2024 7th International Conference on Information and Computer Technologies (ICICT) | 979-8-3503-8562-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICICT62343.2024.00060

Improving Source Code with Assistance from AI — A Pilot Case Study with ChatGPT

Steven McDaniel
Department of Computer Science
Idaho State University
Pocatello, ID, USA
mcdastev@isu.edu

Minhaz F. Zibran
Department of Computer Science
Idaho State University
Pocatello, ID, USA
minhazzibran@isu.edu

***Abstract*—ChatGPT queries were used to provide feedback on five C++ programs selected from various programming assignments for two graduate-level computer science courses – a scientific programming course and an algorithms course. The evaluated software was written by the first author for those courses within the last two years. ChatGPT was asked to evaluate and provide feedback for each program. Specifically, ChatGPT was asked to evaluate the code for strengths and weaknesses and make recommendations for improving (1) execution speed as well as (2) readability and maintainability. A subjective agreement rating was generated by the authors for each strength, weakness, and recommended change provided by ChatGPT. While the overall agreement with the ChatGPT provided feedback was over 90 percent, at times, ChatGPT's recommendations were found misleading.**

***Index Terms*—ChatGPT, Code, Readability, Program, Analysis, Execution Speed, Maintainability**

I. Introduction

Software coding is a highly technical craft that requires logical thinking as well as familiarity with the languages and tools used to develop the code. Errors and warnings provided by compilers, debuggers, and linters can find syntax errors, formatting discrepancies, and non-adherence to coding standards and conventions. However, these tools typically only identify low-level errors that enable modest improvements in overall software quality and do little to instruct the user on good coding practices. Consequently, human feedback in the form of code reviews, paired programming, and the like is used to supplement programming tools. Unfortunately, the costs and delays associated with human feedback limit its usage.

With the recently growing popularity of large language models (LLM) and generative AI (Artificial Intelligence), tools such as ChatGPT [1] are being adopted for assistance in various tasks including computer programming. When a programming/technical difficulty is encountered, a programmer typically consults Google search or websites such as Stack Overflow and Stack Exchange to find a potential solution. But now, with the advent of AI tools such as ChatGPT, many practitioners ask the AI tools with questions explaining the problem. In response, they obtain suggestions or solutions. ChatGPT has also become popular among students dealing with programming problems.

However, little is known about how good the AI-generated solutions are and to what extent the suggestions from the AI tools are really helpful. Therefore, this case study of ours investigates using ChatGPT as a mechanism to provide feedback for improving software quality as a supplement to, or replacement of, human feedback.

The rest of the paper is organized as follows. In Section II, we first discuss the work in the literature relevant to this study. Section III describes the methodology of our study. The results are presented in Section IV. The findings are further discussed and contextualized in Section V. The limitations of this work are addressed in Section VI. Finally, Section VII concludes the paper.

II. Related Work

There are many studies [2]–[15] of code written by humans, but the studies of AI-generated code [16]–[19] are rare. Most analysis of the usefulness of AI for software development is found in articles and blogs posted on the internet rather than academic literature. Exceptions include the work of Piccolo et al. [20] which asserts that many bioinformatics programming tasks can be automated with ChatGPT and Jesse et al. [21] which found that the prevalence of single statement bugs generated by Codex (used in Co-pilot) is up to twice their prevalence in the GitHub repository that was used to train Codex.

The consensus among online reviewers [22]–[28], is that ChatGPT will not replace software engineers but can increase their productivity. Gewirtz [22] asserts that ChatGPT excels in assisting with specific coding tasks rather than building complete applications. Israelsen [23] asserts that ChatGPT can be faster than searching stack overflow for useful code snippets. Savona and Ackerson [25] assert that ChatGPT can simplify and speed up the code development process.

Several online reviews [26]–[28] compare ChatGPT with Copilot for assisting with software development. A review from PeopleTechGroup [26] asserts that ChatGPT excels at brainstorming and providing in-depth coding insights but that code generated by ChatGPT is less reliable than code generated by Co-pilot. Lawton [27] asserts that ChatGPT is better

2769-4542/24/$31.00 ©2024 IEEE
DOI 10.1109/ICICT62343.2024.00060

TABLE I
DESCRIPTION OF THE FIVE CODE SAMPLES USED IN THIS CASE STUDY

Program ID	Course	Assignment/Problem Description	LOC*	Complexity
C1	Scientific Computing (CS 6XXX)	Comparison of accuracy of Simpson and Trapezoidal integration techniques[1]	139	Low
C2		LCG random number generation	69	Medium
C3	Advanced Algorithms (CS 5XXX)	Comparison of Prim's, Kruskal's, and Random Algorithms for finding MSTs	428	High
C4		Comparison of sorting methods[2] quicksort, frequency counting and vector sort	403	High
C5		Priority Queue implementation[3]	226	High

