International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(REVIEW ARTICLE)

Check for updates

PROVEX: A pressure-based learning system for evidence-driven mastery in computer science education

Sankalp Tripathi, Pranita Pradeep Potghan, Ojas Amol Umate * and Shreeya Pratap Salunkhe

Department of Information Technology, MIT School of Computing, MIT-ADT University, Pune, India.

International Journal of Science and Research Archive, 2026, 19(01), 1098-1104

Publication history: Received on 17 March 2026; revised on 25 April 2026; accepted on 27 April 2026

Article DOI: https://doi.org/10.30574/ijsra.2026.19.1.0877

Abstract

The proliferation of passive learning tools, AI-generated solutions, and hint-driven coding platforms has created a critical gap between perceived and actual technical competence among learners. This paper presents PROVEX, a pressure-based adaptive learning system designed for students in computer science and information technology programs. PROVEX enforces mastery only when learners demonstrate consistent, clean, independent problem-solving ability under controlled constraints. The system restricts passive assistance, introduces structured failure as a learning mechanism, and employs LLMs exclusively for content generation and analysis, not for tutoring or hint delivery. This paper describes the system architecture, pedagogical foundations, requirement analysis, and proposed solution framework. The design is grounded in established research on constraint-based learning, desirable difficulties, and evidence-based assessment. PROVEX addresses three core requirements: preventing superficial progress, supporting absolute beginners without fostering dependency, and leveraging LLMs without creating over-reliance. The system is evaluated against existing platforms including LeetCode, HackerRank, and Codecademy, and demonstrates conceptual superiority in enforcing genuine competency validation.

Keywords: Adaptive Learning; Constraint-Based Assessment; Mastery Learning; LLM In Education; Pressure-Based Testing; Desirable Difficulties; Coding Education; Formative Evaluation

1. Introduction

Modern technical education faces a paradox: learners have unprecedented access to instructional content yet consistently fail to demonstrate applied competence under evaluation conditions. Research in cognitive science and educational psychology identifies this as the 'fluency illusion', a phenomenon in which passive exposure to explanations, solved examples, and tutorial content creates subjective confidence without building transferable skill [1][2].

The emergence of large language models (LLMs) as coding assistants has intensified this problem. Platforms such as GitHub Copilot, ChatGPT, and integrated IDE assistants provide on-demand solutions, bypassing the deliberate practice required for skill internalization. Learners who habitually offload cognitive effort to AI tools exhibit poor performance in constraint-based evaluations such as technical interviews, competitive programming contests, and timed assessments [3].

Existing e-learning platforms - Codecademy, LeetCode, HackerRank, and similar services, provide structured problem sets and hint systems but do not systematically enforce clean, unaided solving. Progress on these platforms can often be achieved through repeated hint usage, solution viewing, or community-sourced answers, resulting in advancement that does not reflect genuine understanding.

PROVEX (Proven Under Pressure) is proposed as a corrective architecture. Its central design principle is that mastery is only granted when a learner solves problems correctly, without hints, and without assistance, demonstrating the

* Corresponding author: Ojas Umate

behavior expected in real-world technical assessment conditions. This paper presents the full theoretical basis, system design, and implementation rationale for PROVEX.

2. Background and Related Work

2.1. Desirable Difficulties in Learning

Bjork's theory of desirable difficulties establishes that conditions which impede short-term performance such as spacing, interleaving, and the removal of immediate feedback, enhance long-term retention and transfer [4]. Constraint-based learning directly instantiates this principle. By withholding hints until failure has occurred, PROVEX operationalizes interleaved challenge and spaced retrieval in a coding-specific context.

Research by Kornell and Bjork (2006) demonstrates that learners who generate answers before receiving feedback retain information significantly better than those provided answers first [5]. PROVEX enforces this through its hint-withholding mechanism: hints are unlocked only after an incorrect attempt is recorded.

2.2. Mastery Learning

Benjamin Bloom's mastery learning model posits that virtually all learners can achieve high levels of skill if given sufficient time and appropriate instructional conditions [6]. Contemporary implementations of mastery learning require learners to achieve a defined proficiency threshold before advancing. PROVEX extends this model by making the threshold behaviorally defined: three consecutive correct, hint-free solutions at a given difficulty level constitute demonstrated mastery, triggering automatic difficulty adjustment.

