Utilizing Adaptive Learning with GPT-4 for Introduction to Python: Effects on Accuracy and Time-to-Solution

Diego Santivanez
Engineering Faculty, Software Engineering Program, Universidad Peruana de Ciencias Aplicadas, Lima, Peru
u20211221@upc.edu.pe

Kevin Oliva
Engineering Faculty, Software Engineering Program, Universidad Peruana de Ciencias Aplicadas, Lima, Peru
u201410580@upc.edu.pe

Lenis Wong
Engineering Faculty, Software Engineering Program, Universidad Peruana de Ciencias Aplicadas, Lima, Peru
pcsilewo@upc.edu.pe (corresponding author)

Received: 8 July 2025 | Revised: 16 August 2025, 18 September 2025, and 30 September 2025 | Accepted: 12 October 2025

 DOI: https://doi.org/10.48084/etasr.13246

ABSTRACT

Programming instruction in universities often struggles to adapt to individual needs and sustain motivation. This study presents Code Showdown, a web application that combines adaptive learning with GPT-4 to strengthen Python programming skills through automatically generated challenges and immediate personalized feedback. The development followed four phases: (i) selecting the pedagogical approach, (ii) choosing the language model, (iii) integrating GPT-4 for challenge generation and formative feedback, and (iv) building a scalable web architecture. The proposed approach was evaluated with 40 undergraduates (two groups of 20 individuals) over two weeks. Group E1 studied with traditional resources, whereas Group E2 used Code Showdown in individual and multiplayer modes. Using explicit metrics—Improvement Score (IS, %) and Improvement in Response Time (IRT, %)—Group E2 achieved higher means across difficulty levels: IS of 18.9±4.2% (Easy), 40.7±4.7 % (Intermediate), and 58.6±5.1 % (Hard), and IRT reductions of 32.4±6.3%, 28.0±5.8%, and 28.3±6.1 %, respectively. A t-test showed that these differences were statistically significant (p<0.05). Satisfaction was high (ISO/IEC 25010-based survey), with more than 80% positive ratings and a recommendation score of 4.9±0.3/5. These findings suggest that adaptive, GPT-4-assisted practice may enhance accuracy and efficiency in introductory Python while maintaining engagement through gamification and real-time competition.

Keywords-programming skills; ChatGPT/GPT-4; adaptive learning; gamification; AI in education

I. INTRODUCTION

The traditional approach to teaching programming in university settings has shown significant limitations in adapting to students' individual needs and sustaining their motivation over time. Although strategies such as gamification and intelligent tutoring systems have been introduced to address these gaps, they still present restrictions in terms of personalization and immediate feedback, which are crucial for building solid programming skills [1, 2]. Recent research has explored the use of Artificial Intelligence (AI) and conversational agents, such as GPT-4, to overcome these challenges. Studies report notable improvements in comprehension, engagement, and confidence among students who interact with AI-based tutors, as these systems provide timely guidance and explanations [3, 4]. Nevertheless, many of these implementations fail to maintain long-term engagement or provide feedback sufficiently tailored to the progress of the learner to foster autonomous and sustained learning [1, 2]. Consequently, one of the main challenges in contemporary education is designing technological solutions that not only personalize the learning experience but also adapt content dynamically to the learner's level and sustain motivation across extended periods of study [5]. Achieving this goal requires the

integration of AI models capable of generating relevant content, offering context-aware feedback, and automatically evaluating the user's progress.

Within this educational landscape, diverse pedagogical approaches have demonstrated their capacity to improve learning outcomes when combined with technology. Gamification, for example, transforms passive instruction into active engagement through game mechanics such as points, badges, and achievements, which enhance intrinsic motivation and reduce attrition [6]. Similarly, the flipped classroom model optimizes in-person time for collaborative problem solving and debugging activities, as students review theoretical content beforehand and arrive prepared to apply concepts during class sessions [7, 8]. Metacognitive strategies complement these techniques by encouraging learners to monitor their own understanding, reflect on errors, and adjust problem-solving approaches, strengthening both algorithmic thinking and self-efficacy [9, 10]. These methods have been applied not only to programming but also to mathematics and language learning, demonstrating improvements in performance and conceptual understanding when interactive and adaptive techniques are used [2, 5, 11, 12]. In programming education specifically, combining gamified environments with automated tutorials and immediate feedback has been shown to improve logical reasoning and independent problem-solving skills, while in mathematics, adaptive applications and logic games help students connect abstract operations to more concrete and enjoyable learning experiences [11, 12]. In the area of language acquisition, AI-driven chatbots provide a safe and responsive environment in which learners can practice conversation and grammar, increasing fluency and confidence without the anxiety of making mistakes before a human instructor [2, 3].

The rapid evolution of Large Language Models (LLMs)—including GPT-3, GPT-4, T5, and Bard—has further transformed the design of intelligent educational applications. These models enable content generation, code analysis, natural language feedback, and automated assessment, creating opportunities for highly personalized and scalable learning experiences [3, 5, 13]. GPT-4, in particular, has been recognized for its ability to interpret complex instructions, generate accurate code, and provide nuanced, human-like explanations that support a deeper understanding of programming concepts [3]. Educational chatbots built on these models overcome the limitations of synchronous assistance, allowing students to seek guidance at any time and receive immediate feedback that reinforces motivation and encourages continuous learning [2]. However, the integration of such technologies still presents challenges, including ensuring prompt quality, maintaining pedagogical coherence, and avoiding overreliance on AI outputs. Therefore, effective adoption requires careful instructional design to align these tools with educational objectives and to foster critical, reflective engagement rather than passive dependence [2, 5].

