DevQuizzer: An Adaptive Game-Based Learning System for Personalized Programming Instruction

Femi Elegbeleye[1], Isong Bassey[2]

[1,2]North West University, 2745, Mahikeng Campus, South Africa

E-mail: phernet123@gmail.com[1], isongb77@gmail.com[2]

Abstract. Over the years, balancing personalized support with learner motivation has proven difficult in programming education. Traditional instructional approaches seldom adapt to individual needs, often resulting in uneven engagement and cognitive overload. Gamified settings can enhance motivation, but many lack adaptive mechanisms informed by motivational frameworks such as Self-Determination Theory (SDT). As a result, few systems combine reinforcement learning (RL) with probabilistic learner modelling to deliver scalable, personalized learning pathways. This paper proposes DevQuizzer, an adaptive game-based learning (AGBL) system designed to support personalized programming instruction. The system combines RL and Bayesian Networks (BNs) models to dynamically adjust task difficulty, feedback, and learning pathways according to learner performance and engagement. A dual-mode instructional design: code-first and theory-first, accommodates various learning preferences, while gamified elements sustain motivation. We evaluated AGBL with students from five universities using a mixed-methods approach that combined quantitative measures of usability, engagement, and adaptivity with qualitative insights into learner experiences. Results confirmed the system's reliability and construct validity, with learners reporting high satisfaction, autonomy, and perceived learning gains. The findings show the potential of hybrid RL-BN models to provide transparent, interpretable, and scalable adaptive learning environments for programming education.

Keywords: Adaptive Game-Based Learning, Reinforcement Learning, Bayesian Networks, Programming Education, Personalisation, Self-Determination Theory.

1. Introduction

Computer programming has moved beyond its origins as a specialized skill. In present-day higher education (HE), it is recognized as a core discipline. It now supports computational thinking, problem-solving, and analytical reasoning across disciplines [1], [2]. Students in both computing and non-computing programs increasingly need programming competencies. These skills are indispensable. They ensure effective participation in academic and professional environments increasingly shaped by technology [3], [4], [5]. Programming education involves designing algorithms, implementing solutions in programming languages, and systematically testing and debugging code [1]. Despite its importance, many learners struggle with abstract concepts, complex syntax, and limited opportunities for practice and feedback. These challenges together hinder confidence and performance [2], [3], [4], [6]. Currently, traditional approaches, such as lecture-based teaching and theory-oriented laboratory exercises, often fail to connect conceptual understanding with practical application [7], [8]. Insufficient feedback and differences in prior experience further contribute to disengagement and weak performance, particularly among novice learners and non-computing students [3], [4], [6]. These issues highlight the need for interactive, learner-centred methods that promote engagement and deepen conceptual understanding in programming education [8], [14].

In response, game-based learning (GBL) and adaptive learning systems (ALS) have emerged as strategies to address these barriers. GBL integrates challenges, feedback, rewards, and progressive levels to enhance engagement, persistence, and conceptual understanding [10]. However, many GBL implementations provide uniform experiences that overlook differences in prior knowledge, learning pace, and skill development. ALS complements GBL by personalizing instruction through learner-modelling and data-driven adaptation. It adjusts content, difficulty, and learning pathways to improve comprehension and efficiency [11], [12]. The combination of these approaches is known as adaptive GBL (AGBL). This method combines motivational and adaptive mechanisms to deliver interactive, personalized environments that respond continuously to learner progress [13]. AGBL draws on Self-Determination Theory (SDT). By supporting autonomy and competence, SDT helps maintain continuous engagement in demanding fields like programming [14].

Recent empirical studies show that including adaptive support within educational games improves programming outcomes. For instance, serious games equipped with personalized hints, feedback, and performance-based task selection enhance both efficiency and engagement, particularly for beginner programmers [15], [16]. By tailoring feedback and adjusting task difficulty to individual needs, AGBL offers a structured framework. At the same time, it remains flexible enough to support both skill acquisition and motivation in programming education. Therefore, this paper presents an AGBL framework for teaching programming to undergraduate students, with particular focus on non-computing students. The goal is to improve motivation and engagement. The contributions of this study are as follows:

1) Proposes an AGBL framework that integrates game-based learning mechanisms with adaptive instructional strategies to support individualized learning.
2) Develops and implements DevQuizzer, a system that adjusts task difficulty, feedback, and learning pathways according to learner performance.
3) Empirically evaluates the framework and system to assess their impact on learner engagement, progression, and programming performance in undergraduate education.

The remaining parts of the paper are organized as follows: Section 2 presents the related work, Section 3 discusses the methodology of the study, Section 4 presents the framework design, and Section 5 presents the findings from the evaluation. Section 6 discusses the findings, and Section 7 concludes the paper.

2. Related Works

This section discusses previous articles and related literature on the topic, summarizing major findings, approaches used, and existing research gaps.

2.1. Programming Challenges

Computer programming is a core skill in HE, supporting problem-solving, reasoning, and computational thinking across disciplines [2], [17]. Notwithstanding its importance, many students, particularly beginners, find programming difficult due to the cognitive demands of learning multiple concepts at once. Learners must understand syntax and logic, design algorithms, and use development tools simultaneously, which often leads to cognitive overload, stress, and early disengagement [2], [18]. A further challenge is the abstraction gap [18]. Several students can solve problems conceptually but struggle to translate those solutions into executable programs. Novices often lack the mental frameworks needed to organize code into meaningful structures. As a result, their knowledge remains fragile, easily forgotten, misapplied, or never fully integrated [18], [19]. Moreover, traditional instructional approaches, including lecture-based teaching and theory-focused labs, often fail to bridge this gap. Limited practice opportunities, insufficient feedback, and differences in prior experience all contribute to low confidence and disengagement. These factors also lead to high failure rates, particularly among non-computing students [17], [20].

To deal with these challenges, research has explored active, student-centred approaches that promote engagement and deeper understanding. One example is the use of agile-inspired methods such as Scrum and Kanban. These approaches introduce iterative, collaborative learning cycles that promote teamwork and systemic thinking, leading to improved engagement and more effective conflict resolution [17], [18].

Pair programming supports collaborative problem-solving, resulting in enhanced code quality, peer learning, and greater confidence [2]. In the same vein, flipped classrooms encourage autonomy and deep learning, allowing personalized pacing and higher levels of student activity [18]. These approaches also support social and emotional competencies such as persistence, teamwork, and accountability. Such skills are essential for success in programming and professional practice. However, while these approaches show promise, many studies note that traditional curricula remain rigid. They often fail to accommodate diverse student needs, prior knowledge, and learning speeds [2], [17]. This has created growing interest in adaptive and GBL approaches, which aim to personalize instruction and motivation.

2.2. Adaptive learning in HE

ALS personalizes instruction by adjusting content, difficulty, and feedback according to individual learner characteristics, performance, and behaviour. Unlike traditional "teach-to-the-middle" strategies, which target the average student, ALS accounts for differences in proficiency, prior knowledge, and cognitive capacity [21], [22]. Standard ALS architecture includes three modules. The content module organizes domain knowledge, the instructor module drives adaptation and sequencing, and the learner module models individual knowledge, skills, and cognitive load [21], [22]. Increasingly, these systems incorporate physiological and behavioural data to monitor cognitive effort, ensuring instructional pacing remains responsive to real-time learner needs [22]. For example, Almetnawy et al. [53] identified three learner profiles: low-, medium-, and high-proficiency students, using a proficiency formula (P) that evaluates a student's score (S) against the total number of questions (N), adjusted by the ratio of actual time (τa) to expected time (τe):

$$P = \frac{S}{N} + \left(1 + \frac{\tau a}{\tau e}\right)$$

Based on this calculation, the system determines the difficulty distribution D = (E, M, D), where E is easy, M is moderate, and D is difficult questions [53].

