












Article

Leveraging Large Language Models to Support Authoring Gamified Programming Exercises [†]

Raffaele Montella ¹,
Ciro Giuseppe De Vita ¹,
Gennaro Mellone ¹,
Tullio Ciricillo ¹,
Dario Caramiello ¹,
Diana Di Luccio ¹,
Sokol Kosta ²,
Robertas Damaševičius ³,
Rytis Maskeliūnas ³,
Ricardo Queirós ⁴
and Jakub Swacha ^{5,*}

- ¹ Department of Structures for Engineering and Architecture (DiSt), University of Naples “Parthenope”, 80143 Naples, Italy; raffaele.montella@uniparthenope.it (R.M.); cirogiuseppe.devita@uniparthenope.it (C.G.D.V.); gennaro.mellone1@studenti.uniparthenope.it (G.M.); tullio.ciricillo@studenti.uniparthenope.it (T.C.); dario.caramiello001@studenti.uniparthenope.it (D.C.); diana.diluccio@uniparthenope.it (D.D.L.)
- ² Department of Electronic Systems, Aalborg University, 2450 Copenhagen, Denmark; sok@es.aau.dk
- ³ Center of Excellence Forest 4.0, Faculty of Informatics, Kaunas University of Technology, 51423 Kaunas, Lithuania
- ⁴ Center for Research in Advanced Computing Systems (CRACS), INESC TEC, 4169-007 Porto, Portugal; ricardoqueiros@esmad.ipp.pt
- ⁵ Department of Information Technology in Management, University of Szczecin, 70-453 Szczecin, Poland
- * Correspondence: jakub.swacha@usz.edu.pl
- [†] This paper is an extended version of our paper published in Proceedings of the 25th International Conference on Artificial Intelligence in Education (AIED 2024), Recife, Brazil, 8–12 July 2024 as a part of the Late-Breaking Results track, under the title “GAMAI, an AI-Powered Programming Exercise Gamifier Tool”.

Featured Application: The presented solution can be applied to simplify and hasten the development of gamified programming exercises conforming to the Framework for Gamified Programming Education (FGPE) standard.



Citation: Montella, R.; De Vita, C.G.; 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. *Appl. Sci.* **2024**, *14*, 8344. <https://doi.org/10.3390/app14188344>

Academic Editor: Luis Javier Garcia Villalba

Received: 5 August 2024

Revised: 10 September 2024

Accepted: 13 September 2024

Published: 16 September 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract: Skilled programmers are in high demand, and a critical obstacle to satisfying this demand is the difficulty of acquiring programming skills. This issue can be addressed with automated assessment, which gives fast feedback to students trying to code, and gamification, which motivates them to intensify their learning efforts. Although some collections of gamified programming exercises are available, producing new ones is very demanding. This paper presents GAMAI, an AI-powered exercise gamifier, enriching the Framework for Gamified Programming Education (FGPE) ecosystem. Leveraging large language models, GAMAI enables teachers to effortlessly apply storytelling to describe a gamified scenario, as GAMAI decorates natural language text with the sentences needed by OpenAI APIs to contextualize the prompt. Once a gamified scenario has been generated, GAMAI automatically produces exercise files in a FGPE-compatible format. According to the presented evaluation results, most gamified exercises generated with AI support were ready to be used, with no or minimum human effort, and were positively assessed by students. The usability of the software was also assessed as high by the users. Our research paves the way for a more efficient and interactive approach to programming education, leveraging the capabilities of advanced language models in conjunction with gamification principles.

Keywords: gamification; programming education; educational tools; artificial intelligence

1. Introduction

The rising demand for skilled programmers reflects the pivotal role of programming in various areas of the contemporary world [1]. However, the formidable challenge of learning programming is an obstacle to meeting this demand. The integration of automated assessment tools has shown promise in providing swift feedback to students as they experiment with their code, facilitating a more efficient learning process [2]. In tandem, gamification

has become a powerful strategy to enhance motivation and engagement, intensifying students' efforts to master programming concepts [3]. Together, these approaches form the backbone of the Framework for Gamified Programming Education (FGPE) ecosystem [4]. The practical usefulness of the FGPE relies on the availability of gamified programming exercises covering diverse topics.

In the context of this work, we define a gamified programming exercise as an exercise aimed at testing the student's ability to solve a specified problem, with the use of any common or peculiar programming language feature or features, embedded in a narrative scenario (considered as a gamification element, following, e.g., [5–7]), in which solving the exercise is part of a wider challenge. On the technical level, a gamified programming exercise is a programming exercise enhanced with an additional layer of metadata, defining, e.g., the conditions for accessing the exercise and the rewards for solving it, conforming to a standardized format (e.g., GEdIL [8]), making it possible to use the exercise within a gamified interactive learning environment, such as the FGPE PLE [4].

Although there are open-licensed collections of these exercises [9], more are needed to let students practice the same techniques in different contexts or to provide each student with exercises different from their peers. Creating new gamified exercises remains a significant challenge for educators, teachers, and trainers [4].

This paper, an extended version of our late-breaking work presented at the 25th International Conference on Artificial Intelligence in Education (AIED 2024) [10], addresses this challenge by introducing GAMAI, an artificial intelligence (AI)-powered exercise gamifier integrated into the Framework for Gamified Programming Education (FGPE) ecosystem. GAMAI leverages OpenAI API [11] to streamline the process of creating gamified programming exercises. Through GAMAI, teachers can employ a storytelling approach to articulate the gamified scenario. The system automatically enhances natural language text with essential sentences required by OpenAI API to contextualize the prompt. Once the gamified scenario has been generated, GAMAI further automates the production of specification files that can be easily edited within FGPE AuthorKit [4]. This innovative approach aims to alleviate the burden on educators, facilitating the seamless creation and integration of gamified programming exercises into educational curricula. The novel contribution of this paper is the processing approach to text prompt augmentation. The proposed algorithm decorates the teachers' gamified storytelling, to enable an automatic gamified programming exercise playground.

