EUROPEAN JOURNAL OF STEM EDUCATION

European Journal of STEM Education, 2026, 11(1), Article 42
eISSN: 2468-4368

lectito

Research paper

AI-Enhanced Python Virtual Laboratory Integrated with Moodle LMS: A Qualitative Proof-of-Concept for Computational Thinking Support in Distance Education

Mery Noviyanti[1]*, Sudirman Sudirman[1], Thesa Kandaga[1], Suci Nurhayati[1], Muhamad Galang Isnawan[2]

[1]*Universitas Terbuka*, INDONESIA
[2] *Universitas Nahdlatul Wathan Mataram,* INDONESIA

*Corresponding Author: meryn@ecampus.ut.ac.id
This paper is part of the special issue: AI and STEM: Exploring the Impact of Artificial Intelligence on Science, Technology, Engineering, and Mathematics Education

Citation: Noviyanti, M., Sudirman, S., Kandaga, T., Nurhayati., S & Isnawan, M. G. (2026). AI-enhanced python virtual laboratory integrated with moodle LMS: A qualitative proof-of-concept for computational thinking support in distance education. *European Journal of STEM Education, 11*(1), Article 42. https://doi.org/10.20897/ejsteme/18921

Published: July 8, 2026

ABSTRACT

Programming courses in distance learning institutions face two unresolved problems within existing Learning Management Systems. First, students frequently fail to initiate meaningful programming practice due to technical barriers — installation failures, connectivity issues, and device incompatibilities — that exhaust cognitive resources before algorithmic learning can begin. Second, automated assessment in standard Moodle–VPL configurations evaluates only functional correctness through test-case matching, leaving conceptual misalignments undetected: structurally flawed submissions may pass all test cases, while syntactically minor errors receive no diagnostic feedback. To address both gaps, this study developed and validated an AI-Enhanced Python Virtual Laboratory (AI-PVL) embedding GPT-4 semantic assessment within institutional Moodle–VPL infrastructure, enabling code analysis beyond binary test-case matching. A Participatory Design Research (PDR) methodology was employed across five phases with six lecturers at Universitas Terbuka, Indonesia's national open distance university. Needs analysis (N = 65 students) using Importance–Performance Analysis identified five priority features with gap scores exceeding −2.0, confirming that the primary barrier was the absence of foundational environment affordances. The co-design process yielded three unanticipated innovations: a dual-feedback interface, an instructor-facing misconception frequency map, and co-authored semantic prompt templates. Expert validation via Aiken's V confirmed content and media validity at 84–90% feasibility. An initial proof-of-concept deployment produced observable patterns in student assignment progression across three cycles. The AI-PVL is established as technically feasible and institutionally validated; whether it produces measurable learning gains requires a future adequately powered study.

Keywords: virtual programming laboratory, computational thinking, AI-enhanced feedback, distance education, Moodle LMS

Programming education in distance learning institutions presents a convergence of challenges that classroom-based universities rarely encounter simultaneously. Students must acquire computational thinking skills —

Copyright © 2026 by the author(s). Licensed by Lectito Publications, Netherlands. This is an open-access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

decomposition, pattern recognition, abstraction, and algorithm design (Sudirman, et al., 2026; Wardani et al., 2025; Wing, 2006) — without access to a physical laboratory, without immediate instructor support, and frequently without a stable local development environment. At Universitas Terbuka (UT), Indonesia's sole open distance learning university serving more than 400,000 active students nationwide, these challenges are acute. The Algorithm and Programming course (MATA4110) requires students to write, execute, and debug Python code as part of their assessed learning activities, yet the institution's geographically dispersed student body — many accessing course materials from regions with variable internet connectivity and heterogeneous computing devices — makes the provisioning of a reliable, standardised programming environment a persistent infrastructural challenge (Zabasta et al., 2024). This infrastructural challenge is not unique to UT: research on distance learning in socially diverse contexts consistently demonstrates that the effectiveness of remote educational technologies is profoundly shaped by the socio-technical conditions of the user — including device access, connectivity reliability, and domestic learning environments — making digital equity a prerequisite for, not merely a complement to, effective educational technology design (Sudirman et al., 2021). Needs analysis data collected for this study (N = 65) confirm this picture: 78.5% of enrolled students reported no prior programming experience, and 87.7% relied on personal laptops as their sole computational device, often encountering environment setup failures before any algorithmic learning could begin.

The Virtual Programming Laboratory (VPL) plugin for Moodle has emerged as the most widely adopted institutional response to this challenge, enabling in-browser Python code editing, server-side execution, and automated test-case validation without requiring local software installation (Rodríguez-del-Pino & Hernández-Figueroa, 2022). VPL successfully removes the environment setup barrier that data from this study show is responsible for suppressing early-semester performance: the Importance--Performance Analysis conducted in Phase 1 revealed a gap score exceeding 2.0 points between students' rated importance of reliable code execution and their rated satisfaction with the current environment. However, VPL — and the broader Moodle LMS infrastructure within which it operates — has a fundamental academic weakness that has received insufficient attention in the educational technology literature: its assessment mechanism is architecturally incapable of evaluating conceptual understanding. VPL's reliance on test-case matching produces a binary evaluation paradigm that confirms only whether student code produces the expected output, with no capacity to determine whether the solution reflects understanding of the intended algorithm, uses an appropriate computational approach, or demonstrates the learning objectives the assignment was designed to assess. This is not a configuration limitation that can be resolved through better test case design; it is a structural constraint of output-matching as an assessment philosophy. The practical consequences are well-documented: a student who submits a structurally incorrect solution may receive a passing score if test cases are insufficiently discriminating, while a student who correctly identifies the required algorithmic approach but makes a minor syntactic error receives a failing score with no explanation of what their thinking got right (Ramos et al., 2021; Messer et al., 2024). In a conventional classroom, an instructor can intercept both of these students and provide the conceptual correction that the automated system cannot. In distance learning, where students lack access to real-time instructor feedback and must rely entirely on system-generated responses to guide their learning, this assessment gap is not merely an inconvenience — it is a structural barrier to the development of the computational thinking competencies that programming courses are designed to build (Tan et al., 2025; Keuning et al., 2018).

Recent advances in large language models (LLMs) have opened a substantively different assessment pathway. Studies demonstrate that LLM-based systems can evaluate code at the semantic level – identifying whether a solution addresses the correct problem, uses an appropriate algorithmic approach, and demonstrates the intended computational concepts, rather than merely checking whether it produces the right output (Impey et al., 2024; Tan et al., 2025). GPT-4 in particular has shown reliability approaching human expert judgement in educational code review tasks (Impey et al., 2024). The broader question of whether AI systems can assess student work with the validity, reliability, and fairness expected of human graders has received increasing attention in the international literature. Zacharis and Papadakis (2025), in a study of AI-assisted grading across university coursework, found that while AI assessment tools demonstrate strong potential for consistency and efficiency, significant challenges remain around construct validity and equitable treatment of diverse student responses — findings that directly inform how the semantic assessment layer in the present system was designed and validated. These concerns are consistent with the broader literature on AI in education, which identifies data privacy, algorithmic bias, and the erosion of relational dimensions of teaching as persistent structural challenges that accompany the adoption of AI-driven assessment tools across educational levels and contexts (Acar et al., 2025). This question is not new to programming education specifically: research on novice programming environments has consistently shown that the design of the assessment and feedback interface — not just its technical capability — determines whether students engage productively with automated responses (Papadakis & Orfanakis, 2018). The challenge of designing environments that support genuine learning rather than superficial

 © 2026 by Author/s

compliance applies equally whether the automated feedback is generated by test-case matching or by a large language model (Yıldırım et al., 2024). However, existing LLM-based implementations share a critical structural limitation: they operate as external tools – standalone applications or API services accessed outside the LMS – requiring students and instructors to leave the institutional learning environment, disrupting established workflows and creating adoption barriers that are well-documented in the educational technology literature (Akojie et al., 2022; Khlaif et al., 2024). This adoption barrier is compounded by the uneven distribution of digital readiness across institutions and national contexts: teachers and students in environments with limited digital infrastructure are disproportionately disadvantaged by tools that require stable external connectivity, and the gap between technological capability and actual pedagogical uptake is widest precisely in the contexts — such as open distance universities in developing countries — where such tools are most needed (Amka & Dalle, 2022). To the best of our knowledge, no published system has yet embedded LLM-based semantic code analysis directly within an institutional Moodle – VPL infrastructure in a way that preserves existing gradebook integration, assignment management, and authentication workflows while extending assessment capability to the semantic level – though the pace of development in this area means that parallel work may exist that has not yet reached publication.

Existing AI-enhanced assessment systems share a second limitation beyond their technical architecture: they are typically developed without systematic involvement of the course lecturers who will use them. Nor do they attend sufficiently to the student perspective: research in Indonesian higher education contexts shows that students' relationships with AI tools are mediated by factors including trust, perceived personalization, and alignment with individual learning goals — dimensions that technically capable but pedagogically decontextualised systems consistently fail to address (Sultan et al., 2025; Novianti et al., 2026). Course lecturers possess contextual knowledge that is irreplaceable in the design of educationally sound automated feedback: they know which types of conceptual errors are most common at which points in the curriculum, which Moodle workflows must be preserved for institutional acceptance, and what diagnostic information would actually change their instructional decisions. Designing an AI assessment system without this knowledge risks producing a technically capable but pedagogically misaligned tool — one that generates feedback that is accurate but not actionable, or that is adopted in isolated pilots but fails to diffuse into institutional practice (Fishman et al., 2013). PDR directly addresses this gap by positioning practitioners as *co-designers* rather than end-users, embedding their knowledge into the design specification, the iterative prototype cycle, and the validation criteria (Barab & Squire, 2004; Muller & Druin, 2012). At UT, where six course lecturers carry collective expertise in the specific computational thinking challenges of distance learners in MATA4110, PDR provides the methodological structure to convert this expertise into verifiable design decisions.

This study addresses the intersection of these two gaps, the technical gap in AI-enhanced VPL integration and the methodological gap in practitioner-centred educational technology design, by developing, validating, and conducting an initial proof-of-concept deployment of an AI-Enhanced Python Virtual Laboratory (AI-PVL) embedded within Universitas Terbuka's Moodle LMS infrastructure through a PDR process involving six course lecturers as co-designers across five iterative phases. The system integrates three layers: the Moodle LMS (authentication, grade management, and assignment distribution), the VPL plugin (in-browser code editing and test-case execution), and an OpenAI GPT-4 semantic assessment engine (conceptual alignment analysis and diagnostic feedback generation). To the best of our knowledge, the resulting system represents one of the first documented implementations of LLM-based semantic code assessment embedded directly within an institutional VPL environment, validated through Aiken's V expert review (84% overall feasibility) and deployed in an initial classroom context to document observable assignment progression patterns across three assignment cycles.

