IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 27, Issue 3, Ser. 3 (May. – June. 2025), PP 55-68
www.iosrjournals.org

Code Quality Generated by AI Tools: A Review

Ishika Sharma[1], Dhavleesh Rattan[2]
[1]*Department of Computer Science and Engineering, Punjabi University, India*
[2]*Department of Computer Science and Engineering, Punjabi University, India*

Abstract
The research explores the transformative impact of artificial intelligence (AI) on software development, highlighting the specific advancements in code era, evaluation, and maintenance. It discusses the abilities of numerous AI-powered tools, which includes ChatGPT, GitHub Copilot, and Google Bard, which assist developers with the aid of automating repetitive tasks, generating test cases, and identifying bugs and security vulnerabilities. While these tools enhance productivity and code quality, the paper additionally addresses risks, including the technology of erroneous or inefficient code and security vulnerabilities that can arise from reliance on AI-generated outputs. A systematic review methodology is employed to investigate the present literature on AI-generated code quality, emphasizing the necessity of human oversight for making the best outcomes. The research concludes by way of evaluating the effectiveness of these AI methodologies in enhancing software improvement approaches, outlining fine practices for integrating AI tools into coding practices, and emphasizing the stability between automation and the want for human know-how.

Keywords: *AI Generated Code , Code Quality, Leetcode, Reliability,Metrics and Errors.*

Date of Submission: 06-06-2025 Date of Acceptance: 16-06-2025

I.Introduction

The world of AI is a dynamic and ever-evolving panorama, in which researchers like me are continuously pushing the limits of what's viable. It's a discipline packed with enormous promise and complicated challenges. Artificial intelligence (AI) has turned out to be more and more popular in software improvement to automate obligations and improve performance[2]. AI has the capability to help while developing or retaining software, in the experience that it could produce solutions out of a textual requirement specification, and recognize code to offer tips on how a new requirement will be applied.The intersection of AI and coding has ushered in a new era of software program improvement[13]. AI-powered tools are revolutionizing the way we write, test, and preserve code, mainly to good sized upgrades in code quality and developer productivity.One of the most promising packages of AI in coding is the automated code era. By studying tremendous datasets of code, AI fashions can learn how to generate code snippets, complete capabilities, or maybe entire packages based totally on herbal language descriptions or precise necessities[9]. While this generation continues to be in its early ranges, it has the potential to seriously boost up improvement cycles and decrease human blunders.AI-powered code evaluation tools can analyze code for capacity bugs, security vulnerabilities, and overall performance bottlenecks. These tools can perceive styles and anomalies that might be ignored via human reviewers, mainly to better-great code[18]. Furthermore, AI can generate check cases, automate checking out approaches, and examine test consequences to enhance code reliability.AI can assist builders write more readable, maintainable, and efficient code with the aid of suggesting improvements in code style, naming conventions, and modularity[16]. By analyzing codebases, AI can perceive possibilities for refactoring and optimization, leading to cleaner and more performant code.AI-powered tools are revolutionizing the way we write, check, and keep code, leading to full-size upgrades in code satisfactory and developer productivity. Here are some key AI tools for code quality: ChatGPT, Github Copilot, DevGPT, Tabnine, Google Bard, Google Plam2[5,21].

Table 1 : Comparison of Different AI Tools

AI Tool	Developed By	Launch Year	Purpose	Strengths	Limitations
ChatGPT	OpenAI	2022 (ChatGPT-3)	General-purpose AI, code generation, Q&A	Advanced language model, versatile for many tasks, easy to use UI	Struggles with highly domain-specific code, can produce verbose answers
GitHub	GitHub (powered by	2021	Assists in coding, IDE	Excellent for code	Limited outside of

Copilot	OpenAI Codex)		integration	completion in IDEs, context-aware	specific programming languages, IDE dependent
DevGPT	DevGPT Labs	2023	Code generation and developer workflow automation	Optimized for development, script-based workflows	Limited to specific programming languages and tasks
Tabnine	Tabnine	2019	Code completion for developers	Strong code completion customizable	Not as powerful for natural language tasks
Google Bard	Google	2023	General-purpose AI, also code	Good for data-related queries, solid understanding of Google's own APIs	Still improving on complex programming tasks
Google PaLM 2	Google	2023	Advanced language and code model	Designed to handle complex language tasks, large dataset	Limited availability; mostly experimental as of now