The rest of this paper is organized as follows: Section 2 gives an overview of using AI for the automatic generation of programming exercises; Section 3 describes the Framework for Gamified Programming Education ecosystem targeted by GAMAI, how GAMAI was designed and implemented, and how it operates; a qualitative evaluation of the exercises generated with GAMAI is reported in Section 4; finally, the paper content is summarized in the final Section 5, with concluding remarks and an outline of the next steps planned for future research.

2. Related Work

AI tools are pivotal in enhancing the teaching–learning process of computer programming by providing personalized, adaptive, and interactive learning experiences [12]. In programming education, AI can assist educators and learners in various aspects, including automated feedback and code review [13], adaptive learning paths [14], programming tutoring systems [15], automated assessment and grading [16], code summarization and documentation [17], and code generation [18–21].

Another way of exploiting AI's capabilities is by generating programming exercises. There is a notable interest in such generators, as creating programming exercises that are diversified and challenging for learners can be time-consuming for educators. The generated exercises can be tailored to cater to different skill levels, ensuring a dynamic learning environment that adapts to individual learner needs [22].

Kurdi et al. [23] conducted a systematic review of automatic exercise generation in many different domains, such as geometry, history, logic, programming, and science. It included 93 papers between 2015 and 2019 that tackled the automatic generation of exercises for educational purposes. The study concluded that there needs to be more tools for generating exercises of controlled difficulty and with facilities like enriching question forms and structures, automating template construction, improving presentation, and generating feedback.

Zavala and Mendoza [24] presented a tool (version 1.0) that uses automatic item generation (AIG) to address the problem of creating many similar programming exercises using predefined templates that are used for quizzes. The main goal was to ensure consistency in testing many students with questions of the same difficulty level.

Goliath (version 1.0) is an application that automatically generates programming exercises using a template system. With a Python programming exercise generator, Goliath targets the facilitation of teachers' day-to-day tasks. The tool creates templates, employing two AI models—one for generating the basic text of the statement from keywords, and another for extracting source code.

Agni (version 1.2) [25] serves as a dynamic code playground tailored for learning JavaScript. In recognition of the challenges posed by manual exercise creation, Agni's creators recently introduced a new back end with an exercise generation component powered by the ChatGPT API. This integration automates the exercise creation process by generating statements, solution code, and test cases. The tool supports the IMS LTI specification, allowing seamless integration with learning management systems (LMS) such as Moodle, Blackboard, and Canvas.

ExGen (version 1.0) [26] focuses on generating ready-to-use exercises for the specific difficulty level and concept the student is working on. It leverages the latest advances in LLMs to autogenerate many novel exercises and filter them to ensure they suit students.

TESTed (version 1.0) [27] is an educational testing framework that supports the creation of programming exercises with automated assessment capabilities in a programming-language-independent manner. TESTed combines the advantages of unit testing with output comparison, providing a versatile solution for educational assessment.

Recently, works have been increasingly leveraging pre-trained LLMs for educational purposes [28]. The most recent and relevant work [29] in automatic exercise generation using novel AI technologies (pre-trained LLMs) explored OpenAI Codex (which has been unavailable since March 2023) to create new programming exercises and code explanations. They found many Codex-generated exercises sensible and novel, but others needed clarification regarding problem statements and missing or faulty test cases.

On the other hand, Kasneci et al. [30] discussed the potential benefits, for instance, content generation and personalized learning, as well as challenges, e.g., model biases, system brittleness, etc., of applying LLMs to education. Similarly, Becker et al. [22] elaborated on the educational opportunities of AI code generation and how educators should act quickly given these developments.

Table 1 provides a comparative overview of some of the mentioned tools, based on their generation method, supported formats, and other relevant criteria.

Regarding the generation method, Goliath relies on a template-based approach using specific template fields for customization. Conversely, Agni integrates the ChatGPT API, enabling on-demand customization through dynamic interaction with the language model. ExGen leverages LLMs for exercise generation, while Zavala's tool employs semantic-based automatic item generation (AIG) to create contextual programming exercises dynamically. TESTed uses unit testing, combining the advantages of traditional testing methodologies with generic output comparison.

The tools present several output formats: Goliath generates exercises with DSL-formatted text, ensuring a structured and consistent output. Agni outputs exercises in JSON format, providing a machine-readable and versatile representation. ExGen produces exercises in natural language, maintaining simplicity and accessibility. The remaining tools'

output format varies based on the semantic-based AIG and depending on the unit testing approach used.

Table 1. Comparison of automated programming exercise generation tools.

Tool	Goliath (Version 1.0)	Agni (Version 1.2)	ExGen (Version 1.0)	Zavala (Version 1.0)	TESTed (Version 1.0)
Generation Method	Template-based	ChatGPT API	LLM-based	Semantic-based AIG	Unit Testing
Variability and Customization	Template Fields	On-Demand Customization	On-Demand Customization	LOD Integration	Varies
Output Formats	DSL-formatted text	JSON	Natural language	Varies	Varies
Difficulty Levels and Tags	Difficulty/Tags	Difficulty/Tags	Difficulty	Difficulty	Difficulty
Integration with Environments	No	IMS LTI Integration	API Integration	API Integration	API Integration
Semantic-Based Generation	No	No	No	Yes	No

All tools allow the association of difficulty levels. This feature ensures instructors can customize exercises based on the student’s proficiency levels. In Goliath and Agni, it is possible to categorize exercises according to specific topics or concepts.

Regarding interoperability, Agni integrates with learning environments such as learning management systems (LMS) using IMS LTI specifications. All the remaining tools (except Goliath) offer API integration, allowing easy incorporation into existing educational platforms.

Finally, regarding semantic-based generation, while Goliath, Agni, and TESTed do not explicitly focus on semantic-based generation, ExGen uses recent advances in pre-trained large language models (LLMs) for automatic exercise generation. Zavala’s tool stands out by employing semantic-based AIG with linked open data integration, enhancing the contextual relevance of exercises.

While there are known examples of supporting gamification with AI methods [31], none of the existing tools and frameworks for programming exercise generation known to the authors (including those described above) are capable of generating gamified programming challenges, like GAMAI proposed here. The closest veins of research we were able to identify are the automatic generation of quizzes on the topic of software specifications and testing [32], and the automatic generation of challenges for gamification systems relatively distant in scope from programming education: ear training for music theory classes [33] and sustainable urban mobility [34]. It was this gap in the current research landscape that sparked our motivation to develop GAMAI [10], which is described in the following section.