The computational backbone of many ALS relies on Bayesian networks (BNs) and reinforcement learning (RL) to manage the dynamic, non-linear nature of student interactions. Bayesian approaches, such as Bayesian Knowledge Tracing (BKT), model learner knowledge probabilistically, updating mastery estimates based on observed responses [23], [24]. These models are valuable for cognitive diagnosis and predicting skill acquisition over time, including parameters such as "slipping" or "guessing" [23], [25]. RL, in contrast, is used to frame instruction as a sequential decision problem, mapping learner states to instructional actions to maximize long-term outcomes. Algorithms such as Q-Learning, Actor-Critic methods, and Deep RL enable systems to schedule content dynamically. They adjust difficulty and improve learning paths while balancing knowledge acquisition against cognitive load [26], [27].

Table 1. Summary of RL and BNs integrations

Feature	BNs	RL	Hybrid Integration
Primary goal	Diagnosis: Learner state estimation [23],[24]	Optimization: Best next action selection [26]	BN provides belief state; RL selects content [26],[29]
Handling data	Probabilistic, causal modeling [23],[30]	Trial-and-error with reward feedback [26]	Learner state + reward-based action selection
Common use	Knowledge tracing, skill mastery, dropout prediction [23],[32]	Instruction sequencing, adaptive path planning, resource	Personalized, responsive learning environments

Feature	BNs	RL	Hybrid Integration
		recommendation [26]	
Strength	Interpretability, explainability	Optimization for long-term learning	Combines diagnosis with proactive personalization
Limitations	Binary or simplified states, static updates	Black-box models, cold-start problem [31],[32]	Complexity requires robust data privacy measures [28][31]

A notable strength of combining RL with BNs lies in the probabilistic modeling they enable. BNs provide a "belief state" that represents learner knowledge and supports adaptive decision-making. This state guides the RL agent's selection of optimal instructional actions, enabling ALS to diagnose what a student knows and prescribe the next step most likely to enhance mastery [26], [28], [29]. Table 1 summarizes BNs-RL integration. Several empirical studies have shown substantial gains. For instance, a dual-stream neural architecture combining deep knowledge tracing with cognitive load estimation achieved 87.5% prediction accuracy, improved learning efficiency by 24.6%, and reduced abandoned learning sessions by 43% [22]. Despite their effectiveness, ALS face ongoing challenges, including the explainability of RL recommendations, the cold-start problem for new learners, and privacy concerns related to granular data collection [11], [28], [31], [32]. However, combining probabilistic modelling with RL approaches creates an effective framework for personalized instruction in HE. This approach is especially valuable in programming, where learners differ widely in cognitive load and prior knowledge.

2.3. Game-based learning in HE

In recent years, GBL has become an important approach in programming education, shifting instruction from passive delivery to active, participatory experiences. Kenwright [36] highlighted that GBL transforms traditional curricula into narrative-driven journeys where learners act as participants rather than passive recipients. By utilizing the motivational aspects of games, GBL improves engagement, reduces learning anxiety, and supports active participation. Serious games, in particular, teach programming concepts through interactive experiences, turning learners into problem solvers [33], [34]. The core elements of GBL include gamification mechanics such as points, badges, levels, and leaderboards, which provide immediate feedback and reward progress, helping students remain motivated [33], [35], [39]. Narrative and story-driven modules connect programming tasks to real-life contexts, supporting the understanding of abstract concepts [35], [36]. Collaborative and competitive features, such as coding duels, guilds, or team challenges, promote peer learning and create social learning environments [37].

GBL has been applied across multiple levels of programming education. Introductory games such as Codey and NanoDoc support syntax, iteration, and logic through puzzles and 3D activities, helping beginners reduce anxiety and build foundational skills [38]. For advanced topics, games such as EscapeScript [37] and Gridlock [34] teach web development or system engineering by engaging students in real-world problem solving and logic controller design [39]. Some platforms also integrate adaptive features, including BNs, RL, and intelligent tutoring systems (ITS). These tools provide personalized feedback, adjust difficulty, and guide learning paths [33], [34], [40]. These features help maintain appropriate cognitive challenges and address individual learner needs.

Empirical evidence shows the effectiveness of GBL. Chandrasekara et al. [39] reported a 40% increase in engagement metrics and a 35% improvement in coding competency through gamification elements such as points and leaderboards. Kenwright [36] equally observed enhanced retention and emotional resonance in immersive game-based environments, while Almetnawy et al. [53] proved that role-based progression increased persistence in technical tasks. There are several theoretical frameworks that further support GBL. SDT explains how games satisfy needs for competence, autonomy, and relatedness; Bloom's Taxonomy aligns gamified tasks with cognitive levels; and cognitive load theory guides design

to minimize unnecessary mental effort [40], [41], [53]. In general, GBL integrates game mechanics into educational environments to improve engagement, motivation, and conceptual understanding. In programming education, skills that are often underdeveloped in traditional lecture-based approaches, such as abstract thinking, problem-solving, and continuity, are supported [33], [41]. Table 2 summarizes some of the GBL components and effectiveness in HE.

Table 2. Summary of GBL effectiveness

GBL Component	Instructional Goal	Examples / Mechanisms	Observed Outcomes
Gamification Elements	Motivation, engagement, immediate feedback	Points, badges, XP, levels, leaderboards	Increased participation, sustained effort, higher retention [33],[35]
Narrative & Storytelling	Conceptual understanding, immersion	Story-driven challenges linking programming tasks to real-life scenarios	Deeper understanding, improved problem-solving [34],[36]
Collaborative & Competitive Features	Peer learning, social engagement	Coding duels, guilds, group challenges	Enhanced teamwork, community building, practical coding experience [37]
Introductory Programming Concepts	Foundational syntax and logic	Code puzzles, 3D maze games (Codey, NanoDoc)	Reduced beginner anxiety, better grasp of basic structures [38]
Web Development & Applied Programming	Practical skills, real-world problem solving	HTML/CSS/JS games (EscapeScript)	Improved applied skills, confidence in web development [37],[39]
Advanced Systems & Engineering	Higher-order logic, algorithmic thinking	Logic controller simulations (Gridlock)	Improved system design skills, complex problem solving [34]
Adaptive & Intelligent Feedback	Personalized learning, pacing	BNs, RL, ITS hints	Tailored challenges, optimized cognitive load, better retention [33],[34],[40]

2,4. Adaptive Game-Based Learning in HE

AGBL combines the motivational strengths of GBL with the personalized capabilities of ALS. While GBL improves engagement and continuity, ALS ensures that instructional content, feedback, and task difficulty are tailored to individual learners [21], [33]. This integration prevents meaningless engagement by linking game mechanics to measurable pedagogical progress [36], [39]. Role-based progression, as demonstrated by Almetnawy et al. [53], fulfils psychological needs for competence and autonomy, while incremental challenges sustain motivation. Moreover, AGBL systems combine game environments with learner modelling, adaptive sequencing, and feedback. Intelligent models such as BNs, RL, etc., track knowledge, estimate cognitive load, and plan optimal learning paths in real time [26], [33]. For example, coding puzzles can be dynamically adjusted to maintain optimal challenge, blending motivational and cognitive adaptation [38].

Empirical studies show that AGBL improves programming outcomes compared to GBL or ALS alone. Toukiloglou and Xinogalos [15] and Hare et al. [16] reported gains in efficiency, test scores, and persistence. Chandrasekara et al. [39] observed significant improvements in coding competency and engagement, while Almetnawy et al. [53] demonstrated increased persistence with role-based progression. Despite these benefits, challenges remain in technical complexity, data requirements, and explainability [31], [32]. Evidence is often limited to small-scale studies, with interventions focused mainly on beginners. In addition, real-time adaptation and cold-start problems also continue to hinder system design [26], [33]. These gaps show the need for more integrated, scalable approaches. This study proposes an integrated AGBL system for undergraduate programming that integrates RL and BN to guide instructional pathways. It incorporates game mechanics to strengthen learner engagement and persistence.