This study is guided by the following four research questions, each corresponding to a core phase of the PDR cycle:

RQ1: What are the needs of mathematics education students at Universitas Terbuka for a Python Virtual Laboratory integrated within a Moodle-based e-learning platform?

RQ2: What design model of an AI-Enhanced Python Virtual Laboratory, developed through a participatory co-design process with course lecturers, effectively supports algorithm and programming learning in a distance education context?

RQ3: How valid and feasible is the AI-Enhanced Python Virtual Laboratory as assessed by domain experts across content, media, and functionality dimensions?

RQ4: How does the AI-Enhanced Python Virtual Laboratory perform in an initial classroom deployment, and what observable patterns in student assignment progression does it produce in the Algorithm and Programming course at Universitas Terbuka?

© 2026 by Author/s

METHOD

Research design

This study adopted PDR as its methodological framework, structured across five sequential and iterative phases: Discover, Co-Design, Develop, Validate, and Evaluate. PDR was selected because the central design challenge — embedding AI-driven semantic assessment within an institutional Moodle–VPL environment — is inherently situated within the practical knowledge, pedagogical habits, and workflow constraints of the lecturers who teach the Algorithm and Programming course at Universitas Terbuka (UT). Rather than treating lecturers as end-users who validate a completed system, PDR positions them as *co-designers* with active decision-making authority across the full design cycle (Barab & Squire, 2004; Muller & Druin, 2012).

The research process unfolded across five sequential and interdependent phases, each producing outputs that directly shaped the work of the phase that followed. The study began with a problem diagnosis phase (Phase 1: Discover), in which the research team sought to establish an empirical rather than assumed understanding of what distance learning students at UT actually needed from a virtual programming environment. This phase combined a structured needs survey administered to 65 enrolled students with two rounds of focus group discussions involving all six course lecturers, producing a prioritised design specification grounded in both student-reported gaps and practitioner-observed challenges. This specification then became the primary input to the design phase (Phase 2: Co-Design), in which the six lecturers assumed the role of co-designers across four structured participatory workshops. Rather than reacting to a completed prototype, lecturers made active design decisions — selecting features, rejecting proposals, and originating innovations — that the research team was then obligated to implement. The outputs of this phase were a validated system blueprint and a set of design decision logs that documented the rationale behind each architectural choice. The development phase (Phase 3: Develop) translated this blueprint into a working system through three iterative sprints, each of which closed with a lecturer usability walkthrough that fed directly into the next sprint's revision cycle. This iterative structure ensured that the system that proceeded to validation was one that lecturers had already reviewed and refined, rather than one presented to them for the first time at the point of formal evaluation. The validation phase (Phase 4: Validate) then subjected the co-designed system to independent expert scrutiny, with two domain experts applying Aiken's V index (Aiken, 1985) across 14 indicators covering content validity, interface design, and technical functionality. Only after this independent validation was completed and sub-threshold indicators addressed did the study proceed to the deployment phase (Phase 5: Deploy), in which the system was introduced in a real classroom context through a six-week initial deployment with a comparison group (n = 7) using the conventional Moodle–VPL environment and a deployment group (n = 10) using the fully integrated AI-PVL. This sequencing is not incidental: the descriptive observations from Phase 5 are meaningful only because the validity of what was deployed was established in Phase 4, which depends on the design rigour established in Phases 2 and 3, which depends on the empirical needs profile established in Phase 1. **Table 1** presents the complete phase structure.

Table 1

PDR phase structure: Timeline, participants, activities, and data produced

Phase	Participants	Key Activities	Data Produced
Phase 1 Discover	6 lecturers 65 students	Contextual inquiry; needs survey; IPA workshop; lecturer focus groups (2 rounds × 90 min)	Needs questionnaire data (N=65); IPA matrix; focus group transcripts; design specification document
Phase 2 Co-Design	6 lecturers (co-designers)	4 participatory workshops (3 hr each); wireframe reviews; card-sorting; pedagogical alignment sessions	Design decision logs; annotated wireframes; agreed feature specification; prototype feedback notes
Phase 3 Develop	6 lecturers + dev team	3 iterative development sprints; lecturer usability walkthroughs per sprint; severity-rated bug review	Sprint retrospective logs; usability checklist results; change-request records
Phase 4 Validate	2 expert validators 6 lecturers	Expert validation sessions (Aiken's V); lecturer acceptance review; revision and finalisation	Aiken's V per indicator; feasibility %; radar chart; acceptance confirmation record
Phase 5 Deploy	Comparison n=7 Deployment n=10	Initial classroom deployment; 3 programming assignments (T1–T3) over 6 weeks; assignment rubric scoring	Assignment scores T1–T3; gain score observations; descriptive statistics

© 2026 by Author/s

Participants

Lecturers (co-designers)

The participants, data collection instruments, and analytical procedures described in the sections that follow are each embedded within this five-phase process. Rather than reading them as independent components, they are best understood as the specific operationalisation of each phase: who was involved at each stage, what data were collected to answer each phase's question, and how those data were analysed to produce the inputs that the next phase required. Six lecturers (n = 6) who teach the Algorithm and Programming course (MATA4110) at UT's Faculty of Mathematics and Natural Sciences participated as co-designers throughout Phases 1–4. All six hold a minimum qualification of Magister (S2) in Computer Science, Informatics Education, or a related discipline, and each has a minimum of three years of active teaching experience in the course. Lecturer participation was voluntary; written consent was obtained prior to Phase 1. Participation involved: (a) two rounds of focus group discussions in Phase 1 (approximately 90 minutes each), (b) four co-design workshops in Phase 2 (approximately three hours each), (c) three usability walkthrough sessions in Phase 3 (approximately 60 minutes each), and (d) one acceptance review session in Phase 4 (approximately 90 minutes). Total lecturer involvement across the study was approximately 18 hours per participant over the ten-month period (January–October 2025).

Students — Needs analysis

A total of 65 students (N = 65; 43.1% male, 56.9% female) enrolled in MATA4110 at UT participated in the needs analysis survey during Phase 1. Participants spanned four semester levels: Semester 3 (18.5%), Semester 5 (38.5%), Semester 7 (27.7%), and Semester 9 (15.4%). The age distribution was: 20–25 years (33.8%), 26–30 years (43.1%), 31–35 years (15.4%), and above 35 years (7.7%). Regarding prior coding experience, 78.5% reported no previous programming experience, and 21.5% had taught themselves independently. In terms of device access, 87.7% owned a laptop and 12.3% used a desktop PC. Participation was voluntary and anonymous; no identifying information was collected.

Expert validators

Two domain experts (n = 2) were recruited for the Phase 4 expert validation. Expert 1 specialises in educational technology and learning management system architecture, with more than ten years of experience in Moodle-based system development and instructional design. Expert 2 specialises in programming education and computational thinking assessment, with extensive experience designing and evaluating programming courses in higher education. Both experts were external to Universitas Terbuka and had no prior involvement in the system's design or development, ensuring independent and unbiased validation.

Students — Initial classroom deployment

Seventeen students enrolled in MATA4110 participated in the Phase 5 classroom deployment across two naturally-formed tutorial groups. The comparison group (n = 7) completed the course using the existing conventional Moodle–VPL environment, without access to the AI semantic assessment layer. The deployment group (n = 10) completed the same course content and assignments using the fully integrated AI-Enhanced Python Virtual Laboratory. Group formation was based on existing tutorial group assignments and was not manipulated by the researchers. Because students were not randomly assigned to conditions, pre-existing differences between groups cannot be ruled out as factors in any observed patterns. The baseline score difference at T1 (deployment group M = 72.90 vs. comparison group M = 60.00) confirms that the groups were not equivalent at the outset; all observations should be interpreted with this constraint in mind. No causal or effectiveness claims are made from this deployment.

Data Collection

Phase 1 — Student needs questionnaire

A structured questionnaire comprising 13 indicators across three domains was administered to all 65 students via Google Forms during the first two weeks of the semester. Domain 1: *Python Environment and Editor Usage* (3 indicators) measured familiarity with code editors, Python installation experience, and file management. Domain 2: *Algorithm Coding and Problem-Solving* (5 indicators) measured students' experience with debugging, program construction, real-time feedback, and code execution. Domain 3: *Computational Thinking* (5 indicators) measured self-reported ability in decomposition, pattern recognition, abstraction, algorithm design, and comparative analysis. All items used a five-point Likert scale (1 = *strongly disagree*, 5 = *strongly agree*). Internal consistency was α

© 2026 by Author/s

= .87 (Cronbach's alpha across all 13 indicators), confirmed acceptable prior to analysis. **Table 2** presents the descriptive statistics for each indicator.

Table 2
Descriptive Statistics per Needs Analysis Indicator

Indicator	N Items	Mean	SD	Min	Max	Category
Device and Internet Access	3	3.81	0.92	2	5	High
Python & Editor Experience	3	2.79	1.32	1	5	Moderate
Installs & Environment Management	3	2.05	0.84	1	4	Low
Editor Features & Compiling	4	4.45	0.61	3	5	Very High
Integration & Storage	2	3.88	0.94	2	5	High
Basic Understanding of Algorithms	3	3.54	0.90	2	5	High
Debugging Tools	3	3.39	1.02	2	5	Moderate
Programming	2	2.59	0.81	1	4	Low
Learning Challenges and Support	3	3.85	1.04	2	5	High
Problem Decomposition	3	3.16	1.07	2	5	Moderate
Pattern Recognition	2	2.66	0.55	2	4	Moderate
Abstraction	2	2.25	0.67	1	3	Low
Algorithms & Comparison	3	3.20	1.12	2	5	Moderate

In addition to the survey, an Importance–Performance Analysis (IPA) matrix was constructed by plotting each indicator's mean *importance* score (how critical students rated the need) against its mean *current performance* score (how adequately the existing environment met that need). The IPA matrix directly produced the feature priority tier used in Phase 2 co-design workshops. Indicators in Quadrant I (high importance, low performance) — debugging tools (M = 4.57), code visualisation (M = 4.55), auto-save (M = 4.54), run-code functionality (M = 4.52), and real-time feedback (M = 4.51) — were designated as mandatory features, with a gap score exceeding 2.0 points relative to current performance.

