Exercisify: An AI-Powered Statement Evaluator

Ricardo Queirós
School of Media Arts and Design & CRACS – INESC TEC,
Polytechnic University of Porto, Portugal

Abstract

A growing concern with current teaching approaches underscores the need for innovative paradigms and tools in computer programming education, aiming to address disparate user profiles, enhance engagement, and cultivate deeper understanding among learners This article proposes an innovative approach to teaching programming, where students are challenged to write statements for solutions automatically generated. With this approach, rather than simply solving exercises, students are encouraged to develop code analysis and problem formulation skills. For this purpose, a Web application was developed to materialize these ideas, using the OpenAI API to generate exercises and evaluate statements written by the students. The transformation of this application in H5P and its integration in a LMS gamified workflow is explored for wider and more effective adoption.

2012 ACM Subject Classification Social and professional topics → Computer science education

Keywords and phrases Code generation, Computer Programming, Gamification

Digital Object Identifier 10.4230/OASIcs.ICPEC.2024.19

Funding This work is co-funded by the Erasmus+ Programme of the European Union within the project FGPEPlusPlus, with Agreement Number 2023-1-PL01-KA220-HED-000164696.

1 Introduction

Learning to code can be tough. Understanding programming languages, algorithms, and problem-solving can be overwhelming, specially for novice students [3].

This work explores a different way of teaching programming. Instead of students solving problems, they're given a completed JavaScript solution and asked to write a clear problem statement based on it. This way, rather than focusing solely on implementing solutions, students are challenged to think at a higher level by articulating problem statements. This requires them to analyze, synthesize, and communicate complex ideas, fostering critical thinking and problem-solving skills which are crucial in programming.

By working backward from a provided solution to create a problem statement, students gain a deeper understanding of programming concepts and logic embedded within the solution code. Students are challenged to think critically and analytically as they articulate the problem requirements based on the provided solution, fostering problem-solving skills essential in programming.

For this purpose, Exercisify was created. The workflow is straightforward: 1) a complete source code of a programming exercise is generated and presented to the student; 2) students write a clear problem statement and test case based on the solution given and 3) the problem statement and test cases are automatically evaluated and a scored is delivered. Both the exercise generation and the statement evaluation is supported by the OpenAI API.

The evaluation includes specific criteria to assess various aspects from relatedness of the statement to the solution code, clarity and coherence of the statement, explanation of input parameters return values, test cases correctness and English writing.

The rest of the article is structured in three sections: the second section presents related work on computer learning teaching environments. The following section presents Exercisify and all its components. Finally, the contributions of this article to the scientific community are presented as well as the future work.

2 Literature review

Currently, AI tools play a crucial role in improving the teaching and learning process of computer programming by offering personalized and adaptive learning experiences. In programming education, AI assists educators and learners in various aspects, including providing automated feedback and code review, offering adaptive learning paths, facilitating programming tutoring systems, automating assessment and grading tasks, and aiding in code summarization and documentation.

Another way AI can be used is by generating programming exercises. There is a notable interest in such generators, as creating diversified and challenging programming exercises can be time-consuming for educators. These generated exercises can be tailored to cater to different skill levels, ensuring a dynamic learning environment that adapts to individual learner needs. Several tools have been developed to address the challenges of programming exercise generation.

Kurdi [2] conducted a systematic review of automatic exercise generation in various domains, highlighting the need for tools that offer exercises of controlled difficulty and provide features like enriching question forms and structures, automating template construction, improving presentation, and generating feedback.

Zavala [6] presented a tool that uses Automatic Item Generation (AIG) to create consistent programming exercises using pre-defined templates, ensuring uniformity in testing.

ExGen [5] generates ready-to-use exercises tailored to specific difficulty levels and concepts, leveraging advances in large language models (LLMs) to autogenerate novel exercises and filter them to suit students.

Agni [1] is as a code playground tailored for learning JavaScript which includes a 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, or Canvas.