2.5. The Self-Determination Theory

SDT explains motivation in education through three core needs: autonomy, competence, and relatedness [45]. In programming education, autonomy is promoted by allowing learners to choose tasks and pace [46], [47]. Competence is supported through scaffolded challenges and adaptive feedback [45], [48], and relatedness is promoted via collaboration and peer interaction [46], [49]. When these needs are met, students show intrinsic motivation, leading to continuity, problem-solving, and retention [46], [47], [50]. Figure 1 summarizes SDT's core components adapted to programming. It is the psychological foundation for long-term engagement in technical subjects.

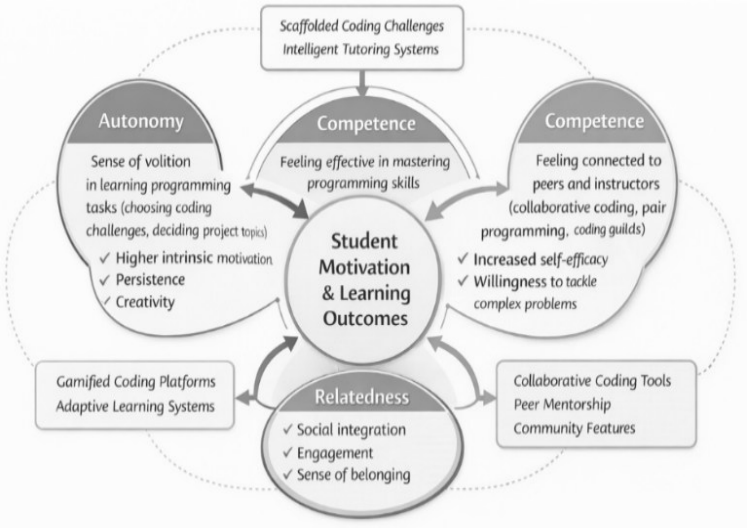

Figure 1. SDT applied to programming.

As shown in Figure 1, instructional design informed by SDT relies on autonomy-supportive teaching, structured challenges with immediate feedback, and collaborative learning communities [48], [50]. In practice, autonomy-supportive teaching involves providing meaningful choices and explaining why a task is relevant. It encourages students to take ownership of their problem-solving process. This approach aligns closely with the capabilities of adaptive and gamified platforms [48]. Competence is strengthened through adaptive difficulty that ensures learners progress at a steady, manageable rate [47]. Relatedness is improved by collaborative projects and mentorship, which reduce the isolation often felt by beginner coders [50]. These principles are not just theoretical but serve as a blueprint for personalizing difficulty and encouraging sustained effort. By applying SDT, programming education can produce self-regulated learners who continue through complex challenges to achieve deeper mastery [51], [52].

3. Methodology

In this study, we designed and evaluated AGBLS to improve engagement, personalization, and motivation in programming. Following the engineering and empirical cycles or design science research (DSR) [56], [57], the study involved two stages: system design and implementation, and empirical evaluation across five universities. The system combined RL and BNs to personalize instruction and sustain engagement. RL was used to optimize task sequencing by adjusting exercise difficulty in real time, while BNs modelled learner knowledge to capture both uncertainty and progression. In the same vein, game-based elements, including levels, rewards, and interactive challenges, were incorporated to support motivation. In this framework, learners could follow either a code-first pathway, which emphasizes practice, or a theory-first pathway, which focuses on conceptual understanding. The system monitors learners' behaviour and guides them toward appropriate tasks, ensuring personalized support for diverse learners.

The system was empirically evaluated involving 172 undergraduate and postgraduate students from computing and non-computing disciplines, as summarized in Figure 2. Participants completed programming exercises and game-based tasks before responding to an online survey assessing usability, engagement, satisfaction, and the impact of adaptive feedback. Responses were recorded on a five-point Likert scale, with completion rates exceeding 90%. Ethical approval was obtained, and participation was voluntary and anonymous. Furthermore, survey data were analysed using IBM SPSS Statistics. Quantitative responses were coded numerically, with reliability assessed using Cronbach's alpha (>0.70). Descriptive statistics, correlation, regression, and factor analysis examined relationships among variables. Likewise, qualitative responses were coded to identify themes such as enjoyment, support, and challenges, with representative quotes used to illustrate findings.

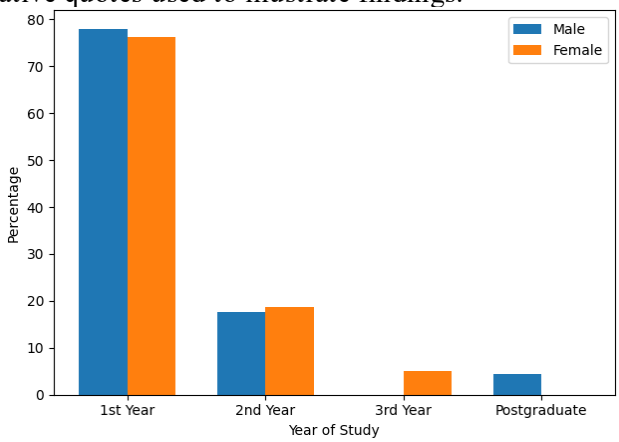

Figure 2. Gender distribution of respondents by year of study

4. AGBLS Framework

The proposed AGBLS framework was developed to support personalized programming instruction by integrating within a gamified environment. The framework addresses challenges of learner engagement, motivation, and progression in programming education, particularly for non-computing students. It applies adaptive learning principles by aligning pedagogical objectives with intelligent algorithms and interactive game mechanics. The requirements of the system were derived from prior literature and needs analysis [5], [54], [55]. Functional requirements included adaptive content delivery, interactive exercises, and learning analytics, while non-functional requirements addressed scalability, security, and usability.

The framework was implemented as DevQuizzer, involving three primary actors: learners, instructors, and administrators. Learners engage with quizzes, coding exercises, adaptive assessments, and a Unity-powered mini-game linked to RL and BN algorithms, receiving immediate feedback and tracking progress through dashboards. Instructors manage content and monitor learner performance, while administrators oversee accounts, configuration, and analytics. Figure 3 illustrates these interactions in the use case model, particularly the learners. After signing in, DevQuizzer consults the adaptive engine,

which retrieves the learner profile and generates personalized activity recommendations. Learner responses are recorded, allowing the adaptive engine to update the learner model, provide feedback, and suggest subsequent activities, forming a continuous adaptive learning cycle.

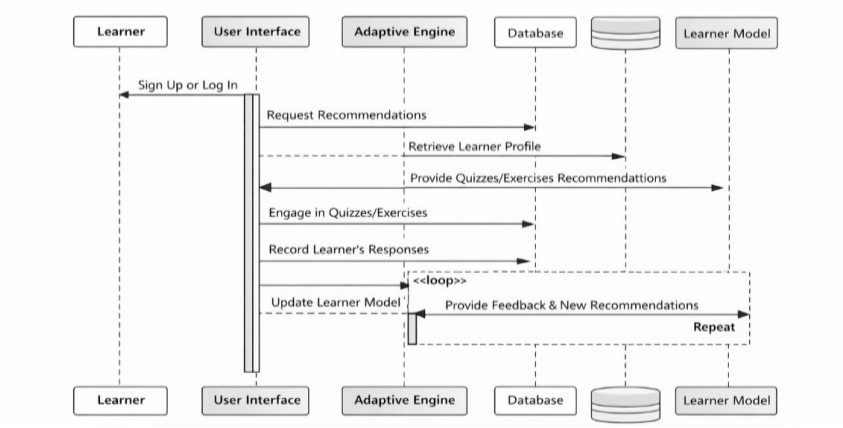

Figure 3. Learner interaction flow

The DevQuizzer architecture combines adaptive learning with gamified activities through a web interface. Learning strategies are personalized through a pedagogical adaptation engine, while the adaptive game engine regulates difficulty and feedback in real time. Again, student information is stored in two models: the learner model (knowledge, skills, performance) and the player model (gaming behaviours, engagement). A content repository provides curriculum-aligned materials, and gamification features such as leaderboards and badges sustain motivation. The monitoring and feedback module updates learner models in real time. Figure 4 presents the proposed AGBLS framework, combining BN-based learner modelling. The RL mechanisms and gamified interaction jointly enable personalized pathways and increased student engagement, as shown in Figure 5.