In this context, this study introduces Code Showdown, an educational web application that combines the GPT-4 language model with adaptive learning techniques to strengthen Python programming skills. The system is designed to generate personalized challenges in real time and to provide immediate, structured feedback, thereby supporting dynamic and motivating learning experiences. Code Showdown distinguishes itself from other AI-enhanced platforms by unifying gamification, adaptive content generation, and automated evaluation within a coherent pedagogical framework, positioning it as a comprehensive solution to the persistent challenges of engagement, personalization, and sustained skill development in programming education.

II. MATERIALS AND METHODS

This section describes the methodological approach adopted for developing the Code Showdown web application. The proposal is based on innovative pedagogical techniques and advanced AI models.

A. Choice of Learning Type

Four pedagogical approaches applicable to the development of programming skills were evaluated: Game-Based Learning (T1), Intelligent Tutoring (T2), Inverted Classrooms (T3), and Adaptive Learning (T4). The selection was made through a multi-criteria analysis, considering five factors: interactivity (C1), personalization (C2), feedback (C3), motivation (C4), and skill development (C5). To determine the relative importance of each criterion, a pairwise comparison matrix was used. A value of "1" was assigned if the criterion in the row was equal to or more important than the criterion in the column. Otherwise, a value of "0" was assigned. Pairwise comparisons were performed independently by the three authors, and the criterion weights were obtained by normalizing row sums to 100%. Evaluator agreement was reached by consensus (κ=0.82). Table I presents the results of this comparison, showing that C2 is the most important criterion (40%).

A comparative analysis (benchmarking) of the four learning techniques was carried out based on the weights obtained in the comparison matrix. Each approach was evaluated against the five defined criteria, and a score from 1 to 10 was assigned according to the level of compliance. Then, a weighting was applied based on the relevance of each criterion. As shown in Table II, T4 achieved the highest weighted score (9.5) due to its ability to adapt content and provide immediate feedback.

TABLE I. COMPARISON MATRIX FOR TECHNIQUES

Criteria	C1	C2	C3	C4	C5	
C1	0	0	1	1	1	30%
C2	1	0	1	1	1	40%
C3	0	0	0	1	1	20%
C4	0	0	0	0	1	10%
C5	0	0	0	0	0	0%

TABLE II. BENCHMARKING OF LEARNING TECHNIQUES

Criteria	T1	T2	T3	T4
C1	9	6	7	10
C2	6	4	5	10
C3	7	5	4	10
C4	8	6	6	9
C5	7	4	6	9
Total weighted	7.2	5.2	5.5	9.5

B. LLM Selection

Five alternatives were evaluated to select the most suitable language model for generating customized challenges and automated feedback in the application: GPT-4 (M1), GPT-3 (M2), Bard (M3), T5 (M4), and Llama 3 (M5). The evaluation was based on seven criteria: contextual understanding (C1), code generation (C2), adaptability to new tasks (C3), processing speed (C4), ease of integration (C5), resource consumption (C6), and community support (C7). To establish the relevance of each criterion, a pairwise comparison matrix was applied, similar to the one used in the previous section, where each criterion was directly compared with the others. Table III shows the results of this comparison. C1 had the highest relative weight (28.6%), followed by C2 (23.8%) and C3 (19.0%). These three criteria were prioritized to ensure the quality, accuracy, and versatility of the challenges generated by the system. Subsequently, benchmarking was performed, in which each model was rated against defined criteria using a scale of 1 to 10. As shown in Table IV, M1 received the highest weighted score (8.8) due to its ability to understand the educational context, generate accurate code, and adapt to different learning situations. These characteristics make GPT-4 the most suitable option for integration as an AI engine in an educational application.

TABLE III. COMPARISON MATRIX FOR LLM

Criteria	C1	C2	C3	C4	C5	C6	C7	Weight
C1	1	1	1	0	1	1	1	28.6%
C2	0	1	1	1	1	1	1	23.8%
C3	0	0	1	1	1	1	1	19%
C4	0	0	0	1	0	1	0	9.5%
C5	0	0	0	0	1	0	0	4.8%
C6	0	0	0	0	0	1	0	4.8%
C7	0	0	0	0	0	1	1	9.5%

TABLE IV. BENCHMARKING FOR LLM SELECTION

Criteria	M1	M2	M3	M4	M5
C1	9	8	7	7	8
C2	10	9	8	6	8
C3	9	7	8	7	8
C4	7	7	8	8	8
C5	8	8	8	7	8
C6	7	7	6	8	7
C7	9	9	7	8	8
Total weighted	8.8	7.9	7.5	6.9	7.9

C. Implementation of GPT-4

GPT-4 was integrated as a central component of the application to support adaptive learning of Python programming. This language model generates personalized challenges and provides immediate feedback adapted to each student's level and performance. The technical implementation is divided into five functional components that facilitate interaction between the user, business logic, and the OpenAI API. Figure 1 illustrates the communication architecture between the main elements of the system: the end user, the web application (front-end and back-end), and the OpenAI API. This visualization clarifies how challenge requests and AI-generated responses flow through the application in a structured, automated manner.

1) Backend Integration

The application's backend was developed with FastAPI, which acts as an intermediary between user actions and processing logic. Requests for challenges or evaluations are transformed into specific prompts and securely sent to the OpenAI API. The model responses are then received, validated, and processed for submission.

2) Generation of Personalized Challenges

When a student requests a new exercise, the system records the difficulty level and required topic. Then, it generates a prompt that directs GPT-4 to create a programming challenge. The prompt includes a clear problem statement and two test cases (input and output), but does not provide the solution. A token limit controls the length of the response. Then, the backend formats and adapts the response to the application's visual environment.