TESTed [4] is an educational testing framework that supports the creation of programming exercises with automated assessment capabilities in a programming-language-independent manner.

Recent work has leveraged pre-trained LLMs for automatic exercise generation using novel AI technologies. However, challenges such as model biases and system brittleness need to be addressed when applying LLMs to education.

While several tools exist for programming exercise generation, there is a gap in tools capable of offering different approaches as the one presented in this article.

3 Implementation

This work introduces a Web application called Exercisify, designed to enhance programming knowledge through a unique approach. In Exercisify, students are tasked with creating exercise statements based on generated solutions. This approach offers a novel methodology that not only challenges their understanding of programming concepts but also fosters critical thinking, problem-solving, and communication skills crucial for real-world software development.

The Exercisify graphical user interface (GUI) is structured into three main areas: the statement and generated solution display on the left, the test case creation area at the top right, and the result display at the bottom right. The left area showcases the generated solution and allows students to compose the exercise statement and initiate evaluation. The

top right section enables students to define test cases comprising input and output data. Finally, the bottom right area presents the evaluation results after the evaluation button is pressed.

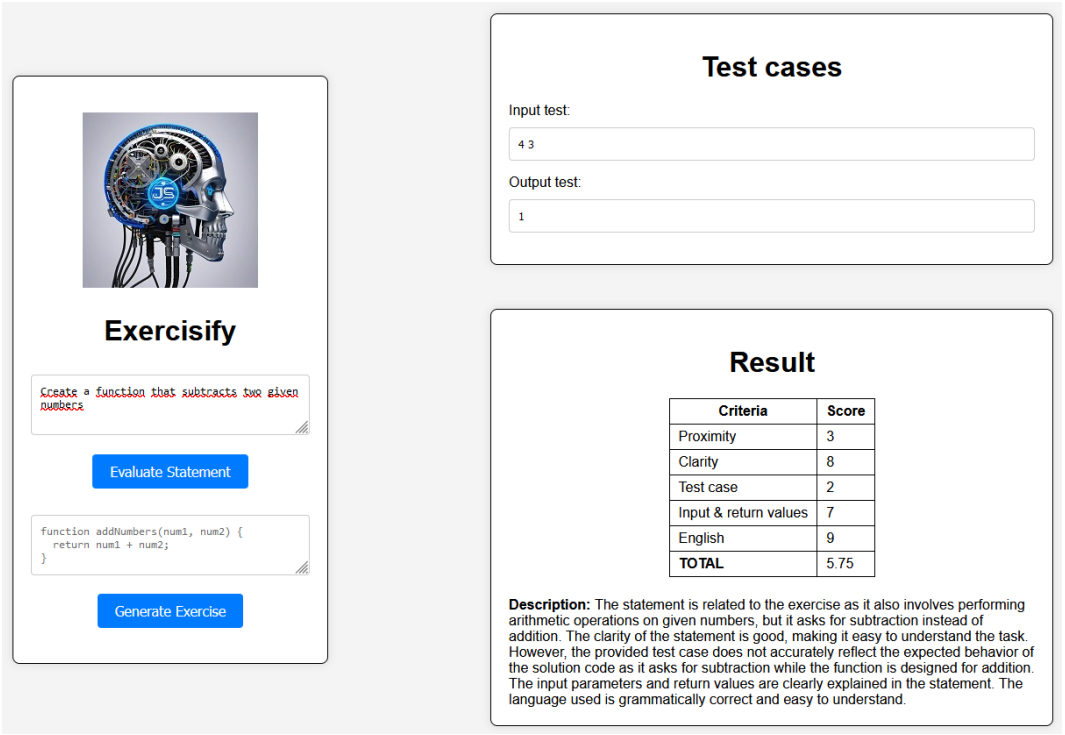

Figure 1 Exercisify GUI.

The Exercisify architecture comprises two primary components: the Exercise Generator and the Statement Evaluator.