3. Solution

3.1. Target Ecosystem: FGPE

The Framework for Gamified Programming Education (FGPE) has succeeded in establishing a technical milieu for incorporating gamification methods into programming education. It covers various aspects, including gamified exercise formats, exemplary collections, and the necessary supporting software [4]. The FGPE is programming-language-agnostic (allowing students to select their preferred programming language to tackle exercises), features multilingual instruction for students (who can effortlessly navigate between languages to obtain exercise requirements in the form most comprehensible to them) and extensive customization—encompassing both the learning content (educators can compose original courses, reuse exercises from the provided open repositories [9], or create brand new exercises, thus tailoring the learning content to the specific needs of the class or individual students) and the gamification rules (educators can select pre-existing gamified courses or create their own, drawing specifications of gamification rules from the provided courses or developing their own ones). It also provides integration with any LTI-compliant [35] learning management system (LMS) or massive open online

course (MOOC), allowing for seamless synchronization of student identities, activities, and learning outcomes.

The key components of the FGPE ecosystem include:

- FGPE AuthorKit: serves the dual purpose of preparing and managing both programming exercises and gamification rules [4].
- GitHub-hosted Open Repository: serves as a centralized hub for gamified programming exercises [9].
- FGPE Gamification Service: processes gamification rules and manages the overall game state [36].
- Mooshak sandbox: executes programs submitted by students and autonomously assesses their performance [37].
- FGPE PLE (Programming Learning Environment): a progressive web app which lets students access gamified exercises, solve them, and receive graded feedback, whereas teachers can use the PLE to organize exercise sets, grant access to students, and monitor their learning progress [4].

The gamified programming exercises within the FGPE ecosystem are encapsulated in educational content in two distinct formats. These formats serve to articulate programming exercises and gamification layers, respectively. The first format, known as YAPeXIL [38], is dedicated to meticulously describing programming exercises, ensuring clarity and precision in their presentation. On the other hand, the second format, GEdIL [8], is specifically tailored for the representation of gamification layers, providing a structured framework for integrating game elements into educational materials.

3.2. Design and Implementation

Figure 1 provides a comprehensive visualization of the overarching Framework for Gamified Programming Exercises (FGPE) architecture, emphasizing the innovative contribution of the proposed AI-powered exercise gamification solution. Designing, implementing, training, and reinforcing a dedicated generative pre-trained transformer-based large language model from scratch was outside of the scope of this research, so we designed GAMAI with leveraging OpenAI API [11] usage in mind. However, we considered a possible scenario in which different resources of a similar kind are used. The GPT Abstraction Layer component abstracts the interface to OpenAI API, enabling FGPE developers to test GAMAI with diverse and different large language model services. GAMAI uses generative AI LLMs to help creators implement exercises and challenges. Although the technical interactions between the GPT abstraction layer and the actual AI service provider are similar for both cases, the prompt engineering is different, as explained below.

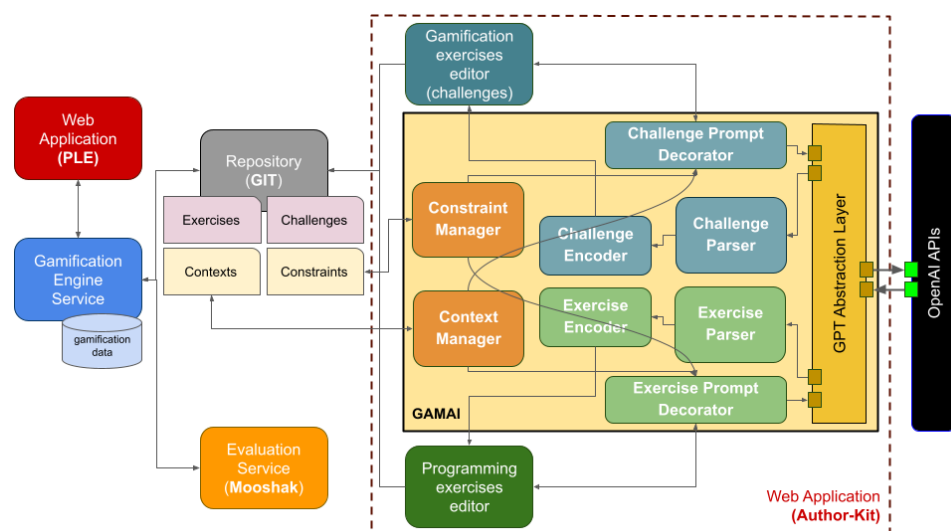


Figure 1. GAMAI placed within the FGPE architecture.

AI-assisted programming exercise creation. GAMAI's main aim is to assist teachers, educators, and trainers in creating gamified exercises for programming education. In this work, the creator is the human interacting with GAMAI to create the gamified exercises and the related gamified scenario. That means the creator's expertise is crucial in setting up the context in which the generative AI model has to imitate human behavior. The OpenAI GPT generative model used to develop the GAMAI prototype enables context settings using the "act as" clause. For example, we can consider a computer science research assistant who has to generate exercises for a first-year introduction to computing programming using the C language for bachelor students in a computer science degree class. The generative AI model must reflect the creator's cultural background and experience, acting appropriately. The AI generative model context setting sentence has to be assembled as a well-formed context selection prompt for the GPT service containing the following *act as* selectors: (i) the background skills of the creator; (ii) the kind of class for which the exercises have to be created; and (iii) the level of the exercises. In a different scenario, the creator could be a full professor of computer science who has to generate exercises for advanced algorithms and data structures using Python programming for a master's degree in computer engineering. Contexts are stored in the project repository and managed by the Context Manager component. For AI-based programming exercises, the context is consistent by assembling it in each session, ensuring exercises created by different creators for the same target audience are consistent.