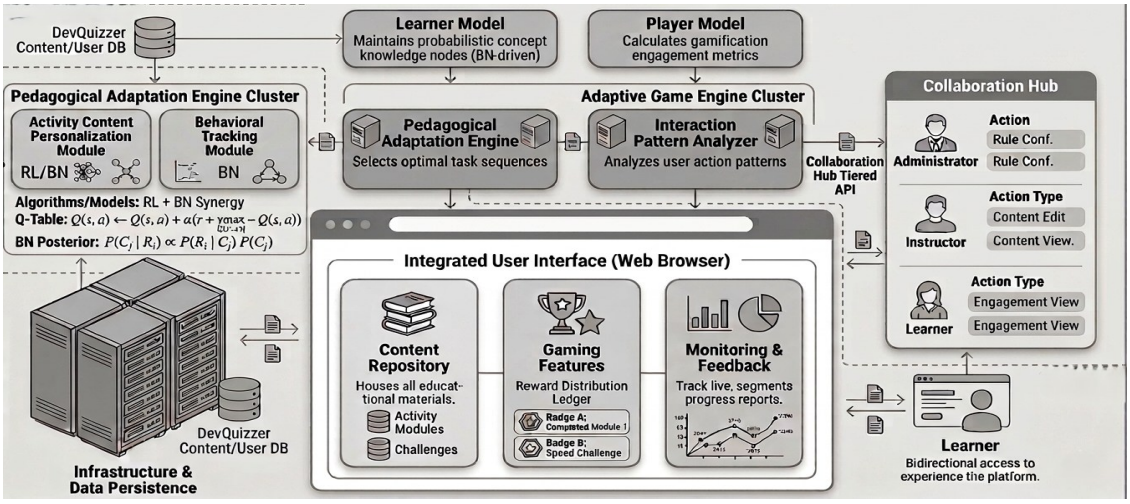

Figure 4. AGBLS framework

As discussed above, RL and BN form the intelligent core of DevQuizzer that allows personalized and adaptive learning. RL functions as the decision-making engine, selecting learning activities or game challenges to maximize learner progress and engagement. It observes the learner's current state, derived from the BN knowledge model, and applies an ε-greedy strategy to balance exploration and exploitation. Task outcomes generate rewards, strengthening successful strategies and guiding future task selection. In

parallel, BN models the learner's evolving knowledge by representing probabilistic relationships between concepts, questions, and responses. Each concept is assigned a prior probability representing the learner's initial understanding. As learners respond to tasks, the BN updates posterior probabilities using Bayes' rule. When a learner's understanding falls below a threshold, revision tasks are recommended; when mastery is demonstrated, progression to advanced concepts occurs.

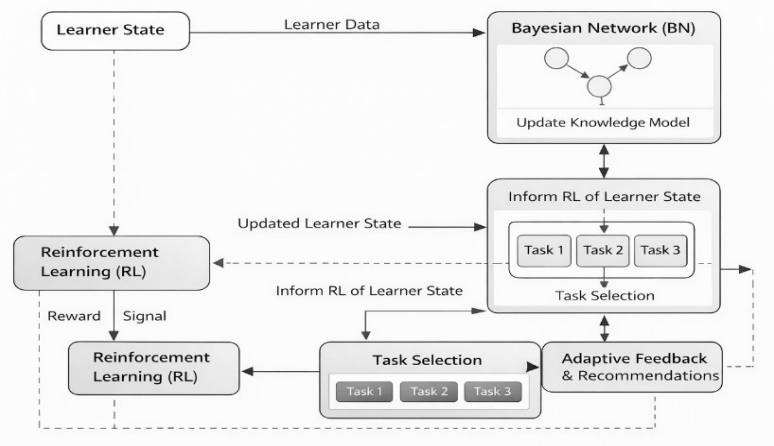

Figure 5. Personalized adaptive learning in DevQuizzer

The hybrid RL-BN approach combines these mechanisms to provide dynamic task sequencing and continuous adaptation. The interaction between them is presented in Figure 5. The BN informs the RL agent about the learner's current knowledge state, which the RL component uses to select the most suitable tasks. Learner interactions are recorded, including correctness, response time, and engagement metrics. These data iteratively update the RL Q-values and BN knowledge model, ensuring the system continuously adapts to each learner's performance. The algorithm is captured in Figure 5.

AGBLS - DevQuizzer

Input:
- Set of learning tasks Q_set
- Learner identifier Learner_ID
- Learning parameters: learning rate (α), discount factor (γ), exploration rate (ε)

Output:
- Optimised personalised learning path (P*)

Steps:
1. Retrieve learner profile, learning history, and task set from the database.
2. Initialise BN to model learner knowledge:
   a. Define nodes for concepts, questions, and responses.
   b. Set prior probabilities using learner history.
   c. Define dependencies based on concept hierarchy.
3. Initialise RL, Q-learning:
   a. Initialise Q-values for all state-action pairs.
   b. Derive initial learner state from BN inference.
4. Repeat for each learning session until mastery threshold is reached:
   a. Observe learner state using BN inference.
   b. Select next task using RL ε-greedy policy.
   c. Present task to learner via game interface.
   d. Receive response and compute reward (correctness, time, engagement, difficulty).
   e. Update Q-values according to RL update rule.
   f. Update BN to reflect learner's new knowledge state.
5. Store learner progress and interaction data for analysis.
6. Generate a performance report.
7. Return optimised personalised P*.

Figure 6. DevQuizzer algorithm

As shown in Figure. 6, DevQuizzer operates through an iterative cycle: learners log in, their profile is retrieved, personalized tasks are generated, and responses are evaluated in real time. Feedback strengthens understanding, and models are updated continuously until mastery is achieved. Performance summaries provide instructors with actionable analytics. This combination ensures that DevQuizzer delivers a

personalized and engaging learning experience, dynamically adjusting task difficulty and sequencing to optimize outcomes.

5. Evaluations

This section reports on the implementation and evaluation of the AGBLS prototype, DevQuizzer. It brings together the architectural and algorithmic elements discussed above, using RL and BN to deliver personalized programming instruction in a gamified setting. The system was evaluated by university students to assess usability, engagement, adaptability, and satisfaction, using a mixed-methods approach.

5.1. The system prototype

The prototype was implemented as a web-based platform that adjusted content, feedback, and task sequencing based on learner performance. RL was used to improve task selection through Q-learning. BN, on the other hand, estimated learner knowledge states from quiz and game responses. As shown in Figure 7, learners engaged with quizzes and Unity-based activities designed to their knowledge level, receiving immediate feedback to strengthen understanding. Their performance and engagement data were recorded to update the learner model and enable continuous adaptation across sessions.

DevQuizzer addresses the cold-start problem through uniform prior initialization [11],[[31],[32]. In this case, all BN knowledge-state nodes were assigned equal prior probabilities at the start of each new learner's first session. This avoids unverified assumptions about previous competence. The BN then updates as the learner responds to an opening diagnostic sequence of three to five calibration tasks, which adjusts conditional probability tables before adaptive sequencing begins. Response patterns from these tasks are sufficient to differentiate beginners from intermediate learners within the first interaction cycle. Incorporating self-reported experience level as an informative prior is one concrete path to reducing cold-start uncertainty in future iterations. In parallel, gamification features such as points, badges, and challenges sustained motivation without altering learning objectives.