*LOC = Line of Code. [1]Warm up assignment. [2]Developed for previous assignment (C3).
[3]some code was instructor-provided which needed modifications for purposes of the assignment.
Note: Course numbers are partially concealed for the sake of annonymity.

suited than Copilot for summarizing complex code or generating a starting template for a specific coding task and that both tools can make developers more productive by automating the writing of mundane, boilerplate code. Finally, Hutsulyak [28] asserts that ChatGPT is useful for the generation of code snippets, code refactoring, and optimization, fixing bugs, API integration, rapid prototyping, framework, and library usage, cross-language code migration and integration as a chatbot.

III. METHODOLOGY

The purpose of this study is to qualitatively evaluate feedback provided by ChatGPT for code written by the author. It is believed that personal familiarity with the evaluated code enables a good assessment of the quality of the feedback. For obtaining programming aid/suggestions, we used ChatGPT version 3.5. We have chosen ChatGPT because of its recent popularity and capability in assisting diverse tasks including automatic generation of source code.

A. Code Samples

For the study, we used five code samples, which were used/authored in two computer science graduate-level classes taught at the Idaho State University. All these code samples were written in C++, they are of different sizes dealing with problems at varying levels of difficulty. These pieces of code were written primarily for completing assignments for the classes. An overview of each of the programs is shown in Table I.

The first author of this paper is the author of the five code samples. He is a highly experienced former software developer with about five years of experience coding in C, eight years of experience coding in C++, and 10 years of experience programming in other programming languages such as C#, javascript, python, Occam, and pascal. Most of the authors' experience with C++ is dated and proceeds many of the improvements made to C++ and associated libraries since 2002.

The focus in writing the code for the Computer Science classes was on providing accurate results with a minimal amount of programming effort. Some level of readability was sought but there was little concern for execution speed, code reusability, and maintainability.

B. Operating with ChatGPT

For each of the five samples, the following three tasks were provided to ChatGPT.

1) Evaluate the *strengths and weaknesses* of the given C++ code.
2) Please recommend changes to the following C++ program to improve *execution speed*.
3) Please recommend changes to the following C++ program to improve *readability and maintainability*.

The responses provided by ChatGPT were essentially a list of strengths, weaknesses, and recommended changes, which in some cases included actual C++ code in addition to a natural language description of the strengths, weaknesses, or recommended changes.

C. Reviewing ChatGPT Responses

The strengths, weaknesses, and recommended changes provided by ChatGPT were compiled into tables with a (shortened) description of the strengths, weaknesses, or recommended changes in the first column augmented with a column for each of the code examples. Highly similar strengths, weaknesses, and recommended changes were merged to reduce the number of rows in the tables.

ChatGPT's recommendations were also reviewed by the authors to determine whether those recommendations made sense. Thus, a (subjective) agreement rating in the likert scale was generated by the authors and included for each coding example that the strength, weakness, or recommended change applied to. The agreement ratings ranged from -2 to +2 using the Likert scale presented in Table II.

TABLE II
THE LIKERT SCALE USED IN THIS CASE STUDY

-2	-1	0	+1	+2
Strongly disagree	Disagree	Neutral	Agree	Srongly agree

IV. RESULTS

Tables III through VI summarize the feedback provided by ChatGPT. Table III and Table IVrespectively present the strengths and weaknesses ChatGPT identified in the five sample programs fed to it and our agreement ratings for each strength or weakness. Table V presents ChatGPT's recommendations for enhancing the execution speed of the programs

TABLE III
AGREEMENT RATINGS IN LIKERT SCALE (-2 TO +2) FOR CHATGPT IDENTIFIED STRENGTHS

Description of Strengths Identified by ChatGPT	Code Samples					
	C1	C2	C3	C4	C5	All
Well-documented with comments	+1	+2	+1	+1		
Provides multiple methods for same function	+1			+1		
Uses the C++ standard library function or template	+2		+2	+1		
Handles invalid input parameters	+1					
Provides a practical demonstration of ...	+1	+1		+1		
Uses a global array which simplifies implementation		+2				
Encapsulation/Modularized into functions			+2		+2	
Dynamic Memory Allocation			+1			
Diversity in Algorithmic Approaches			+2			
Testing function(s)			+2			
Includes timing measurements to assess performance			+2	+2		
Includes thorough explanations of Big O analysis					+2	
Big O analysis considers various scenarios					+2	
Relatively simple and easy to follow					+2	
Utilizes array-based binary heap known for efficient insertion/extraction					+2	
Feedback Count	5	3	7	5	5	25
Agreement Count	5	3	7	5	5	25
Agreement Percentage	100	100	100	100	100	100

