Human-Computer Interaction & Emerging Technologies, Vol. 195, 2025, 324–333
https://doi.org/10.54941/ahfe1006251

PyPro: Think in Code. Grow in Logic!

Elijah Ballou, Osita Odunze, Michael Adeleke, and Naja A. Mack

Morgan State University, Baltimore, MD 21251, USA

ABSTRACT

The accelerating demand for computing professionals, fueled by over 500,000 unfilled positions and a projected need for 1.7 million more by 2030 underscores the urgent need to rethink how computer science (CS) is introduced to learners, particularly during the formative middle school years. Yet, barriers such as limited early exposure, rigid curricula, and misconceptions about the field continue to hinder equitable access and sustained engagement. This concept paper introduces PyPro, a next-generation educational platform envisioned to transform the way students experience programming. PyPro integrates adaptive learning pathways, a conversational AI tutor, gamified challenges, and accessibility-first design to create a dynamic, inclusive, and student-centered environment for Python instruction. Based on the principles of personalized learning and interactive engagement, the platform reimagines CS education as a responsive, exploratory journey rather than a static instructional sequence. By continuously adapting to learner performance, offering on-demand support, and aligning content with student interests and needs, PyPro aims to cultivate computational confidence, deepen conceptual understanding, and promote long-term interest in CS. This paper explores PyPro not just as a tool, but as a conceptual model for how emerging technologies can reshape computer science education into a more equitable, engaging, and empowering experience for all learners.

Keywords: Adaptive learning, Gamification, Personalization, Virtual AI tutor

INTRODUCTION

The tech industry's rapid growth is threatened by a persistent shortage of skilled professionals, with over 500,000 computing jobs currently unfilled and demand expected to rise by 1.7 million roles by 2030. Yet many students face barriers to entering the field, including limited early exposure, lack of foundational education, and widespread misconceptions about computer science (CS). CS is often introduced too late in K–12 education, limiting the development of essential skills (Sullivan and Bers, 2019). Although after-school programs and summer camps offer valuable technical instruction, they often overlook the motivational and contextual factors needed for long-term engagement. Media portrayals further reinforce stereotypes, framing CS as inaccessible or suited only for a select few (Dou et al., 2020).

To address workforce demands, CS education must begin earlier, be engaging and culturally relevant, and extend beyond traditional classrooms. Intelligent Tutoring Systems (ITS) offer a promising solution by providing personalized, adaptive instruction based on students' prior knowledge and learning preferences. ITS dynamically adjust teaching strategies in real-time, supporting student progress and improving outcomes, especially among novice learners (Sarrafzadeh et al., 2008).

PyPro, a web-based platform, applies this model by offering interactive, hands-on coding experiences tailored to K–12 learners. Its real-world challenges, scalable lessons, and responsive feedback help students build both confidence and competence. By supporting early and inclusive CS engagement through adaptive tools, platforms like PyPro can help prepare a more diverse and capable workforce for the future tech landscape.

BACKGROUND

Interactive Learning Environments in Computer Science Education

Interactive learning environments in computer science education are web-based platforms designed to support programming instruction, especially in remote and asynchronous settings. These environments enable students to write, test, and compile code from any location while allowing instructors to monitor progress and provide feedback (Choy and Ng, 2004). Features such as automated grading, integrated analytics, and software agents help address common challenges in distance learning by fostering interaction and streamlining feedback. Platforms like Khan Academy, Replit, EdStem, and Codio demonstrate how interactive tools enhance coding practice, collaboration, and formative assessment, enriching the learning experience in computer science (Morrison and DiSalvo, 2014; Engkamat, Yii, and Gran, 2023; Eyitayo, Botha, and van der Westhuizen, 2021; Allen-Perez et al., 2025; Croft and England, 2019).

Each platform offers distinct strengths. Khan Academy provides JavaScript tutorials and gamified elements but offers limited Python depth (Morrison and DiSalvo, 2014). Replit is praised for its cloud-based, real-time collaboration and intuitive design (Engkamat et al., 2023; Eyitayo et al., 2021). EdStem blends an online IDE with discussion tools and automated feedback to encourage collaborative learning (Allen-Perez et al., 2025), while Codio supplies structured lessons and virtual environments to improve learning outcomes (Croft and England, 2019). PyPro builds upon these models by offering adaptive learning pathways, AI-driven feedback, and gamified progression tailored to K–12 learners, reflecting a shift toward more personalized and student-centered computer science education.