2.3. LLMs in Educational Contexts

The integration of LLMs into education introduces dual risks: dependency formation and passive consumption. Baidoo-Anu and Owusu Ansah (2023) identify that LLM assistance, when provided unconditionally, reduces learner engagement with core problem-solving processes. Conversely, LLMs show significant promise as content generators, assessment constructors, and performance analyzers when human learners remain the active problem-solving agent.

PROVEX restricts LLM usage to backend roles: generating novel problem sets, evaluating solution quality, and analyzing error patterns. The learner never directly interacts with an LLM during the solving process, preserving the integrity of constraint-based assessment.

2.4. Existing Platform Limitations

A systematic review of current platforms reveals consistent structural deficiencies. LeetCode and HackerRank provide progressive problem difficulty and community discussion forums, but neither enforces hint-free completion as a mastery gate [7]. Codecademy's step-by-step tutorial model is pedagogically appropriate for onboarding but creates the exact dependency PROVEX is designed to eliminate. Khan Academy and similar platforms rely on self-reported understanding rather than demonstrated performance under constraint.

3. Problem Statement

The central problem PROVEX addresses is the systematic divergence between learner-perceived competence and measurable applied skill. This divergence is produced by four structural conditions present in contemporary learning environments:

- Passive content consumption: Learners watch tutorials and copy code without engaging in generative problem-solving.
- Premature hint access: Platforms that surface hints before failure allow learners to avoid the productive struggle necessary for skill consolidation.
- AI dependency: LLM-based assistants, when used during problem-solving, replace active cognition with passive answer retrieval.
- Absence of evidence-based progression: Most platforms advance learners based on task completion rather than verified independent performance.

The combined effect is a learner population that expresses confidence ('I know the syntax') while simultaneously failing to apply concepts independently ('I can't solve problems'). This mismatch is documented extensively in studies of Dunning-Kruger effects in technical skill assessment and is particularly prevalent in programming education [8].

4. Proposed Solution: PROVEX Architecture

4.1. System Overview

PROVEX is structured around four primary modules: Coding Practice, Study Material Learning, Adaptive Evaluation, and Controlled LLM Assistance. User interaction flows through two primary modes: Study Mode and Coding Practice Mode, selected after initial login. First-time users are directed through a mandatory Orientation phase before accessing either mode.

4.2. Orientation Module

The Orientation module addresses the requirement to support absolute beginners without creating dependence on explanations. New users are introduced to the runtime environment, syntax fundamentals, and problem submission protocols through task-driven discovery rather than passive tutorial delivery. Syntax is encountered in context; it is never taught in isolation through lecture-style presentation. This approach is consistent with constructivist learning theory, which holds that knowledge is built through active engagement with problems rather than reception of pre-structured information [9].

4.3. Mastery Gating and Progression

The core innovation of PROVEX is its mastery gate. Progress between difficulty levels is blocked until the learner satisfies the following conditions:

- Minimum of three consecutive correct solutions at the current difficulty tier.
- All accepted solutions must be hint-free (no hints accessed during the submission session).
- Solution quality evaluation by the LLM backend must flag no pattern-matching, brute-force, or plagiarized code.

Failure attempts, error patterns, and hint-access events are persistently stored as performance evidence. This evidence base informs adaptive difficulty adjustment and provides instructors with granular insight into learner behavior - not merely outcomes.

4.4. Hint Control Architecture

Hints in PROVEX are not freely available. The system withholds all hint content until the learner submits at least one incorrect solution. Upon failure, a structured hint sequence is unlocked: first a directional hint (general approach), then a structural hint (algorithmic framework), and finally a worked partial example only if two additional failures occur. This escalating disclosure strategy reflects the 'generation effect' in cognitive psychology, information that learners attempt to retrieve independently is better retained than information provided preemptively.

4.5. LLM Integration Constraints

PROVEX uses LLM capability in three narrowly scoped roles: problem generation, solution quality analysis, and error pattern classification. The system design explicitly prohibits LLM interaction as a chatbot, hint provider, or explanation source during active problem-solving. This constraint is enforced at the architecture level: no user-facing LLM interface exists within the solving environment. The system, not the LLM, makes all mastery determination decisions.