TABLE IV
AGREEMENT RATINGS IN LIKERT SCALE (-2 TO +2) FOR CHATGPT IDENTIFIED WEAKNESSES

Description of Weaknesses Identified by ChatGPT	Code Samples					
	C1	C2	C3	C4	C5	All
Use of `#define` debug false for conditional debugging output	+1					
Does not check input parameter validity	+2			+2		
Error handling is minimal/deficient	+2	+2	+2	+2	+2	
Lacks unit tests	+2	+2		+2		
Unnecessary type casting (e.g., using `0.0f` instead of `0.0`)	0					
Magic numbers like `0x01`, `0.5f`, `24` and `100`	+2	+2		+2		
Lacks clear separation between input, computation, and output	+1					
Use of global variables		+1	+1	+1		
Uses a simple modulo operation (%) to limit the range of generated random numbers which can introduce bias		0				
Multiple (non-modular) code segments in a single loop		+1				
Does not check for potential division by zero		+2				
Uses raw arrays and pointers for memory management instead of smart pointers or container classes			+2			
Hardcoded input file			+2			
Hardcoded iterations			+2			
Unexplained algorithm-specific details			+2			
Code duplication			+2			
Inconsistent formatting			+2			
Lack of function prototypes			+1			
Verbose output (prints a lot of information to the console)			+2	+2		
Code does not consider worst-case scenario				+1		
Code includes unused variables				+2		
Inconsistency in memory management				+2		
Commented-out sections of code				+1		
Unnecessary print statements				+1		
Code's performance measurement is somewhat arbitrary				-1		
Execution time for sorting large lists is excessive				-2		
Uses a fixed-size array for the priority queue					+1	
Doesn't employ dynamic memory allocation					+1	
Doesn't employ modern data structures like `std::vector`					+2	
Reset function not particularly useful without queue resizing					0	
The `printQ` function does not provide a comprehensive view of the outputted structure					+1	
Lack of template-based implementation					+2	
Unclear comments					+1	
Utilizes a linear search to find the minimum child					+1	
Feedback Count	7	7	10	13	9	46
Agreement Count	6	6	10	11	8	41
Agreement Percentage	86	86	100	85	89	89

TABLE V
AGREEMENT RATINGS IN LIKERT SCALE (-2 TO +2) FOR CHATGPT RECOMMENDATIONS FOR IMPROVING EXECUTION SPEED

Description of Recommendations Made by ChatGPT for Improving Execution Speed	Code Samples					
	C1	C2	C3	C4	C5	All
Mark small frequently used functions with 'inline'	+2					
Replace function objects with function pointers	-2					
Use `\n` instead of `std::endl` for newlines	+2	+2	+2			
Wrap debug statements in preprocessor directives	+2					
Move the calculation of `dX` outside the loop	-2					
Replace bitwise operations with more readable alternatives instead of (i & 0x01), you can use (i % 2 == 1)	-2					
Avoid using global variables		+2			+1	
Use `constexpr` for constants		+2				
Use pre-increment instead of post-increment		+2				
Use static_cast for type casting		+2				
Minimize I/O operations, especially printing to the console		+2	+2			
Explore more efficient random number generation algorithms		0				
Use profiling tools to analyze performance		+2	+2	+2	+2	
Use reserve for allocating vectors			+2	+2	+2	
Replace raw arrays with smart pointers			+2	+2		
Avoid dynamic memory allocation inside loops			+2			
Use emplace_back instead of push_back when inserting elements into lists			+2	+2	+2	
Optimize logic to reduce conditional statements			+1			
Consider using the C++11 `<random>` instead of `rand()`			+2	+2		
Use a local array instead of dynamic memory allocation				+2		
Consider multi-threading for parallelization				-1		
Consider using a dynamic array instead of a fixed-sized array					-2	
Consider unnesting nested loops					0	
Avoid unnecessary operations					+1	
Consider using a more efficient algorithm for bubble-down operations					+1	
Feedback Count	6	8	9	7	8	38
Agreement Count	3	7	9	6	6	31
Agreement Percentage	50	88	100	86	75	82

