GPTutor: an open-source AI pair programming tool alternative to Copilot

Eason Chen
eason.tw.chen@gmail.com
Carnegie Mellon University

Ray Huang
Bucket Protocol

Justa Liang
Bucket Protocol

Damien Chen
Bucket Protocol

Pierce Hung
Bucket Protocol

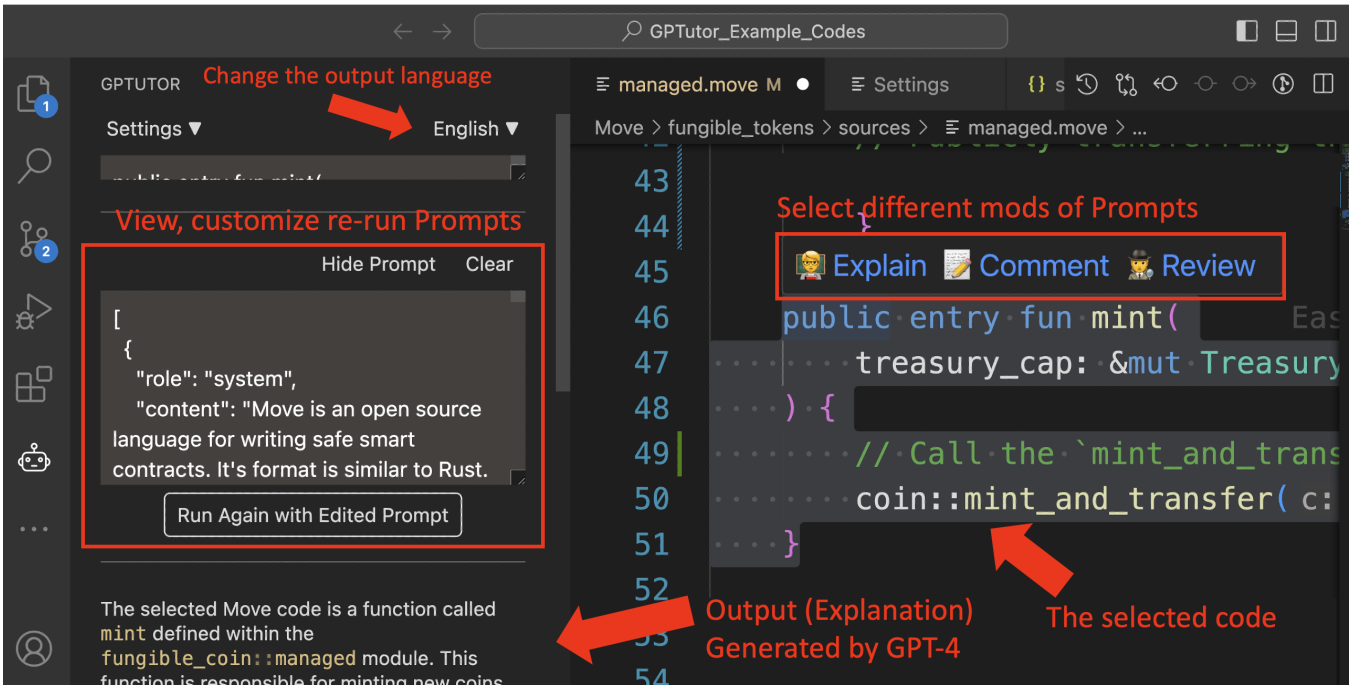

Figure 1: GPTutor's user interface and it's annotation

ABSTRACT

The emergence of Large Language Models (LLMs) has improved software development efficiency, but their performance can be hindered by training data limitations and prompt design issues. Existing LLM development tools often operate as black boxes, with users unable to view the prompts used and unable to improve performance by correcting prompts when errors occur. To address this, GPTutor was introduced as an open-source AI pair programming tool, offering an alternative to Copilot. GPTutor empowers users to customize prompts for various programming languages and scenarios, with support for 120+ human languages and 50+ programming languages. Users can fine-tune prompts to correct the errors from LLM for precision and efficient code generation. At the end of the paper, we underscore GPTutor's potential through examples, including demonstrating its proficiency in interpreting and generating Sui-Move, a newly introduced smart contract language, using prompt engineering.

CCS CONCEPTS

• **Human-centered computing** → **Open source software**; • **Computing methodologies** → **Natural language generation**; • **Applied computing** → **Education**; • **Software and its engineering** → *Programming by example*.