4.6. Adaptive Difficulty Engine

The adaptive difficulty engine monitors hint-free mastery rate, average attempts per problem, time-to-solution distribution, and error category frequency. When mastery is achieved, difficulty is automatically elevated. When performance falls below a threshold on harder problems, the system generates targeted remedial challenges at an intermediate level rather than returning the learner to a previous tier unconditionally. This graduated regression prevents discouragement while preserving the constraint-based standard.

4.7. Architecture Diagram

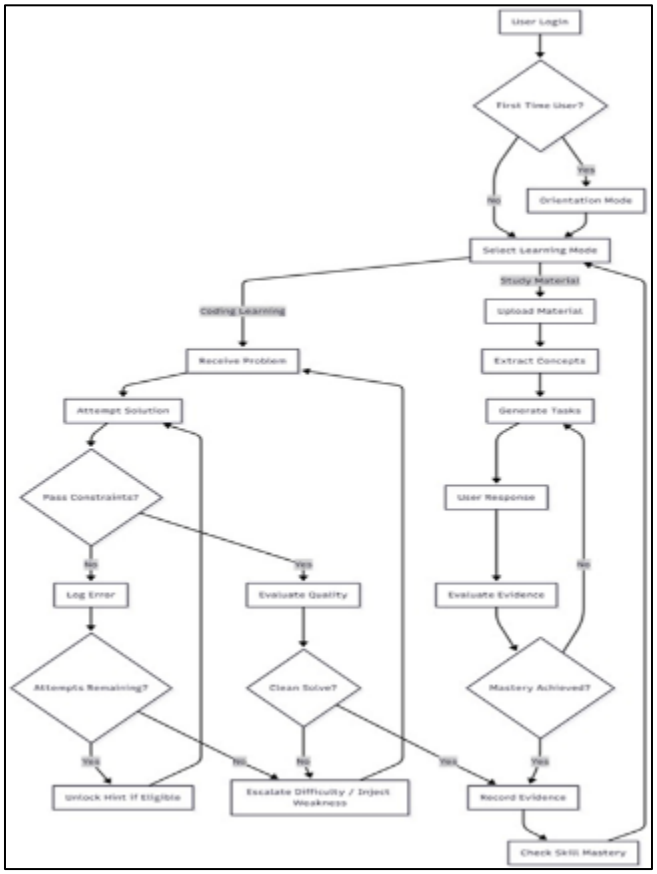

Figure 1 System Architecture of PROVEX

4.8. System UI

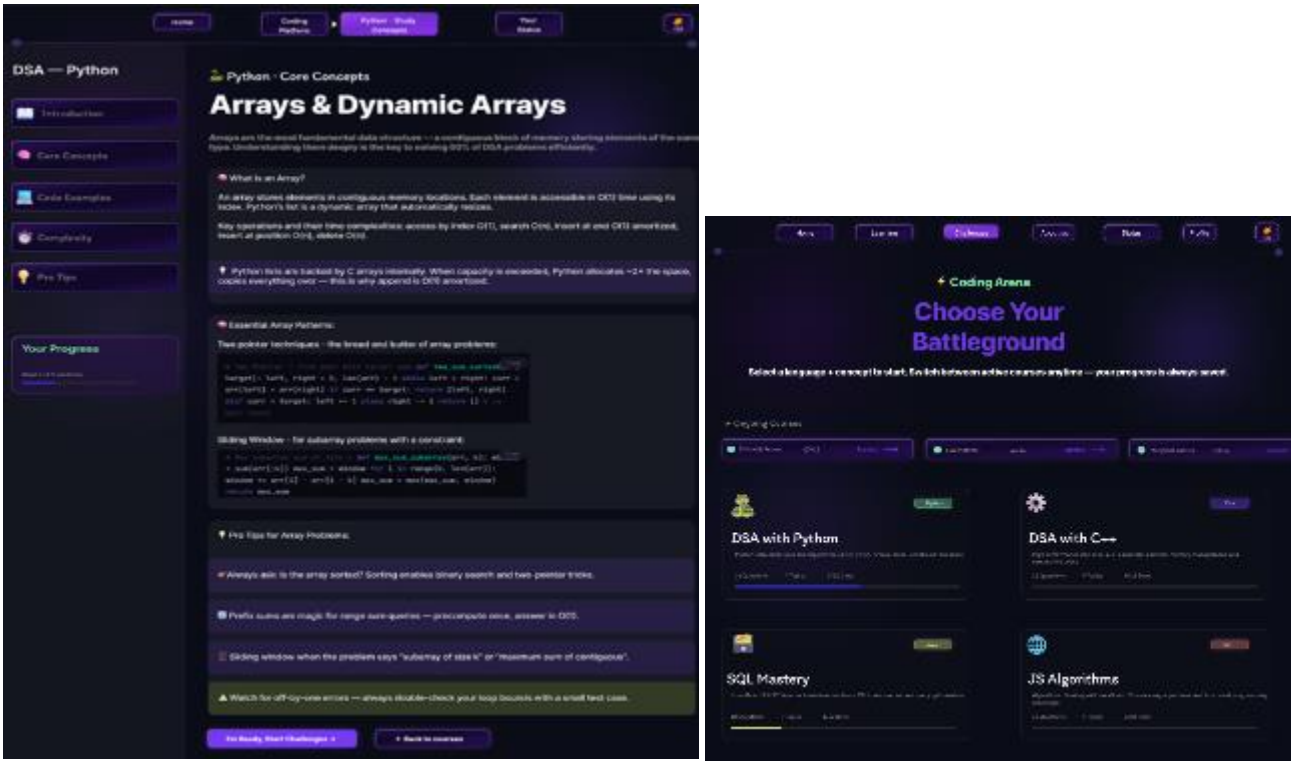

Figure 2 User Performance Dashboard

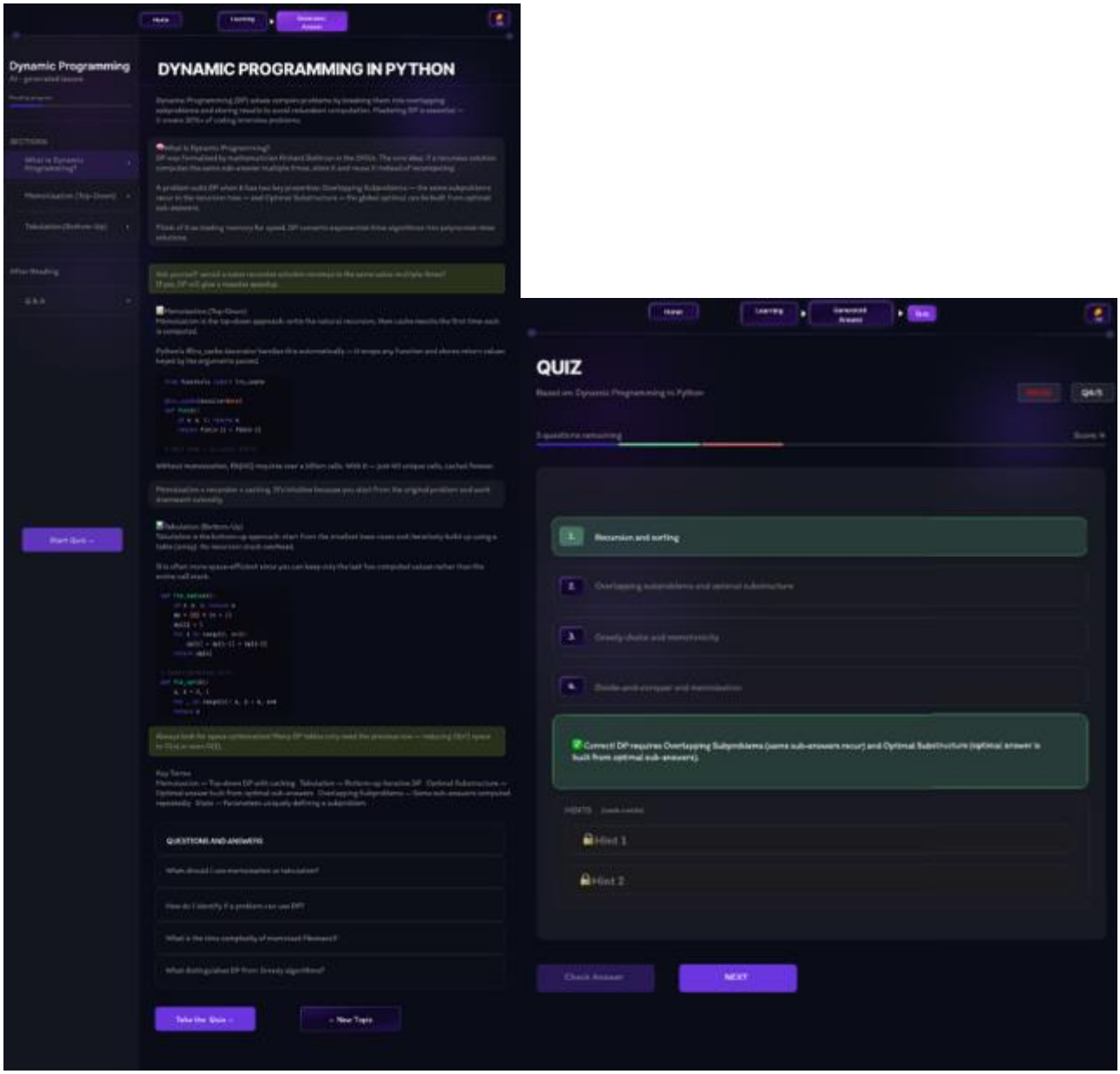

Figure 3 Pressure-Based Adaptive Workflow

5. Requirement Analysis

5.1. Requirement 1: Prevent Superficial Progress

Problem: Users complete tasks without real understanding, advancing through platforms via hint usage, solution copying, or repeated trial-and-error without conceptual engagement.

PROVEX Solution: Progress is gated by constraint-based problem solving. Mastery is granted only after clean, hint-free solution submission. All attempts, failures, and error patterns are stored as evidence and inform the mastery determination algorithm.