3.1 Exercise Generator

The Exercise Generator component is responsible for generating programming solutions using the OpenAI API . The code below starts by defining an asynchronous JavaScript function which is responsible for generating exercise statements using the ChatGPT API from OpenAI. The prompt variable contains the instruction provided to the ChatGPT model. In this case, it instructs the model to generate a very simple function in JavaScript. The apiKey variable holds the authentication key required to access the OpenAI API. It's essential for authenticating requests to the API. The apiUrl variable specifies the endpoint of the OpenAI API that will be used to send requests for text generation. The requestData object contains the data to be sent in the request body to the API. It includes the model to use for text generation (gpt-3.5-turbo), the user's prompt message, and parameters like max_tokens (maximum number of words to generate), temperature (controls randomness), and stop (criteria to stop text generation).Then the fetch function is used to send a POST request to the specified API endpoint with the request headers, such as content type and authorization. The request body contains the requestData object, which is converted to JSON format using JSON.stringify().

Listing 1 Programming solution generation.

```
async function generateExercise() {
  const prompt = "Generate a JavaScript function...";
  const apiKey = "API-KEY";
  const apiUrl = "https://api.openai.com/v1/chat/completions";

  // Data to be sent in the request body
  const requestData = {
    model: "gpt-3.5-turbo",
    messages: [{"role": "user", "content": prompt}],
    max_tokens: 100,
    temperature: 0.7,
    stop: ["\\n"],
  };

  try {
    const response = await fetch(apiUrl, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        "Authorization": `Bearer ${apiKey}`
      },
      body: JSON.stringify(requestData)
    });
    ...
  } catch (error) {
    ...
  }
}
```

Once the solution is generated, it is presented to the student, who is tasked with creating a statement for the exercise based on the solution provided. This manual process encourages students to think critically about the problem and articulate their understanding in writing.

3.2 Statement Evaluator

The Statement Evaluator component assesses the quality and relevance of the exercise statement created by the student in relation to the provided solution. It leverages the ChatGPT API from OpenAI once again to evaluate the relatedness of the statement and the solution based on specific criteria. The key elements of the Statement Evaluator include

1. Evaluation Prompt Creation: The web app generates a prompt for the ChatGPT API, incorporating both the exercise statement created by the student and the provided solution. This prompt guides the API in evaluating the alignment of the statement with the solution based on predefined criteria.
2. ChatGPT API Integration: The Statement Evaluator interacts with the ChatGPT API to assess the exercise statement. The API processes the prompt and offers feedback on the statement's coherence with the solution.

The evaluation criteria used in the Statement Evaluator component is depicted in the following table.

The first criterion prioritizes the alignment between the exercise statement and the provided solution code, ensuring it effectively describes the programming task, algorithms used (if any), and guides students towards the intended solution.

Criteria	Description	Weight
Proximity to solution	How closely does the exercise statement align with the provided solution code?	35%
Clarity of Explanation	How clear and understandable are the instructions provided in the exercise statement?	25%
Test Case Correctness	How accurately do the provided test cases reflect the expected behavior of the solution code?	25%
Input and return values	How effectively does the exercise statement clarify input parameters and expected return values?	10%
English Proficiency	How grammatically correct and fluent is the English language used in the exercise statement?	5%

The clarity of explanation criterion underscores the importance of clear communication in the exercise statement, ensuring students can readily comprehend the task and its requirements.

Including test cases is crucial for validating the correctness of the solution code. By assigning significant weight to this criterion, the evaluation process ensures that the provided test cases accurately reflect the expected behavior of the solution.

Additionally, clarifying input parameters and expected return values helps students grasp the function's purpose and behavior. While not as heavily weighted as other criteria, this aspect still contributes to the overall effectiveness of the exercise statement.

Finally, ensuring grammatical correctness and fluency in the exercise statement is vital for clear communication. While English proficiency is essential, it's appropriately given a lower weight compared to other criteria, as long as the statement remains understandable.