While the nature of the exercises can vary widely, the kind of gamification augmentation is limited by the FGPE core implementations. A typical kind of exercise gamification leverages approaches like, but not limited to, (i) fill the gap; (ii) find the bug; or (iii) complete the code. Finally, the exercise is evaluated by the Mooshak component; thus, it has to be constrained so that the evaluation service can correctly perform scoring. To prepare a prompt considering this constraint, we developed a Constraint Manager component leveraging diverse and different constraints stored in the project repository alongside the Exercises, Challenges, and Contexts above.

AI-assisted gamified scenario creation. To create gamified challenges, the creator's skills could belong to a completely different area, in particular, focused on digital humanities, communications, and professional storytelling. Before each GPT service interaction, a prompt has to be built, performing a decoration stage in which the critical points of the "act as" statement are formed. For example, the creator can be an expert in fantasy saga writing, defining a story's characters and main plot. The plot branching and twists can be mapped on gamified exercises, adding custom fields to the results. Once the students have solved a quest, the story continues along a different branch. The Context Manager and the Constraint Manager components provide the prompt decoration with data from the project repository needed to set the context automatically, ensuring the consistency of the automatically generated content.

The Exercise Prompt Decorator is the component devoted to adding the needed ancillary information to the creator-generated prompt describing the kind of exercise. Listing 1 represents the prompt produced by this component. Line 1 is set by gathering the exercise creator skill from the project setup (default exercise creator skills) or the exercise collection level. Line 2 is directly written by the exercise creator, specifying the number and the kind of exercises to be generated. Line 3 can be assembled automatically using the GAMAI exercise prompt user interface or from details described by the exercise creator. Finally, line 4 is automatically added to retrieve results directly in JSON, to simplify the parsing of the results.

Listing 1. Gamified exercise generation prompt example (1).

- 1 Acting as a research assistant in **computer** science.
- 2 Generate 10 gamified programming exercises of increasing difficulty **for** a first-year introduction to computing programming using the C language **for** bachelor students in a **computer** science degree class.
- 3 The answer to the exercise's correct solution has to be chosen from a **set** of 5.
- 4 Return the result in json.

The Challenge Prompt Decorator is the component devoted to adding the main plot and the kind of quests, with related plot branching and plot twisting, to the creator-generated prompt describing the gamification scenario. Listing 2 represents the prompt produced by this component. As for the Exercise Prompt Decorator, line 1 is gathered from the contexts repository at the project level. In the example, the creator's skills are not technical. Line 2 is written directly by the content author and is the main statement for the AI-assisted world generation. Line 3 is a statement provided by the content creator specifying if each student (in the gamified and real worlds) has to compete alone or in teams. Line 4 states how the quest concludes and if all the participants can win the award or only one. Line 5 is the quest generation statement. The creator writes it using the GAMAI challenges prompt user interface. In line 6, one can observe the kinds of plot branches from one quest to the next. In this example, the quests have to be solved in sequential order. Line 7 instructs the GPT provider to generate the results in JSON. Finally, line 8 defines how the quest has to be connected to the exercise repository, the exercise kind (for example, multiple choice), and the exercise's difficulty level.

Listing 2. Gamified scenario generation prompt example.

- 1 Acting as an expert fantasy storyteller.
- 2 Create a world where young boys and girls in high school attend wizarding schools.
- 3 Each student has to compete alone to solve a quest.
- 4 Each student solving the final quest wins the wizard certificate.
- 5 Generate 10 different **magic** quests of increasing difficulty.
- 6 The wizard student **who** solves the first quest can continue with the second one, and so on.
- 7 Return the result as json.
- 8 Add to each quest the field:
- 9 `''exercise'' : {`
- 10 `''repository'' : ''__world__'',`
- 11 `''kind'' : ''multiple_choices'',`
- 12 `''level'' : ''__iteration__''`
- 13 `}`

Once the Exercise Prompt Decorator and the Challenge Prompt Decorator have provided the decorated prompts, the provided GPT service is invoked and the received AI-generated content has to be parsed by the Exercise Parser and the Challenge Parser. Both components have a similar behavior, interpreting the results provided by the GPT service via the GPT Abstraction Layer. The parsers map the data produced by the GPT services onto FGPE entities, providing semantic consistency. The gamified exercises and scenarios produced by the GPT service could be described without considering the peculiarity of the FGPE ecosystem. The Exercise Encoder component acts as a data sink for the Exercise Parser, producing an exercise representation that is fully compliant with the FGPE gamified exercise schema (YAPExIL [4]). Thanks to this compatibility, the exercise creator can use the AuthorKit tool to fully manage the exercises by adding or removing functional and ancillary parts. This component enables the creator to, for example, deco-

rate the exercise description with diverse and different input dataset providers or custom exercise correctness checkers. The Challenge Encoder plays a pivotal role in the processing pipeline by receiving data generated from the Challenge Parser. Its primary function is to transform its input data into a gamified scenario representation that strictly conforms to the FGPE gamified scenario schema (GEdIL [8]). It ensures a standardized and cohesive structure for the gamified exercises within the framework. As was the case with exercises, AuthorKit affords exercise creators extensive control over the gamified scenarios, providing the flexibility to add or remove both functional and ancillary components, thus allowing for a tailored and dynamic exercise design process. Whether incorporating specific functionalities to enhance the challenge or streamlining the scenario by eliminating non-essential elements, AuthorKit serves as a versatile interface for refining the AI-generated gamified content. In essence, the Challenge Encoder and AuthorKit's Challenge Editor together form a cohesive unit within the FGPE architecture. While the former ensures adherence to the established gamified scenario schema, the latter provides a user-friendly and feature-rich environment for exercise creators to fine-tune and customize gamified scenarios according to their pedagogical objectives and creative preferences. This integrated approach underscores the commitment to standardization and flexibility in designing and implementing gamified programming exercises. As we explore further functionalities and interactions within this architecture, the synergistic relationship between these components becomes increasingly apparent, contributing to the effectiveness and adaptability of the overall GAMAI-enhanced framework.

3.3. Operation

GAMAI seamlessly integrates cutting-edge technologies, focusing on leveraging OpenAI's robust API. The user-facing front end establishes a nuanced dialogue with the back end, which is engineered to handle request processing and integrate with OpenAI's advanced capabilities for exercise creation. As the core computational engine, the back end diligently manages user intent transmission to OpenAI and intelligently interprets the AI-generated responses. The diagram in Figure 2 depicts the complete sequence.