Phase 1 — Lecturer focus groups

Two rounds of focus group discussions were conducted with all six lecturers, each lasting approximately 90 minutes, facilitated by the lead researcher using a semi-structured protocol. Round 1 explored: (a) current challenges in assessing student code submissions at scale in a distance learning context, (b) specific types of conceptual errors that test-case matching fails to detect, and (c) existing Moodle and VPL workflows that must be preserved. Round 2, conducted two weeks later after preliminary analysis of Round 1 transcripts, presented emerging design themes for member verification and explored: (d) what information AI-generated feedback should surface to students, and (e) what instructor-facing analytics would support pedagogical decision-making. All sessions were audio-recorded and transcribed verbatim. Thematic analysis was applied using an inductive coding procedure (Braun & Clarke, 2006), with inter-rater reliability established between two independent coders (Cohen's $\varkappa$ = .81, indicating strong agreement).

Phase 2 — Co-design workshop documentation

Four participatory co-design workshops were conducted fortnightly with all six lecturers, each lasting approximately three hours. Each workshop used structured facilitation tools: (a) printed wireframes of proposed interface screens for annotation and critique, (b) feature card-sorting exercises in which lecturers ranked candidate features by pedagogical priority, and (c) scenario-based walkthroughs in which lecturers assumed the roles of both student and instructor to evaluate design proposals in situ. At the close of each workshop, a design decision log was completed, recording which features were accepted, modified, or rejected and the rationale given by the lecturer group. Between workshops, revised medium-fidelity prototypes were circulated to lecturers via a shared annotation platform; written comments were systematically recorded and classified by priority before the next session. These logs and annotations constitute the primary qualitative data corpus of Phase 2.

© 2026 by Author/s

Phase 4 — Expert validation instrument

A structured validation rubric comprising 14 indicators across three dimensions was administered to both expert validators independently: (a) *Content and Material Validity* (5 indicators: curriculum alignment, accuracy of AI feedback, CT assessment criteria, assignment relevance, learning objective coverage), (b) *Media and Interface Design Validity* (5 indicators: usability, navigation clarity, cognitive load, accessibility, visual design), and (c) *Functionality and Practicality Validity* (4 indicators: Moodle integration stability, workflow coherence, response latency, scalability). Each indicator was rated on a five-point scale. Aiken's V was computed for each indicator as $V = \frac{\sum s}{n.c(r-1)}$, where s is the sum of deviation scores from the minimum rating, n is the number of raters (2), c is the number of categories (5), and r is the maximum rating (5). An indicator was classified as valid when V ≥ 0.75 (Aiken, 1985). Feasibility percentage per dimension was computed as the proportion of indicators in that dimension reaching the validity threshold. Table 3 presents the validation results.

Table 3

Expert validation results — Aiken's V index and feasibility by dimension

Validation Dimension	No. Indicators	V Range	% Feasibility	Decision
Content & Material (CT alignment, feedback accuracy)	5	0.750–1.000	90%	Valid — proceed to pilot
Media & Interface Design (usability, cognitive load)	5	0.750–1.000	90%	Valid — minor UI revision noted
Functionality & Practicality (LMS integration, workflow)	4	0.500–1.000	80%	1 indicator (Moodle auth) flagged for technical fix
Overall	14	0.500–1.000	84%	Feasible — approved for controlled deployment

Phase 5 — Assignment rubric for classroom deployment

Student assignment performance was recorded through three progressively complex Python programming assignments (T1, T2, T3), administered at Weeks 2, 4, and 6 of the six-week deployment period. Each assignment required students to solve an algorithmic problem that explicitly engaged the four CT components specified in the course learning objectives, as operationalised by Wing (2006): *decomposition* (breaking the problem into solvable sub-tasks), *pattern recognition* (identifying structural regularities in the problem), *abstraction* (identifying the essential features of the solution), and *algorithm design* (producing a correct, efficient, and clearly structured Python solution). Assignment scores were derived using a validated rubric aligned with these four CT components, scored on a 0–100 scale. Two independent raters scored all submissions; inter-rater reliability was established prior to analysis (Cohen's κ = .84, indicating strong agreement). In cases of discrepancy exceeding 5 points, a third rater adjudicated.

The deployment group additionally received AI-generated semantic feedback at each submission point. This feedback was produced by the Layer 3 OpenAI GPT-4 API component, which analysed submitted code against the assignment learning objectives and identified conceptual misalignments — cases where code executed correctly but failed to address the intended algorithmic task — providing diagnostic commentary beyond the binary pass/fail output of conventional test-case matching.

ANALYSIS

Needs analysis

Descriptive statistics (mean, standard deviation, minimum, maximum) were computed for each of the 13 needs indicators and aggregated at the domain level using SPSS v.26. Category thresholds followed the conversion scale: Very High (≥ 4.20), High (3.40–4.19), Moderate (2.60–3.39), and Low (< 2.60). The IPA matrix was constructed in Microsoft Excel by computing the mean importance score and mean current performance score for each indicator, plotting them on a two-dimensional quadrant, and identifying the priority tier for each feature based on quadrant membership.

Focus group transcripts were analysed using inductive thematic analysis (Braun & Clarke, 2006) in six stages: familiarisation, initial code generation, theme development, theme review, theme definition, and report production. NVivo 14 was used to manage the coding process. Inter-rater reliability (κ = .81) was established by

© 2026 by Author/s

having a second researcher independently code 30% of the transcript corpus, followed by structured discussion to resolve disagreements.

Expert validation

Aiken's V was calculated for each of the 14 validation indicators using the formula stated in Section 2.3.4. Feasibility classification followed the scale: *Very Feasible* (≥ 85%), *Feasible* (70–84%), *Sufficiently Feasible* (55–69%), and *Not Feasible* (< 55%). A radar chart was produced to visualise convergence between the two validators across the three validation dimensions. Indicators failing to reach V ≥ 0.75 were documented with the validator's rationale and addressed in a targeted revision cycle prior to pilot deployment.

Classroom deployment observation

Phase 5 data were examined through two descriptive steps to document observable patterns in student assignment progression. This analysis is not intended as a test of effectiveness; given the small, naturally-formed groups (comparison n = 7; deployment n = 10) and the absence of random assignment, no inferential claims about the system's causal impact can be supported. The purpose of the quantitative records here is descriptive: to document what patterns were visible during the deployment and to provide a basis for future hypothesis generation in adequately powered studies. Step 1 — Descriptive statistics. Mean, standard deviation, minimum, and maximum were computed for each group at each time point (T1, T2, T3). Score distributions were examined to identify patterns in assignment progression across the six-week period. Step 2 — Gain score observation. Net change was calculated as T3 score − T1 score for each student and averaged by group. Gain scores are reported as descriptive observations only, interpreted in relation to each group's starting point to provide context for the patterns observed. All analyses were conducted in SPSS v.26. Table 4 presents the descriptive results.

Table 4

Assignment score descriptive statistics by group and time point (T1–T3)

Group	Task	n	Mean	SD	Min	Max
Comparison	T1	7	60.00	8.43	50	72
Comparison	T2	7	66.28	6.99	58	78
Comparison	T3	7	68.71	7.04	60	80
Deployment	T1	10	72.90	15.52	46	96
Deployment	T2	10	74.50	28.00	0	97
Deployment	T3	10	79.20	16.27	55	100

Ethical considerations

This study was approved by the Institutional Ethics Committee of Universitas Terbuka prior to data collection. All participants — lecturers, student informants, expert validators, and pilot students — provided written informed consent before participating. Consent forms explained the purpose of the study, the nature of participation at each phase, the voluntary nature of involvement, the right to withdraw at any time without consequence, and how data would be stored, used, and reported.

Student assignment scores collected in Phase 5 were anonymised prior to analysis by replacing names with participant codes. No individual student is identifiable in any reported result. For Phase 1 survey participants, no identifying information was collected; responses were submitted anonymously via Google Forms. Lecturer focus group and co-design workshop data were pseudonymised; direct quotations used in reporting are attributed to participant codes (L1–L6) rather than to individuals. Expert validator identities are disclosed only at the level of their professional domain (educational technology; programming education) without naming individuals, consistent with their consent agreements.

Student performance data submitted to the AI assessment engine (Layer 3, OpenAI GPT-4 API) were processed in compliance with OpenAI's data usage policies and UT's institutional data protection framework, which aligns with Indonesian Government Regulation on Personal Data Protection (PP No. 71/2019). Beyond local regulatory compliance, the data handling protocols adopted in this study were designed to meet the principles underlying major international data privacy frameworks, including the European Union's General Data Protection Regulation (GDPR) and the OECD Privacy Guidelines — frameworks that, while not directly binding in Indonesia, represent internationally recognised standards for the ethical handling of personal data in research contexts (European Parliament, 2016; OECD, 2013). Three specific measures were implemented to

 © 2026 by Author/s

operationalise these principles. First, data minimisation: code submissions processed by the API were stripped of all personally identifiable information at the point of transmission, including student names, identification numbers, and any biographical details, ensuring that the data sent to the commercial API contained only the code text and the assignment prompt necessary for semantic analysis. Second, purpose limitation: the OpenAI API was used exclusively for the specific assessment function described in this study; no data were used for model training, profiling, or any purpose beyond generating diagnostic feedback for the assignment in question, consistent with OpenAI's API data usage policies for enterprise and research contexts, which explicitly exclude API-submitted data from being used to train OpenAI models by default. Third, storage limitation: API interaction logs were retained for a maximum of 90 days on institutional servers and then permanently deleted, with no data transferred to or retained by third parties beyond this period. Institutions considering analogous deployments in jurisdictions with stricter data sovereignty requirements — including EU member states subject to GDPR — should note that additional data processing agreements with the API provider may be required, and that open-weight local LLM deployment represents an alternative architecture that eliminates third-party data transfer entirely.

RESULTS

The results are organised according to the four research questions, each corresponding to a phase of the PDR cycle. The sections are cumulative: the needs profile from RQ1 informed the design priorities in RQ2; the system produced in RQ2 was the one validated in RQ3; and the validated system is the one whose initial classroom deployment is documented in RQ4. The results are organised sequentially: each phase's findings are presented in relation to the phase that preceded it.

What students actually need

The needs analysis addressed a foundational question: what do distance learning students at Universitas Terbuka require from a virtual programming environment? The findings were more specific than the existing literature anticipates. The primary unmet needs were not advanced features but basic affordances — reliable code execution, debugging support, auto-save, and real-time feedback — without which programming practice cannot begin.

Table 5 presents descriptive statistics and Importance–Performance Analysis (IPA) classifications for all 13 need indicators across three domains. The column to pay closest attention to is not the mean importance score — which is high across the board — but the *gap score*: the difference between how much students value each feature and how adequately the current environment delivers it. A large negative gap score is a diagnostic signal that the environment is actively failing students on something they consider critical.

