the answer. Its evaluation plan seeks stronger evidence than most reviewed studies. Future work will finish AI integration, multi-language support, testing and final integration, and then assess the platform with learners.

REFERENCES

[1] T. Crow, A. Luxton-Reilly, and B. Wuensche, "Intelligent tutoring systems for programming education: A systematic review," in Proc. ACE '18, 2018, doi: 10.1145/3160489.3160492.

[2] J. Katona and K. I. Katonané Gyönyörű, "Artificial intelligence in programming education: A systematic review of intelligent, adaptive, and generative systems, learner outcomes, and implementation challenges," Heliyon, vol. 12, Art. no. e45361, 2026.

[3] N. Meißner, S. Speth, N. Krieger, and S. Becker, "Boosting student motivation through game-based learning in programming education with Gamify-IT," in Proc. 57th ACM Tech. Symp. Comput. Sci. Educ. (SIGCSE TS), 2026, doi: 10.1145/3770762.3772508.

[4] L. Rodrigues, A. M. Toda, W. Oliveira, P. T. Palomino, A. P. Avila-Santos, and S. Isotani, "Gamification works, but how and to whom? An experimental study in the context of programming lessons," in Proc. ACM SIGCSE '21, Mar. 2021.

[5] M. Kazemitabaar, R. Ye, X. Wang, A. Z. Henley, P. Denny, M. Craig, and T. Grossman, "CodeAid: Evaluating a classroom deployment of an LLM-based programming assistant that balances student and educator needs," in Proc. CHI Conf. Human Factors

International Journal of
Innovation In Electronics and Computer Information Technology
Peer Reviewed, Referred & Indexed Journal
ISSN: 3143-4231 www.ijiecit.com Original Research Paper

Comput. Syst. (CHI '24), Honolulu, HI, USA, 2024, doi: 10.1145/3613904.3642773.

[6] S. Jacobs and S. Jaschke, "Evaluating the application of large language models to generate feedback in programming education," arXiv:2403.09744, Mar. 2024.

[7] Z. Zhan, L. He, Y. Tong, X. Liang, S. Guo, and X. Lan, "The effectiveness of gamification in programming education: Evidence from a meta-analysis," Comput. Educ.: Artif. Intell., vol. 3, Art. no. 100096, 2022.

[8] G. Polito and M. Temperini, "A gamified web based system for computer programming learning," Comput. Educ.: Artif. Intell., vol. 2, Art. no. 100029, 2021.

[9] D. Maryono, Sajidan, M. Akhyar, Sarwanto, B. T. Wicaksono, and N. P. T. Prakisya, "NgodingSeru.com: An adaptive e-learning system with gamification to enhance programming problem-solving skills for vocational high school students," Discover Educ., vol. 4, Art. no. 157, 2025, doi: 10.1007/s44217-025-00581-9.

[10] H. Ying and W. A. J. Wan Yahaya, "AI-mediated feedback in gamified programming education: Effects on vocational students' achievement and motivation," Front. Educ., vol. 11, 2026, doi: 10.3389/feduc.2026.1846699.

[11] H. J. Barbosa Rocha, P. C. A. R. Tedesco, and E. de Barros Costa, "On the use of feedback in learning computer programming by novices: A systematic literature mapping," Inform. Educ., vol. 22, no. 2, pp. 209–232, 2023, doi: 10.15388/infedu.2023.09.

[12] R. Parvathy, M. G. Thushara, and J. M. Kannimoola, "Automated code assessment and feedback: A comprehensive model for improved programming education," IEEE Access, 2025, doi: 10.1109/ACCESS.2025.3554838.

[13] J. C. Paiva, R. Queirós, J. P. Leal, J. Swacha, and F. Miernik, "An open-source gamified programming learning environment," in Proc. ICPEC, OASIcs, 2021, doi: 10.4230/OASIcs.ICPEC.2021.5.

[14] K.-Y. Lin, M.-H. Li, F. Lo, H.-C. Huang, K. Matsuno, and R. Watanabe, "Adaptive learning with human factors and artificial intelligence: Associations with training effectiveness in programming education," Int. J. Ind. Ergon., vol. 110, Art. no. 103834, 2025.

[15] H. Farhood, M. Nyden, A. Beheshti, and S. Muller, "Artificial intelligence-based personalised learning in education: A systematic literature review," Discover Artif. Intell., vol. 5, Art. no. 331, 2025, doi: 10.1007/s44163-025-00598-x.