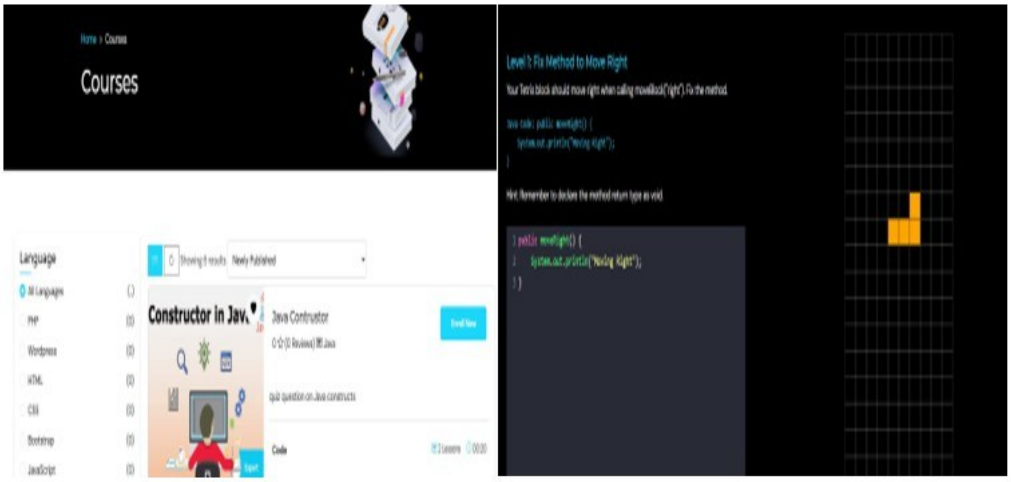

Figure 7. Learner interaction with DevQuizzer

In the same vein, instructors used the system to create and manage content, assign activities, and monitor learner progress through summary analytics. Administrators managed user accounts and oversaw overall system operation. Unity handled the game-based components, while the backend system managed adaptive decision-making, content management, and data storage. This separation ensured that game elements enhanced engagement, while adaptive logic remained consistent across learning activities. Based on its operation, DevQuizzer effectively supported personalized programming instruction.

5.2. Empirical findings

To validate that DevQuizzer effectively supported personalized programming instruction, this subsection presents the results of the evaluation of the AGBLS prototype. Results combine quantitative survey data

with qualitative learner feedback to assess system effectiveness. Quantitative analysis results are summarized in Table 3 and Figure 8. The results showed a weak but significant correlation between prior programming experience and preference for a code-first approach ($r = 0.171$, $p < 0.05$). A stronger correlation was found between prior exposure to game-based systems and programming experience ($r = 0.483$, $p < 0.01$). These findings suggest that learners with programming experience tend to prefer hands-on interaction over theory-driven instruction. No meaningful correlations were observed for gender or level of study. Furthermore, regression analysis confirmed that prior programming experience did not significantly predict learning preference ($\beta = -0.045$, $p = 0.263$, $R^2 = 0.007$). The small coefficient and low explanatory power indicate that prior experience alone does not account for differences in learning preference. The indication is that learner behaviours within the system are influenced by multiple interacting factors rather than a single background variable.

Instrument validity and reliability were carefully assessed. Exploratory factor analysis confirmed that the data were suitable for factor extraction, with a high KMO value (0.887) and a significant Bartlett's Test of Sphericity ($\chi^2 = 1146.084$, $p < 0.001$). Five factors were identified, collectively accounting for 64.86% of the variance, corresponding to engagement, adaptivity, usability, and motivation. Still, Table 3 and Figure 8 present representative loadings, showing that items related to engagement, system adaptivity, navigation, and recommendation intent loaded strongly on their respective constructs. Reliability analysis also indicated that the main questionnaire scale had strong internal consistency ($\alpha = 0.807$). By contrast, the demographic subscale showed lower reliability ($\alpha = 0.393$). Internal consistency metrics such as Cronbach's alpha measure co-variation among items tapping a common latent construct. Demographic variables like age, gender, and level of study share no such latent dimension. According to Streiner [58], applying alpha to categorically distinct attributes is therefore methodologically inappropriate as a quality indicator. Therefore, the low coefficient reflects construct heterogeneity, not instrument failure. These demographic variables were not used as inputs to either the BN or the RL components. They served descriptive and stratification purposes only, so learner modeling accuracy in early sessions was unaffected, and adaptation was driven entirely by task-response data from session onset. These results support the robustness of the instrument while providing transparency about its construct validity and internal consistency.

Table 3. Summary of quantitative analysis

Analysis Category	Variable / Measure	Statistics(s)	Significance
Correlation Analysis	Prior programming experience	$r = 0.171$	$p < 0.05$
	Game-based system experience	$r = 0.483$	$p < 0.01$
Regression Analysis	Predictor: Prior programming experience	$\beta = -0.045$, $t = -1.124$	$p = 0.263$
	Model fit	$R^2 = 0.007$; Adjusted $R^2 = 0.002$	$F(1,170) = 0.263$
Construct Validity (Factor Analysis)	KMO measure	0.887	Sampling adequacy acceptable
	Bartlett's test of sphericity	$\chi^2 = 1146.084$	$p < 0.001$ (factorability supported)
	Total variance explained	64.86%	Five-factor structure supported
	Factor loading range	0.626 – 0.787	Adequate item loading
Reliability Analysis	Main questionnaire scale (12 items)	Cronbach's $\alpha = 0.807$	Good reliability

Analysis Category	Variable / Measure	Statistics(s)	Significance
	Demographic scale (3 items)	Cronbach's α = 0.393	Low reliability
Sample Size	Effective responses	n = 172	—

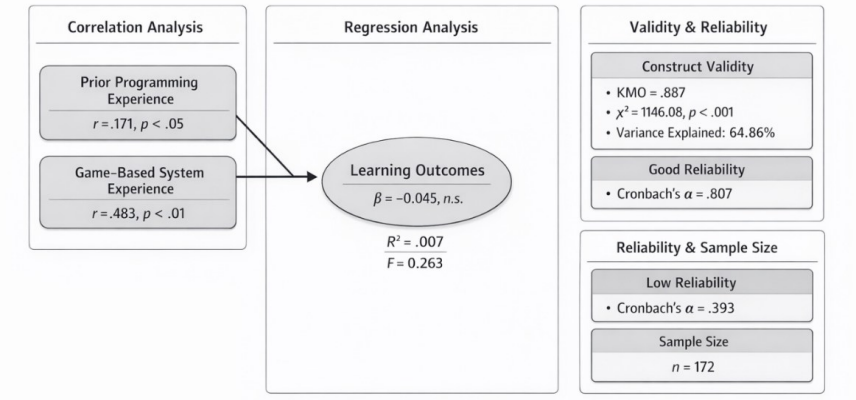

Figure 8. Summary of results

In the same vein, qualitative analysis of open-ended responses provided insights into learner experiences with the AGBLS. Usability-related themes, visualized in Figure 9, included system enhancement, no change needed, content quality, and user experience. The most frequent theme, system enhancement, reflected learners' emphasis on technical optimization, such as performance speed, stability, and scalability, along with requests for additional game-based features. The no change needed theme indicated general satisfaction with the system's functionality and intuitive design. The content quality theme highlighted the desire for a broader range of programming tasks and deeper conceptual material, while the user experience theme emphasized interface clarity and ease of navigation.

Learners also compared the AGBLS with traditional classroom learning. Three themes emerged: positive experience, traditional preference, and flexibility. Positive experiences were most commonly reported, with learners describing the system as engaging, practical, and supportive of self-paced learning. Some learners expressed a preference for traditional settings, valuing face-to-face interaction and instructor presence. Flexibility was frequently noted, with learners appreciating the ability to repeat tasks and control their learning pace. In general, both quantitative and qualitative findings indicate that the AGBLS supports personalized, engaging programming instruction. Learner behaviours are influenced by multiple factors, while feedback shows adaptability, usability, and flexibility. This indicates that DevQuizzer provides a responsive, learner-centred environment.