Table 5

Descriptive Statistics and IPA Classification for All 13 Need Indicators (N = 65)

Indicator	Items	Mean	SD	Category	IPA Quadrant	Gap Score
Device and Internet Access	3	3.81	0.92	High	Keep Up	−0.44
Python & Editor Experience	3	2.79	1.32	Moderate	Low Priority	−0.31
Installs & Environment Management	3	2.05	0.84	Low	Low Priority	−0.52
Editor Features & Compiling	4	4.45	0.61	Very High	Concentrate	−2.18
Integration & Storage	2	3.88	0.94	High	Concentrate	−2.36
Basic Understanding of Algorithms	3	3.54	0.90	High	Keep Up	−0.61
Debugging Tools	3	3.39	1.02	Moderate	Concentrate	−2.16
Programming	2	2.59	0.81	Low	Low Priority	−0.48
Learning Challenges and Support	3	3.85	0.94	High	Concentrate	−2.31
Problem Decomposition	3	3.16	1.07	Moderate	Low Priority	−0.74
Pattern Recognition	2	2.66	0.55	Moderate	Low Priority	−0.61
Abstraction	2	2.25	0.67	Low	Low Priority	−0.58
Algorithms & Comparison	3	3.20	1.12	Moderate	Low Priority	−0.67

© 2026 by Author/s

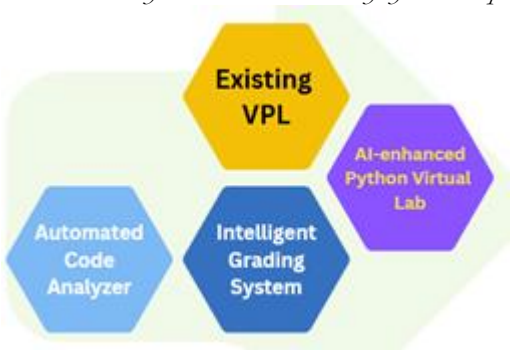
The pattern that emerges from Table 5 is not random. The five indicators with gap scores exceeding −2.0 — Auto-Save (−2.36), Real-Time Feedback (−2.31), Code Visualisation (−2.24), Run/Execute Code (−2.23), and Debugging Tools (−2.16) — are not five unrelated preferences. They are five symptoms of the same underlying problem: students cannot reliably run, observe, fix, or save their code in the current environment. In other words, the most unmet needs are precisely the ones that make programming practice possible in the first place. This finding matters because it reframes the design problem. The challenge is not to enrich an already-functional learning environment with additional features; it is to repair a broken one.

This interpretation is further supported by the domain-level picture. Domain B (Algorithm Coding and Problem-Solving) returns the highest aggregated mean (M = 3.74), indicating that students are acutely aware of what active coding practice requires. Yet Domain A (Python Environment and Editor Usage) and Domain C (Computational Thinking) both fall at Moderate level (M = 2.88 and 2.82 respectively) — not because students value them less, but because unmet foundational needs in Domain B systematically crowd out the cognitive bandwidth required for the higher-order skills that Domain C represents. Students who spend the first two weeks of a semester troubleshooting installation errors are not failing to develop computational thinking because they lack ability; they are failing because the environment has not yet allowed them to start. This is the structural problem that the AI-enhanced PVL was designed to interrupt — a point that becomes the organising logic for every design decision that follows in the co-design phase.

From needs to design

If the needs analysis identified the structural problem, the co-design workshops with the six course lecturers were where that problem was translated into a specific architectural solution. Three months of bi-weekly workshops did not simply produce a list of features — they produced a set of design decisions that would not have been reachable without practitioner knowledge, and that distinguish this system from every comparable VPL implementation in the published literature (Figure 1).

Figure 1
AI-enhanced Python virtual laboratory system components



The most consequential of these decisions concerned how AI feedback should be presented to students. The research team's initial proposal — replacing the standard VPL test-case result screen with AI-generated semantic analysis — was rejected unanimously in the first workshop. Lecturers' reasoning, documented verbatim in the design decision log, was precise: removing the test-case result would deprive students of the immediate, objective signal they use to confirm that their code *runs at all*, and that AI analysis presented in isolation would be treated with scepticism because it carries no verifiable ground truth. The dual-feedback interface that replaced this proposal — test-case output on the left, AI semantic analysis on the right, always presented together — is not a compromise. It is a pedagogically superior solution that no one on the technical team had considered, because no one on the technical team had watched students navigate a debugging session under time pressure with unreliable connectivity.

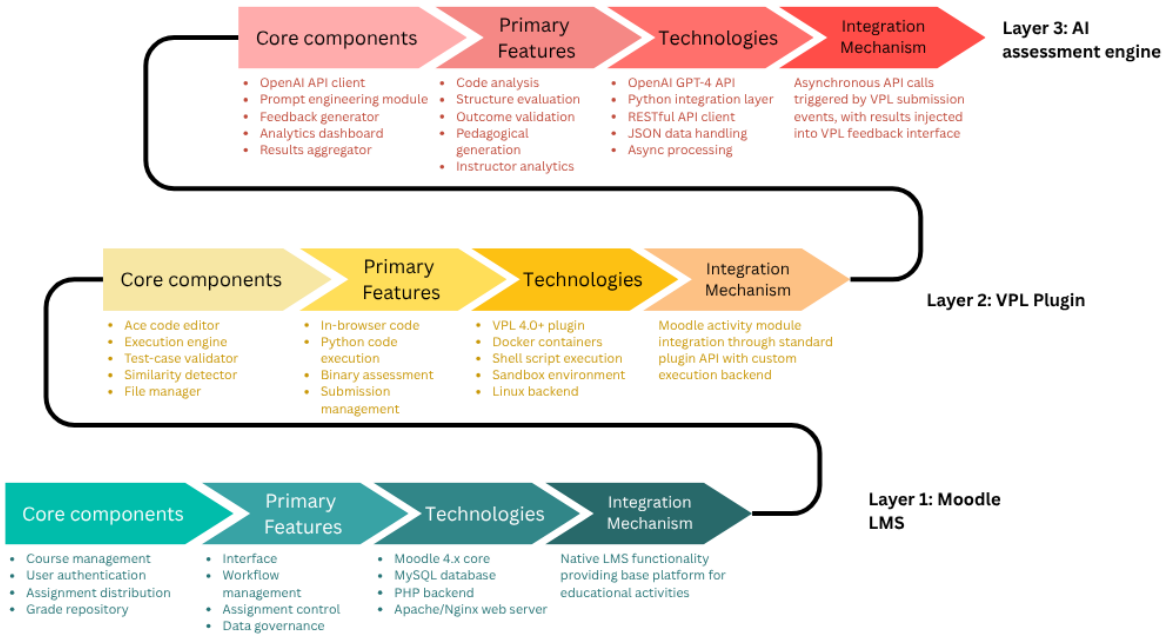
The second critical decision concerned what the instructor dashboard should show. Early prototypes displayed individual student performance in a conventional gradebook layout. Lecturers rejected this not because the data were wrong, but because they were answering the wrong question. At UT's scale, reviewing individual submissions is impossible; what lecturers need to know is whether the same conceptual error is appearing across the cohort — because that pattern signals a gap in instruction, not a gap in any individual student. The revised dashboard therefore aggregates AI diagnostic outputs into a *misconception frequency map*, showing which algorithmic errors appear most prevalently at each submission point across the entire group. This shift — from monitoring students to monitoring learning — represents a qualitative change in what automated assessment can do for an instructor, and it was only possible because the people who would use the dashboard were in the room when it was designed.

© 2026 by Author/s

The third decision addressed the AI assessment engine itself. Rather than allowing the research team to author the prompt templates that would be sent to the GPT-4 API, lecturers co-authored them across two workshops, reviewing and annotating 12 sample AI outputs before approving any prompt structure. What emerged from this process was a template architecture in which each prompt explicitly specifies the assignment's targeted CT component, the expected algorithmic approach, and the criteria by which conceptual misalignment — not just execution failure — would be identified. The pedagogical specificity of these templates is what separates the system's feedback from generic LLM code review: it analyses student work against *this assignment's* learning objectives, not against abstract correctness criteria (See **Figure 2**).

Figure 2

Three-layer architecture specifications



Taken together, these three co-design decisions — dual feedback, misconception mapping, and co-authored prompts — constitute the theoretical core of the system's novelty, and they lead directly to the question of whether the resulting design is, in fact, valid.

Expert validation

A system designed through practitioner participation is not automatically a *valid* system — practitioner knowledge, however valuable, is not the same as domain expertise applied through a standardised evaluation protocol. The expert validation phase therefore serves a distinct function: it subjects the co-designed system to independent, structured scrutiny from two specialists who had no involvement in its creation, and whose judgements carry no social obligation toward the design team. The results of that scrutiny, summarised in **Table 6,** are unambiguous in their main finding and instructive in their one qualification.

Table 6

Expert validation results

Validation Dimension	Items	V Range	% Valid	% Feasibility	Decision
Content & Material	5	0.75–1.00	100%	90%	Valid — proceed to pilot
Media & Interface Design	5	0.75–1.00	100%	90%	Valid — minor UI annotation noted
Functionality & Practicality	4	0.50–1.00	75%	80%	1 indicator below threshold — revised pre-pilot

