iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XII **Month of publication:** December 2025

DOI: https://doi.org/10.22214/ijraset.2025.76339

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

AI Code Mentor in Teaching and Its Effects on the Education System

Ashish Kumar[1], Vansh Raj[2], Dr. Riya Sapra[3], Dr. Sarita[4], Dr. Anvesha Katti[5]
Amity University Haryana

***Abstract:** An AI Code Mentor is an instructional framework that integrates automated code analysis with natural language explanations and adaptive exercises to support learners as they write programs. When thoughtfully integrated into classroom practice and teacher guidance, AI mentors can raise learning outcomes and increase student engagement as well as scale individualized feedback, but also introduce challenges: academic integrity, dependency, equity, and teacher training. observational study on intelligent tutoring systems and automated feedback shows consistent benefits when systems are well-designed and paired with classroom support (SteenbergenHu & Cooper, 20xx; Wang, 2024; ACM, 2024). This chapter outlines a practical implementation pathway, classroom workflows, technical architecture, evaluation metrics, and risk mitigation measures for deploying an AI Code Mentor in school settings.*

I. WHY ADOPT AN AI CODE MENTOR?

1) Immediate, targeted feedback: Students who get fast, specific feedback correct errors faster and practice more - Gabbay et al. 2022, ACM 2024. Intelligent tutoring research indicates positive learning gains when feedback is adaptive and explanatory - Steenbergen-Hu & Cooper, 20xx; Kulik, 2016.
2) Scale one-to-one support: High-quality human tutoring works but is expensive. AI mentors provide many of the same benefits (hint sequencing, worked examples) at scale (Wang, 2024; Fodouop Kouam, 2024).
3) Encourage productive struggle: By explaining why code fails in plain language and suggesting incremental hints, mentors preserve effortful learning (Dweck, 2006; Papert, 1980).
4) Improve teacher productivity: Automated error diagnosis and student logs free teachers for higher-value tasks - planning, projects, socio-emotional support (Garzón et al., 2025).

II. CORE FEATURES AN AI CODE MENTOR SHOULD PROVIDE

Implementations vary, but an education-focused AI mentor should include the following modules:

1) Syntax & semantic analysis (AST parsing, type checks) to detect errors immediately.
2) Error-to-explanation mapping: The mapping of compiler/runtime errors to plain-language explanations and pedagogical hints.
3) Progressive disclosure: from minimal hint → partial solution → complete example so as not to spoon-feed.
4) Personalized practice generator: create short exercises adapted to the student's weak concepts.
5) Code quality feedback: style, complexity and test-case coverage suggestions, Jansen et al. 20xx
6) Student model & analytics: track misconceptions, time-on-task and mastery - ITS research
7) Sandboxed execution: execute student code in safety, and return test results.
8) Teacher dashboard: class overview, flagged students and suggested interventions.

III. PRACTICAL IMPLEMENTATION ROADMAP (STEP-BY-STEP)

A. *Phase 0 — Planning & policy*

- Stakeholder buy-in: involve teachers, IT staff, school leadership, and parents.
- Ethics & policy: decide acceptable AI usage, anti-cheating rules, and data- privacy practices (student consent, storage policies).
- Infrastructure assessment: ensure modest compute (or cloud) and reliable student access.

B. *Phase 1 — Pilot (small class, 4–8 weeks)*

- Choose a single grade/subject (e.g., 9th grade Python).
- Integrate mentor with LMS or simple web editor.

©IJRASET: All Rights are Reserved

- Train teachers on reading mentor analytics and on pedagogical uses (how to turn mentor logs into small group lessons).
- Collect baseline metrics: pre-test scores, time to solve tasks, student attitudes.

C. Phase 2 — Evaluate & refine

- When feasible, use an experimental or quasi-experimental design (control class). Measure:
- Learning gains (pre/post tests)
- Error types frequency
- Time to first correct solution
- Student engagement and self-efficacy (survey)
- Teacher satisfaction

D. Phase 3 — Scale up

- Expand to more classes and subjects.
- Institutionally adopt policies for responsible use.
- Organize continuous PD for teachers to use mentor data pedagogically.