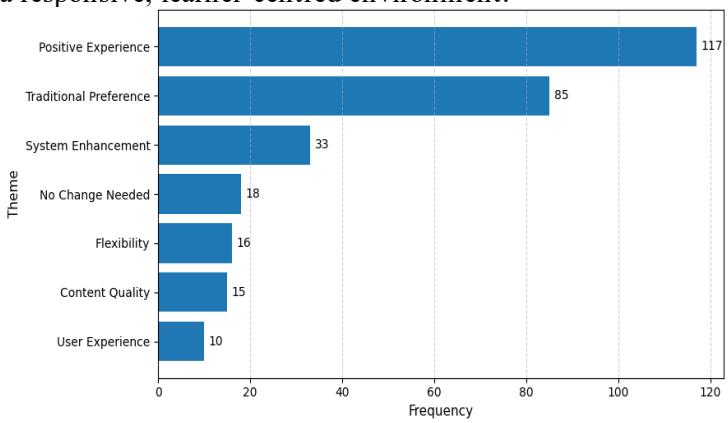

Figure. 9. Thematic analysis

6. Discussion

This study developed and evaluated AGBLS, DevQuizzer, to improve engagement, motivation, and conceptual understanding. The findings show that combining RL and BNs within a gamified learning environment supports adaptive programming education. The combination of probabilistic knowledge modelling and reward-based task sequencing fosters learner motivation, engagement, and personalized progression. This builds on the DSR approach [56], shows how iterative design and evaluation are applied to adaptive learning in practice.

Quantitative analyses revealed that adaptability is shaped by multiple interacting factors rather than a single variable such as prior programming experience. Correlation and regression results confirmed that learner engagement, system design, and motivational drivers collectively influence adaptive outcomes. This finding reinforces the importance of multi-agent approaches, where RL optimises task sequencing, and BN captures probabilistic knowledge relationships, providing a more holistic representation of learner progress than traditional rule-based methods. Validity and reliability analyses further confirmed that the evaluation instrument effectively captured the multidimensional nature of learner experience, with factor analysis and Cronbach's alpha ($\alpha = 0.807$) supporting effectiveness. In parallel, the qualitative findings enriched these results, revealing that learners valued system improvement, content diversity, and intuitive design, while appreciating flexibility and autonomy. Themes related to usability, content quality, and user experience stressed the importance of continuous system improvement. Comparisons with traditional learning showed that while some learners preferred classroom interaction, most valued the autonomy, flexibility, and self-paced progression offered by the adaptive system.

These findings can be interpreted through SDT, which identifies autonomy, competence, and relatedness as critical for sustained motivation [45-50]. Autonomy was supported by learners' ability to choose between Code-First and Theory-First pathways and progress independently. Competence was reinforced through adaptive feedback and progressive mastery of tasks, while relatedness was fostered through gamified and collaborative elements. Table 4 compares recognised AGBL components and their reported consequences with DevQuizzer's empirical findings, showing alignment with prior evidence and SDT constructs. Autonomy was supported through the choice of code-first or theory-first pathways and flexible progression. Likewise, competence was strengthened by adaptive sequencing and immediate feedback, while relatedness was fostered through gamified elements and collaborative engagement. Thus, the RL-BN effort within a gamified environment shows a distinctive contribution compared to existing systems.

Table 4. Alignment of AGBL components with literature, DevQuizzer findings, and SDT constructs

AGBL Component	Purpose	Mechanism	Observed Outcomes	DevQuizzer	Alignment with Literature	SDT Construct Supported
Game Mechanics	Motivation, engagement	Points, levels, leaderboards, story context	Increased participation, lower anxiety, sustained effort [33],[35]	Points, badges, interactive challenges; learners reported enjoyment, persistence, reduced frustration	Matches literature: gamification sustained motivation and effort	Relatedness (shared progress, collaborative play); Autonomy (choice of engagement)
Learner Modeling	Personalization	BNs, cognitive diagnosis models	Accurate assessment of knowledge state, tailored interventions [23],[24]	BN tracked probabilistic knowledge states; revision tasks for weak areas, advanced tasks for mastery	Confirms BN effectiveness in personalization	Competence (accurate assessment, tailored support)

AGBL Component	Purpose	Mechanism	Observed Outcomes	DevQuizzer	Alignment with Literature	SDT Construct Supported
Adaptive Sequencing	Optimal learning path	RL, dynamic difficulty adjustment	Balanced cognitive load, improved retention, better performance [22],[26]	RL with ε-greedy strategy; tasks sequenced dynamically; learners reported achievable progression	Aligns with literature: balanced load and improved persistence	Competence (progressive mastery); Autonomy (learner-directed pathways)
Intelligent Feedback	Scaffolding & guidance	ITS hints, automated recommendations	Reduced errors, timely remediation, enhanced problem-solving [40]	Immediate feedback after each task; qualitative themes highlighted clarity and confidence-building	Strong alignment: feedback reduced frustration and supported mastery	Competence (confidence, mastery through feedback)
Real-Time Adaptation	Continuous optimization	Integration of BN + RL	Maintains engagement, reduces dropout, ensures mastery [26],[29]	Hybrid RL–BN model updated learner profiles and Q-values continuously; sustained motivation and flexible progression	Directly confirms literature: continuous personalization and mastery	Autonomy (flexible progression); Competence (sustained mastery); Relatedness (engagement through shared gamified setting)

However, limitations include dependence on stable internet connectivity and device specifications, the current focus on programming content, and the need for instructor training to fully utilize adaptive features. Greater flexibility in task customisation would enhance pedagogical control. These point to future improvements, including expanded content coverage, collaborative features, and further enhancement of the RL-BN algorithms to strengthen personalization. Table 5 compares our DevQuizzer against four established AGBLS across basic technical and pedagogical dimensions in its explicit integration of RL and BN to deliver dynamic, interpretable, and gamified personalization [53-55]. This includes Knewton, Cognitive Tutor, Smart Sparrow, and Duolingo.

Table 5. Technical comparison of DevQuizzer with established ALS [53-55]

Feature	Knewton	Cognitive Tutor	Smart Sparrow	Duolingo	DevQuizzer
Adaptation algorithm	Item Response Theory + Bayesian inference	Model tracing (rule-based)	Branching logic (rule-based)	Spaced repetition + A/B testing	RL (Q-learning, ε-greedy) + BN
Learner modeling	Probabilistic (item-level)	Knowledge component tracing	Path-based state tracking	Performance history	Probabilistic knowledge states via BN
Feedback granularity	Item-level hints	Step-level procedural hints	Pathway redirection	Correct/incorrect + streak	Task-level immediate feedback with remediation

					paths
Real-time adaptation	Yes	Partial (session-level)	Yes	Yes	Yes (continuous Q-value + BN update)
Gamification	None	Minimal	None	Extensive (surface-level)	Integrated (points, badges, challenges tied to learning objectives)
Interpretability	Low (black-box scoring)	Moderate	Low	Low	Moderate (BN state readable by instructors)
Domain	Multi-domain	Mathematics / STEM	Multi-domain	Language learning	Programming education
Cold-start handling	Placement test	Pre-assessment	Branching entry point	Level selector	Uniform prior + calibration tasks

As presented in Table 5, no listed platform combines reward-optimized task sequencing with probabilistic knowledge state estimation in a domain-specific gamified environment. Knewton's Bayesian component operates at the item-response level and does not model knowledge state transitions across a session. Cognitive Tutor's rule-based tracing is interpretable but cannot adjust sequencing outside predefined knowledge component maps. Duolingo's gamification is extensive but pedagogically decoupled from its adaptive logic, as its motivational mechanics do not change task selection based on knowledge state. DevQuizzer's distinguishing feature is the tight coupling between the BN knowledge state and the RL reward signal. In this case, task selection at every step is both probabilistically informed and reward- optimized. This makes it particularly suitable for programming education, where immediate feedback, scaffolded challenges, and motivational reinforcement are critical. However, whether this coupling generalizes to other procedural STEM domains remains unresolved and warrants further empirical investigation.

In general, DevQuizzer effectively implements the AGBLS framework and aligns with SDT principles. By combining adaptive algorithms with gamified learning, the system offers a flexible, student-centred approach. This design increases engagement, motivation, and self-directed learning in programming. These findings contribute to a scalable framework for adaptive educational technology. They also guide future research on multi-agent systems for personalized learning.