Intelligent Tutoring Systems in Computer Science

Intelligent Tutoring Systems (ITS) expand on interactive learning environments by providing adaptive, personalized instruction that simulates one-on-one tutoring. These systems leverage AI and cognitive models to tailor feedback and guidance based on individual student needs, promoting deeper conceptual understanding and long-term retention. Unlike general-purpose coding platforms, ITS focus on scaffolding problem-solving skills, offering step-by-step hints, real-time feedback, and error analysis. Systems like CTAT, JavaTutor, PythonTutor, ASK-ELLE, and CodeWorkout demonstrate various approaches to intelligent tutoring in computer science, each with strengths and limitations in content coverage, adaptability, and instructional depth.

CTAT enables educators to build tutors using example-tracing or cognitive models without programming expertise, though creating model-based tutors is resource-intensive (Aleven et al., 2016; Koedinger et al., 2004). JavaTutor enhances Java learning through interactive, dialogue-based tutoring but lacks cross-language support (Boyer et al., 2010; Sykes & Franek, 2003). PythonTutor offers code visualization across multiple languages but lacks adaptive feedback (Guo, 2013). ASK-ELLE provides structural code feedback in Haskell but may constrain diverse solutions (Gerdes et al., 2017; Olmer, Heeren, & Jeuring, 2014). CodeWorkout promotes fluency through drill-based exercises in Java and Python, though it could be strengthened with more open-ended challenges (Edwards & Murali, 2017). Collectively, these ITS illustrate the potential and trade-offs of AI-driven instruction in programming education.

Gamification

Gamification involves incorporating game-like elements into non-gaming contexts to increase motivation, engagement, and skill acquisition. In educational settings, particularly programming education, gamification has proven effective in sustaining learner interest and promoting deeper participation. Defined as "the process of game-thinking and game mechanics to engage users and solve problems" (Zichermann, as cited in Arnold, 2014), this approach integrates elements such as points, badges, levels, and interactive challenges to enhance learning experiences. As programming often requires sustained focus and iterative problem-solving, gamified platforms can provide the motivational scaffolding necessary to keep students engaged.

ClassCode and CodeCombat exemplify how gamification is applied in programming education. ClassCode offers interactive tutorials and real-time progress tracking to help students build JavaScript proficiency but lacks adaptive feedback and debugging support, which can limit deeper conceptual learning (Suzuki, Kato, & Yatani, 2020). CodeCombat presents a gamified, level-based environment where students use Python to solve progressively difficult challenges. It is particularly effective for learners transitioning from block-based to text-based coding. However, it has limited coverage of advanced programming concepts and requires a subscription and stable internet connection, which may restrict its accessibility in underserved regions (Choi & Choi, 2024). These tools highlight both the promise and the limitations of gamified learning in programming education.

Virtual Pedigogical Agents

AI-powered virtual tutors and Pedagogical Agents as Learning Companions (PALs) offer adaptive, personalized instruction and simulate peer-like interactions to enhance learning. These agents provide cognitive and emotional support, particularly valuable in distance and hybrid learning contexts, by serving as scalable alternatives to direct teacher engagement. To be effective, PALs must demonstrate authentic behaviors, appropriate communication styles, and responsive feedback, which contribute to perceived realism and learner trust (Yakubu et al., 2025; Kim, 2004).

Zhang et al. (2024) conducted a systematic review of pedagogical agents in K–12 education, identifying critical design elements such as agent realism, communication strategies, and instructional methods that drive student motivation and engagement. The review also highlights implementation challenges, including ethical risks, data privacy concerns, algorithmic bias, and integration into traditional classrooms. In parallel, Mojjada et al. (2024) examine the evolution of AI virtual tutors in higher education, tracing the transition from rule-based systems to advanced tools leveraging machine learning, natural language processing, and real-time analytics. These systems offer benefits like 24/7 access and personalized feedback but also face limitations in recognizing emotional cues, ensuring equitable access, and maintaining human-centered learning in increasingly automated environments.

Adaptive Learning

Adaptive learning integrates advanced technology with pedagogical strategies to personalize education in both K–12 and higher education settings. These systems dynamically adjust content, pacing, and feedback based on individual learner performance, offering benefits such as accelerated learning, targeted remediation, enhanced metacognitive skills, mastery-based progression, and interactive engagement. Researchers increasingly recognize adaptive learning as a dual innovation—both technological and instructional that supports differentiated instruction and promotes deeper, student-centered learning experiences (Mojjada et al., 2024).