The subsequent phase involves the back end transitioning from a relay to an intelligent parsing engine, proficiently dissecting the received messages and extracting essential components for exercise creation. The back end synthesizes exercises, strictly adhering to the YAPEXIL format. A critical milestone is achieved, as the back end molds OpenAI's raw output and dispatches a curated list of exercises to the front end. This achievement signifies the completion of a complex orchestration and delivers a tangible educational resource to end-users, featuring exercises tailored to computer-science concepts. The user-centric front end acts as the interactive interface, empowering users to influence the exercise generation process actively. Acknowledging user agency and integrating it into the system, the interface allows users to articulate bespoke requests directly to OpenAI API, injecting personalized dimensions into the exercise generation pipeline. This user-initiated input allows imbuing exercises with contextual relevance, aligning educational content precisely with computer science students' nuanced demands and preferences. This meticulously designed system stands at the forefront of AI-driven educational technology in computer science, redefining the way educational content is created. The intricate interplay of components, spanning front-end interactions, back-end orchestration, and AI-output parsing, holds transformative potential for the pedagogical aspects of computer science education by automating the process with precision, customization, and engagement.

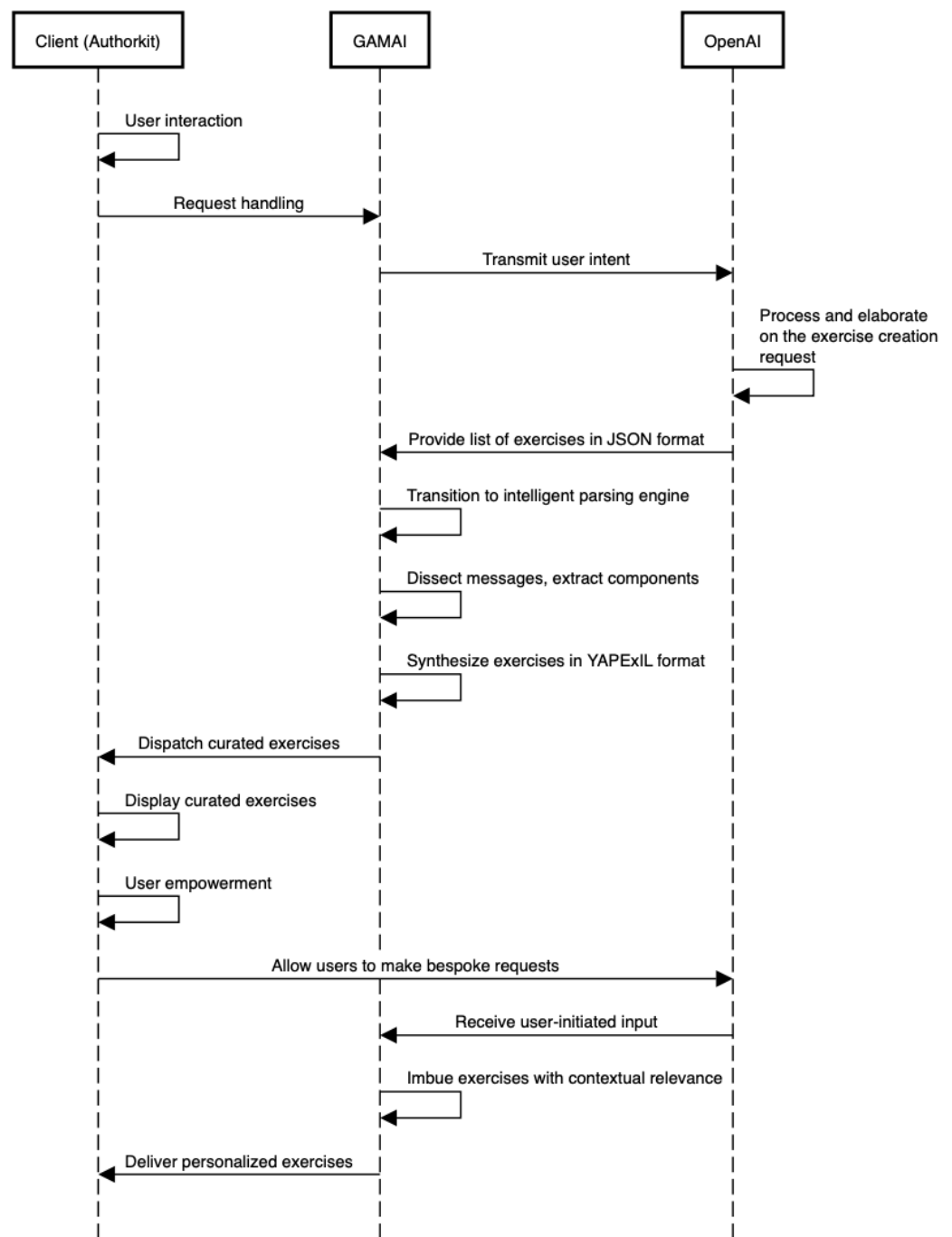


Figure 2. GAMAI sequence diagram while creating an exercise using OpenAI API.

4. Evaluation and Discussion

4.1. Evaluation Methods and Procedures

Although generating a set of exercises automatically is much less time-consuming for the creators than developing them manually, the automatically generated exercises often need to be refined to meet expectations, which still requires some human effort.

Listing 3 contains one of the exercises produced as the result of the prompt described in Listing 4.

Listing 3. Example of an AI-generated gamified exercise (1).

```

1  ...
2  {
3    "question": "Which of the following is the correct way to
      declare an integer variable named 'count' in C?",
4    "options": [
5      {"value": "int count;", "correct": true},
6      {"value": "integer count;", "correct": false},
7      {"value": "count int;", "correct": false},
8      {"value": "declare count as int;", "correct": false},
9      {"value": "variable count is int;", "correct": false}
10   ]
11 },
12 ...