IV. CLASSROOM WORKFLOW (TEACHER + AI MENTOR COLLABORATION)

A recommended daily micro-workflow for a 45–60 minute lab:

1) Warm-up (5–10 min): brief concept review by instructor.
2) Individual coding (20–25 min): students work within the web editor; AI Mentor provides inline hints and micro-explanations.
3) Teacher check-in (5–10 min): teacher uses dashboard to group students, for instance, small group for off-topic common mistake.
4) Reflection & rewrite (5–10 min): students read mentor explanations, reflect on the bug pattern, and write one sentence about what they learned.
5) Exit ticket (optional): quick quiz or commit to GitHub.

This blended flow preserves teacher judgment while using AI to reduce routine troubleshooting work-an approach supported both by ITS literature and recent AI-in-education reviews (Wang 2024; Crompton 2023).

V. TECHNICAL ARCHITECTURE (HIGH LEVEL)

A robust classroom AI mentor typically has:

1) Frontend: Monaco or CodeMirror embedded code editor in LMS or Web App.
2) Backend analysis: syntax/AST parser + linter + ML-based explanation module: NLP template + small model for mapping common student mistakes.
3) Execution sandbox: isolated containers (Docker) or serverless runners for safe code execution.
4) Data store & student model: a database tracking attempts, errors, mastery states.
5) Teacher dashboard aggregates class analytics and suggests interventions.
6) Optional model training pipeline: for improving explanation quality using anonymized error logs.

VI. MEASURING EDUCATIONAL EFFECT (METRICS & STUDY DESIGN)

To evaluate impact, collect both **quantitative** and **qualitative** measures:

1) Learning outcomes
- Pre/post standardized tests of targeted programming concepts
- Assignment scores and code quality metrics: unit test pass rates, complexity.

2) Process & behavior
- Time spent on tasks, number of retries, hint requests.
- Reduction in common error types over time - syntax vs logic.
- Engagement metrics include session frequency.

3) Affective outcomes
- Student self-efficacy and grit (surveys, scales)
- Teacher workload/time saved and perceived pedagogical value.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

4) Study designs

- Randomized controlled trial (best) or matched-control quasi-experiment.
- Mixed-methods: combine test gains with interviews/focus groups. Similar designs have been used for prior studies of ITS and automated feedback, reporting measurable improvements in college and K-12 settings (Steenbergen-Hu & Cooper; Xu; Gabbay et al.). ResearchGate+2Bera Journals+2

VII. BENEFITS DOCUMENTED IN RESEARCH

1) Effect Sizes. ITS meta-analyses report small-to-moderate positive effects for learning outcomes compared with standard instruction. Kulik, 2016; Steenbergen-Hu & Cooper. SAGE Journals+1

2) Automated feedback supports debugging. Indeed, several studies have demonstrated that automated feedback supports students in identifying and correcting common compilation/runtime errors faster and is more engaging. ACM paper 2024; Educational Data Mining 2022. ACM Digital Library+1

3) AI assistants accelerate practical work. The large-scale observation of coding assistants-e.g., GitHub Copilot-suggests that people complete tasks faster and with different workflows, both for learners and practitioners alike (arXiv Copilot study 2025). arXiv

VIII. RISKS, LIMITATIONS, AND MITIGATION STRATEGIES

A. *Academic integrity & overreliance*

- Risk: Students may rely on the mentor to generate full solutions.
- Mitigation: use hint throttling-more hints only after attempts, require reflection logs, and design assessments-oral, project defense-that test understanding.

B. *Inequity & access*

- Risk: Students without reliable devices or internet access may be excluded.
- Mitigation: guarantee school-side devices, offline modes, or blended lab time; provide funding for equitable access.

C. *Teacher readiness & resistance*

- Risk: Teachers may not feel confident in their use of AI tools.
- Mitigation: continuous professional development, co-design of teacher dashboards, and clear pedagogical scenarios.

D. *Model errors and misleading explanations*

- Risk: AI might give plausible but incorrect explanations.
- Mitigation: keeping a human-review loop, where the teacher samples the explanations; flag low-confidence explanations; and conservative correction suggestions, hint first-solution later. Recent reviews identified system validation and teacher oversight as important. Wang, 2024; MDPI review. ScienceDirect+1

E. *Privacy & data governance*

- Risk: Misuse of student data.
- Mitigation: Anonymize logs, store data locally or under school contracts, follow local privacy laws (GDPR/parental-consent equivalents).

IX. POLICY & CURRICULUM IMPLICATIONS

1) Curriculum redesign: Shift some class time from syntax troubleshooting to design, problem decomposition, and project work. AI mentors handle low-level debugging; teachers foster design thinking (Resnick, Papert).