The Intelligent Tutoring System for the Text Structure Strategy (ITSS) exemplifies adaptive learning in K–12 literacy education. ITSS helps struggling readers in grades 4–7 develop comprehension skills through tasks such as identifying signal words, classifying text structures, and rewriting passages, supported by pop-up hints and automated analysis. Studies report gains of up to two grade levels in reading proficiency. However, its reliance on a fixed sequence of tasks and static feedback limits adaptability to novel learner errors (Atun, 2020). In mathematics, ASSISTments provides middle school students with scaffolded hints and instant feedback, adjusting problem difficulty in real time. This approach led to a 75% improvement in standardized test scores compared to traditional homework (Roschelle, Feng, Murphy, and Mason, 2016). Despite its success, its emphasis on routine practice can limit creative exploration and problem-solving.

Emerging platforms like PyPro expand on these foundations by offering adaptive support for programming education across age groups. Using real- time learner data, PyPro reorganizes tasks and provides context-sensitive hints tailored to student performance. Unlike systems focused solely on accuracy or speed, PyPro encourages experimentation through optional "quests," promoting deeper conceptual understanding and building confidence through iterative, discovery-based learning.

Personalization

Personalized learning tailors instruction to individual students' needs, interests, and learning preferences, promoting flexibility, differentiation, and student agency. Supported by adaptive technologies and real-time performance tracking, personalization has been shown to enhance engagement and improve learning outcomes (Zia, 2024). As educational systems seek scalable and inclusive strategies, personalization is emerging as a central paradigm in both K–12 and higher education.

The SPARCS program exemplifies a personalized, problem-based approach to integrating computer science into middle school education. Designed to support teachers with limited computer science backgrounds, SPARCS improved educator confidence, content knowledge, and collaborative teaching practices. However, it encountered implementation barriers, including high support demands, limited scalability, and insufficient data on student outcomes (Siy et al., 2017). PyPro builds on this model by offering modular lessons, scaffolded tools, and AI-driven feedback that adapt in real time to student needs, creating a more scalable and data-informed framework for personalized CS instruction.

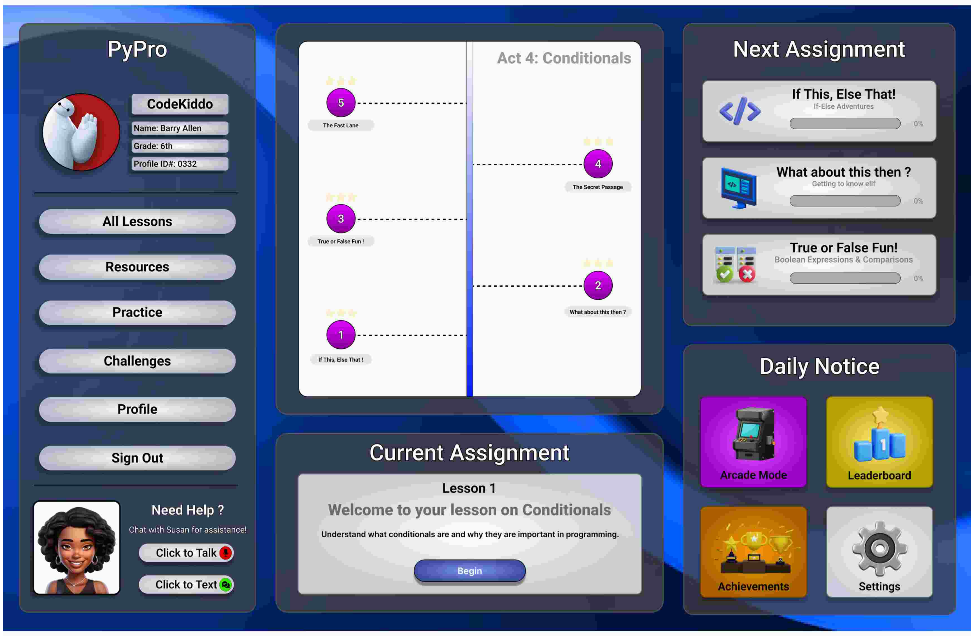
Similarly, an intelligent tutoring system designed by Brad'ač et al. (2022) uses expert systems and fuzzy logic to create personalized study plans for English language learners on Moodle. It adapts to the prior knowledge and sensory preferences of students using the VARK model. While innovative, the system is constrained by Moodle's limited adaptivity and the rigidity of rule-based logic. PyPro addresses these limitations with machine learning refined pathways, dynamic content delivery, and reduced instructor overhead. Its modular infrastructure enables broader subject integration, real-time data collection, and media-rich customization extending personalization beyond static rules into responsive, learner-centered experiences.