3) Evaluation of Responses and Feedback

After a student submits their code, the system generates a new prompt that requests a detailed evaluation from GPT-4. This evaluation includes a correctness analysis, Python best practices, refactoring suggestions, and motivational comments. This provides pedagogical feedback that validates the response and promotes continuous improvement.

4) Data Warehousing and Continuous Learning

All information related to student performance is stored in a PostgreSQL database. This includes solved exercises and system responses. This database enables longitudinal tracking of each user's progress based on their scores and response times, promoting a personalized learning experience.

5) Interaction with the Frontend

The user interface, built with React, allows real-time visualization of the proposed challenges, coding space, feedback messages, and progress metrics. Communication between the frontend and backend is managed through a REST API, ensuring a smooth, interactive, and synchronized experience with the responses generated by GPT-4.

D. Web Application Construction

The web application was designed under a five-layer modular architecture to ensure scalability, maintainability, and efficiency in data processing. This organization facilitates smooth interaction between the components of presentation, business logic, integration with AI, data storage, and infrastructure management. Figure 2 shows the physical architecture of the application.

1) Presentation Layer (Frontend)

The user interface was developed using React, offering an interactive, responsive, and end-user-oriented experience. Through this layer, students can access the application's functionalities, such as requesting challenges, viewing feedback, and participating in competitions. Two modes of use were implemented: individual mode and multiplayer mode, allowing both autonomous practice and real-time competition with other users.

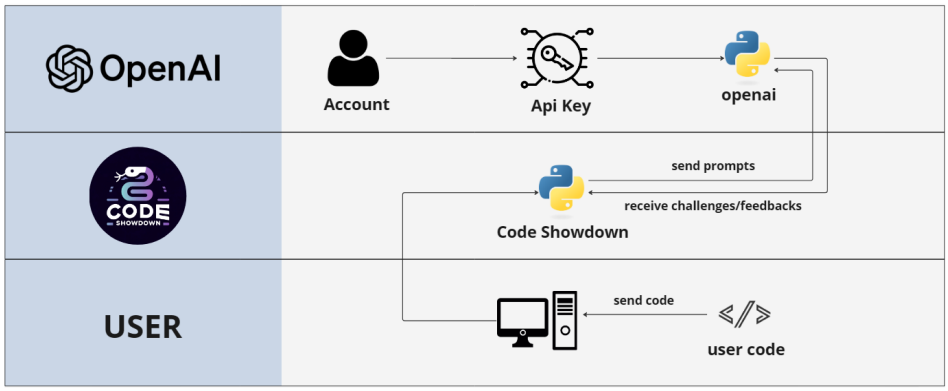

Fig. 1. User-system interaction architecture.

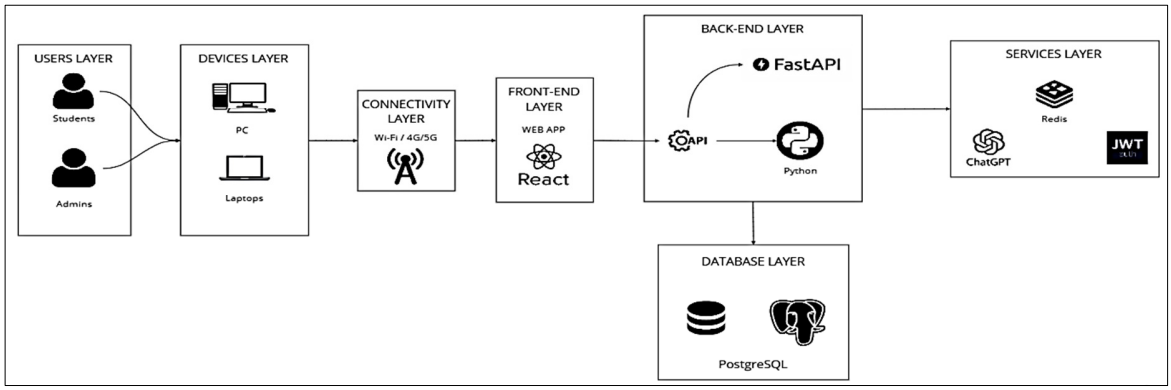

Fig. 2. Physical architecture diagram.

2) Application Layer (Backend)

The backend was implemented with FastAPI and structured into microservices to ensure clear separation of responsibilities. Among the main services, the API Controller manages requests and coordinates interactions across the different modules. The Challenge Service is responsible for generating exercises and handling parameters such as topic and difficulty level, while the Feedback Service manages automatic evaluation and delivers personalized feedback. The Profile Service oversees user data, including level progress and configuration. In addition, the Competition Service supports real-time competition logic, incorporating live chat between participants. Finally, the Statistics Service records and analyzes student performance, and the Authentication Service ensures secure access through credential verification.

3) AI Integration Layer

This layer acts as a bridge between the internal services and the OpenAI API, taking care of prompt generation, response processing, and content validation before presenting it to the user. Adaptive logic runs in this layer to customize challenges based on the learner's history.

4) Storage Layer

A PostgreSQL relational database was used for data storage, managing key information such as user profiles, challenges generated, scores obtained, and feedback history. Its use allows longitudinal tracking of each student's performance, facilitating adaptive decision-making within the application.

5) Infrastructure and Orchestration Layer

Docker was adopted as the containerization tool to facilitate system deployment across different environments. Each component of the architecture—including the frontend, backend, AI services, and database—was encapsulated in independent containers. Once the construction of the web application was completed, four main functionalities consolidated its educational proposal: (i) The personalized generation of programming challenges relies on adaptive prompts to provide students with exercises tailored to their knowledge level and configuration, enabling progressive learning according to individual needs; (ii) The automated feedback mechanism delivers immediate responses after students submit their solutions, including suggestions for improvement, code refactoring, and good programming practices, thus facilitating self-assessment and autonomous learning; (iii) The real-time competition feature allows participants to create or join competition rooms, set challenges, compete simultaneously, and interact through integrated chat, enhancing motivation through social engagement; And (iv) the user-level statistics module stores data on performance, such as scores achieved, resolution times, and participation frequency, enabling each student to visualize progress and identify areas for improvement.