[16] K. Ishaq, A. Alvi, M. Ikram ul Haq, F. Rosdi, A. N. Choudhry, A. Anjum, and F. A. Khan, "Level up your coding: A systematic review of personalized, cognitive, and gamified learning in programming education," PeerJ Comput. Sci., 2024, doi: 10.7717/peerj-cs.2310.

[17] S. Liénardy, L. Leduc, D. Verpoorten, and B. Donnet, "Café: Automatic correction and feedback of programming challenges for a CS1 course," in Proc. Australasian Comput. Educ. Conf. (ACE '20), Melbourne, Australia, Feb. 2020.

[18] L. Benotti, M. C. Martínez, and F. Schapachnik, "A tool for introducing computer science with automatic formative assessment," IEEE Trans. Learn. Technol., vol. 10, 2017.

[19] M. Tasadduq, M. S. Khan, R. M. A. Nawab, M. H. Jamal, and M. T. Chaudhry, "Exploring the effects of gamification on students with rote learning background while learning computer programming," Comput. Appl. Eng. Educ., 2021, doi: 10.1002/cae.22428.

[20] D. Azcona, I.-H. Hsiao, and A. F. Smeaton, "Personalizing computer science education by leveraging multimodal learning analytics," in Proc. IEEE Frontiers Educ. Conf. (FIE), 2018.

[21] C. J. Butz, S. Hua, and R. B. Maguire, "A web-based Bayesian intelligent tutoring system for computer programming," Web Intell. Agent Syst., vol. 4, pp. 77–97, 2006.

[22] K. Chrysafiadi, M. Virvou, G. A. Tsihrintzis, and I. Hatzilygeroudis, "Evaluating the user's experience, adaptivity and learning outcomes of a fuzzy-based intelligent tutoring system for computer programming for academic students in Greece," Educ. Inf. Technol., vol. 28, pp. 6453–6483, 2023, doi: 10.1007/s10639-022-11444-3.

[23] S. Schez-Sobrino, C. Gómez-Portes, D. Vallejo, C. Glez-Morcillo, and M. Á. Redondo, "An intelligent tutoring system to facilitate the learning of programming through the usage of dynamic graphic visualizations," Appl. Sci., 2020.

[24] C. C. Y. Yang, M. Li, and A. Y. Q. Huang, "Leveraging large language models to enhance self-regulated learning in programming education with explainable AI," J. Comput. Assist. Learn., 2026.

[25] C. Li, X. Lee, and X. Wu, "Provide personalized programming learning for individuals based on large language models," Alexandria Eng. J., vol. 132, pp. 396–406, 2025.

[26] E. Zabala and H. S. Narman, "Development and evaluation of an AI-enhanced Python programming education system," in Proc. IEEE Conf., Marshall Univ., Huntington, WV, USA, 2024.

[27] H. Güner and E. Er, "AI in the classroom: Exploring students' interaction with ChatGPT in programming learning," Educ. Inf. Technol., vol. 30, pp. 12681–12707, 2025, doi: 10.1007/s10639-025-13337-7.

International Journal of
Innovation In Electronics and Computer Information Technology
Peer Reviewed, Referred & Indexed Journal
ISSN: 3143-4231 www.ijiecit.com Original Research Paper

[28]G. Jošt, V. Taneski, and S. Karakatič, "The impact of large language models on programming education and student learning outcomes," Appl. Sci., vol. 14, no. 10, Art. no. 4115, 2024, doi: 10.3390/app14104115.

[29]D. Kohen-Vacs, M. Usher, and M. Jansen, "Integrating generative AI into programming education: Student perceptions and the challenge of correcting AI errors," Int. J. Artif. Intell. Educ., vol. 35, pp. 3166–3184, 2025, doi: 10.1007/s40593-025-00496-4.

[30]X. Ye, W. Zhang, Y. Zhou, X. Li, and Q. Zhou, "Improving students' programming performance: An integrated mind mapping and generative AI chatbot learning approach," Humanit. Soc. Sci. Commun., vol. 12, Art. no. 558, 2025, doi: 10.1057/s41599-025-04846-4.

[31]P. Silva and E. Costa, "Assessing large language models for automated feedback generation in learning programming problem solving," Proc. Mach. Learn. Res., vol. 273, pp. 116–124, 2025 (iRAISE Workshop, AAAI 2025).