```

Listing 4. Gamified exercise generation prompt example (2).

```

1  Acting as a seasoned full professor in computer science.
2  Generate 100 gamified programming exercises of increasing difficulty
      for a
3  ...
4  The answer to the exercise's correct solution has to be chosen from a
      set of 5.
5  Return the result in json.

```

Because, to the best of our knowledge, a standardized procedure for evaluating the quality of automatically generated human language exercise texts has not yet been made available in the literature, we had to define a tentative protocol as follows:

Considering as exercise *text* all the human language content, assigning a score, in parentheses, for each needed fix, we classified the efforts as follows:

1. We set up a team of experts composed of five assistant professors in computer science for human-based computer-generated text evaluation.
2. We set up the evaluation grid in Table 2 to score the automatically generated exercise texts. Our effort was focused on trying to homogenize the evaluation provided by the team of experts.
3. In order to assess the human effort needed to make the automatically generated gamified exercises usable, we choose to generate 10 exercises for each of the following prompts as part of line three of Listing 4:
 - first-year introduction to computer programming using the C language for bachelor students in a computer science degree class [1CCS];
 - first-year introduction to computer programming using the Python language for bachelor students in a computer science degree class [1PCS];
 - first-year introduction to computer programming using the Python language for bachelor students in an environmental science degree class [1PES];
 - first-year introduction to computer programming using the Python language for bachelor students in a law degree class [1PLS];
 - third-year object-oriented computer programming using the Java language for bachelor students in a computer engineering degree class [3JCE];
 - third-year object-oriented computer programming using the Java language for bachelor students in a computer science degree class [3JCS].
4. Finally each member evaluated all the AI automatically generated programming exercise texts, scoring each one with a value from 0 to 5; then, for each exercise, the mean score was evaluated and then rounded to the closest integer.

The AI-generated exercise types (1CCS, 1PCS, 1PES, 1PLW, 3KCE, and 3JCS) were chosen to match the evaluation team's direct experiences in dealing with the related topics.

We are aware that this could have been a source of bias, but at the current stage of our studies, we preferred to maximize the overall consistency in exercise quality evaluation. The results discussed in the following section have been grouped by exercise type.

Table 2. The evaluation grid for AI-generated programming exercise texts.

Score	Evaluation
0	The exercise can be used as is
1	The text is unclear, it needs rearrangement
2	The text must be completely rewritten, but the exercise is formerly correctly generated
3	The text is correct, but there are issues from a technical point of view
4	The exercise, although correct in the text part, is too easy or too hard considering the intended difficulty level
5	The exercise cannot be used as is; it must be dropped or completely rewritten; for example, a code snippet is entirely missing

4.2. Results and Discussion

The experiment results (Table 3) show that most generated exercises were ready to be used with no or minimum human effort to make them comply with the expectations (61.7%).

Nevertheless, there were non-neglectable cases in which the code snippet had to be manually added, especially for high-difficulty-level exercises (28.3%).

Cases of entirely useless automatic generations were uncommon (10.0%).

Another point to focus on is that the generated exercise quality was related to the kind of programming language. Most of the provided C-language exercises were ready to be used, whereas more Python exercises needed the human creator's intervention.

Table 3. Human effort needed to ensure usability of created gamified exercises. Experimental results.

Prompt	0	1	2	3	4	5	Score
1CCS	5	4	0	1	0	0	7
1PCS	4	0	0	4	2	0	20
1PES	6	0	0	4	0	0	12
1PLW	3	1	2	3	0	1	19
3JCE	4	3	0	3	0	0	12
3JCS	6	1	0	0	3	0	13
%	40.0	21.67	3.33	25.0	8.33	1.67	

The improvement in the speed of the exercise generation and gamification layer creation is beyond doubt.

Nevertheless, some discussion about using the FGPE and GAMAI in a real-world production environment is needed. As shown by the example given in this section, the text generated by the GPT service could be affected by inaccuracies, biases, and intellectual property issues.

While checking each automatically generated exercise is a task that could be considered mandatory for the content creator, and tedious but not complicated to perform, the same cannot be said about the quality check verification of the world created for the gamification scenarios' automatic generation. Arguably, due to the strategy used by Open AI to train and reinforce their models, they are strongly influenced by popular culture, causing a potential issue regarding copyright management.

For example, in Listing 5, there is a field named “wizarding_schools” that conveys the name of three different schools of wizardry in the fantasy world of Eldoria [39]. In the used prompt (Listing 2), no mention was made of this entity, while the result produced is conceptually and numerically similar to literature protected by intellectual property rights. Note that this particular problem has also been observed in other use cases of Open AI’s models [40].

Listing 5. Example of an AI-generated gamified exercise (2).

```

1  {
2    "world": {
3      "name": "Eldoria",
4      "description": "A magical realm where wizarding schools are
                    scattered across mystical landscapes, and young sorcerers
                    engage in solo quests to prove their magical prowess."
5    },
6    "wizarding_schools": [...],
7    "quests": [
8      {
9        "name": "The Enchanted Amulet",
10       "description": "Retrieve the Enchanted Amulet hidden within
                    the Forbidden Forest, guarded by mystical creatures and
                    protected by a powerful enchantment.",
11       "exercise": { "repository": "__world__", "kind": "
                    multiple_choice", "level": "__iteration__" }
12     },
13     ...

```

5. Conclusions

This paper presents our findings from applying a large language model based on a generative pre-trained transformer to help generate gamified exercises designed for programming education, in a form compatible with the FGPE ecosystem. Given the complexity of designing, implementing, training, and reinforcing such natural language processing models, we opted to leverage the capabilities of a commercially available robust API service.

The results of the evaluation of the quality of the generated exercises presented in Section 4 are positive, with most of the exercises being usable as they are or only requiring slight corrections.

We are nonetheless aware of the limitations of our research. Threats to the validity and generalizability of the presented results stem from both the limited number of the evaluated exercises (60 in total) and the subjective character of the experts’ evaluation. It is possible that another batch of generated exercises could contain more flaws, and another group of experts could evaluate the same exercises more harshly.

Despite these research limitations, with GAMAI, we have successfully demonstrated that the time needed for developing gamified programming exercises can be significantly reduced by employing LLMs, which motivates further research on this topic.