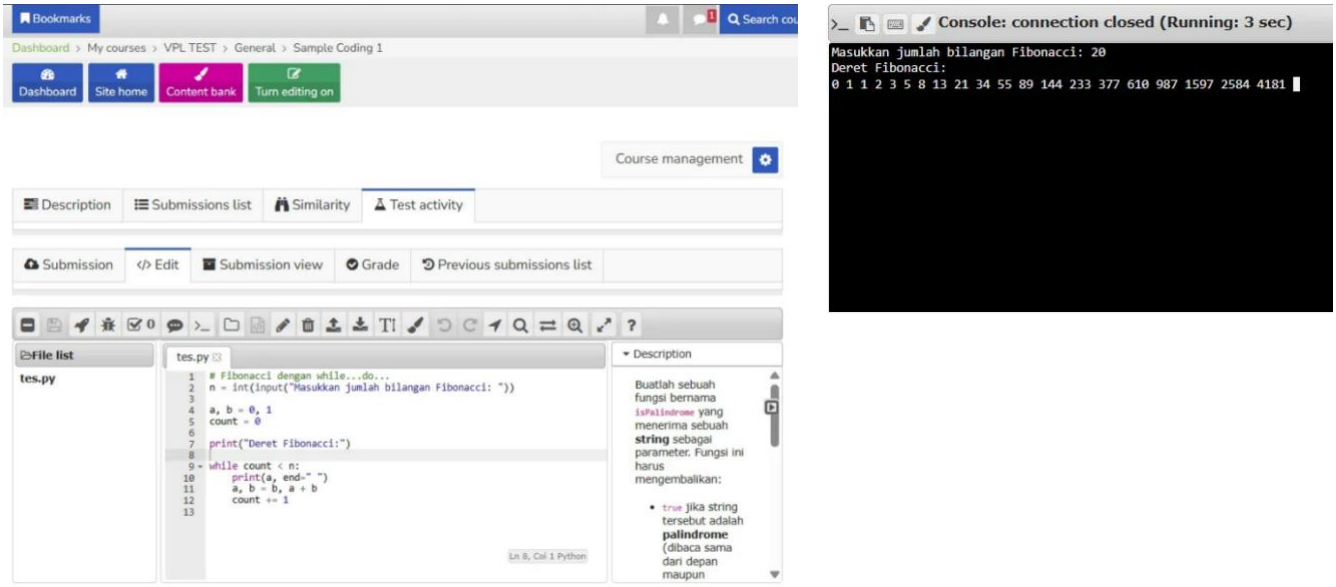
The main finding is that the system's content accuracy and interface design are both valid, and both are valid to the same degree. Every indicator in the Content and Material dimension reached $V \geq 0.75$ (range 0.75–1.00; feasibility 90%), as did every indicator in the Media and Interface Design dimension (identical range and feasibility). The fact that these two dimensions — evaluated from different perspectives by different specialists

© 2026 by Author/s

— produced identical profiles is not a coincidence. It reflects something the co-design process was specifically designed to achieve: coherence between what the system teaches and how it presents that teaching (See **Figure 3**).

Figure 3

VPL interface integration with moodle



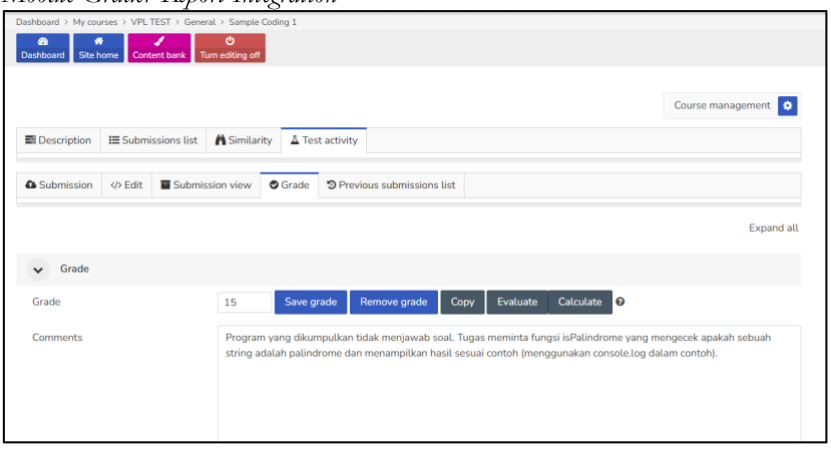
When the pedagogical expert rates the AI feedback as conceptually accurate, and the educational technology expert rates the interface as cognitively accessible, and both arrive independently at the same validity level, it confirms that the system works as an integrated whole rather than as two components that happen to be packaged together.

The one qualification is in the Functionality and Practicality dimension, where one indicator — Moodle multi-user authentication stability — fell below threshold at V = 0.50, reducing the dimension's valid indicator rate to 75% and its overall feasibility to 80%. It would be misleading, however, to treat this as evidence of a design failure. Both validators explicitly classified the limitation as a *server-configuration issue* rather than a conceptual or architectural problem, meaning the system's design logic is sound but one component of its deployment environment required technical remediation. That remediation was completed before the classroom deployment commenced, and the technical validator confirmed its resolution in a follow-up review session. The overall feasibility rating of 84%, classified as Feasible on the adopted scale, is therefore a conservative estimate of the system's readiness: the validated version that students in the deployment group actually used was the remediated one, not the version that generated the sub-threshold score.

What the validation results establish is that the co-design process produced a system whose pedagogical logic is sound, whose interface supports rather than burdens learning, and whose technical implementation, once a specific and addressable configuration issue was resolved, was ready for initial classroom deployment. The following section documents the observable patterns from that deployment (see Figure 4).

Figure 4

Moodle Grader Report Integration



© 2026 by Author/s

Classroom deployment observations

The classroom deployment ran for six weeks across three assignment cycles, with a comparison group using the conventional Moodle–VPL environment (n = 7) and a deployment group using the AI-enhanced PVL (n = 10). Both groups recorded higher mean scores at T3 than at T1. The score patterns across the three time points are described below as observational records from the deployment period (See **Table 7**).

Table 7

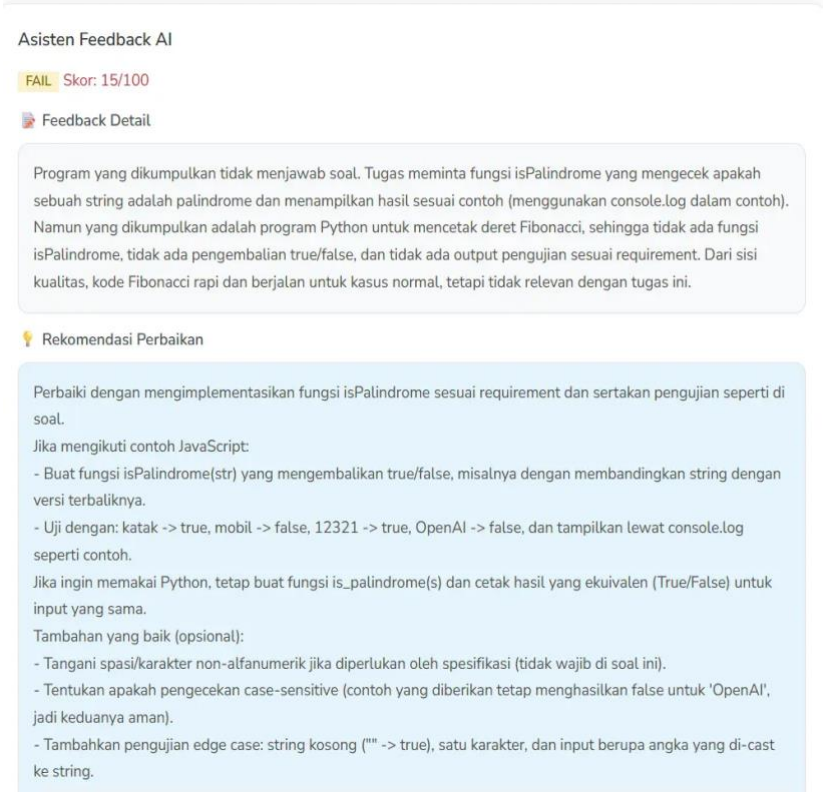
Assignment score descriptive statistics by group and time point (T1–T3)

Group	Task	n	Mean	SD	Min	Max	Trajectory note
Comparison	T1	7	60.00	8.43	50	72	Starting mean: 60.00
Comparison	T2	7	66.28	6.99	58	78	Interval gain: +6.28 points
Comparison	T3	7	68.71	7.04	60	80	Interval gain: +2.43 points
Deployment	T1	10	72.90	15.52	46	96	Starting mean: 72.90
Deployment	T2	10	74.50	28.00	0	97	Wide SD: one submission error (score=0); see note
Deployment	T3	10	79.20	16.27	55	100	Final mean: 79.20; 8 of 10 students showed increasing scores

The comparison group recorded a mean of 60.00 at T1, rising to 66.28 at T2 and 68.71 at T3. The largest single-interval gain occurred between T1 and T2 (+6.28 points), with a smaller gain between T2 and T3 (+2.43 points). Three of the seven students in this group showed consistently increasing scores across all three time points. The deployment group recorded a mean of 72.90 at T1 — 12.90 points above the comparison group at the same time point — rising to 74.50 at T2 and 79.20 at T3. Eight of the ten students in this group showed consistently increasing scores across T1–T3. The difference in starting points between the two groups was substantial and reflects pre-existing differences between the naturally-formed tutorial groups; no causal attribution is made. These trajectory patterns are documented as observational data for future hypothesis development (See **Figure 5**).

Figure 5

AI semantic feedback example



The submitted program does not answer the question. The task requires a function called **isPalindrome** that checks whether a string is a palindrome and displays the result according to the example (using console.log in the example)

Fix it by implementing the **isPalindrome** function according to the requirements and include the test cases as stated in the question.

© 2026 by Author/s

Gain score observations

Table 8 presents a summary of net score change from T1 to T3 for each group. The comparison group recorded a mean gain of +8.71 points across the six-week period.

Table 8
Gain score summary (T1 to T3)

Comparison	Mean T1	Mean T3	Gain
Control	60.00	68.71	8.71
Experiment	72.90	79.20	6.30
Between-groups at T1	60.00	72.90	12.90
Between-groups at T3	68.71	79.20	10.49

The deployment group recorded a mean gain of +6.30 points from a higher starting baseline. These figures are reported as descriptive observations only. The difference in gain scores is not interpreted as evidence of differential learning outcomes; the baseline gap between groups at T1 (12.90 points), the small group sizes, and the absence of random assignment make any comparative effectiveness claim unwarranted. The gain scores are recorded here to document the observable patterns in this initial deployment for reference in future adequately powered studies.

DISCUSSION

The four research questions in this study were sequentially dependent: the needs profile from Phase 1 determined the design priorities addressed in Phase 2; the system produced in Phase 2 was the one subjected to expert validation in Phase 3; and the validated system was the one deployed in Phase 4, whose observable score patterns are discussed here. The discussion that follows preserves this structure, treating each finding not in isolation but in relation to the phase that preceded it and the question it was designed to answer (Djam'an et al., 2025; Sies et al., 2025; Sudirman et al., 2026).

