ADAPTIVE LEARNING PLATFORM FOR PROGRAMMING EDUCATION

Naveen Gowda M Y
School of Computing and Information Technology
Reva University
Bengaluru, India

Sachidanand Shivade
School of Computing and Information Technology
Reva University
Bengaluru, India

Jagadeesh Pujari
School of Computing and Information Technology
Reva University
Bengaluru, India

N Prabudh
School of Computing and Information Technology
Reva University
Bengaluru, India

Prof. Karthik Kashyap
Assistant Professor
School of Computing and Information Technology
Reva University
Bengaluru, India

***Abstract*—**Programming education for beginners often suffers from rigid curricula, limited visualization, and a lack of personalized feedback, resulting in poor conceptual understanding and high dropout rates. This paper presents an AI-Enhanced Progressive Programming Tutor, a web-based learning platform that delivers adaptive instruction, real-time feedback, and interactive visualizations to support novice programmers.

The proposed system integrates an adaptive learning engine that tracks learner progress and understanding levels, a visual code execution and debugging environment, and an intelligent conversational assistant. The platform provides a structured default curriculum while enabling on-demand generation of personalized instructional content based on individual learner needs. By dynamically adjusting content difficulty, pedagogy, and explanations using learner analytics, the system emphasizes concept-driven learning over rote memorization.

The proposed approach aims to bridge the theory–practice gap in programming education, improve learner engagement, and strengthen foundational programming skills. The paper outlines the system architecture, learning workflow Models' knowledge decays over time, and experimental evaluation strategy, demonstrating the educational potential of combining artificial intelligence with modern programming pedagogy.

Keywords—artificial intelligence, programming education, adaptive learning, intelligent tutoring systems, code visualization.

I. INTRODUCTION

Programming has become a foundational skill across multiple disciplines; however, learning to program remains challenging for beginners. Traditional programming education often emphasizes syntax and memorization, while neglecting conceptual clarity and problem-solving skills. As a result, learners struggle with abstract concepts such as control flow, memory management, and debugging. These challenges contribute to high attrition rates in computer science education and discourage potential learners.

Recent advances in artificial intelligence and educational technology provide an opportunity to redesign programming education. Intelligent tutoring systems and adaptive learning platforms can personalize instruction, provide immediate feedback, and accommodate diverse learning styles. However, many existing platforms offer static learning paths, limited visualization, or generic assistance that fails to address individual learner needs.

This paper makes the following key contributions:

1. A Bayesian Knowledge Tracing-based adaptive learning model for concept-level knowledge tracking.
2. Integration of a forgetting curve to model knowledge decay and improve long-term retention.
3. An AI-driven personalized remediation system that generates lessons from learner mistakes.
4. A multi-modal learning framework combining concept learning, query-based interaction, and project-based learning.
5. A dynamically adaptive learning path based on learner performance and mastery levels.

To the best of our knowledge, this is one of the first systems to integrate probabilistic knowledge tracing, cognitive decay modelling, and AI-based tutoring in a unified programming education platform.

II. PROBLEM STATEMENT

Despite the availability of numerous online programming courses and tools, there is a lack of integrated platforms that offer adaptive learning, conceptual visualization, and personalized feedback simultaneously. Existing systems often adopt a one-size-fits-all approach, which does not account for variations in learner pace, prior knowledge, or cognitive style.

This limitation results in weak foundational understanding, learner frustration, and reduced confidence. The absence of real-time visualization tools further compounds the problem, as beginners find it difficult to mentally simulate code execution and debug effectively. Addressing these issues is critical to improving programming literacy and developing a skilled workforce capable of meeting the demands of the growing technology sector.

III. RELATED WORK AND EXISTING SYSTEMS

A. Traditional Programming Learning Platforms

Early programming learning platforms primarily focused on delivering static educational content such as text-based tutorials, recorded lectures, and predefined coding exercises. These systems provided learners with foundational knowledge and practice opportunities but lacked interactivity and personalization. Most platforms followed a fixed curriculum, where all learners progressed through the same sequence of topics regardless of their prior knowledge or learning pace.

While these systems were effective for content delivery, they placed a significant cognitive burden on beginners, who often struggled to understand abstract concepts like control flow and memory management without guided support. Additionally, the absence of real-time feedback and conceptual visualization limited learners' ability to debug and comprehend program behaviour effectively.

B. Rule-Based and Structured Programming Tutors

To improve learning outcomes, rule-based programming tutors were introduced, offering structured guidance and automated feedback. These systems evaluate user code based on predefined rules, test cases, and expected outputs, providing hints or corrections when errors occur. Many platforms also incorporate step-by-step problem-solving approaches and predefined learning paths.