III. EXPERIMENTAL VALIDATION

This section describes the design of the experimental process used to evaluate the effectiveness of the Code Showdown web application in strengthening programming skills with Python. The validation was carried out in Lima, Peru, through two main phases: an initial and a final evaluation. At the conclusion of the final phase, a structured survey was applied to collect participants' perceptions of their experience with the application.

The sample consisted of 40 students from the Peruvian University of Applied Sciences, belonging to the Faculty of Software Engineering, Systems, and Computer Science. The participants were equally distributed into two experimental groups of 20 students each. Both groups were exposed to three scenarios with varying levels of difficulty: Easy, Intermediate, and Hard. The first group studied Python using traditional methods, while the second group used the Code Showdown web application. To measure the impact of the application, two quantitative metrics were defined and applied to both groups: the Improvement in Response Time (IRT), which quantifies the reduction in the time needed to solve exercises, and the Improvement in Score (IS), which evaluates the increase in the accuracy or quality of the students' solutions. In addition, qualitative data on user experience were collected through a survey administered at the end of the intervention. Students from different academic disciplines, levels of programming experience, or institutional backgrounds may interact differently with adaptive learning systems and AI-generated feedback. Cultural and technological factors could influence user engagement and performance. Future research should include broader and more diverse populations across public and private institutions, as well as international settings, to evaluate the platform's adaptability, robustness, and effectiveness on a larger scale.

Figure 3 shows the methodological design followed to assess Python learning in two groups of students. After a common initial assessment, both groups participated in an independent experiment with similar stages of instruction, recording metrics and solving exercises. Finally, data were collected to measure the impact of each approach.

A. Experiment 1 (E1): Study of Python in a Traditional Way

This experiment served as a baseline to compare the effect of traditional methods on Python learning. The students studied for two weeks autonomously with materials such as notes and tutorials, without technological mediation or automatic feedback. The path followed is reflected on the left side of Figure 3. Progress was evaluated using IRT and IS metrics after solving exercises at the end of the process.

B. Experiment 2 (E2): Study of Python with the App

The second group used the Code Showdown application for two weeks, which offers hands-on challenges, immediate feedback, and single- and multiplayer game modes. This experience, depicted on the right side of Figure 3, allowed real-time progress monitoring and encouraged participation. The IRT and IS metrics allowed measuring the improvement achieved compared to the traditional study group.

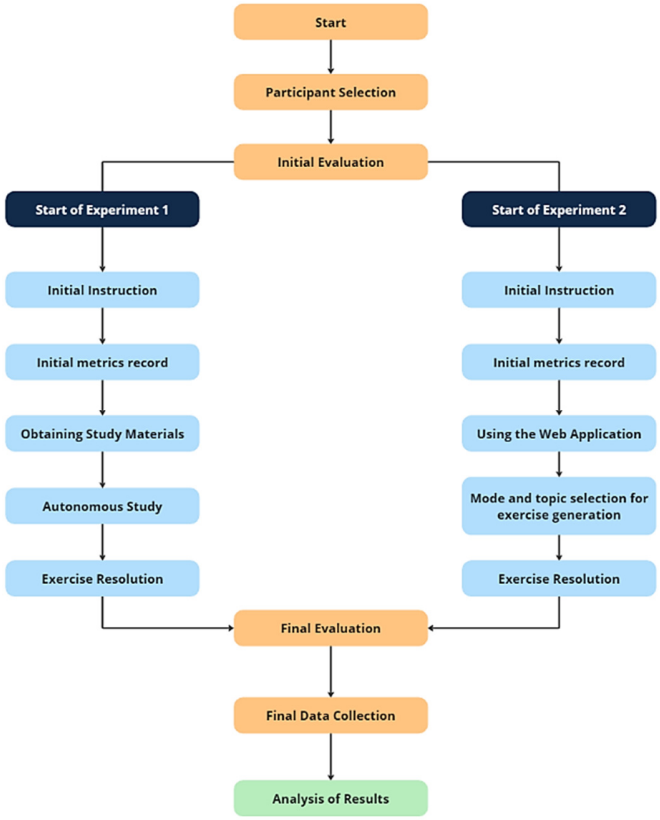

Fig. 3. User-system interaction architecture.

C. Survey Design

A structured survey was applied at the end of the study, aiming to collect the perceptions of the 40 students about the Code Showdown web application. This survey was designed following the guidelines of the ISO/IEC 25010 standard, focused on the quality of software products, and was composed exclusively of closed questions. The questions were organized into three main categories. Category O (Operability) evaluates the ease of use, the navigation within the interface, and the clarity of the visual elements. Category F (Functionality) aims to measure the appropriateness of the challenges generated in relation to the student's level, as well as the influence of the multiplayer mode on the interest in continuing practicing. Category L (Learnability) addresses the perception of improvement in programming skills after using the application, as well as the motivational impact of the gamified approach. All questions were answered using a Likert-type scale with five levels: 1 = "Strongly disagree," 2 = "Disagree," 3 = "Neutral," 4 = "Agree," and 5 = "Strongly agree" (see Table V).