System Overview

PyPro is a conceptual educational platform aimed at enhancing Python programming instruction for middle school students through a dynamic, inclusive, and student-centered learning environment. By incorporating key elements such as **gamification**, **adaptive learning**, a **virtual AI tutor**, **accessibility**, and **personalized instruction**, PyPro seeks to increase student engagement, support diverse learning needs, and cultivate sustained interest in computer science.

Adaptive Learning

PyPro reimagines adaptive learning by offering a dynamic and responsive framework that personalizes programming education to the evolving needs and abilities of each student. Moving beyond the limitations of a uniform instructional path, PyPro introduces an intelligent system that adjusts in real time, tailoring content delivery, pacing, and support to match individual learner profiles. When students face difficulties, the platform intervenes with scaffolded assistance ranging from contextual hints and step-by-step walkthroughs to targeted practice exercises, creating an environment that nurtures persistence, builds confidence and supports concept mastery.



Name: Barry Allen
Grade: 6th
Profile ID#: 0332

Lesson 1
Welcome to your lesson on Conditionals
Understand what conditionals are and why they are important in programming.

If This, Else That!
If-Else Adventures
0%

What about this then ?
Getting to know elif
0%

True or False Fun!
Boolean Expressions & Comparisons
0%

Figure 1: The PyPro student dashboard at a glance.

For learners who progress quickly, PyPro offers a suite of advanced challenges that promote critical thinking, creative exploration, and higher-order problem solving. These tasks are designed not only to maintain engagement but to deepen students' understanding through complexity and experimentation. By continuously adapting to performance data, PyPro ensures instruction remains appropriately rigorous and relevant, reinforcing foundational skills while expanding advanced competencies.

This adaptive model fosters a growth-oriented learning culture where every student regardless of their starting point is met with instruction that is both supportive and ambitious. The result is a personalized, student-driven experience that empowers learners to build lasting proficiency, develop computational confidence, and engage deeply with the discipline of computer science.

Virtual AI Tutor

PyPro's virtual AI tutor transforms programming education by offering interactive, adaptive, student-centered guidance. Learners engage in real-time conversations to receive hints, clarify concepts, and tackle challenges with tailored support. Acting as an evolving mentor, the tutor adapts its strategies based on student growth, fostering autonomy, critical thinking, and problem-solving skills. By emphasizing deep understanding over rote learning, PyPro promotes a personalized, mastery-based approach that evolves with each learner.

Gamification

Gamification is a central element of PyPro's instructional design, strategically employed to transform the process of learning to code into an engaging and immersive experience. By embedding game-based mechanics such as points, badges, and progress levels into the educational framework, PyPro reframes programming from a routine academic task into a compelling, interactive journey. Learners embark on coding 'quests', ascend leaderboards, and earn achievements as they master core concepts, strengthening skills through a progression of meaningful milestones. A distinguishing feature of PyPro is its 3rd-person mini-gameplay, where students control avatars navigating narrative-driven environments that demand real-time application of programming logic, including loops, conditionals, and debugging. This fusion of storytelling and problem-solving fosters deeper conceptual retention. Moreover, features like daily practice streaks and unlockable content cultivate consistent engagement by blending structured instruction with elements of exploration and play. Collectively, these gamified strategies sustain motivation, bolster confidence, and nurture long-term interest in computer science.

Accessibility

PyPro is built on a foundation of accessibility and equity, ensuring that every student, regardless of ability, background, or learning style, has a meaningful opportunity to engage with coding. The platform features a user friendly interface designed with universal design principles, offering adjustable text sizes, high contrast themes, and text to speech functionality to support visually impaired learners. For students with motor impairments, PyPro includes alternative input methods such as keyboard navigation, switch access, and voice commands, enabling smoother, more independent interaction with the platform. Customizable pacing options empower learners to move through material at a speed that suits their individual needs, reducing anxiety and promoting mastery at every stage. Closed captions and visual prompts support deaf and hard of hearing students, as well as English language learners, ensuring that all content is accessible and clearly communicated. By thoughtfully integrating these features, PyPro removes barriers and creates a supportive, inclusive environment where all learners can thrive, explore coding with confidence, and realize their full potential.