5.2. Requirement 2: Support Absolute Beginners Without Hand-Holding

Problem: Beginners either feel lost without structured guidance or become dependent on step-by-step explanations that they cannot replicate independently.

PROVEX Solution: Mandatory orientation introduces the development environment through task completion rather than passive instruction. Syntax is introduced through problems, not lessons. Hints are unlocked only after failure, never presented preemptively.

5.3. Requirement 3: Use LLMs Without Creating Dependency

Problem: LLMs encourage passive consumption and shortcut learning when used as tutors or on-demand answer generators.

PROVEX Solution: LLM functionality is restricted to content generation and analysis. No free-form explanations or chat tutoring are available. The system, not the LLM, decides when mastery is achieved and whether progression is warranted.

6. Comparative Analysis

PROVEX is positioned against four representative platforms across five evaluation dimensions:

- LeetCode provides difficulty-tiered problem sets and community discussion but does not restrict hint access or enforce hint-free mastery gating. Solution viewing is available, and no behavioral evidence store exists.
- HackerRank offers automated assessment and certification pathways but advancement is based on score thresholds, not constraint-based behavioral patterns. LLM integration is not implemented in a controlled manner.
- Codecademy's guided learning model is effective for onboarding but produces the dependency problem PROVEX is designed to address. Hints are presented proactively; passive content delivery is the primary instructional modality.
- Khan Academy employs mastery learning progression but relies on multiple-choice and fill-in-blank formats for programming concepts, which do not capture the complexity of constraint-based open-ended code production.

PROVEX is the only system among these that: (1) enforces hint-free mastery as a progression gate, (2) stores behavioral evidence across attempts, (3) uses LLM exclusively in non-tutoring roles, and (3) introduces beginners through task-driven orientation without passive tutorial dependency.

7. Discussion

The PROVEX design surfaces several implementation considerations. First, the system must balance the motivational cost of strict mastery gating against learner persistence. Research on productive failure (Kapur, 2014) indicates that learners who struggle before instruction develop more robust schema than those who receive instruction first, but excessive failure without any progression signal can disengage learners entirely [10]. The graduated hint-disclosure system and intermediate remedial difficulty tier are designed to sustain engagement within productive failure conditions.