7. Conclusion

This study presented the design, implementation, and evaluation of DevQuizzer, an AGBL system that combined the strengths of RL and BN within a gamified programming environment. In the proposed framework, adaptive sequencing, real-time feedback, and reward mechanisms with dual instructional modes: code-first and theory-first, are integrated. This design supports diverse learner preferences and personalized learning pathways. The implemented system was evaluated using online surveys, and findings demonstrated strong empirical performance. In particular, quantitatively, it showed high learner satisfaction with adaptivity, usability, and engagement, while qualitative feedback accentuated flexibility, autonomy, and motivational design. These results confirm the effectiveness of the proposed RL-BN

adaptive framework and its practical implementation in supporting personalized programming instruction. The evaluation further revealed clear alignment with SDT. Learner-controlled pathways supported autonomy, adaptive feedback, and structured challenges strengthened competence, and gamified progress indicators enhanced relatedness. These features contributed to improved motivation, engagement, and programming learning outcomes.

Despite these contributions, the study faces several limitations. These include reliance on stable internet and device performance, concentration on programming content, and limited scope for instructor adaptation. Future work will extend the framework to additional subject domains. It will also improve accessibility under varied connectivity conditions. In addition, enhanced collaborative and instructor-controlled features will be introduced. These findings show that DevQuizzer and the hybrid RL-BN, AGBL framework provide a scalable and effective approach to personalized programming education. They also show strong potential for broader application across adaptive digital learning environments.

8. Acknowledgement

This research was supported by the Unit for Data Science and Computing and the Department of Computer Science at the North-West University Mafikeng campus.

9. References

[1] C. Tikva and E. Tambouris, "A systematic mapping study on teaching and learning Computational Thinking through programming in higher education," Thinking Skills and Creativity, vol. 41, p. 100849, 2021.

[2] H. Belmar, "Review on the Teaching of Programming and Computational Thinking in the World," Frontiers in Computer Science, vol. 4, p. 997222, 2022.

[3] K. Parajuli, "Mixed Method Study of Experiences of Non-Computer Science Majors in Introductory Computer Science Courses," Virginia Tech, 2024.

[4] C.-W. Tsai et al., "The effects of online peer-facilitated learning and distributed pair programming on students' learning," Computers & Education, p. 104849, 2023.

[5] A. Mbiada, B. Isong, and F. Lugayizi, "A Comparative Study of Computer Programming Challenges of Computing and Non-Computing First-Year Students," The Indonesian Journal of Computer Science, vol. 12, no. 4, 2023.

[6] C. S. Cheah, "Factors contributing to the difficulties in teaching and learning of computer programming: A literature review," Contemporary Educational Technology, vol. 12, no. 2, p. ep272, 2020.

[7] G. Gonzalez Araujo, "Using Apprenticeship and Product-Based Learning to Improve Programming Outcomes in Introductory Computer Science Courses," UC Merced, 2024.

[8] F. Xu and A.-P. Correia, "Adopting distributed pair programming as an effective online collaborative learning approach in computing education: a systematic literature review," Education Tech Research Dev, 2022.

[9] S. C. Santos, P. Azevedo, M. Borba, and M. Brito, "Introductory Programming in Higher Education: A Systematic Literature Review," in International Conference on Peripheral cysts (ICPEC 2022), 2022.

[10] M. Qian and K. R. Clark, "Game-based Learning and 21st century skills: A review of recent research," Computers in Human Behavior, vol. 63, pp. 50-58, 2016.

[11] S. G. Essa, T. Celik, and N. E. Human-Hendricks, "Personalised adaptive learning technologies based on machine learning techniques to identify learning styles: A systematic literature review," IEEE Access, vol. 11, pp. 48392-48409, 2023.

[12] J. Dai, X. Gu, and J. Zhu, "Personalised recommendation in the adaptive learning system: The role of adaptive testing technology," Journal of Educational Computing Research, vol. 61, no. 3, pp. 523-545, 2023.

[13] D. Tsatsou, N. Vretos, and P. Daras, "Adaptive game-based learning in multi-agent educational settings," Journal of Computers in Education, vol. 6, pp. 215-239, 2019.
[14] T. K. Chiu, "Applying the self-determination theory (SDT) to explain student engagement in online learning during the COVID-19 pandemic," Journal of Research on Technology in Education, vol. 54, no. sup1, pp. S14-S30, 2022.
[15] Toukiloglou P, Xinogalos S. A Systematic Literature Review on Adaptive Supports in Serious Games for Programming. Information. 2023; 14(5):277. https://doi.org/10.3390/info14050277.
[16] R. Hare, Y. Tang, and C. Zhu, "Combining Gamification and Intelligent Tutoring Systems for Engineering Education," 2023 IEEE Frontiers in Education Conference (FIE), College Station, TX, USA, 2023, pp. 1-5, doi: 10.1109/FIE58773.2023.10343378.
[17] López-Alcarria A, Olivares-Vicente A, Poza-Vilches F. A Systematic Review of the Use of Agile Methodologies in Education to Foster Sustainability Competencies. Sustainability. 2019; 11(10):2915. https://doi.org/10.3390/su11102915.
[18] Pócsová J, Bednárová D, Bogdanovská G, Mojžišová A. Implementation of Agile Methodologies in an Engineering Course. Education Sciences. 2020; 10(11):333. https://doi.org/10.3390/educsci10110333.
[19] Brown NCC, Wilson G. Ten quick tips for teaching programming. PLoS Comput Biol 14(4): e1006023, 2018. https://doi.org/10.1371/journal.pcbi.1006023.
[20] V. Kamat, "Agile Manifesto in Higher Education," 2012 IEEE Fourth International Conference on Technology for Education, Hyderabad, India, 2012, pp. 231-232, doi: 10.1109/T4E.2012.49.
[21] Li, H., Xu, T., Zhang, C., Chen, E., Liang, J., Fan, X., Li, H., Tang, J., & Wen, Q. (2024). Bringing Generative AI to Adaptive Learning in Education. ArXiv. 2024. https://arxiv.org/abs/2402.14601.
[22] Tong, C., Ren, C. Deep knowledge tracing and cognitive load estimation for personalized learning path generation using neural network architecture. Sci Rep 15, 24925 (2025). https://doi.org/10.1038/s41598-025-10497-x.
[23] Adel Ihichr, Omar Oustous, Younes El Bouzekri El Idrissi, and Ayoub Ait Lahcen. "A Systematic Review on Assessment in Adaptive Learning: Theories, Algorithms and Techniques". International Journal of Advanced Computer Science and Applications (IJACSA) 15.7 (2024). http://dx.doi.org/10.14569/IJACSA.2024.0150785.
[24] B. Tian, C. Wang, and H. Hong, "A Survey of Personalized Adaptive Learning Systems," 2023 2nd International Conference on Artificial Intelligence and Computer Information Technology (AICIT), Yichang, China, 2023, pp. 1-6, doi: 10.1109/AICIT59054.2023.10277850.
[25] M. Murtaza, Y. Ahmed, J. A. Shamsi, F. Sherwani, and M. Usman, "AI-Based Personalized E-Learning Systems: Issues, Challenges, and Solutions," in IEEE Access, vol. 10, pp. 81323-81342, 2022, doi: 10.1109/ACCESS.2022.3193938.
[26] Riedmann, A., Schaper, P. & Lugrin, B. Reinforcement Learning in Education: A Systematic Literature Review. Int J Artif Intell Educ 35, 2669-2723 (2025). https://doi.org/10.1007/s40593-025-00494-6.
[27] Singla, A., Rafferty, A. N., Radanovic, G., & Heffernan, N. T. Reinforcement learning for education: Opportunities and challenges [Workshop]. In International Conference on Educational Data Mining (EDM), 2021.
[28] Wu, Siyu et al. "A Comprehensive Exploration of Personalized Learning in Smart Education: From Student Modeling to Personalized Recommendations." ArXiv abs/2402.01666 (2024): n. pag.
[29] Padakandla, S. A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments. ArXiv, 2020. https://doi.org/10.1145/3459991.
[30] Kitson, N.K., Constantinou, A.C., Guo, Z. et al. A survey of Bayesian Network structure learning. Artif Intell Rev 56, 8721–8814 (2023). https://doi.org/10.1007/s10462-022-10351-w.
[31] Q. Bin, M. F. Zuhairi and J. Morcos, "A Comprehensive Study on Personalized Learning Recommendation In E-Learning System," in IEEE Access, vol. 12, pp. 100446-100482, 2024, doi: 10.1109/ACCESS.2024.3428419