D. Metrics

The following metrics were used to validate the performance of the proposed system. Let S_0 and S_f be initial/final scores and T_0 and T_f be the initial/final times:

$$IRT(\%) = \frac{T_0 - T_f}{T_0} \times 100 \quad (1)$$

$$IS(\%) = \frac{S_f - S_0}{S_0} \times 100 \quad (2)$$

IRT represents the percentage reduction in the time required by a student to solve a programming challenge. A positive IRT value indicates faster solutions. *IS* indicates the percentage increase in the grade obtained by the student.

TABLE V. POST-EXPERIMENT SURVEY QUESTIONS

Category	Question	
O	Q1	How would you rate your user-friendliness experience with the Code Showdown application?
O	Q2	Was navigation through the application easy and intuitive?
O	Q3	Did the application work without technical problems during use?
O	Q4	Did you find the visual design of the application pleasing?
O	Q5	Was the feedback from the system clear and helpful?
F	Q6	Were the programming challenges appropriate to your level?
F	Q7	Did the difficulty of the challenges adapt as you configured them?
F	Q8	Did the multiplayer mode influence your interest in practicing more?
L	Q9	Do you consider that you improved your Python skills after using the application?
L	Q10	Did you feel motivated to continue programming thanks to the gamified approach?
L	Q11	Would you recommend Code Showdown to other students learning Python?
L	Q12	Compared to other methods you have used to learn to program, did you find this experience more useful?

IV. RESULTS AND DISCUSSION

A. Experiment E1: Traditional Method

During the initial evaluation, the results of the students who participated under the traditional programming learning approach were compiled. Table VI shows the scores obtained and the resolution times recorded by difficulty level: Easy, Intermediate, and Hard.

TABLE VI. INITIAL AND FINAL RESULTS OF E1

Initial Test				Final Test		
Student	**Difficulty**	**Score**	**Time (min.)**	**Difficulty**	**Score**	**Time (min.)**
Student 1	Easy	370	47	Easy	378	39
Student 2	Easy	340	38	Easy	332	33
Student 3	Easy	390	44	Easy	388	35
Student 4	Easy	320	42	Easy	378	39
Student 5	Easy	300	50	Easy	400	33
Student 6	Easy	310	41	Easy	388	35
Student 7	Easy	350	45	Easy	310	37
Student 8	Easy	280	36	Easy	408	35
Student 9	Easy	360	48	Easy	360	31
Student 10	Intermediate	320	39	Intermediate	350	38
Student 11	Intermediate	300	40	Intermediate	340	28
Student 12	Intermediate	300	35	Intermediate	362	36
Student 13	Intermediate	270	37	Intermediate	318	30
Student 14	Intermediate	330	46	Intermediate	330	32
Student 15	Intermediate	290	43	Intermediate	370	29
Student 16	Intermediate	250	36	Intermediate	300	31
Student 17	Hard	210	37	Hard	350	34
Student 18	Hard	290	41	Hard	350	38
Student 19	Hard	300	48	Hard	310	29
Student 20	Hard	180	38	Hard	218	30

After two weeks, a final evaluation was carried out at the end of the experimental stage. In this phase, the same classification was followed by difficulty levels. From the data obtained, (1) and (2) were applied to calculate the percentage of IRT and IS scores by difficulty level. Table VII summarizes the percentage results obtained at this stage. The results indicate a moderate improvement in the scores obtained, especially in the more difficult exercises. However, the reduction in resolution time was more limited, suggesting that the traditional approach allowed some evolution in accuracy, but without a significant impact on students' time efficiency.

TABLE VII. PERCENTAGE RESULTS OF E1

Difficulty	IS (%)	IRT (%)
Easy	12.24 ± 19.22	18.04 ± 10.90
Intermediate	15.53 ± 8.95	17.93 ± 14.15
Hard	27.95 ± 27.11	19.02 ± 15.09
Overall	16.54 ± 19.32	18.20 ± 11.91

B. Experiment E2: Use of the Code Showdown Application

The second experiment was developed in a controlled academic environment to evaluate the use of the Code Showdown web application as a support tool for learning programming. The experiment was carried out in the laboratories of the Universidad Peruana de Ciencias Aplicadas, with the participation of 20 students from the Faculty of Software Engineering, Systems, and Computer Science. After a first introductory session, the participants continued using the application for a period of two weeks, integrating it into their daily study routines both individually and collaboratively. Figure 4 shows a student using the application during the development of this experiment.

The steps followed by the students to interact with the application during each session of the experiment are described below. Before the start of each session, participants accessed the application interface to create a competition room, where they could configure the name of the room, the number of exercises, and the time limit per exercise (see Figure 5a). Once the room was created, the student could request that the system generate programming challenges automatically according to the selected difficulty (Figure 5b); subsequently, the system displayed a summary of the configuration (Figure 5c). During the competition, the system displayed the exercise statement along with the expected input and output, as well as the remaining countdown time (Figure 5d). In case they needed help, students could consult contextualized hints (Figure 5e), which explained key concepts of the exercise. Upon submitting their solution, the system automatically generated an immediate feedback report (Figure 5f), which detailed the fulfillment of the problem requirements and assigned a quantitative score (from 0 to 100) based on the quality of the submitted solution.

Fig. 4. A student using Code Showdown (Experiment 2).