II.Background

Leetcode

Programmers frequently utilize LeetCode, an online platform that offers a variety of programming problems and challenges, to improve their skills. LeetCode provides a trustworthy database of programming problems arranged according to their concepts and degree of difficulty. These issue categories include "easy," "medium," and "hard," and the topics cover a wide spectrum, including databases, shell scripting, algorithms, concurrency, and more. It is important to note that our contribution does not include classifying problems according to their level of difficulty or subject. For more detail, readers who are interested in learning more about the distinctive problem classification should contact the LeetCode platform. Every programming problem usually has pattern inputs, outputs, and a problem statement. With the help of LeetCode's inbuilt editor and compiler, users may compare the accuracy and efficiency of their code with a collection of predefined test references. Furthermore, LeetCode gives error warnings, monitors the popularity of submissions, and, according to submission scores, assigns people to fantastic performance ranks. For our test, we use LeetCode as a source of many programming issues and challenges[3].

Working od Leetcode

- Select a Problem: Pick a problem from a list of categories or levels of complexity.
- Write Code: Write your code and check it using the online editor.
- Run test cases: To make sure your code functions as intended, test it using sample inputs.
- Submit Solution: To verify if the code passes all check cases, submit it.
- Examine and Learn Skills: Examine discussions to learn new techniques or view solutions from other users.

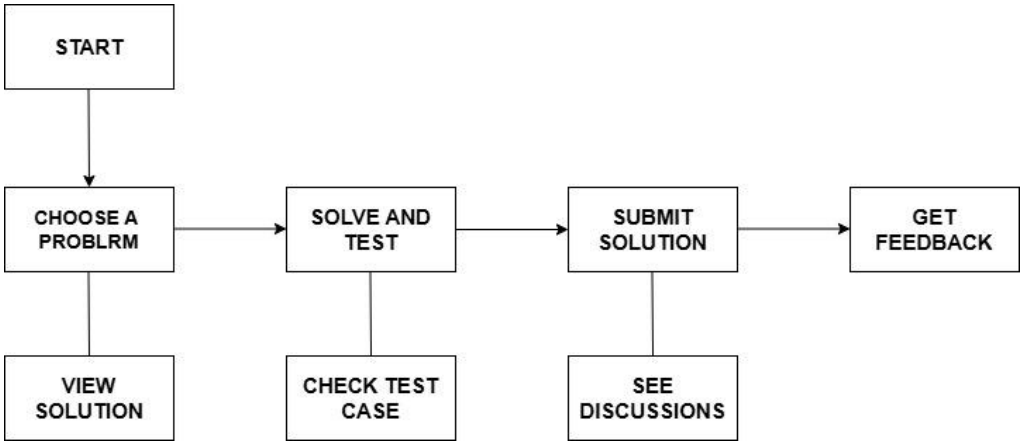

Fig.1: Working of leetcode

Code Quality

Code quality is the degree to which the code is well-written, responsibly developed, and maintained. Good code satisfaction indicates that the code is easy to analyse, dependable, and plays well. It wants to be free of bugs and security issues, avoid needless complexity, and adhere to best practices. High-quality code is more useful for long-term projects since it is also easier to update, test, and scale across instances. ChatGPT, GitHub Copilot, and other AI coding tools can boost productivity by automating repetitive tasks and making code recommendations, but they also often cause specific issues with code quality. The biggest problem is duplication; AI-generated code may be complex and contain extra strains that make it larger than required.Another frequent problem is inefficiency, as AI tools may generate code that functions but isn't performance-optimized, which might slow down packages or consume more resources. Lack of context is also a problem; given that AI models create code primarily based on statistical patterns rather than actual understanding, they will pass over the larger image, producing code that technically works however doesn't completely match project requirements or best practices.

AI-generated code may not always adhere to security best practices, resulting in vulnerabilities. AI can potentially recommend insecure patterns simply because it sees comparable styles in its training data, without filtering for security considerations. Error rates are every other problem, as AI tools may generate code that has small, hard-to-note bugs, mainly in complex projects. These small issues can emerge as complicated if no longer cautiously reviewed and corrected via a developer. Researchers advise for the use of AI-generated code as a starting point, but ensure that human oversight is important to ensure code quality, security, and proper optimization, specially in vital software program development contexts[3].

III. Review Method

Planning and Review

The framework for the study's questions, the databases that were searched, and the techniques for locating and verifying the evidence are all included in the review methodology. Identifying original research, applying inclusion and exclusion criteria, and synthesizing the findings are all part of the assessment process. The protocol, which was designed in the remaining phase, was developed by one of the authors, evaluated by the other authors, and then finalized by discussion, overview, and iteration in order to eliminate researcher bias. A thorough search of electronic databases has been conducted, and more study is recommended. Additionally, a number of the top conference proceedings and software program engineering publications that are not accessible through electronic search had to be manually searched. A total of 189 articles were found using both manual and computerized search methods.

