1. Executive Summary

This review examines state-of-the-art (SOTA) systems in AI-assisted programming education published between 2023 and 2025. The analysis focuses on technical architectures (backend/database), AI methodologies, and quantitative performance metrics. The literature reveals a strong industry consensus on **Python-based backends** and **"scaffolding" pedagogies** (providing hints rather than solutions). Crucially, the review identifies a significant gap in current research regarding cost-effective scalability, which the proposed project addresses through the novel integration of **Local LLMs (Ollama)**.

2. Technical Architecture Trends

2.1. Backend Frameworks

The literature demonstrates a clear standard for backend development in Intelligent Tutoring Systems (ITS), driven by the need for seamless integration with AI libraries.

- **Python Dominance:** The majority of successful implementations utilize Python micro-frameworks due to their native compatibility with AI SDKs.
    - **Flask:** Widely used for bridging frontend interfaces with prompt engineering logic, as seen in the *Tutor Builder* system by Calò & MacLellan. [1]
    - **FastAPI:** Preferred for systems requiring high-performance asynchronous processing, such as the *Intelligent Deep-Learning Tutor* developed by Roldán-Álvarez & Mesa. [2]
- **Strategic Alignment:** The proposed project's selection of **Django/Flask** is strongly supported by these findings, offering superior compatibility compared to Node.js alternatives for AI orchestration.

2.2. Data Management and Persistence

Educational platforms require flexible data structures to handle unstructured data such as chat logs, code snapshots, and affective state tracking.

- **NoSQL Adoption:** Non-relational databases are the standard for storing conversational history. Roldán-Álvarez & Mesa successfully utilized **MongoDB** to store student queries, system answers, and user satisfaction scores [2].
- **Strategic Alignment:** The proposal's consideration of **MongoDB** aligns perfectly with SOTA approaches for handling the diverse, unstructured data generated during student-AI mentorship sessions.

3. Performance and Pedagogical Validity

3.1. "AI as Mentor" vs. "AI as Solver"

A critical aspect of the proposed project is restricting AI to a "guidance" role. The literature provides empirical evidence validating this approach over direct solution generation.

- **Scaffolding Success:** Phung et al. demonstrated that systems designed to generate **hints** (scaffolding) rather than direct fixes achieve outcomes comparable to human tutors. Their *GPT4HINTS* system achieved **~95% precision** in feedback quality [3].
- **Reduced Anxiety:** Fan et al. found that AI-assisted pair programming significantly reduced programming anxiety ($p < .001$) and improved intrinsic motivation compared to individual work [4].
- **Detection of Misuse:** Karnalim et al. highlighted the need for detecting AI misuse, developing a system that identifies "code anomalies" with **89% precision** in controlled environments, reinforcing the need for platforms that control AI output [5].

3.2. Quantitative Impact on Student Success

Reviewed case studies confirm that properly integrated AI tools lead to measurable academic improvement.

- **Grade Improvement:** Timcenko reported that in a class using AI assistance, the average grade rose to **7.67** (on a 7-point scale), compared to **6.08** in previous years without AI [4].
- **Performance Scores:** Fan et al. reported that students using AI assistance outperformed individual programmers with mean performance scores of **~84 vs. ~75** ($p < .001$) [1].
- **Efficiency:** For instructor-facing tools, Calò & MacLellan showed that AI assistance reduced the time required to build complex tutor interfaces by **68%** [5].

4. Gap Analysis: The Case for Local LLMs

While the literature demonstrates the effectiveness of AI tutors, it also highlights a significant barrier to widespread adoption: **Reliance on Proprietary APIs.**

- **The Cost/Privacy Barrier:** Most reviewed studies, including Phung et al. [2] and Pankiewicz & Baker [6], rely on the **OpenAI API (GPT-4)**. This introduces high

operational costs and potential data privacy concerns, limiting scalability for educational institutions.

- **The Proposed Innovation:** The project proposal addresses this specific gap by introducing **Local LLM integration (Ollama)**. By enabling the use of open-source models like Llama or Mistral, the proposed platform offers a **sustainable, cost-effective, and privacy-compliant** alternative to the API-dependent systems found in current research.

5. Conclusion

The proposed **"AI-Assisted Programming Platform"** is an advancement of the current body of research. It aligns with state-of-the-art findings by adopting the **Python/NoSQL** architecture [3][5] and the **"scaffolding" pedagogy** [2]. Furthermore, it innovates by solving the critical **"accessibility vs. cost"** dilemma identified in the literature through the support of **Local LLMs**.

References

[1] T. Calo and C. MacLellan, “Towards Educator-Driven Tutor Authoring: Generative AI Approaches for Creating Intelligent Tutor Interfaces,” in *L@S 2024 - Proceedings of the 11th ACM Conference on Learning @ Scale*, Association for Computing Machinery, Inc, Jul. 2024, pp. 305–309. doi: 10.1145/3657604.3664694.

[2] D. Roldan-Alvarez and F. J. Mesa, “Intelligent Deep-Learning Tutoring System to Assist Instructors in Programming Courses,” *IEEE Transactions on Education*, vol. 67, no. 1, pp. 153–161, Feb. 2024, doi: 10.1109/TE.2023.3331055.

[3] T. Phung *et al.*, “Automating Human Tutor-Style Programming Feedback: Leveraging GPT-4 Tutor Model for Hint Generation and GPT-3.5 Student Model for Hint Validation,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Mar. 2024, pp. 12–23. doi: 10.1145/3636555.3636846.

[4] G. Fan, D. Liu, R. Zhang, and L. Pan, “The impact of AI-assisted pair programming on student motivation, programming anxiety, collaborative learning, and programming performance: a comparative study with traditional pair programming and individual approaches,” *Int J STEM Educ*, vol. 12, no. 1, Dec. 2025, doi: 10.1186/s40594-025-00537-3.

[5] O. Karnalim, H. Toba, and M. C. Johan, “Detecting AI assisted submissions in introductory programming via code anomaly,” *Educ Inf Technol (Dordr)*, vol. 29, no. 13, pp. 16841–16866, Sep. 2024, doi: 10.1007/s10639-024-12520-6.