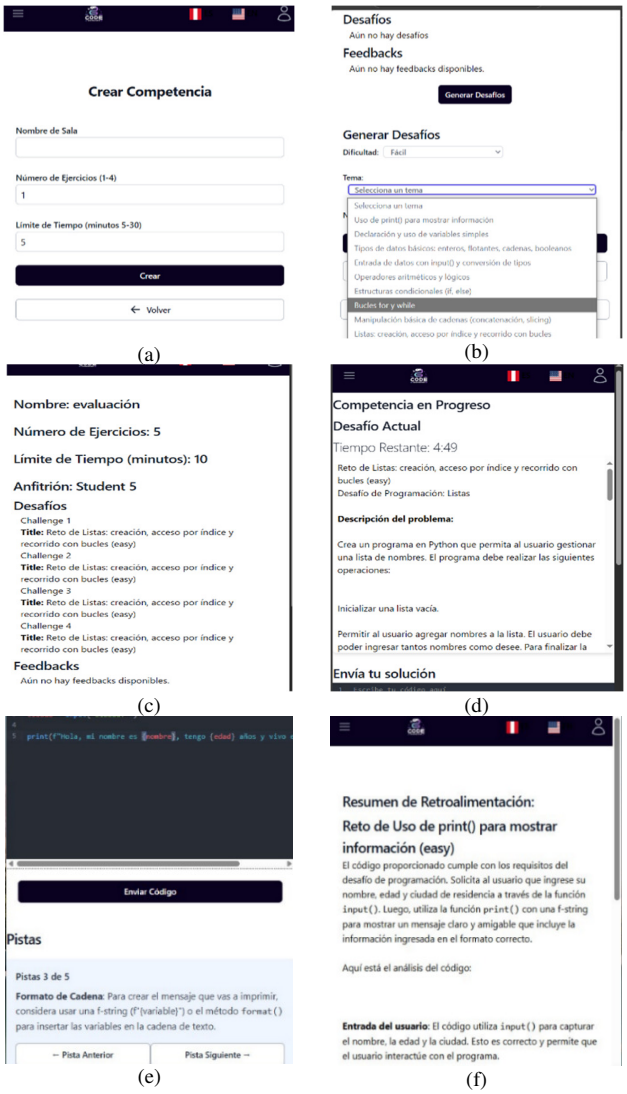

Fig. 5. Code Showdown application interfaces.

Table VIII presents the results of the second experiment, which shows the initial and final results of the students' performance across different difficulty levels, indicating an overall improvement in their scores and the time taken to solve the exercises. Additionally, Table IX provides a percentage breakdown of the results, highlighting the improvements in both IS and IRT after using the application. The results reflect a notable improvement in student performance after using the Code Showdown web application. Both the increase in scores and the reduction in solving times show a positive impact on the understanding and execution of programming tasks. The direct comparison between the two experiments is presented below to analyze the differential effect of the evaluated approaches.

TABLE VIII. INITIAL AND FINAL RESULTS OF E2

Initial Test				Final Test		
Student	**Difficulty**	**Score**	**Time (min.)**	**Difficulty**	**Score**	**Time (min.)**
Student 21	Easy	380	48	Easy	500	23
Student 22	Easy	340	45	Easy	400	25
Student 23	Easy	390	50	Easy	440	38
Student 24	Easy	300	47	Easy	400	27
Student 25	Easy	370	50	Easy	450	26
Student 26	Easy	280	43	Easy	330	30
Student 27	Easy	310	46	Easy	380	39
Student 28	Easy	300	42	Easy	300	34
Student 29	Easy	355	49	Easy	400	41
Student 30	Intermediate	315	44	Intermediate	350	32
Student 31	Intermediate	290	46	Intermediate	330	34
Student 32	Intermediate	250	43	Intermediate	400	31
Student 33	Intermediate	260	44	Intermediate	300	33
Student 34	Intermediate	320	47	Intermediate	470	28
Student 35	Intermediate	280	46	Intermediate	420	36
Student 36	Intermediate	240	40	Intermediate	450	29
Student 37	Hard	190	35	Hard	330	28
Student 38	Hard	260	42	Hard	310	30
Student 39	Hard	280	48	Hard	380	28
Student 40	Hard	170	39	Hard	350	30

TABLE IX. PERCENTAGE RESULTS OF E2

Difficulty	**IS (%)**	**IRT (%)**
Easy	18.90 ± 27.18	32.49 ± 12.49
Intermediate	40.67 ± 24.33	27.99 ± 5.87
Hard	58.63 ± 38.89	28.33 ± 9.57
Overall	34.46 ± 28.17	30.06 ± 10.91

C. Comparison of Results

In order to evaluate the differential impact between the two approaches applied, the percentage values of IS and IRT obtained in both experiments were compared. Table X summarizes the results obtained by level of difficulty.

TABLE X. COMPARISON OF RESULTS BY DIFFICULTY

Difficulty	**IS (E1)**	**IS (E2)**	**IRT (E1)**	**IRT (E2)**
Easy	9.14%	19.4%	20.0%	32.62%
Intermediate	13.01%	39.0%	19.2%	28.0%
Hard	23.06%	52.0%	22.56%	29.9%

Overall, experiment E2, based on the use of the Code Showdown application, achieved an average score improvement of 36.8%, compared to the 15.7% obtained in Experiment E1, which represents more than double the improvement over the traditional approach. As for solving

time, E2 achieved an average reduction of 30.2%, while E1 achieved a reduction of 20.6%. These results reflect that the use of the application not only allowed students to solve the challenges more accurately but also more efficiently, especially at intermediate and advanced difficulty levels.

These differences are visually illustrated in Figure 6, which shows the percentage increase in IS by difficulty level, and in Figure 7, which presents the percentage reduction in IRT.

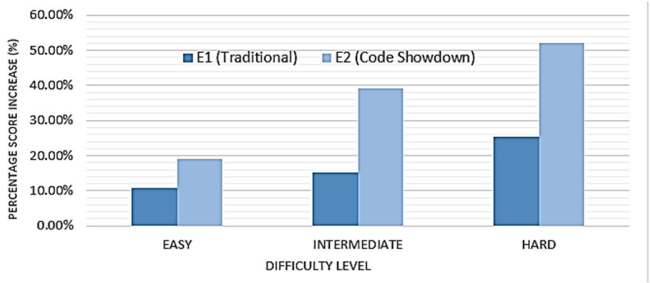

Fig. 6. IS (%) by difficulty level for Experiments 1 and 2 (mean ± SD; n=20/group).