The needs analysis finding that most directly shapes the rest of the study is not any single indicator mean but the pattern formed by the five indicators with gap scores exceeding −2.0: debugging tools, code visualisation, auto-save, code execution, and real-time feedback. These five features are not enrichment capabilities that would improve an already-functioning environment; they are foundational affordances without which programming practice cannot sustainably begin, and the current environment at Universitas Terbuka fails to provide all five reliably. This reframes the design problem in a way that the VPL literature — which predominantly discusses virtual programming laboratories in terms of feature enhancement rather than feature provision — does not adequately address (Ramos et al., 2021; Rodríguez-del-Pino & Hernández-Figueroa, 2022; Isnawan et al., 2025). The IPA gap score of −2.23 for basic code execution, recorded at a university where 78.5% of enrolled students have never previously written a line of Python, is not a preference statement; it is empirical evidence that a substantial proportion of learners are being stopped at the threshold of the learning encounter before any algorithmic reasoning can begin. This pattern is consistent with the updated cognitive load framework articulated by Sweller et al. (2019), which distinguishes between extraneous load — arising from poorly designed learning environments — and germane load — the cognitive resource available for schema construction and genuine learning. When environment failures consume working memory through installation errors, connectivity problems, and lost work, germane load approaches zero, and with it the student's capacity to develop the computational thinking competencies that constitute the course's stated learning objectives (Berssanette & de Francisco, 2022). The domain-level data make this dynamic visible in a second way: Domain C (Computational Thinking, M = 2.82) returned the lowest aggregate need score not because students undervalue higher-order skills but because students who cannot yet reliably execute their code have not reached the cognitive availability to identify CT development as a discrete need they can articulate and rate. The AI-PVL was designed specifically to interrupt this cycle — removing the extraneous load of environment management so that germane load becomes available for algorithmic understanding — and the design decisions that produced the system's specific architecture can be traced directly back to this diagnosis.

The co-design findings illuminate a distinction that the participatory design literature has named but rarely demonstrated with concrete examples: the difference between practitioners who *confirm* researcher decisions and practitioners who *determine* them (Cumbo & Selwyn, 2022; Coenraad et al., 2022). In this study, the two design innovations that most distinguish the AI-PVL from comparable VPL implementations — the dual-feedback interface and the misconception frequency map — were neither anticipated by the research team nor derivable from the prior literature. Both emerged from lecturers exercising design authority in response to problems they

 © 2026 by Author/s

had observed over multiple semesters of teaching the Algorithm and Programming course at UT. The dual-feedback interface arose from a specific pedagogical concern that no technical specification could have captured: that students in this institutional context use the test-case output not only as an assessment signal but as a credibility anchor, the objective, verifiable confirmation that their code runs at all, and that AI semantic analysis presented without this anchor would be treated with scepticism because learners would have no prior basis for calibrating their trust in the system. This concern is empirically supported by Kinder et al.'s (2025) randomised trial with 269 pre-service teachers, which found that adaptive LLM feedback was associated with stronger learning outcomes than static feedback, but only under conditions where learners had a basis for evaluating the feedback's reliability. The dual-feedback architecture provides exactly that basis — not by explaining the AI's reasoning but by ensuring that students can always verify whether their code runs before evaluating what the AI says about whether it solves the right problem. The misconception frequency map addresses a second problem that the lecturers named precisely and the literature has identified but rarely operationalised: the difference between individual-level and cohort-level assessment intelligence. Yan et al. (2024), in their systematic scoping review of 118 studies on LLM applications in education, identify the generation of cohort-level diagnostic intelligence as one of the most significant unexplored potentials of LLM-based assessment systems — information that could inform instructional decisions at the course level rather than the student level. The misconception map addresses this gap directly, aggregating AI diagnostic outputs across all submissions at each time point to show lecturers not which students are struggling but what the cohort is systematically misunderstanding, because that pattern is what informs instructional decisions at scale. Both innovations were produced through what Coenraad et al. (2022) term co-invention rather than co-validation: design features that emerged from the encounter between research knowledge and practitioner knowledge in a structured deliberative setting, and that neither party could have specified independently.

The expert validation results communicate through their structure as much as through their overall figure. The 84% overall feasibility rating is best understood not as a single number but as the composite of three dimensionally distinct assessments that converge in an analytically significant way. Content and Material validity (90% feasibility, V range 0.75–1.00) and Media and Interface Design validity (90% feasibility, identical V range) achieved identical profiles from two validators who assessed entirely different dimensions of the system from entirely different professional vantage points. When an educational technology expert and a programming education specialist independently assign identical validity levels to a system's pedagogical accuracy and its interface design respectively, it indicates that the system's content and its presentation were developed in genuine alignment rather than as separately optimised components assembled after the fact (Wilson et al., 2012). This coherence is the expected — though not guaranteed — outcome of a co-design process in which the same practitioners who specified what the semantic feedback should convey also specified how it should be displayed. The validation data therefore confirm that the co-design process achieved its intended design goal: the two dimensions of the system that bear most directly on whether lecturers will integrate it into their teaching and whether students will engage with its output are both independently certified as sound. The one qualification, the Functionality and Practicality dimension (80% feasibility), is attributable entirely to a single sub-threshold indicator ($V = 0.50$) concerning Moodle multi-user authentication stability under concurrent load — a server-configuration issue that both validators explicitly characterised as a deployment concern rather than a conceptual design flaw. The underlying integration architecture was judged sound by both validators; what required remediation was one component of its deployment environment, and that remediation was confirmed resolved before the classroom deployment commenced. For institutions considering analogous deployments, this profile carries a concrete practical message: the system's pedagogical logic and interface design have been independently validated; the remaining adoption barrier is institutional IT capacity for server provisioning and load configuration, which is significant but not novel for any institution already operating Moodle at scale.

The classroom deployment produced observable score patterns that are consistent with the system functioning as designed, though no effectiveness claims are drawn from these data. The deployment group entered the study with a substantially higher mean score than the comparison group (72.90 vs. 60.00), a difference that reflects pre-existing group characteristics rather than any intervention effect. Both groups improved across the six-week period. The deployment group showed a higher proportion of students with consistently increasing scores across all three time points (8 of 10) compared with the comparison group (3 of 7). These patterns are noted as observational findings that may inform the design of future hypothesis-testing studies.

The rationale for investing in a future adequately powered study rests on what the existing literature has established about the limitations of test-case-based feedback. Research has documented a persistent ceiling on the learning gains that test-case-based systems produce: feedback that confirms only whether code output matches an expected value gives students no information about what their algorithmic thinking got wrong, generating a pattern where learners optimise for test-case passage rather than for conceptual understanding

© 2026 by Author/s

(Messer et al., 2024; Keuning et al., 2018). The AI-PVL was designed specifically to address this limitation by providing semantic feedback at the conceptual level. Whether this approach produces measurable learning gains in the present institutional context is a question the current deployment cannot answer; it constitutes the primary hypothesis for future investigation, informed by evidence that conceptually specific feedback can outperform static automated responses under controlled conditions (Kinder et al., 2025; Yan et al., 2024).

Limitations

Six constraints materially limit what this study has established. First, the initial classroom deployment involved 17 students across two naturally-formed groups with a substantial baseline difference (12.90 points at T1). The small group sizes and absence of random assignment mean that no effectiveness claims can be drawn from this deployment. The descriptive observations reported here should be treated as design-informing data for a future hypothesis-testing study, not as evidence of the system's impact on learning outcomes. Second, all data were collected within a single course at a single institution, with its specific Moodle configuration, open-distance delivery model, and Indonesian student demographic profile; the design principles that emerged from the PDR process are offered as transferable, but whether they produce equivalent effects in institutions with different LMS infrastructures, student preparation levels, subject disciplines, or instructional cultures is an empirical question that multi-site, multi-course investigation must answer (Cumbo & Selwyn, 2022). Third, the six-week deployment period is too short to determine whether the assignment score patterns observed reflect durable learning development. Wing (2006) argues that genuine CT development is evidenced by the transferability of skills across problem domains — a quality that a three-assignment pilot cannot assess. Longitudinal follow-up across multiple semesters is required before claims about lasting CT development can be made. Fourth, the consistency and reliability of GPT-4-generated feedback was not independently audited in this study. While the co-authored prompt templates were designed to constrain the model's outputs to assignment-specific criteria, LLM responses can vary across identical or near-identical inputs, and the extent to which feedback quality remained stable across all 30 student submissions (10 students × 3 assignments) is unknown. Future studies should incorporate systematic human evaluation of a sample of AI-generated feedback outputs to verify alignment with intended pedagogical criteria (Yan et al., 2024). Fifth, the possibility of a novelty or engagement effect in the deployment group cannot be excluded. Students using a newly developed AI-enhanced system may have invested greater effort or attention simply because the environment was novel, rather than because the semantic feedback was pedagogically effective. Without a condition that controlled for novelty — for example, a group using an AI system that provided generic rather than assignment-specific feedback — the contribution of the semantic feedback mechanism itself cannot be isolated from the motivational effect of a new tool. Sixth, the system's semantic assessment layer depends entirely on the OpenAI GPT-4 API — a commercial service whose pricing, availability, and data governance policies sit outside institutional control; for distance learning institutions in regions with unreliable internet connectivity, constrained budgets, or data sovereignty regulations, this dependency represents an adoption barrier not resolved by the anonymisation protocols implemented here, and future development should investigate open-weight local LLM deployment as an alternative architecture, with recent evidence suggesting that appropriately prompted open-source models can provide programming feedback of sufficient quality for introductory course contexts (Koutcheme et al., 2024).

Implications

For distance learning institutions, the most immediately actionable implication is architectural: the three-layer integration model described here — Moodle as the institutional backbone, VPL as the execution environment, and an LLM as the semantic assessment layer — preserves existing gradebook workflows, assignment management, and instructor practices rather than replacing them, directly addressing the workflow disruption that Awidi and Paynter (2022) identify as the primary driver of technology adoption resistance in higher education settings comparable to UT. For programming education researchers, this study raises mechanistic questions that its design cannot answer: whether semantic feedback changes students' problem-solving strategies, reduces ineffective revision cycles, or alters metacognitive monitoring — questions that Yan et al. (2024) identify as the most significant gap in the LLM-in-education literature and that require learning analytics data, specifically revision histories, time-on-task measures, and submission patterns, that were not collected in this pilot but should be a design priority for subsequent studies.

For educational technology developers more broadly, the co-design methodology demonstrated here — particularly the finding that lecturers co-invented design features that neither practitioners nor researchers could have anticipated independently — provides a replicable template for developing AI-enhanced educational tools that are institutionally grounded and pedagogically sound (Barab & Squire, 2004; Fishman et al., 2013); this methodology is most urgent in contexts where the practitioners who will use a system possess contextual

 © 2026 by Author/s

knowledge — about student behaviour, institutional constraints, and workflow dependencies — that no amount of literature review can substitute for. This framing is consistent with emerging scholarship that positions AI not merely as an assessment tool but as a component of a structured learning and assessment ecosystem — one in which questions of accuracy, validity, and ethical deployment must be addressed at the design level rather than retrofitted after implementation (Raza et al., 2025). The present study's approach of embedding AI semantic assessment within existing institutional infrastructure, co-designed with practitioners and validated through independent expert review, represents one operationalisation of this ecosystem model. Recent work on AI-enhanced micro-credentials further illustrates how generative AI can be integrated into the design, development, and delivery of learning experiences in ways that improve both efficiency and accessibility — a direction relevant to open distance learning institutions such as UT that serve large and geographically dispersed student populations (Kovilpillai et al., 2025).