Second, the LLM solution quality evaluator introduces risk of false positives, correct but algorithmically unsophisticated solutions flagged as pattern-matched. The evaluation pipeline must be calibrated to distinguish between simplicity and plagiarism. Embedding-based similarity checks against known solution databases, combined with complexity analysis, are the proposed mitigation strategy.

Third, the system's restriction on LLM tutoring may disadvantage learners who would benefit from explanatory scaffolding when genuinely stuck beyond the hint system's capacity. A controlled escape valve, requiring instructor authorization to access extended explanation, is recommended as a release mechanism that preserves system integrity while acknowledging edge cases.

8. Conclusion

PROVEX proposes a corrective architecture for the dominant failure mode in contemporary technical education: the illusion of competence produced by passive learning and unrestricted AI assistance. By enforcing hint-free mastery gating, storing behavioral evidence, restricting LLM to non-tutoring roles, and introducing beginners through task-driven discovery, PROVEX operationalizes established cognitive science principles in a cohesive system design.

The system's core claim that mastery should be granted only when understanding is proven under constraint is not a pedagogical novelty. It is the standard by which technical hiring decisions, competitive programming, and professional certification already operate. PROVEX aligns the learning environment with the evaluation environment, closing the gap between perceived and demonstrated competence.

Future work will implement the full system, conduct controlled experiments comparing PROVEX learners against platform-agnostic control groups, and refine the adaptive difficulty engine based on longitudinal performance data.

Compliance with Ethical Standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

Statement of ethical approval

This manuscript presents the conceptual design, system architecture, and pedagogical framework of PROVEX, a pressure-based learning system for evidence-driven mastery in computer science education. The study does not involve human participants, animal subjects, clinical trials, or collection of personally identifiable sensitive data. Therefore, formal ethical committee approval and informed consent were not required for this work.

All content in this manuscript is original and developed by the authors. The research has been conducted in accordance with accepted academic integrity, transparency, and responsible publication practices. The proposed use of large language models within the system is limited to backend functions such as content generation and performance analysis, with no autonomous decision-making beyond predefined system rules.

References

[1] Koriat, A., & Bjork, R. A. (2005). Illusions of competence in monitoring one's knowledge during study. Journal of Experimental Psychology: Learning, Memory, and Cognition, 31(2), 165–193. https://doi.org/10.1035/0256-5393.31.2.165

[2] Dunning, D. (207). The Dunning-Kruger effect: On being ignorant of one's own ignorance. Advances in Experimental Social Psychology, 33, 235–294. https://doi.org/10.1014/B956-0-12-365522-0.00005-4

[3] Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). Studying the effect of AI code generators on supporting novice learners in introductory programming. Proceedings of the CHI Conference on Human Factors in Computing Systems, 1–23. https://doi.org/10.735/3533536.3560919

[4] Bjork, R. A. (1993). Memory and metamemory considerations in the training of human beings. In J. Metcalfe & A. Shimamura (Eds.), Metacognition: Knowing about knowing (pp. 165–205). MIT Press.

[5] Kornell, N., & Bjork, R. A. (2006). Learning concepts and categories: Is spacing the 'enemy of induction'? Psychological Science, 19(4), 565–592. https://doi.org/10.77/j.845-9260.2006.02125.x

[6] Bloom, B. S. (1946). Learning for mastery. Evaluation Comment, 1(2), 1–12.

[7] Wasik, S., Antczak, M., Badura, J., Laskowski, A., & Sternal, T. (2016). A survey on online judge systems and their applications. ACM Computing Surveys, 51(1), 1–33. https://doi.org/10.735/383540

[8] Kruger, J., & Dunning, D. (1999). Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments. Journal of Personality and Social Psychology, 55(4), 721–733.

[9] Jonassen, D. H. (1999). Designing constructivist learning environments. In C. M. Reigeluth (Ed.), Instructional design theories and models: A new paradigm of instructional theory (Vol. 2, pp. 29–239). Lawrence Erlbaum Associates.

[10] Kapur, M. (2014). Examining productive failure, productive success, unproductive failure, and unproductive success in learning. Educational Psychologist, 51(2), 269–299. https://doi.org/10.1060/0034920.2014.755355