Our future research will focus on two primary avenues. First, we aim to refine the components responsible for prompt decoration and parsing the generated text. The ultimate goal is to greatly minimize or, if possible, completely eliminate the need for human intervention within the editing process, thereby streamlining the workflow. Second, we envision the creation of a formally defined test suite to evaluate the performance of student learning with gamified exercises generated with the support of GAMAI, comparing it with the performance of students learning with manually developed gamified exercises. Such a research would provide valuable insights into the efficacy of incorporating automatically generated gamified content into educational settings.

Overall, our research paves the way for a more efficient and interactive approach to programming education, leveraging the capabilities of advanced language models in conjunction with gamification principles.

Author Contributions: Conceptualization, R.M. (Raffaele Montella) and J.S.; methodology, R.M. (Raffaele Montella); software, T.C. and D.C.; validation, C.G.D.V. and G.M.; investigation, R.M. (Raffaele Montella), C.G.D.V. and G.M.; resources, C.G.D.V., G.M. and D.D.L.; data curation, C.G.D.V., G.M. and D.D.L.; writing—original draft preparation, R.M. (Raffaele Montella), D.D.L., J.S., R.Q., R.D. and R.M. (Rytis Maskeliunas); writing—review and editing, R.M. (Raffaele Montella), J.S. and S.K.; visualization, R.M. (Raffaele Montella), C.G.D.V., G.M. and J.S.; supervision, R.M. (Raffaele Montella); project administration, J.S.; funding acquisition, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was co-funded by the European Union, grant number 2023-1-PL01-KA220-HED-000164696.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data presented in this study are available on request from the first author.

Acknowledgments: The authors would like to thank José Carlos Paiva for his support in software development.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Vicuña, C. What Are Europe's Most Needed Software Roles and Skills? Trends to Look Out For. 2022. Available online: <https://digital-skills-jobs.europa.eu/en/latest/news/what-are-europes-most-needed-software-roles-and-skills-trends-look-out> (accessed on 10 September 2024).
2. Paiva, J.C.; Leal, J.P.; Figueira, Á. Automated assessment in computer science education: A state-of-the-art review. *ACM Trans. Comput. Educ. (TOCE)* **2022**, *22*, 1–40. [CrossRef]
3. Zhan, Z.; He, L.; Tong, Y.; Liang, X.; Guo, S.; Lan, X. The effectiveness of gamification in programming education: Evidence from a meta-analysis. *Comput. Educ. Artif. Intell.* **2022**, *3*, 100096. [CrossRef]
4. Paiva, J.C.; Queirós, R.; Leal, J.P.; Swacha, J.; Miernik, F. Managing gamified programming courses with the FGPE Platform. *Information* **2022**, *13*, 45. [CrossRef]
5. Marczewski, A. 52 Gamification Mechanics and Elements. 2015. Available online: <https://www.gamified.uk/user-types/gamification-mechanics-elements> (accessed on 10 September 2024).
6. Werbach, K.; Hunter, D. *For the Win: How Game Thinking Can Revolutionize Your Business*; Wharton Digital Press: Philadelphia, PA, USA, 2012.
7. Chou, Y.K. *Actionable Gamification: Beyond Points, Badges, and Leaderboards*; CreateSpace: Scotts Valley, CA, USA, 2015.
8. Swacha, J.; Paiva, J.C.; Leal, J.P.; Queirós, R.; Montella, R.; Kosta, S. GEdIL—Gamified Education Interoperability Language. *Information* **2020**, *11*, 287. [CrossRef]
9. FGPE Consortium. Repository of Gamified Exercises. 2021. Available online: <https://github.com/jcpaiva-fgpe?tab=repositories> (accessed on 10 September 2024).
10. Montella, R.; Giuseppe De Vita, C.; Mellone, G.; Ciricillo, T.; Caramiello, D.; Di Luccio, D.; Kosta, S.; Damasevicius, R.; Maskeliunas, R.; Queiros, R.; et al. GAMAI, an AI-Powered Programming Exercise Gamifier Tool. In *Artificial Intelligence in Education: Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky—25th International Conference on Artificial Intelligence in Education (AIED 2024), Recife, Brazil, 8–12 July 2024, Proceedings, Part I*; Olney, A.M., Chounta, I.A., Liu, Z., Santos, O.C., Bittencourt, I.I., Eds.; Springer Nature: Cham, Switzerland, 2024; pp. 485–493. [CrossRef]
11. OpenAI. OpenAI API. 2020. Available online: <https://openai.com/index/openai-api/> (accessed on 10 September 2024).
12. Freitas, T.C.; Costa Neto, A.; Pereira, M.J.V.; Henriques, P.R. NLP/AI Based Techniques for Programming Exercises Generation. In Proceedings of the 4th International Computer Programming Education Conference (ICPEC 2023), Dagstuhl, Germany, 26–28 June 2023. [CrossRef]
13. Messer, M.; Brown, N.C.C.; Kölling, M.; Shi, M. Automated Grading and Feedback Tools for Programming Education: A Systematic Review. *ACM Trans. Comput. Educ.* **2023**, *24*, 10. [CrossRef]