Future research priorities include a hypothesis-testing study with adequate statistical power ($n \geq 42$ per condition, random assignment, and baseline equivalence verification), a comparative study of local versus API-dependent LLM deployment for programming feedback, and a longitudinal study examining whether assignment score patterns observed in this initial deployment are sustained and transferable across multiple semesters.

CONCLUSION

This study set out to address a problem that is simultaneously technical, pedagogical, and institutional: how to provide distance learning students with programming assessment that goes beyond whether their code runs, without dismantling the Moodle infrastructure that universities have already built. Through a PDR process involving six course lecturers as co-designers, the study developed, validated, and conducted an initial proof-of-concept deployment of an AI-Enhanced Python Virtual Laboratory that embeds GPT-4 semantic assessment directly within the Moodle–VPL environment. The system does not require students or instructors to adopt a new platform; it extends what already exists.

Four findings emerge from the study, each corresponding to a research question. First, the needs analysis ($N = 65$) revealed that distance learners at Universitas Terbuka are not struggling with a lack of features but with a lack of the most foundational ones — reliable code execution, debugging, auto-save, and real-time feedback — whose absence prevents learning from beginning at all. Second, the co-design process demonstrated that practitioner knowledge is not simply a source of requirements but a generative force: the dual-feedback interface, misconception frequency map, and co-authored prompt templates — the system's three most distinctive innovations — all emerged from lecturers exercising design authority, not from the research team's technical expertise. Third, expert validation confirmed that the system is content-valid and interface-valid (84% overall feasibility; Aiken's $V \geq 0.75$ for 90% of indicators), with the single sub-threshold indicator reflecting a resolved server-configuration issue rather than a design flaw. Fourth, the initial classroom deployment produced an observable score pattern in which a higher proportion of deployment group students showed consistently increasing scores across all three assignment cycles (8 of 10) compared with the comparison group (3 of 7). This pattern is documented as a descriptive observation from the proof-of-concept deployment, not as evidence of effectiveness.

Taken together, these findings establish three things: that the infrastructure gap and assessment gap facing distance programming learners at Universitas Terbuka are real and empirically documented; that a co-designed AI-enhanced system addressing both gaps is technically feasible and independently validated; and that an initial classroom deployment produced score patterns consistent with the system functioning as intended. What remains to be established — through a future adequately powered study with random assignment and baseline equivalence — is whether the system produces measurable improvements in learning outcomes. As a proof-of-concept, this study does not answer that question, but it provides the architectural model, the validated instrument, and the observational baseline that make answering it possible.

Acknowledgements

The authors gratefully acknowledge the support of Universitas Terbuka, through funding from the Kemendiktisainstek (Kementerian Pendidikan Tinggi, Sains dan Teknologi) programme, which made this research possible. We also received support from the Enhancing Quality Education for International University Impacts and Recognition (EQUITY) Program — THE University Impact Ranking 2025. We extend our sincere appreciation to the six course lecturers of MATA4110 (Algorithm and Programming) at the Faculty of Mathematics and Natural Sciences, Universitas Terbuka, whose expertise, time, and commitment as co-designers were indispensable to every phase of this study. Their practitioner knowledge shaped the system's architecture in ways that no technical specification could have anticipated. We also thank the 65 students who participated in

© 2026 by Author/s

the Phase 1 needs analysis and the 17 students who participated in the Phase 5 classroom deployment for their willingness to contribute their data and experiences to this research.

Funding

This research was supported by Universitas Terbuka through funding from the Kemendiktisainstek (Kementerian Pendidikan Tinggi, Sains dan Teknologi) programme (Ministry of Higher Education, Science and Technology of the Republic of Indonesia). The funding body had no role in the design of the study, the collection, analysis, or interpretation of data, or in the writing of the manuscript.

Ethical Statement

This study was conducted in accordance with the ethical principles of Universitas Terbuka's Institutional Ethics Committee. All participants — including course lecturers, student informants, expert validators, and students in the classroom deployment phase — provided written informed consent prior to participation. Consent forms disclosed the purpose of the study, the voluntary nature of involvement, the right to withdraw without consequence, and the transmission of anonymised code submissions to the OpenAI GPT-4 API for semantic feedback generation. Student data were anonymised prior to analysis and no individual participant is identifiable in any reported result.

Competing interests

The authors declare that there are no competing interests regarding the publication of this article. The research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The funding body (Kemendiktisainstek / Universitas Terbuka) had no role in study design, data collection or analysis, decision to publish, or preparation of the manuscript.

Author contributions

This study required collaborative expertise across participatory design research, system development, data analysis, and theoretical interpretation. Author contributions are described using the CRediT (Contributor Roles Taxonomy) framework: MN: conceptualisation, methodology, investigation (phase 1–5 data collection), writing – original draft, writing – review & editing, and project administration; SS: conceptualisation, methodology, formal analysis, writing – review & editing, and supervision; TK: methodology, software (system architecture and vpl integration), validation, writing – review & editing; SN: formal analysis (qualitative thematic analysis, phase 1–2), data curation, writing – review & editing; MGI: conceptualisation, writing – review & editing, resources (expert review of methodology and theoretical framing). All authors have read and approved the final version of the manuscript. * Corresponding author.

AI disclosure

The authors used OpenAI GPT-4 as a functional component of the AI-Enhanced Python Virtual Laboratory (AI-PVL) system developed and evaluated in this study. Specifically, GPT-4 was integrated via API as the semantic assessment engine (Layer 3) to generate diagnostic feedback on student code submissions. This use is described in full in the Method section (Section 2.3.5) and constitutes the subject of the research, not an assistive tool in its preparation.

Regarding manuscript preparation: the authors used AI-assisted language tools for grammar checking and phrasing refinement during manuscript revision. All intellectual content, argumentation, data interpretation, and conclusions are solely the work of the authors. The authors take full responsibility for the accuracy and integrity of the manuscript as submitted.

Biographical Sketches

Mery Noviyanti is a Professor in the Department of Mathematics Education, Faculty of Teacher Training and Education, Universitas Terbuka, Indonesia. Her research interests include mathematics education in open and distance learning contexts, e-learning-based instruction, mathematical knowledge for teaching, early childhood mathematics, and AI-enhanced pedagogy. Her recent publications include studies on non-Euclidean geometry teaching through e-learning based on the Theory of Didactic Situations, gender equity in STEM distance learning, and dialogic learning in AI-integrated education. She is the corresponding author of this study. ORCID: 0000-0003-4729-7959.

Sudirman Sudirman is an Associate Professor in the Department of Mathematics Education at Universitas Terbuka, Indonesia. His research interests include MOOC, augmented reality, geometry teaching, geometric

 © 2026 by Author/s

thinking, and ethnomathematics. He has published more than 62 articles in national and international journals covering mathematical connections, STEAM in mathematics education, participatory design research, and didactical engineering. He served as the conceptual lead and supervisor for this study. ORCID: 0000-0002-1696-5160.

Thesa Kandaga is a lecturer in Mathematics Education at Universitas Terbuka, Indonesia. His research interests include mathematics education, mathematical literacy, brain-based learning, and technology-enhanced mathematics instruction. His published work includes studies on improving students' mathematical literacy, culturally contextual e-module design, and AI applications in mathematics learning. In this study, he contributed to the design, development, and technical integration of the VPL environment. ORCID: 0009-0002-2596-2508.

Suci Nurhayati is a lecturer in Mathematics Education at Universitas Terbuka, Indonesia. She holds a Master's Degree in Mathematics Education from the Indonesia University of Education (UPI) and has extensive prior teaching experience at the junior high school, senior high school, and vocational school levels. Her research interests include distance mathematics education, mathematical literacy, gender equity in STEM distance learning, and online tutorial evaluation. In this study, she contributed to qualitative data analysis. ORCID: 0009-0002-4235-5478.

Muhamad Galang Isnawan is a lecturer in the Department of Mathematics Education at Universitas Nahdlatul Wathan Mataram, Mataram, Indonesia. His research interests include didactical design research, STEM education, lesson study, and participatory approaches to educational technology development. He has published in the European Journal of STEM Education and other international outlets, with work on TPACK-incorporated learning devices and STEAM-based courses for prospective mathematics teachers. He contributed theoretical framing, methodological review, and manuscript critique to this study. ORCID: 0000-0002-3799-3435.

REFERENCES

Acar, E., Deiri, Y., & Yigit, F. (2025). A focused review of artificial intelligence in education: Evolution and challenges. *Journal of Interdisciplinary Research in Artificial Intelligence and Society*, *1*(1), Article 3. https://doi.org/10.20897/jirais/17640

Aiken, L. R. (1985). Three coefficients for analyzing the reliability and validity of ratings. *Educational and Psychological Measurement*, *45*(1), 131–142. https://doi.org/10.1177/0013164485451012

Akojie, P., Laroche, I., & Schumacher, J. (2022). Moving from face-to-face instruction to virtual instruction in the COVID-19 pandemic: Narratives of K-12 teachers. *American Journal of Qualitative Research*, 6(1), 59–72. https://doi.org/10.29333/ajqr/11457

Amka, A., & Dalle, J. (2022). The satisfaction of the special need' students with e-learning experience during COVID-19 pandemic: A case of educational institutions in Indonesia. *Contemporary Educational Technology*, 14(1), ep334. https://doi.org/10.30935/cedtech/11371

Awidi, I. T., & Paynter, M. (2022). An evaluation of the impact of digital technology innovations on students' learning: Participatory research using a student-centred approach. *Technology, Knowledge and Learning*, *29*, 65–89. https://doi.org/10.1007/s10758-022-09619-5

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. *Journal of the Learning Sciences*, *13*(1), 1–14. https://doi.org/10.1207/s15327809jls1301_1

Berssanette, J. H., & de Francisco, A. C. (2022). Cognitive load theory in the context of teaching and learning computer programming: A systematic literature review. *IEEE Transactions on Education*, *65*(3), 440–449. https://doi.org/10.1109/TE.2021.3127215

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Campbell, D. T., & Stanley, J. C. (1963). *Experimental and quasi-experimental designs for research*. Rand McNally.

Chan, A. T. S., Cao, J., & Liu, C.-K. (2004). VPL: An online distance learning platform for virtual programming laboratory. *International Journal of Computer Processing of Oriental Languages*, *17*(1), 41–59.