Research Questions

Finding and categorizing the body of literature on AI tools, AI-generated code, and the quality of code produced by AI tools was the primary objective of this systematic review. A collection of research questions was required in order to plan the review. The precise research questions are listed in the table.

Source of Information

For a thorough and comprehensive review of the literature, a broad viewpoint is required. To improve the likelihood of finding the right articles, the exact selection of databases must be selected before the study begins. suggests doing a thorough search of electronic resources; the databases listed below were examined:

- ACM Digital Library (https://dl.acm.org/)
- IEEE eXplore (https://ieeexplore.ieee.org/)
- ScienceDirect (https://www.sciencedirect.com/)
- Springer (www.springerlink.com)
- Wiley Interscience (www3.interscience.wiley.com)

Table 2 : Research Questions and Motivation

RQ	QUESTIONS	MOTIVATION
1	What are the different AI tools used for code generation? Are these codes reliable?	The main objective of these questions are to know about the AI tools which are used to generate codes for different programming languages.
1.1	What are the popular programming languages which are used to generate code through AI?	
1.2	From where the researchers used the datasets?	

2	What are the different parameters to check the quality of code generated by AI?	The motive of these questions are to know the code quality of the AI generated codes. How we can check the quality or what are the parameters or metrics used to check the quality. The motive is to find out the different errors which occur in these codes and how we can fix those errors using AI tools.
2.1	What are the different metrics used to check the quality of AI generated code?	
2.2	What kind of errors or the issues occur in AI generated code?	
2.3	Is there any tool which can be used to fix the errors in AI generated code?	
3	Is there any similarity between the AI generated code and code generated by the developer?	Motivation of these questions are to check the similarity between the Ai generated code and the code generated by the developer. The study will tell us about which AI tool is better and how.
3.1	Is code generated by AI helpful to everyone?	
3.2	Which AI tool is better for coding?	

Search Criteria

The terms "AI GENERATED CODE" and "QUALITY" are included in the abstract of almost every search. It is a lengthy and complex procedure. It identifies the specific search strategy from unique online sources. We made an effort to retrieve as much important information as we could. To ensure that our study was complete, we conducted a thorough database search. Still, for various reasons, some of the acknowledged research articles were not included in the preset search approach. The search term is no longer in the abstract, the article title is exclusive, and so on. In order to finish the evaluation process, these studies are covered in the database using keyword search.

Table 3 : Search Strings

S.No.	Resources	Keyword	Dates	#
1	https://ieeexplore.ieee.org/	AI -Generated Code,Quality	All Dates	53
2	https://www.sciencedirect.com/	AI -Generated Code,Quality	All Dates	25
3	https://dl.acm.org/	AI -Generated Code,Quality	All Dates	76
4	www.springerlink.com	AI -Generated Code,Quality	All Dates	35
5	https://onlinelibrary.wiley.com/search/advanced	AI -Generated Code,Quality	All Dates	0

Inclusion and Exclusion Criteria

Using titles, irrelevant papers were manually filtered out in the first step. There are a lot of study publications that aren't applicable in our context.Studies that focused on AI-generated code quality concerns generally were suitable to be included in the review. Both professional and student software development studies were considered. To make the database search complete, the systematic review covered both qualitative and quantitative research papers that were published up until and including 2011 beginning with the digital library's founding date. Only English-language studies were included. Technical reports were a part of our research. The exclusion at various phases is seen in Fig 3. Studies whose primary focus was not on the quality of codes produced by AI tools were eliminated. To keep our research database consistent, research articles that appeared frequently in various databases and e-resources were removed one at a time. Position papers that indicated future directions are mentioned in the conclusion and future work section, but they were not included in the literature review. Prior to their extended publication in journals, some of the articles were first presented at conferences. Such early research was not included. Using the references of the discovered articles, all missing relevant papers were manually located. As Fig. 3 illustrates, After our search yielded more than 189 papers, we selected 48 papers based on their titles and 39 papers based on their abstracts. After reading all 39 of these papers, a final list of 24 papers was chosen based on the inclusion and exclusion criterion.