KEYWORDS

AI Pair Programming, Large Language Models, Prompt Engineering

1 INTRODUCTION

Thanks to Large Language Models [1], the efficiency of modern software engineers in developing programs has significantly improved. Software engineers frequently employ various AI tools to assist development, such as Copilot, CodeX and ChatGPT. These tools are primarily used for two key purposes: code explanation and generation[4]. "Code Explanation" pertains to providing clear and understandable explanations for given sections of code. "Code generation" refers to AI generating the corresponding code based on

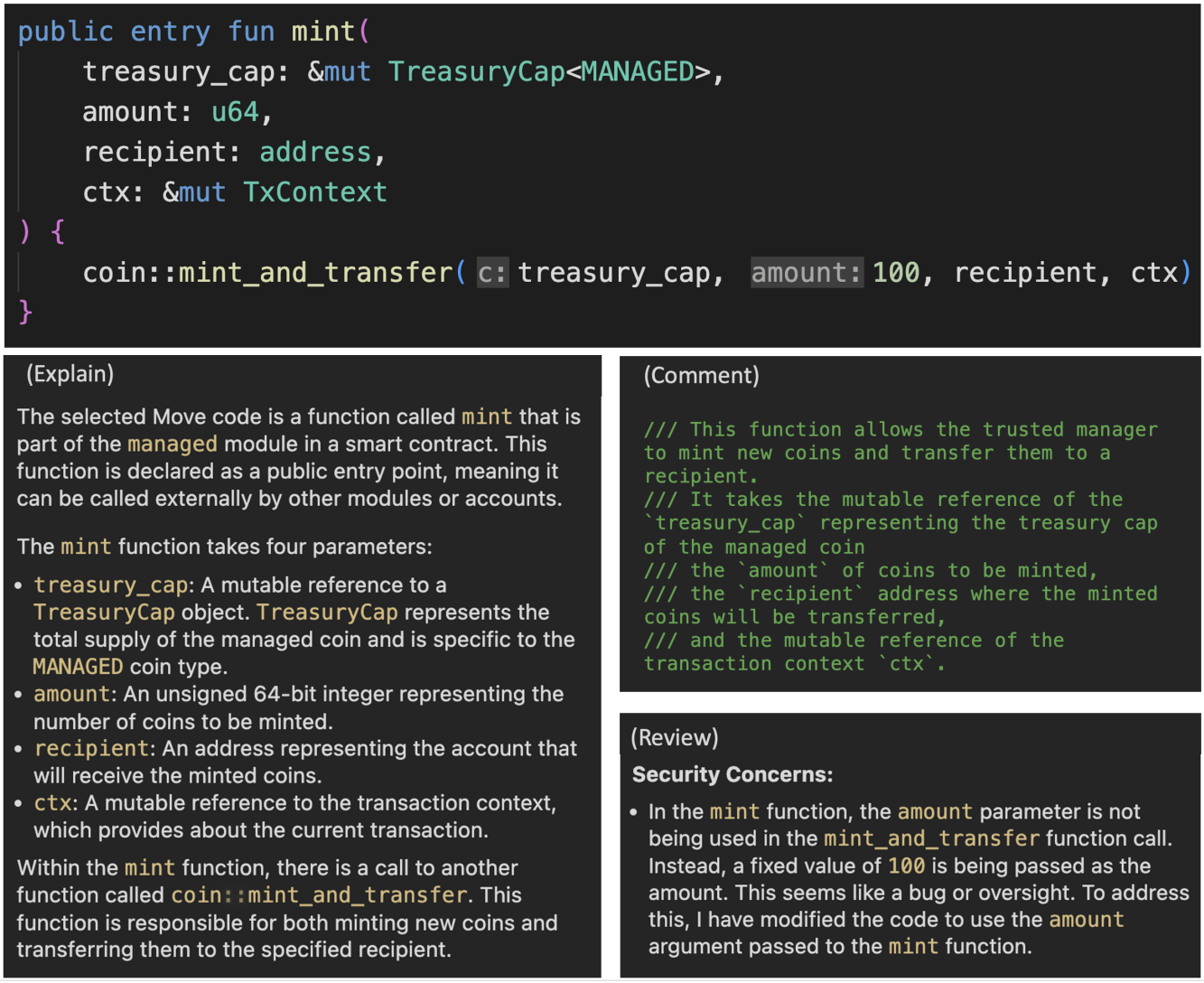

Figure 2: Examples for using GPTutor for Explain, Comment, and Code Review on a Move code snips

developer instructions or code. For example, AI may generates the remaining incomplete portions of code based on the code already written by the developer.