Coenraad, M., Palmer, J., Eatinger, D., Weintrop, D., & Franklin, D. (2022). Using participatory design to integrate stakeholder voices in the creation of a culturally relevant computing curriculum. *International Journal of Child-Computer Interaction*, *31*, Article 100353. https://doi.org/10.1016/j.ijcci.2021.100353

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Lawrence Erlbaum Associates.

Cumbo, B., & Selwyn, N. (2022). Using participatory design approaches in educational research. *International Journal of Research & Method in Education*, *45*(1), 60–72. https://doi.org/10.1080/1743727X.2021.1899167

Djam'an, N., Samah, N. A., & Nasrullah, N. (2025). Post-pandemic online learning in Indonesian mathematics education: The perception-achievement relationship. *International Journal of Didactic Mathematics in Distance Education*, *3*(1), 1–21. https://doi.org/10.33830/ijdmde.v3i1.13088

© 2026 by Author/s

European Parliament. (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data (General Data Protection Regulation)*. Official Journal of the European Union, L 119, 1–88. https://eur-lex.europa.eu/eli/reg/2016/679/o

Fishman, B., Penuel, W. R., Allen, A., Cheng, B. H., & Sabelli, N. (2013). Design-based implementation research: An emerging model for transforming the relationship of research and practice. *National Society for the Study of Education Yearbook*, *112*(2), 136–156.

Impey, C., Wenger, M., & Austin, C. (2024). Comparative evaluation of large language models in educational assessment: Performance analysis across multiple contexts. *Computers & Education: Artificial Intelligence*, *5*, Article 100165. https://doi.org/10.1016/j.caeai.2023.100165

Isnawan, M. G., Alsulami, N. M., Rasilah, R., Sukarma, I. K., & Lavicza, Z. (2025). Didactic design research through lesson study activities: STEM-based courses for representative abilities of prospective mathematics teachers. *European Journal of STEM Education*, *10*(1), Article 12. https://doi.org/10.20897/ejsteme/16758

Keuning, H., Jeuring, J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education*, *19*(1), 1–43. https://doi.org/10.1145/3231711

Khlaif, Z. N., Sanmugam, M., Joma, A. I., Odeh, A., & Barham, K. (2024). Factors influencing teachers' technostress experienced in using generative artificial intelligence chatbots: A qualitative study. *International Journal of Human–Computer Interaction*, 1–16. https://doi.org/10.1080/10447318.2024.2316614

Kinder, A., Briese, F. J., Jacobs, M., Dern, N., Glodny, N., Jacobs, S., & Leßmann, S. (2025). Effects of adaptive feedback generated by a large language model: A case study in teacher education. *Computers and Education: Artificial Intelligence*, *8*, Article 100349. https://doi.org/10.1016/j.caeai.2024.100349

Kovilpillai, J. J. S., Singh, A. D., Hamdan, A., McKenna, K., & Raza, F. A. (2025). AI enhanced micro-credentials for efficiency and accessibility: Using generative AI to improve the design, development, and delivery of micro-credentials. In *Integrating micro credentials with AI in open education*. https://doi.org/10.4018/979-8-3693-5488-9.ch008

Koutcheme, C., Dainese, N., Sarsa, S., Hellas, A., Leinonen, J., & Denny, P. (2024). Open-source language models can provide feedback: Evaluating LLMs' ability to help students using GPT-4-as-a-judge. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education* (Vol. 1, pp. 52–58). Association for Computing Machinery. https://doi.org/10.1145/3649217.3653612

Leon, A. C., Davis, L. L., & Kraemer, H. C. (2011). The role and interpretation of pilot studies in clinical research. *Journal of Psychiatric Research*, *45*(2), 626–629. https://doi.org/10.1016/j.jpsychires.2010.10.008

Messer, M., Brown, N. C. C., Kölling, M., & Shi, M. (2024). Automated grading and feedback tools for programming education: A systematic review. *ACM Transactions on Computing Education*, *24*(1), 1–43. https://doi.org/10.1145/3636515

Muller, M., & Druin, A. (2012). Participatory design: The third space in HCI. In J. Jacko (Ed.), *Human–computer interaction handbook* (3rd ed., pp. 1125–1153). CRC Press.

Novianti, I., Krisnadi, E., Kandaga, T., Nurmawati, & Sudirman, S. (2026). Personalization imperative: Unpacking student-AI relationships through a mixed-methods lens in Indonesian higher education. *International Journal of Learning Teaching and Educational Research*, *25*(1), 750–778. https://ijlter.myres.net/index.php/ijlter/article/view/2685

OECD. (2013). *The OECD privacy guidelines*. Organisation for Economic Co-operation and Development. https://www.oecd.org/digital/ieconomy/privacy-guidelines.htm

Papadakis, S., & Orfanakis, V. (2018). Comparing novice programming environments for use in secondary education: App Inventor for Android vs. Alice. *International Journal of Technology Enhanced Learning*, *10*(1–2), 44–72. https://doi.org/10.1504/IJTEL.2018.088333

Pankiewicz, M., & Baker, R. S. (2024). Navigating compiler errors with AI assistance: A study of GPT hints in an introductory programming course. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education* (Vol. 1, pp. 94–100). Association for Computing Machinery. https://doi.org/10.1145/3649217.3653608

Ramos, M. A., Giménez, R., Casañ, M. J., & Fernández-Fabuel, D. (2021). Perceptions about the usability of the Moodle's virtual programming lab (VPL) in subjects of programming. *Applied Sciences*, *11*(14), Article 6564. https://doi.org/10.3390/app11146564

Raza, F. A., Singh, A. D., Kovilpillai, J. J. S., Hamdan, A., & Rajaratnam, V. (2025). Safeguarding integrity in AI-enhanced education: Stakeholder perspectives on accuracy, validity, and ethics in ASEAN. *European Journal of STEM Education*, *10*(1), Article 22. https://doi.org/10.20897/ejsteme/17307

Rodríguez-del-Pino, J. C., & Hernández-Figueroa, Z. J. (2022). VPL: Virtual programming lab for Moodle. *SoftwareX*, *18*, Article 101090. https://doi.org/10.1016/j.softx.2022.101090

 © 2026 by Author/s

Sies, A. A., Román, G. S., & Juárez-López, J. A. (2025). The impact of augmented reality on the learning of polyhedral: An approach based on didactical engineering and instrumental genesis. *Polyhedron International Journal in Mathematics Education*, *3*(2), 86–109. https://doi.org/10.59965/pijme.v3i2.263

Sudirman, S., Yaniawati, P., Mellawaty, & Indrawan, R. (2021). Augmented reality application: What are the constraints and perceptions of the students during the COVID-19 pandemic's 3D geometry learning process? *Journal of Physics: Conference Series*, *1783*(1), Article 012007.

Sudirman, Rodríguez-Nieto, C. A., Hidayat, R., Isnawan, M. G., Pauzan, M., Yumiati, Martadiputra, B. A. P., & Faizah, S. (2026). Operationalizing didactical situation-based online learning to support eighth-grade students' mathematical reasoning and understanding in geometry: Participatory design research. *Journal on Mathematics Education*, *17*(1), 43–68. https://doi.org/10.22342/jme.v17i1.pp43-68

Sudirman, S., Martadiputra, B. A. P., Faizah, S., Rodríguez-Nieto, C., Soeharto, S., Lavicza, Z., Isnawan, M. G., Sembiring, M. G., Runisah, R., & Suprihatin, R. J. (2026). Integrating STEAM-based MOOC as a supplement to didactical and pedagogical practices in school mathematics: A design-based research. *Journal of Technology and Science Education*, *16*(1), Article 125. https://doi.org/10.3926/jotse.3633

Sullivan, G. M., & Feinn, R. (2012). Using effect size — or why the p value is not enough. *Journal of Graduate Medical Education*, *4*(3), 279–282. https://doi.org/10.4300/JGME-D-12-00156.1

Sultan, Y., Dautova, G., & Dalle, J. (2025). Examining the relationship among artificial intelligence literacy, cultural literacy, and intercultural communication proficiency of philology students. *Journal of Ethnic and Cultural Studies*, 12(5), 345–362. https://doi.org/10.29333/ejecs/2839

Sweller, J., van Merriënboer, J. J. G., & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review*, *31*(2), 261–292. https://doi.org/10.1007/s10648-019-09465-5

Tan, M., Gao, S., Chen, X., Kok, G., Xiao, W., Zhang, D., & Xie, Q. (2025). A systematic review of automated feedback systems for programming education: Opportunities, challenges, and future directions. *ACM Transactions on Computing Education*, *25*(1), 1–38. https://doi.org/10.1145/3654676

Thabane, L., Ma, J., Chu, R., Cheng, J., Ismaila, A., Rios, L. P., Robson, R., Thabane, M., Giangregorio, L., & Gafni, A. (2010). A tutorial on pilot studies: The what, why and how. *BMC Medical Research Methodology*, *10*, Article 1. https://doi.org/10.1186/1471-2288-10-1

Wardani, E. K., Retnasari, L., & Ghiffari, M. A. N. (2025). AR snakes and ladders game innovation to promote learning to live together in elementary schools. *International Journal of Mathematics and Sciences Education*, *3*(2), 57–70. https://doi.org/10.59965/ijmsed.v3i2.264

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Yan, L., Sha, L., Zhao, L., Li, Y., Martinez-Maldonado, R., Chen, G., Li, X., Jin, Y., & Gašević, D. (2024). Practical and ethical challenges of large language models in education: A systematic scoping review. *British Journal of Educational Technology*, *55*(1), 90–112. https://doi.org/10.1111/bjet.13370

Yıldırım, Z., Öztürk, G., Şahinoğlu, H. Ü., & Yıldırım, E. (2024). An ethical use guideline for AI-supported assignments: Design and effectiveness analysis. *ISophos: International Journal of Information, Technology and Philosophy*, 7(13), 34–50

Zabasta, A., Kunicina, N., Chaiko, Y., Zhiravecka, A., Ribickis, L., Patlins, A., Arefyev, A., Ziravecka, K., Galkina, A., Zabasta, I., & Kunicins, A. (2024). Shared modeling as a technology of e-learning in STEM higher education: Experience and prospects. *Sustainability*, *16*(5), Article 2095. https://doi.org/10.3390/su16052095

Zacharis, G., & Papadakis, S. (2025). Can AI grade like a human? Validity, reliability, and fairness in university coursework assessment. *Educational Process: International Journal*, *19*, Article e2025591. https://doi.org/10.22521/edupij.2025.19.591

© 2026 by Author/s