TABLE VI
AGREEMENT RATINGS IN LIKERT SCALE (-2 TO +2) FOR CHATGPT RECOMMENDATIONS FOR IMPROVING READABILITY AND MAINTAINABILITY

Description of Recommendations Made by ChatGPT	Code Samples					
	C1	C2	C3	C4	C5	All
Consistent naming conventions	+1					
Change `#define` to `constexpr`	+1					
Use consistent formatting, spacing and indentation	+1	+1	+1			
Add more comments	+1	+1		+1	+1	
Move function declarations before the main function	+1					
Improve variable names	+2					
Replace bitwise AND operator with modulo comparison operator	-2					
Use descriptive and meaningful names for functions and variables		+1	+1	+1	+1	
Encapsulate (related vars/funcs/params) into a class		+2	+2		+1	
Encapsulate algorithm steps/logic within functions		+2	+2	+1	+2	
Use constants instead of magic numbers		+2	+2		+2	
Use `std::` before each identifier instead of namespace `std`			0			
Use smart pointers instead of raw pointers			+2			
Use const for read-only variables			+2	+2		
Replace C-Style arrays with standard containers such as `std::vector`			+2	+2		
Consider using range-based for loops			+1	+1		
Instead of using new and delete, use smart pointers				+2		
Use const references to avoid unnecessary copying when passing arguments				+2		
Initialize variables when declaring them				+2		
Use `std::vector` instead of dynamic arrays				+2	+2	
Refactor bubble-up and bubble-down operations for better understanding					+1	
Modularize Test Functions					+1	
Feedback Count	7	6	10	10	8	41
Agreement Count	6	6	10	10	8	40
Agreement Percentage	84	100	100	100	100	98

and our agreement ratings. Similarly, ChatGPT's recommendations for enhancing the readability and maintainability of the programs along with our agreement ratings are presented in Table VI. An empty cell in the tables implies the absence of the corresponding strength, weakness, or recommendation for the corresponding program. For example, as seen in Table III, the strength "well-documented with comments" was identified by ChatGPT in programs C1, C2, C3, and C4, but not in C5.

A individual row in the tables includes a description of the feedback (strength, weakness, or recommended change) provided by ChatGPT and indicates which of the five code samples (i.e., C1, C2, C3, C4, and C5) the feedback applies to. An agreement rating indicates the degree to which the authors agree with the ChatGPT provided feedback ranging from strongly disagree (-2) to strongly agree (+2).

Similar feedback descriptions were merged. The number of distinct feedback items/descriptions were 15 identified strengths, 22 recommendations for improving readability and maintainability, 25 recommendations for improving execution speed, and 34 identified weaknesses. Given the imbalance between the number of identified strengths and the number of identified weaknesses, one might assume that ChatGPT found the provided code examples to be lacking in quality. This might be due to the fact that the evaluated code, though accurate for the assigned problem, was not production quality.

An agreement percentage was computed by counting the number of feedback items that the authors agreed or strongly agreed with ChatGPT and dividing by the total number of feedback items. The agreement percentage ranged from 82 percent for identified weaknesses, 89 percent for execution speed recommendations, 98 percent for readability and maintainability recommendations to 100 percent for identified strengths. From the above statistics, it is apparent that the authors were generally impressed with the feedback provided by ChatGPT but found the recommendations for improving execution speed to be less agreeable than the other feedback provided.

V. Discussion of Results

The authors learned a few C++ constructs and features that they were not familiar with despite years of experience with C and C++. Most of these appeared to be features added in C++11 and C++17 which the first author had not yet encountered in the several graduate CS classes he took that used C++ starting in 2019. These features include the C++11 `<random>` library, object functors, the '`inline`' keyword, the '`constexpr`' keyword, and the emplace_back insertion mode. From the authors' perspective, the pedagogical aspect of using ChatGPT provided significant value particularly since the disclosed features appeared to be immediately applicable to the evaluated code.