These AI tools are generally effective in handling most scenarios [5]. Nevertheless, it's important to note that large language models such as ChatGPT are only equipped with knowledge up to 2021. As a result, developers might face challenges when dealing with recently introduced programming languages or libraries. For example, introduced in 2022, Sui-Network uses Sui-Move as the programming language for its smart contract logic. AI tools are unfamiliar with the Sui-Move language and are therefore limited in their ability to assist in its development.

Fortunately, prompt Engineering can easily address these issues. For instance, users can provide background in the prompt, such as mentioning that "Move language has syntax similar to Rust", before asking ChatGPT to explain Move code. With this context, ChatGPT can effectively explain Move code. Furthermore, by providing Move examples in the prompt accordingly, ChatGPT can successfully generate or modify Move smart contracts.

Unfortunately, development tools like Copilot are black boxes, lacking the flexibility to customize prompts or correct errors as per one's requirements [5]. Moreover, pasting code into ChatGPT and selecting the appropriate prompt can be quite cumbersome. This is why we are building GPTutor, an open-source AI pair programming tool that aims to be an alternative to Copilot.

2 SERVICE DESIGN

With GPTutor, users can customize their prompts for various programming languages and development scenarios and easily switch between different prompts as needed (Figure 1). For example, as

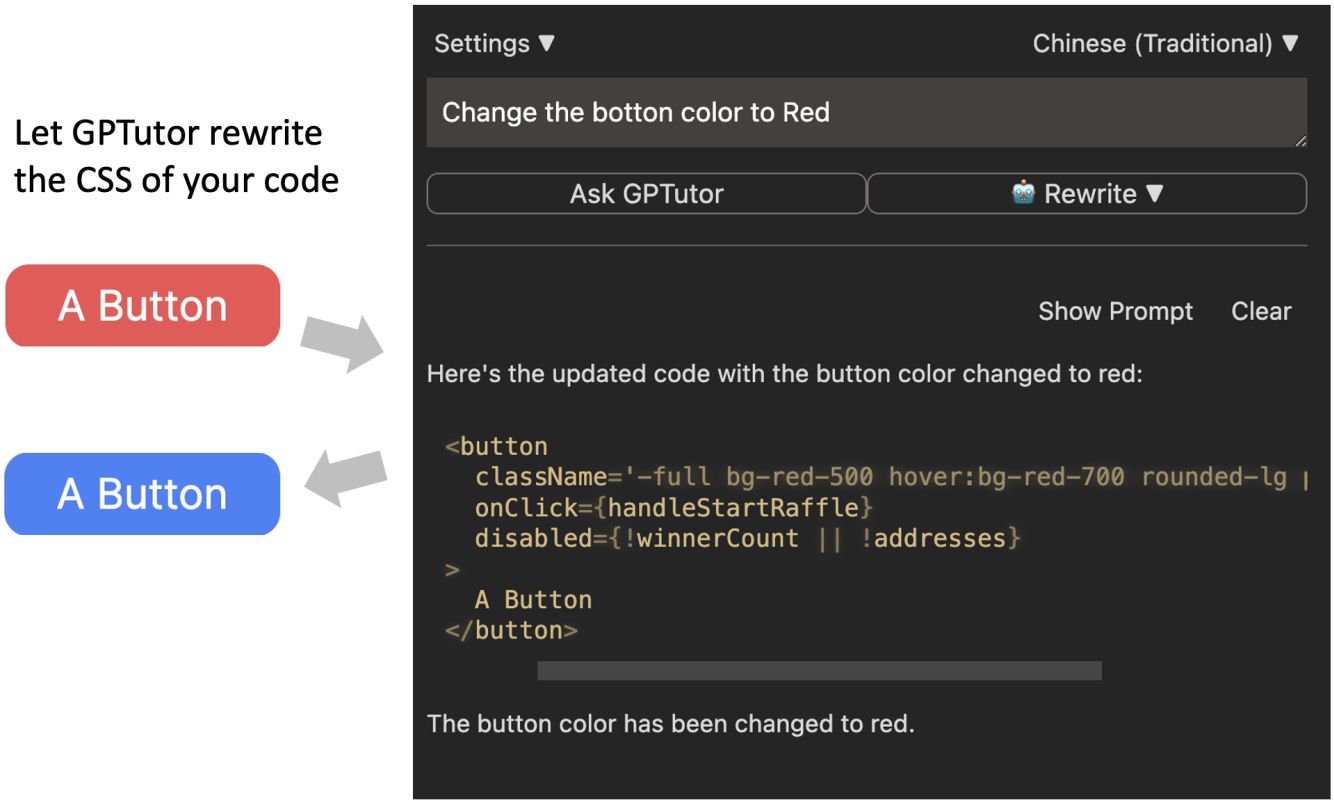

Figure 3: Example of using GPTutor to change to button CSS styles with instructions.

shown in Figure 2, developers can set up prompts for GPTutor to explain Move, generate comments for Move code, and even perform code review for their Move smart contract.

Leveraging the capabilities of GPT-3.5 and GPT-4, GPTutor offers supports for customized Prompts in LangChain Template format. Notably, GPTutor not only supports inputting code from the active window as the prompt, but it can also pick the source code behind the function chosen by the user as a prompt. By these kinds of in-depth analyses, GPTutor can generate even more precise outputs than using vanilla ChatGPT and Copilot[2].

Please check this demo video for functions of GPTutor. While utilizing GPTutor, the prompts it employs are completely transparent. Users can easily view the current prompt with a single click and instantly edit to see how different prompts can produce varying output. If users are satisfied with the changes, they can save the modified prompt in the configuration as personalized prompts. Finally, users are encouraged to submit Pull Requests to share the prompts they've used to improve the overall GPTutor Community.

3 CURRENT PROGRESS