3.3 H5P transformation

Integrating Exercisify into an LMS workflow can significantly enhance the teaching and learning experience in programming education. Two common methods for seamless integration are through Learning Tools Interoperability (LTI) and H5P (HTML5 Package).

Learning Tools Interoperability (LTI) is a standard protocol that allows learning systems, such as LMS platforms, to integrate with external tools and content.

H5P (HTML5 Package) is a content creation tool that allows educators to develop and share interactive content across several platforms.

LTI integration facilitates standardized access within LMS platforms, enabling single sign-on, centralized management, and data exchange for features like grade synchronization. H5P integration, on the other hand, offers customization flexibility, content embedding, reusability, and support for interactive elements. The choice depends on factors such as desired integration level, instructional goals, and LMS platform capabilities.

By integrating Exercisify with the LMS, student performance scores from exercise evaluations can be automatically transported to the LMS gradebook. This eliminates the need for manual score entry by instructors, saving time and ensuring accuracy in grading. At the same time, students receive real-time feedback on their exercise submissions, including scores and performance metrics. This immediate feedback loop promotes continuous learning and allows students to track their progress throughout the course.

One of the benefits of the LMS integration is the capability of including Exercisify in the LMS gamification workflow. Here are some examples:

- Unlocking Content: Gamification elements such as unlocking content based on performance scores can be implemented within the LMS. Students can progress through course materials and access additional resources or levels as they achieve certain performance milestones in Exercisify exercises.

- Earning Badges and Achievements: Students can earn badges or achievements within the LMS for achieving specific scores or completing exercises in Exercisify. These gamified incentives incentivize engagement, encourage mastery of programming concepts, and add an element of fun to the learning process.
- Competition and Leaderboards: the LMS can enable the creation of leaderboards based on Exercisify performance scores. Students can compete with peers, track their rankings, and strive to improve their standings, fostering a sense of healthy competition.

4 Conclusion and Future Work

In conclusion, the development of the Exercisify web app and the integration of AI-based exercise generation and statement evaluation can empower students to actively engage in learning, fostering critical thinking and problem-solving skills.

In regard of future directions, the most obvious is to validate the Exercisify tool with real users. This validation process could involve conducting user testing sessions with students and educators to gather feedback on usability, effectiveness, and overall user experience. Additionally, collecting data on user performance and learning outcomes through the tool can provide insights into its impact on programming education.

Additional future directions:

- Enhanced AI Integration: continuously update and refine the AI models used for exercise generation and evaluation to improve accuracy.
- Expanded Exercise Types: diversify the range of supported exercise types to cover several programming languages and problem-solving scenarios.
- H5P support: finalize the transformation of the Web app to the H5P package.

References

1 Yannik Bauer, José Paulo Leal, and Ricardo Queirós. Can a Content Management System Provide a Good User Experience to Teachers? In *4th International Computer Programming Education Conference (ICPEC 2023)*, volume 112 of *Open Access Series in Informatics (OASIcs)*, pages 4:1–4:8, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

2 Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30, 2019.

3 José Carlos Paiva, Ricardo Queirós, José Paulo Leal, Jakub Swacha, and Filip Miernik. Managing gamified programming courses with the FGPE platform. *Information*, 13(2):45, 2022.

4 Niko Strijbol, Charlotte Van Petegem, Rien Maertens, Boris Sels, Christophe Scholliers, Peter Dawyndt, and Bart Mesuere. Tested – an educational testing framework with language-agnostic test suites for programming exercises. *SoftwareX*, 22:101404, 2023.

5 Nguyen Binh Duong Ta, Hua Gia Phuc Nguyen, and Gottipati Swapna. Exgen: Ready-to-use exercise generation in introductory programming courses. In *Proceedings of the 31st International Conference on Computers in Education Conference*, pages 1–10, Matsue, Shimane, Japan, december 4-8 2023.

6 Laura Zavala and Benito Mendoza. 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, pages 14–19, New York, NY, USA, 2018. Association for Computing Machinery.