A. Potentially Misleading Feedback

The one area that the authors found ChatGPT to be potentially misleading was in regard to recommendations to improve execution speed. Potentially misleading recommendations included:

1) Replacing function objects with function pointers. A search of the internet revealed opposing recommendations since function objects are typically inlined by the compiler and therefore faster.
2) Moving the calculation of `dX` outside the loop. All 3 instances of `dX` in three separate functions were already outside the loops in which they are used. This raises the question of whether or not ChatGPT was aware of the loop nesting. Or, being a generative AI tool, did it simply find a comment that appeared to match, and thus ChatGPT naively included it in its recommendation?
3) Replace `(i \& 0x01)` with `(i \% 2 == 1)`. While the order of precedence in the second statement is not a problem for execution accuracy, in the authors' experience a bitwise AND operation is much faster than the alternative modular arithmetic followed by a logical comparison operation. However, the latter is possibly more readable to human programmers of varying levels of experience.
4) Consider using a dynamic array instead of a fixed-sized array. In the authors' experience, dynamic allocation is much slower than the allocation of local variables.

B. High Agreement Percentages

Despite the potentially misleading recommendations to improve execution speed which resulted in an agreement rate of 82 percent, the authors agreed with a high percentage of the feedback provided by ChatGPT. For example, agreement with identified weaknesses was 89 percent, and agreement with recommendations for readability and maintainability was 98 percent.

Overall the agreement rate was 91.3 percent. Further study is needed to determine if the corpus of publications related to software readability and maintainability is much higher than software execution speed resulting in better recommendations for readability and maintainability than with execution speed. However, a Google search of those phrases resulted in more hits for "software execution speed" (1.4 billion) than for "software readability and maintainability" (1.59 million).

VI. Threats to Validity

The results of this case study are drawn based on ChatGPT's feedback on only five code samples developed as academic assignments for two graduate-level courses. This small size of samples and fairly small (in terms of the number of lines of code) programs dealing with academic assignments may not represent real-world source code developed in the industry. This can be viewed as a limitation of this study. Nevertheless, these small programs were found effective in evaluating ChatGPT's assessment of source code and recommendations for improvements. Another possible threat to the validity of this case study is in equating agreement ratings with feedback quality. The agreement ratings were generated by the authors only. However, the authors' long experience in the field minimizes this threat.

VII. Conclusion

Generative AI tools such as ChatGPT have become popular among software developers and students seeking assistance in dealing with programming problems. In this paper, we have presented a case study on ChatGPT's recommendations for improving programs' execution speed, readability, and maintainability. Five code samples written in C++ were fed to ChatGPT and its feedback was analyzed.

ChatGPT appears to provide feedback on software quality that can reduce the need for human-provided feedback. While many of ChatGPT's recommendations made sense, some recommendations were counter-intuitive. We have noticed that ChatGPT's feedback typically favored recommendations for human readability of source code over technical efficiency. Therefore, as with any advisor, recommendations provided by ChatGPT should be vetted and not blindly followed.

In the future, we plan to extend this case study to a large empirical study with many code samples drawn from diverse sources and with in-depth quantitative and qualitative analyses.

Acknowledgement

This work is supported in part by the ISU-CAES (Center for Advanced Energy Studies) Seed Grant at the Idaho State University (ISU), USA.

References

[1] OpenAI, *ChatGPT (version 3.5) Large language model*. https://chat.openai.com, Verified: Feb 2024.

[2] M. Zibran and C. Roy, "Conflict-aware optimal scheduling of code clone refactoring," *IET Software*, vol. 7, no. 3, pp. 167–186, 2013.

[3] M. Islam and M. Zibran, "How bugs are fixed: Exposing bug-fix patterns with edits and nesting levels," in *35th ACM/SIGAPP Symposium on Applied Computing*, 2020, pp. 1523–1531.

[4] M. Islam and M. Zibran, "What changes in where? an empirical study of bug-fixing change patterns," *ACM Applied Computing Review*, vol. 20, no. 4, pp. 18–34, 2021.

[5] M. Islam and M. Zibran, "A comparative study on vulnerabilities in categories of clones and non-cloned code," in *10th IEEE Intl. Workshop on Software Clones*, 2016, pp. 8–14.

[6] M. Islam, M. Zibran, and A. Nagpal, "Security vulnerabilities in categories of clones and non-cloned code: An empirical study," in *11th ACM/IEEE Intl. Symposium on Empirical Software Engineering and Measurement*, 2017, pp. 20–29.

[7] M. Zibran, R. Saha, C. Roy, and K. Schneider, "Evaluating the conventional wisdom in clone removal: A genealogy-based empirical study," in *28th ACM/SIGAPP Symposium on Applied Computing*, 2013, pp. 1123–1130.