Personalization

Personalization is central to PyPro, creating meaningful, motivating learning experiences tailored to individual interests, skill levels, and preferences. Students can pursue guided learning paths or open-ended challenges in areas like game development, chatbots, and creative coding. Personalized tutorials, project-based challenges, and real-time progress tracking foster engagement, while adaptive recommendations ensure students stay challenged without overwhelm. By centering learning around the student, PyPro builds autonomy, confidence, and long-term mastery.

CONCLUSION

In conclusion, PyPro represents a forward-thinking approach to Python learning, combining adaptive learning, an interactive AI tutor, gamification, and accessibility features to offer a truly engaging and inclusive educational experience. As a proof of concept, the platform demonstrates the potential of leveraging innovative tools to address the need for accessible coding education, particularly for students from underrepresented backgrounds. Looking forward, PyPro will continue to evolve with key updates. The adaptive learning system will become even smarter, offering increasingly personalized challenges that respond to each student's unique learning journey. The AI tutor will incorporate more advanced natural language processing, enabling a more conversational and context-aware approach to guiding students. New collaborative features will encourage teamwork by allowing students to work together on projects, promoting peer learning and the development of collaborative problem-solving skills. Additionally, accessibility will be expanded with multi-language support and enhanced assistive tools to ensure inclusivity for all learners.

To ensure that PyPro meets the real-world needs of both students and educators, we will engage in a participatory design process. Feedback from teachers and students will be incorporated during the development phases, providing critical insights that will shape the evolution of the platform. Furthermore, we will conduct a usability study to evaluate the effectiveness of PyPro's features and user experience, identifying areas for improvement in terms of engagement, accessibility, and overall usability. This study will help ensure that PyPro remains intuitive, effective, and inclusive, catering to a diverse range of learning styles and needs. With continuous testing, user feedback, and iterative development, PyPro will bridge the gap between student interest in coding and long-term success in computer science.

REFERENCES

Aleven, V., McLaren, B. M., Sewall, J., Van Velsen, M., Popescu, O., Demi, S., Ringenberg, M., & Koedinger, K. R. (2016). Example-tracing tutors: Intelligent tutor development for non-programmers. *International Journal of Artificial Intelligence in Education, 26*, 224–269.

Allen-Perez, G., Millan, L., Nghiem, B., Wu, K., Shah, A., & Soosai Raj, A. G. (2025). An analysis of students' testing processes in CS1. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education* (Vol. 1, pp. 46–52). ACM.

Arnold, B. J. (2014). Gamification in education. *Proceedings of the American Society of Business and Behavioral Sciences, 21* (1), 32–39.

Atun, H. (2020). Intelligent tutoring systems (ITS) to improve reading comprehension: A systematic review. *Journal of Teacher Education and Lifelong Learning, 2* (2), 77–89. Retrieved from https://dergipark.org.tr/en/pub/tell/issue/58491/757329.

Boyer, K., Ha, E. Y., Phillips, R., Wallis, M., Vouk, M., & Lester, J. (2010). Dialogue act modeling in a complex task-oriented domain. In *Proceedings of the SIGDIAL 2010 Conference* (pp. 297–305).

Brad'ač, V., Smolka, P., Kotyrba, M., & Prdek, T. (2022). Design of an intelligent tutoring system to create a personalized study plan using expert systems. *Applied Sciences, 12* (12), 6236.

Choi, W. C., & Choi, I. C. (2024). Investigating the effect of the serious game CodeCombat on cognitive load in Python programming education. In *2024 IEEE World Engineering Education Conference (EDUNINE)* (pp. 1–6). IEEE.

Choy, S.-O., & Ng, S.-C. (2004). An interactive learning environment for teaching and learning of computer programming. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies* (pp. 848–849). IEEE.

Croft, D., & England, M. (2019). Computing with Codio at Coventry University: Online virtual Linux boxes and automated formative feedback. In *Proceedings of the 3rd Conference on Computing Education Practice* (pp. 1–4). ACM.