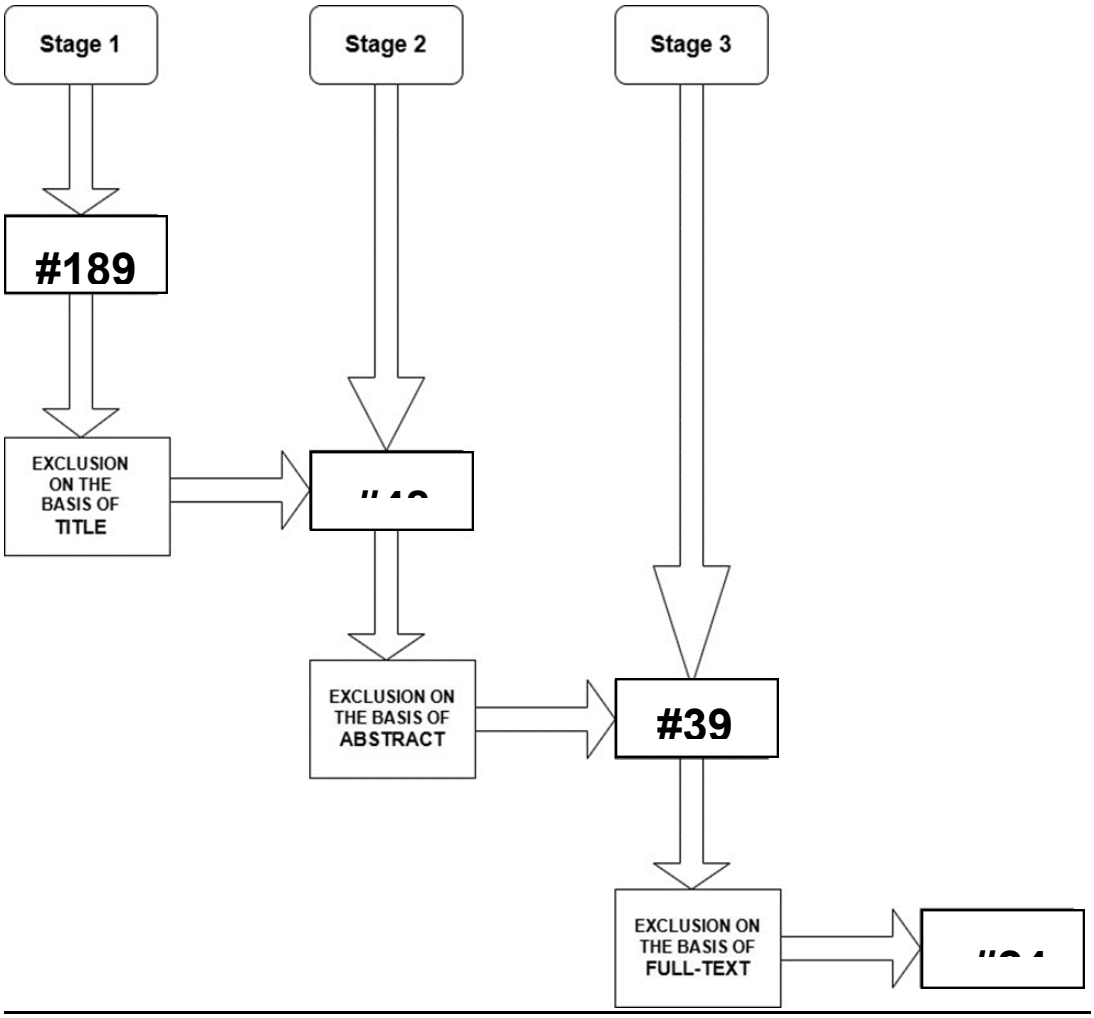

Fig 2 : Study selection procedure.

IV.Results

RQ1 What are the different AI tools used for code generation? Are these codes reliable?