[8] M. Islam and M. Zibran, "On the characteristics of buggy code clones: A code quality perspective," in *12th IEEE Intl. Workshop on Software Clones*, 2018, pp. 23 – 29.

[9] D. Alwad, M. Panta, and M. Zibran, "An empirical study of the relationships between code readability and software complexity," in *27th International Conference on Software Engineering and Data Engineering*, 2018, pp. 122–127.

[10] M. Islam and M. Zibran, "Towards understanding and exploiting developers' emotional variations in software engineering," in *SERA*, 2016, pp. 185–192.

[11] M. Islam and M. Zibran, "Exploration and exploitation of developers' sentimental variations in software engineering," *Internation Journal of Software Innovation*, vol. 4, no. 4, pp. 35–55, 2016.

[12] M. Islam and M. Zibran, "Sentiment analysis of software bug related commit messages," in *27th Intl. Conference on Software Engineering and Data Engineering*, 2018, pp. 3–8.

[13] M. Islam and M. Zibran, "Leveraging automated sentiment analysis in software engineering," in *MSR*, 2017, pp. 203–214.

[14] A. Champa, M. Rabbi, M. Zibran, and M. Islam, "Insights into female contributions in open-source projects," in *20th IEEE International Conference on Mining Software Repositories*, 2023, pp. 357–361.

[15] M. Rabbi, A. Champa, C. Nachuma, and M. Zibran, "SBOM generation tools under microscope: A focus on the npm ecosystem," in *In Proceedings of ACM Symposium on Applied Computing (SAC)*, 2024, pp. 1233–1241.

[16] A. I. Champa, M. F. Rabbi, C. Nachuma, and M. F. Zibran, "ChatGPT in action: Analyzing its use in software development," in *Proceedings of the 21st International Conference on Mining Software Repositories (MSR)*, 2024, p. 5 pages (to appear).

[17] C. Nachuma, M. Rabbi, A. Champa, and M. Zibran, "Analyzing chatgpt assistance in programming," in *22nd IEEE/ACIS International Conference on Software Engineering, Management and Applications (SERA)*, 2024, p. 6 pages (to appear).

[18] M. F. Rabbi, A. I. Champa, M. F. Zibran, and M. R. Islam, "AI writes, we analyze: The ChatGPT python code saga," in *Proceedings of ACM International Conference on Mining Software Repositories (MSR)*, 2024, p. 5 pages (to appear).

[19] A. Clark, D. Igbokwe, S. Ross, and M. Zibran, "A quantitative analysis of quality and consistency in ai-generated code," in *7th IEEE International Conference on Software and System Engineering (ICoSSE)*, 2024, p. 5 pages (to appear).

[20] S. Piccolo, P. Denny, A. Luxton-Reilly, S. Payne, and P. Ridge, *Many bioinformatics programming tasks can be automated with ChatGPT*. https://arxiv.org/abs/2303.13528, Verified: Feb 2024.

[21] K. Jesse, T. Ahmed, P. Devanbu, and E. Morgan, *Large Language Models and Simple, Stupid Bugs*. https://doi.org/10.48550/arXiv.2303.11455, Verified: Feb 2024.

[22] D. Gewirtz, *How to use ChatGPT to write code*. https://www.zdnet.com/article/how-to-use-chatgpt-to-write-code/, Verified: Feb 2024.

[23] A. Israelsen, *How to use ChatGPT to write code*. https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding, Verified: Feb 2024.

[24] Z. Larson, *ChatGPT Isn't Coming for Your Coding Job*. https://www.wired.com/story/chatgpt-coding-software-crisis/, Verified: Feb 2024.

[25] H. Savona and D. Ackerson, *ChatGPT Writes Code: Will It Replace Software Developers?* https://semaphoreci.com/blog/chatgpt-software-developers, Verified: Feb 2024.

[26] P. Group, *GitHub Copilot vs ChatGPT Which Is The Go To Developers*. https://peopletech.com/blogs/github-copilot-vs-chatgpt-which-is-the-go-to-developers/, Verified: Feb 2024.

[27] G. Lawton, *GitHub Copilot vs. ChatGPT: How do they compare?* https://www.techtarget.com/searchenterpriseai/tip/GitHub-Copilot-vs-ChatGPT-How-do-they-compare, Verified: Feb 2024.

[28] O. Hutsulyak, *ChatGPT vs Copilot: Which And When To Use*. https://www.techmagic.co/blog/github-copilot-vs-chatgpt/, Verified: Feb 2024.