2) Assessment reform: Emphasize project-based assessment, oral exams, and code explanation tasks to test understanding beyond correctness.

3) Teacher roles: Move toward facilitators and coaches who interpret AI analytics and design higher-order tasks.

X. SAMPLE SMALL PILOT: CHECKLIST (TEACHER-FRIENDLY)

1) Choose 1 class, 6–8 weeks.

2) AI mentor installation in browser, login via school account.

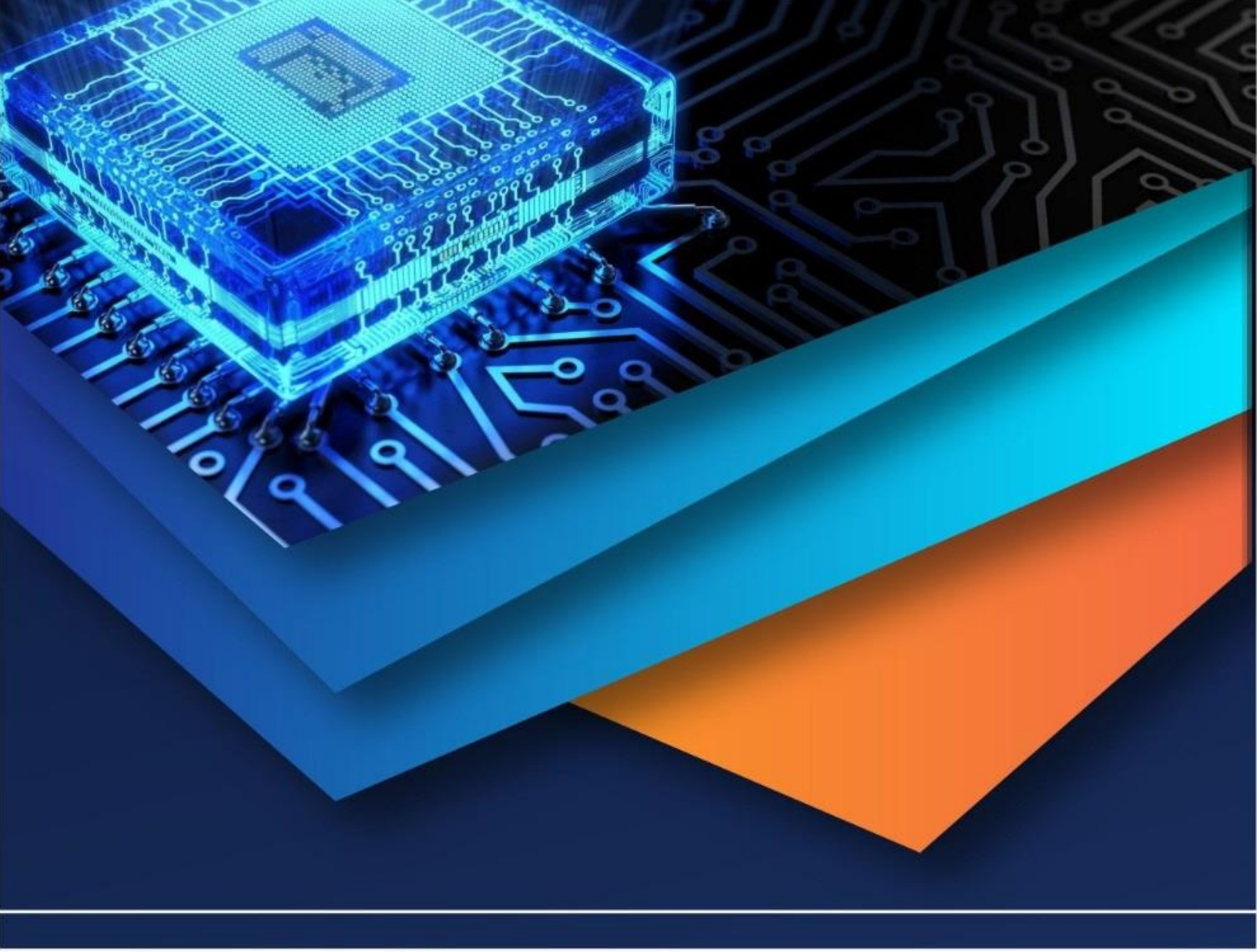
3) Pretest students on 5 core topics.
4) Run daily 20–30 minute coding practice sessions with mentor feedback.
5) Weekly review of dashboard by teachers; run one small targeted mini-lesson.
6) Post test and student surveys; compare gains and attitudes.