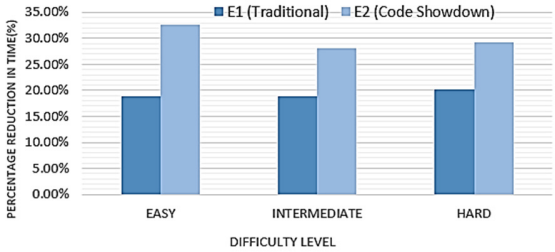

Fig. 7. IRT (s) by difficulty level for Experiments 1 and 2 (mean ± SD; n=20/group).

Across difficulty levels, the app-assisted group E2 outperformed the traditional group E1 in both accuracy and efficiency. In E2, mean IS reached 18.9% (Easy), 40.7% (Intermediate), and 58.6% (Hard) with corresponding IRT reductions of 32.4%, 28.0%, and 28.3%. In E1, mean IS was 12.2%, 15.5%, and 28.0%, with IRT reductions of 18.0%, 17.9%, and 19.0%, respectively. Paired-samples effect sizes were large for scores (E2 overall d=1.61) and very large in absolute magnitude for time (E2 overall $|d|$=2.42), reinforcing the practical significance of the improvements.

D. Survey

At the end of the experimental phase, a structured survey was applied to the 20 participants of Experiment E2 to collect their perception of the learning experience with the Code Showdown application. Regarding the Operability of the system, questions Q1 to Q5 measured aspects related to ease of use, navigation, technical stability, visual design, and clarity of feedback. As shown in Figure 8(a), the results were consistently high, with an overall mean of 4.4. Item Q5, related to the usefulness of feedback, presented a mean of 4.8 and was the highest rated item within this category, indicating a strong positive perception regarding the feedback generated by the system.

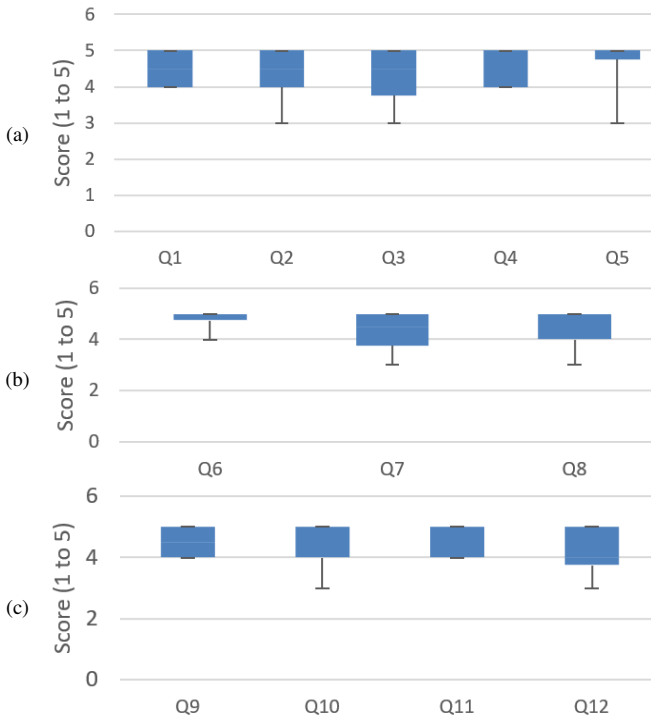

Fig. 8. Results of the survey on Operability (a), Functionality (b), and Learnability (C).

The greatest dispersion was observed in Q3 (technical performance), with a mean of 4.3, suggesting some punctual experiences of instability during the use of the system. Regarding Functionality, Figure 8(b) presents the results of questions Q6 to Q8, aimed at evaluating the appropriateness of the challenges, the adaptability of the difficulty, and the impact of the multiplayer mode on motivation. This dimension obtained an overall mean of 4.5, with Q6 standing out with an average of 4.8, reflecting that the majority of students felt that the challenges were well matched to their level. Questions Q7 and Q8 recorded averages of 4.3 and 4.5, respectively, with greater variability, indicating that the perception of adaptive difficulty and motivation generated by the competitive mode may depend on individual factors such as previous experience or learning preferences. Regarding Learnability, Figure 8(c) summarizes the scores for questions Q9 to Q12. This category had the highest mean among the three dimensions (4.6). Q11 achieved a mean of 4.9, standing out as the highest rated item in the entire survey. This data shows strong student support. The greatest dispersion was found in Q12 (comparison with other methods), with a mean of 4.3, which reflects the diversity of previous experiences among the participants.

V. LIMITATIONS

Despite the positive results obtained, three limitations must be acknowledged:

- The use of GPT-4 introduces significant computational costs that may affect the scalability of the system, particularly due to the token-based pricing model of the GPT-4 API.

- Although the application does not process sensitive personal data, it stores basic user information such as email addresses and encrypted passwords. Although this approach minimizes privacy risks, ensuring compliance with data protection regulations remains essential.
- The effectiveness of the system largely depends on the quality of prompt design. Prompt formulation is critical to obtaining coherent and pedagogically relevant responses from GPT-4, and even small variations in the wording can significantly influence the generated output.