[32] Munir H, Vogel B, Jacobsson A. Artificial Intelligence and Machine Learning Approaches in Digital Education: A Systematic Revision. Information. 2022; 13(4):203. https://doi.org/10.3390/info13040203.
[33] Toukiloglou P, Xinogalos S. A Systematic Literature Review on Adaptive Supports in Serious Games for Programming. Information. 2023; 14(5):277. https://doi.org/10.3390/info14050277.
[34] R. Hare, Y. Tang, and C. Zhu, "Combining Gamification and Intelligent Tutoring Systems for Engineering Education," 2023 IEEE Frontiers in Education Conference (FIE), College Station, TX, USA, 2023, pp. 1-5, doi: 10.1109/FIE58773.2023.10343378.
[35] G. Ahma and A. Kadriu, "Exploring the Impact of Gamification Strategies on Student Engagement and Learning Outcomes in Education," 2025 20th Annual System of Systems Engineering Conference (SoSE), Tirana, Albania, 2025, pp. 1-6, doi: 10.1109/SoSE66311.2025.11083792.
[36] Kenwright, B. Exploring the Power of Creative AI Tools and Game-Based Methodologies for Interactive Web-Based Programming. ArXiv, 2023. https://arxiv.org/abs/2308.11649.
[37] K. P. M, K. Selvi, M. Shrinidhii, M. P. S, R. S. N., and M. G, "EscapeScript - Gamified Coding Platform for Empowering Future Coders," 2024 2nd International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, 2024, pp. 552-557, doi: 10.1109/ICSCSS60660.2024.10625310.
[38] T. Ivanković, T. Jagušt, A. V. Božić, and N. Hoić-Božić, "Gamified and Adaptive Learning System Codey for Learning Programming," 2025 IEEE Global Engineering Education Conference (EDUCON), London, United Kingdom, 2025, pp. 1-5, doi: 10.1109/EDUCON62633.2025.11016339.
[39] S. Chandrasekara, D. Hewavitharana, M. Weerasinghe, B. Gayasri, D. Wijendra, and D. De Silva, "Gamifying Coding Education for Beginners: Empowering Learners with HTML, CSS, and JavaScript," 2025 International Research Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka, 2025, pp. 1-7, doi: 10.1109/SCSE65633.2025.11030977.
[40] Montella R, De Vita CG, Mellone G, Ciricillo T, Caramiello D, Di Luccio D, Kosta S, Damaševičius R, Maskeliūnas R, Queirós R, et al. Leveraging Large Language Models to Support Authoring Gamified Programming Exercises. Applied Sciences. 2024; 14(18):8344. https://doi.org/10.3390/app14188344.
[41] Aydin, M., Karal, H., & Nabiyev, V. Examination of adaptation components in serious games: a systematic review study. Educ Inf Technol 28, 6541–6562 (2023). https://doi.org/10.1007/s10639-022-11462-1.
[42] Toukiloglou, P., & Xinogalos, S. (2023). Adaptive Support With Working Examples in Serious Games About Programming. Journal of Educational Computing Research, 61(4), 766-789. https://doi.org/10.1177/07356331231151393.
[43] S. N. J, L. M. Naik, M. C, R. P, and S. S. Revankar, "Guild Based Online Coding Platform with Gamification Features," 2025 Global Conference in Emerging Technology (GINOTECH), PUNE, India, 2025, pp. 1-5, doi: 10.1109/GINOTECH63460.2025.11077053.
[44] Imran, H. (2022). An Empirical Investigation of the Different Levels of Gamification in an Introductory Programming Course. Journal of Educational Computing Research, 61(4), 847-874. https://doi.org/10.1177/07356331221144074.
[45] Gupta, A.& Behl, A. Self-Determination Theory: A review. In S. Papagiannidis (Ed), TheoryHub Book, 2025. Available at https://open.ncl.ac.uk / ISBN: 9781739604400.
[46] Bureau, J. S., Howard, J. L., Chong, J. X. Y., & Guay, F. (2022). Pathways to Student Motivation: A Meta-Analysis of Antecedents of Autonomous and Controlled Motivations. Review of educational research, 92(1), 46-72. https://doi.org/10.3102/00346543211042426.
[47] Yurou Wang, Hui Wang, Shengnan Wang, Stefanie A. Wind, Christopher Gill, A systematic review and meta-analysis of self-determination-theory-based interventions in the education context, Learning and Motivation, Vol.87,2024, https://doi.org/10.1016/j.lmot.2024.102015.

[48] Neufeld, A. Putting Self-Determination Theory into Practice: A Practical Tool for Supporting Medical Learners' Motivation. The clinical teacher, 22(4), 2025, e70140. https://doi.org/10.1111/tct.70140.

[49] Chirkov, V. I. A cross-cultural analysis of autonomy in education: A self-determination theory perspective: A self-determination theory perspective. Theory and Research in Education, 7(2), 253-262. https://doi.org/10.1177/1477878509104330.

[50] Yang, Y., Chen, J., & Zhuang, X. Self-determination theory and the influence of social support, self-regulated learning, and flow experience on student learning engagement in self-directed e-learning. Frontiers in Psychology, 16, 1545980, 2025. https://doi.org/10.3389/fpsyg.2025.1545980.

[51] Alsuhaymi, A. O., & Atallah, F. A. Sustainable Education in the Age of Artificial Intelligence and Digitalization: A Value-Critical Approach. Sustainability, 18(3), 1257, 2026. https://doi.org/10.3390/su18031257.

[52] Qi Xia, Thomas K.F. Chiu, Min Lee, Ismaila Temitayo Sanusi, Yun Dai, Ching Sing Chai, A self-determination theory (SDT) design approach for inclusive and diverse artificial intelligence (AI) education, Computers & Education, Volume 189, 2022, https://doi.org/10.1016/j.compedu.2022.104582.

[53] H. Almetnawy, A. Orabi, A. R. Alneyadi, T. Ahmed, and A. Lakas, "An Adaptive Intelligent Tutoring System Powered by Generative AI," 2025 IEEE Global Engineering Education Conference (EDUCON), London, United Kingdom, 2025, pp. 1-10, doi: 10.1109/EDUCON62633.2025.11016362.

[54] Elegbeleye, F., & Isong, B. Comparative Evaluation of Adaptive Learning Models for Trustworthy Personalised Education. Studies in Learning and Teaching, 6(3), 663-674, 2026. https://doi.org/10.46627/silet.v6i3.698.

[55] Elegbeleye Femi Abiodun1, Bassey Isong. “A Systematic Review of Challenges in Teaching and Learning Computer Programming Modules. The Indonesian Journal of Computer Science (IJCS), Vol. 14 No. 1, 2025.

[56] R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering. Springer, 2014.

[57] vom Brocke, J., Hevner, A., Maedche, A. Introduction to Design Science Research. In: vom Brocke, J., Hevner, A., Maedche, A. (eds) Design Science Research. Cases. Progress in IS. Springer, Cham. 2020. https://doi.org/10.1007/978-3-030-46781-4_1.

[58] Streiner, D. L. Starting at the Beginning: An Introduction to Coefficient Alpha and Internal Consistency. Journal of Personality Assessment, 80(1), 99-103, 2003. https://doi.org/10.1207/S15327752JPA8001_18.