Although these systems introduce a level of interactivity and structured learning, they remain inherently rigid. Feedback is often limited to syntax or output correctness rather than deeper conceptual understanding. When learners deviate from expected solution patterns, the systems may fail to provide meaningful guidance. Furthermore, these tutors typically do not adapt dynamically to individual learning styles or progress, resulting in a one-size-fits-all learning experience.

C. Intelligent and AI-Based Programming Education Systems

Recent advancements in artificial intelligence have led to the development of intelligent programming education systems that offer personalized learning experiences. These systems leverage machine learning, natural language processing, and learner analytics to provide adaptive content, automated feedback, and conversational AI assistance. Tools such as code visualization platforms and AI-powered assistants help learners understand program execution, debug errors, and receive on-demand explanations.

Despite these improvements, many existing AI-based systems focus on isolated features such as code generation, chatbot assistance, or visualization tools, rather than providing a fully integrated learning environment. Additionally, some systems risk over-reliance on AI-generated solutions, which may hinder active learning and problem-solving skills. Real-time adaptation to learner behaviour and seamless integration of multiple pedagogical strategies remain key challenges in current solutions.

D. Summary of Limitations in Existing Systems

From the above analysis, it is evident that existing systems suffer from the following limitations:

- Lack of integration between personalization, visualization, and intelligent assistance.
- Limited adaptability to individual learner pace and understanding.
- Insufficient support for conceptual learning and debugging skills.
- Over-reliance on static content or isolated AI features.

These limitations highlight the need for a unified system that combines adaptive learning, real-time visualization, and intelligent guidance, which is addressed by the proposed AI-Enhanced Progressive Programming .

IV. NOVEL CONTRIBUTION AND PROPOSED APPROACH

The proposed AI-Enhanced Progressive Programming Tutor introduces several novel contributions beyond existing systems:

• **Bayesian Knowledge Tracing-Based Personalization:** The system models learner knowledge using probabilistic BKT instead of static scoring.

•**Concept-Level Knowledge Tracking:** Tracks mastery per concept rather than overall performance.

• **Forgetting Curve Integration:** Model knowledge decays over time and requires revision.

• **AI-Driven Personalized Remediation:** Generates lessons based on user mistakes instead of generic feedback.

• **Multi-Modal Learning System:** Includes Concept Mode, Query Mode, and Project Mode.

• **Adaptive Learning Path:** Dynamically reorders topics based on learner performance.

V. SYSTEM ARCHITECTURE AND METHODOLOGY

The AI-Enhanced Progressive Programming Tutor consists of four core components: an adaptive learning engine, a visual code execution environment, an intelligent chatbot assistant, and a learner analytics dashboard. The adaptive learning engine analyzes learner interactions, performance metrics, and progression patterns to dynamically adjust lesson content, difficulty level, and instructional strategy. The visual execution environment allows learners to step through code line by line, observing variable changes and memory allocation in real time. The chatbot assistant provides contextual hints and explanations without revealing complete solutions, promoting active learning. The analytics dashboard presents progress reports and identifies areas requiring additional practice. Learner knowledge is updated using Bayesian Knowledge Tracing

as follows:

$$P(K_t|Correct) = \frac{P(K_{t-1})(1-p_{slip})}{P(K_{t-1})(1-p_{slip}) + (1-P(K_{t-1}))p_{guess}}$$

where P(Kt) represents the probability that the learner has mastered the concept at time t.

A. *Adaptive Learning Engine*

The adaptive engine maintains a learner model capturing current proficiency, misconceptions, and engagement indicators. Based on this model, it selects appropriate problems, explanations, and pedagogical approaches, such as visual, inquiry-based, or concept-first explanations. Rule-based and data-driven strategies can be combined to renew recommendations as more learner data is collected.

B. *Intelligent Chatbot Assistant*

The chatbot assistant, powered by natural language understanding and generation, allows learners to ask questions in plain language about code behavior, error messages, and concepts. It offers hints, explanations, and analogies while avoiding direct provision of complete solutions, encouraging productive struggle and self-explanation. The assistant can also escalate complex issues to human instructors by generating summaries of the learner's difficulties and interaction history.

C. *Learner Analytics Dashboard*

The analytics dashboard aggregates performance data, time-on-task, hint usage, and visualization interactions to provide insights for both learners and instructors. Learners can monitor their progress, identify weak topics, and plan targeted practice. Instructors can use aggregated analytics to detect risk, and evaluate the effectiveness of specific activities.