VI. CONCLUSION

This study presented the development of an educational web application called Code Showdown that uses AI to improve the learning of the Python programming language. Forty Peruvian university students participated in the experiment, using the application to interact with personalized challenges and receive immediate feedback. GPT-4 served as the main engine, and adaptive learning techniques were employed. GPT-4 allowed personalized learning through exercise generation and feedback adapted to the level of the learner. The gamified experience and the multiplayer mode increased motivation and autonomous learning, supported by quantitative metrics and a high rating in the final survey. Its adaptability to different learning styles makes it a complementary and scalable tool for traditional teaching. The experimental results demonstrated notable improvements in student performance, with a score increase of up to 52.22% at the "Hard" level and a time reduction of 32.62% at the "Easy" level. The group using Code Showdown outperformed the traditional teaching group across all metrics. Survey results also showed high satisfaction, with average scores of 4.5 in Functionality, 4.4 in Operability, and 4.6 in Learnability. Future work should extend the application to other programming languages, such as Java, C++, and JavaScript, and apply advanced machine learning techniques to further personalize students' progress. Additionally, future research can compare the application's impact in different academic disciplines.

ACKNOWLEDGMENT

The authors thank the professors and students of the Faculty of Software, Systems, and Computer Engineering of the Universidad Peruana de Ciencias Aplicadas for their valuable participation and collaboration during the validation of this study, and the Research Department for its support through the UPC-Expost-2025-2 incentive.

ETHICS STATEMENT

All participants provided their informed consent before participating in the study. Student data were anonymized and processed according to institutional ethical guidelines. No personally identifiable information was collected or stored, and participation had no impact on academic evaluation. The anonymized datasets generated and analyzed during this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] I. Cingillioglu, U. Gal, and A. Prokhorov, "AI-experiments in education: An AI-driven randomized controlled trial for higher education research," *Education and Information Technologies*, vol. 29, no. 15, pp. 19649–19677, Oct. 2024, https://doi.org/10.1007/s10639-024-12633-y.

[2] Y. Jing, H. Wang, X. Chen, and C. Wang, "What factors will affect the effectiveness of using ChatGPT to solve programming problems? A quasi-experimental study," *Humanities and Social Sciences Communications*, vol. 11, no. 1, Feb. 2024, Art. no. 319, https://doi.org/10.1057/s41599-024-02751-w.

[3] K. Hartley, M. Hayak, and U. H. Ko, "Artificial Intelligence Supporting Independent Student Learning: An Evaluative Case Study of ChatGPT and Learning to Code," *Education Sciences*, vol. 14, no. 2, Jan. 2024, Art. no. 120, https://doi.org/10.3390/educsci14020120.

[4] D. López-Fernández, A. Gordillo, R. Lara-Cabrera, and J. Alegre, "Comparing effectiveness of educational video games of different genres in computer science education," *Entertainment Computing*, vol. 47, Aug. 2023, Art. no. 100588, https://doi.org/10.1016/j.entcom.2023.100588.

[5] B. Idrisov and T. Schlippe, "Program Code Generation with Generative AIs," *Algorithms*, vol. 17, no. 2, Jan. 2024, Art. no. 62, https://doi.org/10.3390/a17020062.

[6] J. San Martin, W. Romero, J. L. Castillo-Sequera, and L. Wong, "Talki: A Mobile Application to Improve English Learning of High School Students in Peru utilizing Virtual Reality and Gamification," *Engineering, Technology & Applied Science Research*, vol. 14, no. 5, pp. 17472–17481, Oct. 2024, https://doi.org/10.48084/etasr.8223.

[7] O. Dieste, E. R. Fonseca, G. Raura, and P. Rodríguez, "Efectividad del Test-Driven Development: Un Experimento Replicado," *Revista Latinoamericana de Ingenieria de Software*, vol. 3, no. 3, July 2015, Art. no. 141, https://doi.org/10.18294/relais.2015.141-147.

[8] T. Kosar, D. Ostojić, Y. D. Liu, and M. Mernik, "Computer Science Education in ChatGPT Era: Experiences from an Experiment in a Programming Course for Novice Programmers," *Mathematics*, vol. 12, no. 5, Feb. 2024, Art. no. 629, https://doi.org/10.3390/math12050629.

[9] F. Jiang and D. Shangguan, "Researching and designing educational games on the basis of 'self-regulated learning theory,'" *Frontiers in Psychology*, vol. 13, Nov. 2022, Art. no. 996403, https://doi.org/10.3389/fpsyg.2022.996403.

[10] Q. Fu, Y. Zheng, M. Zhang, L. Zheng, J. Zhou, and B. Xie, "Effects of different feedback strategies on academic achievements, learning motivations, and self-efficacy for novice programmers," *Educational technology research and development*, vol. 71, no. 3, pp. 1013–1032, June 2023, https://doi.org/10.1007/s11423-023-10223-2.

[11] Y. Hanggara, A. Qohar, and Sukoriyanto, "The Impact of Augmented Reality-Based Mathematics Learning Games on Students' Critical Thinking Skills," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 18, no. 07, pp. 173–187, Apr. 2024, https://doi.org/10.3991/ijim.v18i07.48067.

[12] Y. Li, D. Chen, and X. Deng, "The impact of digital educational games on student's motivation for learning: The mediating effect of learning engagement and the moderating effect of the digital environment," *PLOS ONE*, vol. 19, no. 1, Jan. 2024, Art. no. e0294350, https://doi.org/10.1371/journal.pone.0294350.

[13] C. G. Hidalgo Suarez, V. A. Bucheli-Guerrero, and H. A. Ordóñez-Eraso, "Artificial Intelligence and Computer-Supported Collaborative Learning in Programming: A Systematic Mapping Study," *Tecnura*, vol. 27, no. 75, pp. 175–206, Jan. 2023, https://doi.org/10.14483/22487638.19637.