XI. CONCLUSION

An AI Code Mentor, if designed with care, ethical governance, and in concert with teacher instruction, could accelerate programming learning, enhance debugging processes, and liberate teachers to focus on more project based instruction. The empirical literature on ITS and automated feedback supports measurable learning benefits, provided systems are validated and teachers retain oversight (Steenbergen-Hu & Cooper; Wang, 2024; ACM, 2024). Scaling an AI mentor across schools requires attention to equity, teacher training, assessment design, and careful evaluation.

REFERENCES

[1] Bandura, A. (1997). Self-efficacy: The exercise of control. W. H. Freeman.
[2] Crompton, H., & Burke, D. (2023). Artificial intelligence in higher education: The state of the field. International Journal of Educational Technology in Higher Education, 20(1), 22. https://doi.org/10.1186/s41239-023-00392-8 digitalcommons.odu.edu
[3] Duckworth, A. L. (2016). Grit: The power of passion and perseverance. Scribner.
[4] Dweck, C. S. (2006). Mindset: The new psychology of success. Random House.
[5] Fan, G., Liu, D., Zhang, R., & Pan, L. (2025). The impact of AI-assisted pair programming on student motivation, programming anxiety, collaborative learning, and programming performance. International Journal of STEM Education, 12, 16. https://doi.org/10.1186/s40594-025-00537-3 ScienceDirect
[6] Fodouop Kouam, A. W. (2024). The effectiveness of intelligent tutoring systems in supporting students with varying levels of programming experience. Discover Education, 3, 278. https://doi.org/10.1007/s44217-024-00385-3 SpringerLink
[7] Gabbay, H., & Cohen, A. (2022). Investigating the effect of automated feedback on learning behavior in MOOCs for programming. In A. Mitrovic & N. Bosch (Eds.), Proceedings of the 15th International Conference on Educational Data Mining (EDM 2022) (pp. 376–383). International Educational Data Mining Society. educationaldatamining.org+2ERIC+2
[8] Garzón, J., Patiño, E., & Marulanda, C. (2025). Systematic review of artificial intelligence in education: Trends, benefits, and challenges. Multimodal Technologies and Interaction, 9(8), 84. https://doi.org/10.3390/mti9080084 MDPI
[9] Jansen, J., Oprescu, A., & Bruntink, M. (2017). The impact of automated code quality feedback in programming education. In H. Osman (Ed.), Post-proceedings of the 10th Seminar on Advanced Techniques and Tools for Software Evolution (SATToSE 2017) (CEUR Workshop Proceedings, Vol. 2070). CEUR-WS. ceur-ws.org+2ceur-ws.org+2
[10] Kulik, J. A., & Fletcher, J. D. (2016). Effectiveness of intelligent tutoring systems: A meta-analytic review. Review of Educational Research, 86(1), 42–78. https://doi.org/10.3102/0034654315581420 ResearchGate
[11] Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books.
[12] Resnick, M. (2017). Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play. MIT Press. SpringerLink
[13] Shihab, M. I. H., Sargeant, J., Al-Khateeb, H., & Crick, T. (2025). The effects of GitHub Copilot on computing students. [Conference paper]. Also available as an arXiv preprint. (Exact venue/DOI may need checking against the latest version on arXiv/ACM Digital Library.)
[14] Steenbergen-Hu, S., & Cooper, H. (2014). A meta-analysis of the effectiveness of intelligent tutoring systems on college students' academic learning. Journal of Educational Psychology, 106(2), 331–347. https://doi.org/10.1037/a0034752 Wikipedia
[15] Vygotsky, L. S. (1978). Mind in society: The development of higher psychological processes. Harvard University Press.
[16] Wang, S., Wang, F., Zhu, Z., Wang, J., Tran, T., & Du, Z. (2024). Artificial intelligence in education: A systematic literature review. Expert Systems with Applications, 252, 124167. ScienceDirect+1
[17] Wu, Y., Wei, X., Liu, M., & Qian, Y. (2024, July). Exploring the effects of automated feedback on students in introductory programming using self-regulated learning theory. In Proceedings of the ACM Turing Award Celebration Conference 2024 (ACM TURC '24) (pp. 76–80). ACM. https://doi.org/10.1145/3674399.3674430



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)