VI. RESULTS AND DISCUSSION

Preliminary evaluations indicate that learners using adaptive and visualization-supported instruction demonstrate improved conceptual understanding and debugging ability compared to traditional learning methods. Personalized feedback and visual execution reduce cognitive load and help learners build mental models of program behavior. The integration of AI-driven assistance further enhances learner confidence and engagement. The system's web-based architecture ensures accessibility and scalability, making it suitable for academic institutions, self-learners, and career switchers. The results suggest that combining AI with pedagogically sound design principles can significantly enhance programming education.

VII. EXPERIMENTAL SETUP AND EVALUATION

A preliminary experimental setup is designed to evaluate the effectiveness of the proposed AI-Enhanced Progressive Programming Tutor in improving conceptual understanding and learner engagement. The experiment targets undergraduate students and beginners with limited prior programming experience.

Participants are divided into two groups. The control group follows a traditional learning approach using static tutorials and standard coding exercises, while the experimental group uses the proposed platform with adaptive learning, visualization, and AI-assisted guidance enabled. Both groups are provided with identical learning objectives and assessment tasks. The evaluation focuses on multiple metrics, including pre-test and post-test scores, task completion time, debugging accuracy, and learner engagement levels.

Conceptual understanding is measured using problem-solving questions that require an explanation of code behavior rather than syntax recall. Learner interaction data, such as hint usage, visualization interaction, and progress consistency, are also analyzed. The experimental duration spans multiple learning sessions covering fundamental programming concepts such as variables, control flow, and functions. Comparative analysis of results is used to assess improvements in learning outcomes, conceptual clarity, and learner confidence.

Feature	**Traditional**	**Rule-Based**	**AI Systems**	**Proposed System**
Personalization	No	No	Limited	BKT-based
Visualization	No	No	yes	yes
Real-time Adaptation	No	No	Limited	yes
Forgetting Model	No	No	No	yes
Concept Tracking	No	No	Limited	yes

VIII. SOCIAL AND ENVIRONMENTAL IMPACT

The proposed platform democratizes access to quality programming education by providing an affordable, accessible, and self-paced learning environment. It supports diverse learner populations, including students from underserved communities. As a digital solution, the system minimizes the use of physical resources and supports remote learning, contributing to reduced environmental impact.

IX. CONCLUSION AND FUTURE WORK

This paper presented an AI-Enhanced Progressive Programming Tutor designed to address key challenges in programming education. By integrating adaptive learning, real-time visualization, and intelligent assistance, the platform promotes deep conceptual understanding and personalized learning experiences. Future work includes large-scale empirical evaluation, support for multiple programming languages, enhanced learning analytics, and integration of advanced generative AI models for richer instructional support.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to the faculty members of the School of Computing and Information Technology, Reva University, for their guidance and support throughout the preparation of this research paper. The authors also thank the institution for providing the necessary resources and facilities to carry out this work.

REFERENCES

[1] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Modeling and User-Adapted Interaction*, vol. 4, no. 4, pp. 253–278, 1994..

[2] R. S. J. d. Baker, A. T. Corbett, and V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian Knowledge Tracing," in *Proc. 9th Int. Conf. Intelligent Tutoring Systems (ITS)*, 2008, pp. 406–415.

[3] Z. A. Pardos and N. T. Heffernan, "Modeling individualization in a Bayesian networks implementation of Knowledge Tracing," in *Proc. 18th Int. Conf. User Modeling, Adaptation, and Personalization (UMAP)*, 2010, pp. 255–266.

[4] R. Pelánek, "Bayesian knowledge tracing, logistic models, and beyond: An overview of learner modeling techniques," *User Modeling and User-Adapted Interaction*, vol. 27, no. 3–5, pp. 313–350, 2017.

[5] K. VanLehn, "The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems," *Educational Psychologist*, vol. 46, no. 4, pp. 197–221, 2011

[6] B. S. Bloom, "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring," *Educational Researcher*, vol. 13, no. 6, pp. 4–16, 1984.

[7] P. Brusilovsky and E. Millán, "User models for adaptive hypermedia and adaptive educational systems," in *The Adaptive Web*, Lecture Notes in Computer Science, vol. 4321, Springer, 2007, pp. 3–53.

[8] N. J. Cepeda *et al.*, "Distributed practice in verbal recall tasks: A review and quantitative synthesis," *Psychological Bulletin*, vol. 132, no. 3, pp. 354–380, 2006.

[9] [9] E. Kasneci *et al.*, "ChatGPT for good? On opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023.

[10] [10] M. T. H. Chi *et al.*, "Learning from human tutoring," *Cognitive Science*, vol. 25, no. 4, pp. 471–533, 2001.