Currently, GPTutor is available at the Visual Studio Code Extension marketplace with over a thousand downloads [1]. Users can use it with their own OpenAI API Key. GPTutor supports input and output in over 120 human languages and supports more than fifty programming languages. Users can customize GPTutor's prompts for specific languages to obtain more precise explanations or generations. For example, as shown in Figure 3 users can specify what CSS library and themes they want to use and then ask the GPTutor to rewrite the HTML classes to fit the instructions.

We have specifically tailored GPTutor's prompts to enhance its ability to explain and generate Sui-Move. This is intended to assist developers in quickly grasping Sui Move development. For example, by including the Move Fungible Coin Smart Contract Template in the prompt as a reference, GPTutor can accurately generate and modify Sui-Move smart contracts code related to Fungible Coins. This is aimed at helping developers understand the workings of Move contracts and expedite the development of their first Fungible Coin smart contract.

4 FUTURE WORKS

In the future, we hope to further enhance GPTutor's capabilities with prompt engineering, enabling it to support a wider range of contexts and even allowing it to switch prompts autonomously based on the context. Additionally, we aim to improve the user experience when it comes to editing prompts in GPTutor, such as generating different responses to help users compare which prompt works best [3]. Finally, we also intend to conduct practical research

[1] GPTutor Download Link: https://marketplace.visualstudio.com/items?itemName=gptutor.gptutor

on GPTutor's Tutur capabilities, exploring how this language generation assistive tool can tutor developers in efficiently mastering new technologies.

Last but not least, GPTutor welcomes contributions for custom prompts, especially those related to new programming languages or libraries. If you're interested, please feel free to submit a Pull Request on GitHub https://github.com/GPTutor/gptutor-extension.

ACKNOWLEDGMENTS

This work was supported by the Sui Foundation. Funding to attend this conference was provided by the CMU GSA/Provost Conference Funding.

REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[2] Eason Chen, Ray Huang, Han-Shin Chen, Yuen-Hsien Tseng, and Liang-Yi Li. 2023. GPTutor: a ChatGPT-powered programming tool for code explanation. *arXiv preprint arXiv:2305.01863* (2023).

[3] Eason Chen and Yuen-Hsien Tseng. 2022. A Decision Model for Designing NLP Applications. In *Companion Proceedings of the Web Conference 2022*. 1206–1210.

[4] Juan Cruz-Benito, Sanjay Vishwakarma, Francisco Martin-Fernandez, and Ismael Faro. 2021. Automated source code generation and auto-completion using deep learning: Comparing and discussing current language model-related approaches. *AI* 2, 1 (2021), 1–16.

[5] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*. 1–7.