[32]I. Estévez-Ayres, P. Callejo, M. Á. Hombrados-Herrera, C. Alario-Hoyos, and C. Delgado Kloos, "Evaluation of LLM tools for feedback generation in a course on concurrent programming," Int. J. Artif. Intell. Educ., vol. 35, pp. 774–790, 2025, doi: 10.1007/s40593-024-00406-0.

[33]A. Theeb, N. A. Abdullah, A. Fenjan, and P. Keikhosrokiani, "A lightweight intelligent feedback system for student code submissions using large language models," Discover Educ., vol. 5, Art. no. 371, 2026, doi: 10.1007/s44217-026-01400-5.

[34]L. Na Nongkhai, J. Wang, A. Wynn, and T. Mendori, "Evaluating adaptive and generative AI-based feedback and recommendations in a knowledge-graph-integrated programming learning system," Comput. Educ.: Artif. Intell., vol. 10, Art. no. 100526, 2026.

[35]S. Pozdniakov, J. Brazil, S. Abdi, A. Bakharia, S. Sadiq, D. Gašević, P. Denny, and H. Khosravi, "Large language models meet user interfaces: The case of provisioning feedback," Comput. Educ.: Artif. Intell., vol. 7, Art. no. 100289, 2024.

[36]P. Delgado-Pérez and I. Medina-Bulo, "Customisable and scalable automated assessment of C/C++ programming assignments," Comput. Appl. Eng. Educ., vol. 28, no. 6, pp. 1449–1466, Nov. 2020.

[37]A. Bernik, D. Radošević, and A. Čep, "A comparative study of large language models in programming education: Accuracy, efficiency, and feedback in student assignment grading," Appl. Sci., vol. 15, no. 18, Art. no. 10055, Sep. 2025, doi: 10.3390/app151810055.

[38]M. Yousef, K. Mohamed, W. Medhat, E. H. Mohamed, G. Khoriba, and T. Arafa, "BeGrading: Large language models for enhanced feedback in programming education," Neural Comput. Appl., vol. 37, pp. 1027–1040, 2025, doi: 10.1007/s00521-024-10449-y.

[39]F. Portela, "A generative artificial intelligence tool to correct programming exercises," in Proc. Int. Comput. Programming Educ. Conf. (ICPEC), OASIcs, 2025, doi: 10.4230/OASIcs.ICPEC.2025.7.

[40]"CodeTraining: An authoring tool for a gamified programming learning environment," conference paper (authors and venue to be completed from the source document).

[41]M. Videnovik, T. Vold, L. Kiønig, A. Madevska Bogdanova, and V. Trajkovik, "Game-based learning in computer science education: A scoping literature review," Int. J. STEM Educ., vol. 10, Art. no. 54, 2023, doi: 10.1186/s40594-023-00447-2.

[42]J. C. Paiva, R. Queirós, J. P. Leal, J. Swacha, and F. Miernik, "Managing gamified programming courses with the FGPE platform," Information, 2022.

[43]S. H. Mousavi, S. Kolahdouz Rahimi, and L. Samimi Dehkordi, "A model-driven framework for gamification of learning introductory programming," Hum. Behav. Emerg. Technol., vol. 2025, Art. no. 2420221, 2025, doi: 10.1155/hbe2/2420221.

[44]N. I. Prasetya, N. Saurina, and A. Ardiansyah, "Developing a serious game-based platform to support programming learning," J. Educ. Technol. Develop. Exchange, vol. 19, no. 2, pp. 1–19, 2026, doi: 10.18785/jetde.1902.01.

[45]F. Portela, "A new and interactive teaching approach with gamification for motivating students in computer science classrooms," in Proc. ICPEC, OASIcs, 2020, doi: 10.4230/OASIcs.ICPEC.2020.19.

[46]L. R. Murillo-Zamorano, J. Á. López Sánchez, A. L. Godoy-Caballero, and C. Bueno Muñoz, "Gamification and active learning in higher education: Is it possible to match digital society, academia and students' interests?," Int. J. Educ. Technol. High. Educ., vol. 18, Art. no. 15, 2021, doi: 10.1186/s41239-021-00249-y.

[47]Y.-J. Chen, Z.-P. Chen, C.-H. Lai, and C.-W. Peng, "Generative artificial intelligence-based gamified programming teaching system: Promoting peer competition and learning motivation," Eng. Proc., 2025 (presented at SSIM 2024, Taichung, Taiwan).