14. Shaka, M.; Carraro, D.; Brown, K.N. Personalised Programming Education with Knowledge Tracing. In Proceedings of the 2023 Conference on Human Centered Artificial Intelligence: Education and Practice (HCAIep '23), New York, NY, USA, 14–15 December 2023; p. 47. [CrossRef]
15. Le, N.; Strickroth, S.; Gross, S.; Pinkwart, N. A Review of AI-Supported Tutoring Approaches for Learning Programming. In *Advanced Computational Methods for Knowledge Engineering*; Springer: Cham, Switzerland, 2013; pp. 267–279. [CrossRef]
16. Combéfis, S. Automated Code Assessment for Education: Review, Classification and Perspectives on Techniques and Tools. *Software* **2022**, *1*, 3–30. [CrossRef]
17. Edelblut, P. Realizing the Promise of AI-Powered, Adaptive, Automated, Instant Feedback on Writing for Students in Grade 3–8 with an IEP. In *Adaptive Instructional Systems: Second International Conference, AIS 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, 19–24 July 2020, Proceedings*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 283–292. [CrossRef]
18. Svyatkovskiy, A.; Deng, S.K.; Fu, S.; Sundaresan, N. Intellicode compose: Code generation using transformer. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, New York, NY, USA, 8–13 November 2020; pp. 1433–1443.
19. Chakraborty, S.; Ahmed, T.; Ding, Y.; Devanbu, P.T.; Ray, B. Natgen: Generative pre-training by “naturalizing” source code. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, New York, NY, USA, 14–16 November 2022; pp. 18–30.
20. Kazemitabaar, M.; Chow, J.; Ma, C.K.T.; Ericson, B.; Weintrop, D.; Grossman, T. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. *arXiv* **2023**, arXiv:abs/2302.07427. [CrossRef]
21. Jin, J.; Kim, M. GPT-Empowered Personalized eLearning System for Programming Languages. *Appl. Sci.* **2023**, *13*, 12773. [CrossRef]
22. Becker, B.A.; Denny, P.; Finnie-Ansley, J.; Luxton-Reilly, A.; Prather, J.; Santos, E.A. Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1, New York, NY, USA, 15–18 March 2023; pp. 500–506.
23. Kurdi, G.; Leo, J.; Parsia, B.; Sattler, U.; Al-Emari, S. A Systematic Review of Automatic Question Generation for Educational Purposes. *Int. J. Artif. Intell. Educ.* **2019**, *30*, 121–204. [CrossRef]
24. Zavala, L.; Mendoza, B. On the Use of Semantic-Based AIG to Automatically Generate Programming Exercises. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18), New York, NY, USA, 21–24 February 2018; pp. 14–19. [CrossRef]
25. Bauer, Y.; Leal, J.P.; Queirós, R. Can a Content Management System Provide a Good User Experience to Teachers? In *Proceeding of the 4th International Computer Programming Education Conference (ICPEC 2023), Vila do Conde, Portugal, 26–28 June 2023*; Peixoto de Queirós, R.A., Teixeira Pinto, M.P., Eds.; Open Access Series in Informatics (OASIs); Schloss-Dagstuhl-Leibniz Zentrum für Informatik: Dagstuhl, Germany, 2023; Volume 112, pp. 4:1–4:8. [CrossRef]
26. Ta, N.B.D.; Nguyen, H.G.P.; Swapna, G. ExGen: Ready-to-use exercise generation in introductory programming courses. In Proceedings of the 31st International Conference on Computers in Education Conference, Matsue, Japan, 4–8 December 2023; pp. 1–10.
27. Strijbol, N.; Van Petegem, C.; Maertens, R.; Sels, B.; Scholliers, C.; Dawyndt, P.; Mesuere, B. TESTed—An educational testing framework with language-agnostic test suites for programming exercises. *SoftwareX* **2023**, *22*, 101404. [CrossRef]
28. Damasevicius, R.; Sidekerskiene, T. AI as a Teacher: A New Educational Dynamic for Modern Classrooms for Personalized Learning Support. In *Advances in Educational Technologies and Instructional Design*; IGI Global: Hershey, PA, USA, 2024; pp. 1–24. [CrossRef]
29. Sarsa, S.; Denny, P.; Hellas, A.; Leinonen, J. Automatic generation of programming exercises and code explanations using large language models. In Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1 (ICER 2022), Lugano, Switzerland, 7–11 August 2022; pp. 27–43.
30. Kasneci, E.; Seßler, K.; Küchemann, S.; Bannert, M.; Dementieva, D.; Fischer, F.; Gasser, U.; Groh, G.; Günemann, S.; Hüllermeier, E.; et al. ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education. *Learn. Individ. Differ.* **2023**, *103*, 102274. [CrossRef]
31. Swacha, J.; Gracel, M. Machine Learning in Gamification and Gamification in Machine Learning: A Systematic Literature Mapping. *Appl. Sci.* **2023**, *13*, 11427. [CrossRef]
32. Vos, T.E.; Fraser, G.; Martinez-Ortiz, I.; Prada, R.; Silva, A.R.; Prasetya, I. Tutorial on a Gamification Toolset for Improving Engagement of Students in Software Engineering Courses. In Proceedings of the 32nd Conference on Software Engineering Education and Training (CSEE&T 2020), Munich, Germany, 9–12 November 2020; pp. 1–3.
33. Pesek, M.; Vučko, Z.; Savli, P.; Kavčič, A.; Marolt, M. Troubadour: A Gamified e-Learning Platform for Ear Training. *IEEE Access* **2020**, *8*, 97090–97102. [CrossRef]
34. Khoshkangini, R.; Valetto, G.; Marconi, A.; Pistore, M. Automatic generation and recommendation of personalized challenges for gamification. *User Model. User-Adapt. Interact.* **2021**, *31*, 1–34. [CrossRef]
35. IMS Global Learning Consortium. Learning Tools Interoperability Core Specification. 2019. Available online: <http://www.imsglobal.org/spec/lti/v1p3/> (accessed on 10 September 2024).

36. Paiva, J.C.; Haraszczuk, A.; Queirós, R.; Leal, J.P.; Swacha, J.; Kosta, S. FGPE Gamification Service: A GraphQL Service to Gamify Online Education. In *Trends and Applications in Information Systems and Technologies*; Springer: Cham, Switzerland, 2021; Volume 4, pp. 480–489. [CrossRef]
37. Leal, J.P.; Silva, F. Mooshak: A Web-based multi-site programming contest system. *Softw. Pract. Exp.* **2003**, *33*, 567–581. [CrossRef]
38. Paiva, J.C.; Queirós, R.; Leal, J.P.; Swacha, J. Yet another programming exercises interoperability language. In Proceedings of the 9th Symposium on Languages, Applications and Technologies (SLATE 2020), Dagstuhl, Germany, 13–14 July 2020.
39. Done, K. *Encyclopedia Eldoria*; Comstar Media: Dover, UK, 2005.
40. Reddit. What Is AI's Obsession with "Eldoria?". 2024. Available online: https://www.reddit.com/r/ChatGPT/comments/1bb2mzm/what_is_ais_obsession_with_eldoria/ (accessed on 10 September 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.