Dou, R., Bhutta, K., Ross, M., Kramer, L., & Thamotharan, V. (2020). The effects of computer science stereotypes and interest on middle school boys' career intentions. *ACM Transactions on Computing Education, 20* (3), 1–15.

Edwards, S. H., & Murali, K. P. (2017). CodeWorkout: Short programming exercises with built-in data collection. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 188–193).

Engkamat, A., Yii, M. L., & Gran, S. S. (2023). Replit: A simple approach to real-time collaborative coding.

Eyitayo, O. (2021). Online compiler, IDE, interpreter, and REPL for multiple programming languages.

Gerdes, A., Heeren, B., Jeuring, J., & Van Binsbergen, L. T. (2017). Ask-Elle: An adaptable programming tutor for Haskell giving automated feedback. *International Journal of Artificial Intelligence in Education, 27*, 65–100.

Guo, P. J. (2013). Online Python Tutor: Embeddable web-based program visualization for CS education. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (pp. 579–584).

Kim, Y. (2004). *Pedagogical agents as learning companions: The effects of agent affect and gender on student learning, interest, self-efficacy, and agent persona* (Doctoral dissertation, The Florida State University).

Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B., & Hockenberry, M. (2004). Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In *International Conference on Intelligent Tutoring Systems* (pp. 162–174). Springer.

Mojjada, H., Chand, N. G., Lakshmipriyanka, A., & Sravya, T. (2024). The evolution of AI virtual tutors in modern higher education.

Morrison, B. B., & DiSalvo, B. (2014). Khan Academy gamifies computer science. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 39–44). ACM.

Nye, B. D., Graesser, A. C., & Hu, X. (2014). AutoTutor and family: A review of 17 years of natural language tutoring. *International Journal of Artificial Intelligence in Education, 24*, 427–469.

Olmer, T., Heeren, B., & Jeuring, J. (2014). Evaluating Haskell expressions in a tutoring environment. *arXiv preprint arXiv:1412.4879*.

Roschelle, J., Feng, M., Murphy, R. F., & Mason, C. A. (2016). Online mathematics homework increases student achievement: A randomized field trial of ASSISTments. *AERA Open, 2* (4), 1–12. https://doi.org/10.1177/2332858416673968

Rus, V., D'Mello, S., Hu, X., & Graesser, A. (2013). Recent advances in conversational intelligent tutoring systems. *AI Magazine, 34* (3), 42–54.

Sarrafzadeh, A., Alexander, S., Dadgostar, F., Fan, C., & Bigdeli, A. (2008). "How do you know that I don't understand?" A look at the future of intelligent tutoring systems. *Computers in Human Behavior, 24* (4), 1342–1363.

Siy, H., Dorn, B., Engelmann, C., Grandgenett, N., Reding, T., Youn, J.-H., & Zhu, Q. (2017). SPARCS: A personalized problem-based learning approach for developing successful computer science learning experiences in middle school. In *2017 IEEE International Conference on Electro Information Technology (EIT)* (pp. 611–616). IEEE.

Sullivan, A., & Bers, M. U. (2019). Computer science education in early childhood: The case of Scratch Jr. *Journal of Information Technology Education: Innovations in Practice, 18*, 113.

Suzuki, R., Kato, J., & Yatani, K. (2020). ClassCode: An interactive teaching and learning environment for programming education in classrooms. *arXiv preprint arXiv:2001.08194*.

Sykes, E. R., & Franek, F. (2003). An intelligent tutoring system prototype for learning to program Java™. In *Proceedings of the 3rd IEEE International Conference on Advanced Technologies*.

Yakubu, M. A., Sain, Z. H., Lawal, U. S., & Hakim, M. A. R. (2025). Students' perceptions of artificial intelligence as a virtual tutor and self-efficacy in learning. *Indonesian Journal of Artificial Intelligence (IJAI), 1* (1), 1–11.

Zhang, S., Jaldi, C. D., Schroeder, N. L., L'opez, A. A., Gladstone, J. R., & Heidig, S. (2024). Pedagogical agent design for K–12 education: A systematic review. *Computers & Education*, 105165.

Zia, A. S. (2024). Personalized learning in K–12 education: Strategies, challenges, and future directions. *Zeal Journal of Multidisciplinary Research, 1* (2), 32–38.