Several AI technologies, like GitHub Copilot, ChatGPT, Google Palm2, and Tabnine, are used to generate code. These AI-powered tools let programmers create code more quickly by producing whole blocks of code, finishing code snippets, or making suggestions in response to natural language queries. They use massive amounts of pre-written code to look up best practices, syntax, and programming patterns[6]. AI-generated code isn't always ideal, but it may help with ongoing tasks and save time. Reliability is dependent upon both task issues and input quality. Code produced by AI tools may appear to be accurate, but it may contain hidden errors, safety risks, or delays. Therefore, it's critical that developers examine, test, and modify AI-generated code to meet their unique requirements and ensure quality[14]. Nikolaos Nikolaidis et al (2024) assesses the capacity of AI-powered coding tools, ChatGPT and Copilot, in solving coding problems. In this research the researcher has discussed the effectiveness of ChatGPT and Copilot and how to improve the performance and quality of code[15]. Autumn Clark et al (2024) analyzes the greatness and consistency of code generated by using ChatGPT, using the DevGPT dataset. The code generated by ChatGPT is of good quality[4]. Md Fazle Rabbi et al.(2024) This has a look at analyzing the quality and security of code written with ChatGPT's help (1,756 snippets). They compared code generated from scratch (ChatGPT-generated) to code modified from user input (ChatGPT-changed)[17].Ionut Daniel Fagadau et al. (2024) investigates the effect of prompt engineering on the quality of code generated by Copilot, a generative AI tool[6]. Vincenzo Corso et al.(2024) study compares 4 AI-based total code assistants: GitHub Copilot, Tabnine, ChatGPT, and Google Bard. These tools can generate code, however they rarely produce perfect, ready-to-use code[5]. Santiago Aillon et al (2023) explores the use of ChatGPT3.5 to develop a mobile app using Flutter. The AI tool became capable of generating useful code, but its effectiveness trusted the complexity of the task and the quality of the prompts[1].Yunhe Feng† et al (2023) investigates the usage of ChatGPT for the code era by way of analyzing social media posts on Twitter and Reddit. The researchers also created a dataset of ChatGPT prompts and generated code, which became public. By comparing the code quality

and the usage of Flake8, they received insights into ChatGPT's limitations and ability[7]. Muhan Guo(2024) explores ChatGPT's potential to generate complicated Java code for web development, especially a user login system. The experiments showed that ChatGPT can produce high quality, readable, and functional code[8]. Zhijie Liu et al.(2024) analyzes the exception of code generated by way of ChatGPT, a large language model. The researchers evaluated code in 5 programming languages for correctness, complexity, and security[12]. Rabimba Karanjai et al (2023) evaluated the overall performance of ChatGPT and Palm2 in generating smart contracts. The researchers found that even as each model can produce compilable code, they regularly introduce security vulnerabilities[10].Wendy Mendes et al.(2024) explores the reports of software program developers using AI code assistants like Tabnine, GitHub Copilot, and ChatGPT. Take a look at observed that these tools can extensively enhance development speed, code quality, and focus[14]. Haoquan Zhou et al (2023) explores the use of GitHub Copilot in data analysis. Researchers performed a user examination with data scientists to recognize how powerful prompts may be used to generate code[22]. Burak Yetistiren et al. (2022) investigates the quality of code generated by means of GitHub Copilot, a brand new AI device for programmers. The researchers evaluated the code's validity (does it compile?), correctness (does it work as supposed?), and efficiency (how nicely does it perform?). GitHub Copilot achieved a 91.5% successful rate in generating valid code[21]. WEI WANG et al.(2024) evaluates the effectiveness of ChatGPT in assisting software program development tasks[19]. Mohammad Amin Kuhail et al. studies assessed ChatGPT 3.5's effectiveness in solving coding issues and surveyed programmers' perspectives on AI tools. ChatGPT 3.5 excelled at easy problems however struggled with more difficult ones, especially people with lower popularity[11].Alessio Bucaioni et al. This research examines how nicely ChatGPT can write computer code. The researchers conducted experiments where they gave ChatGPT diverse programming issues . They discovered that ChatGPT may want to solve easy and medium problems , but struggled with harder ones[3].

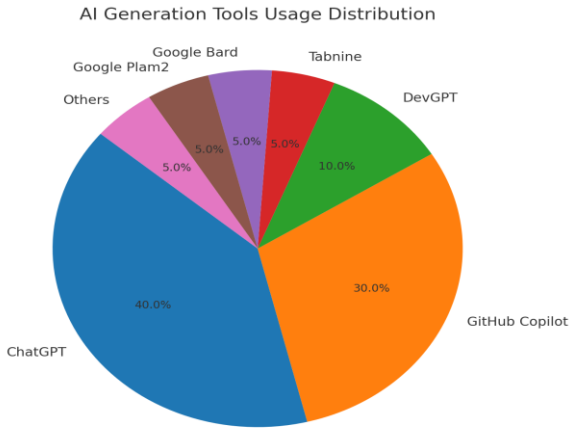

Fig 3 : AI Code Generation Tools

RQ1.1 What are the popular programming languages which are used to generate code through AI?

AI code generation technologies perform best when used with widely used programming languages that have plenty of training data. The languages that are most frequently implemented include TypeScript, Python, JavaScript, Java, and C++. Python is a popular choice for AI-driven development because of its extensive use in data science and machine learning. Because JavaScript is widely used in both front-end and back-end web projects, it is essential for web development and is strongly supported by many AI tools. While C++ is commonly used for performance-critical jobs like systems programming and game development, Java is a dependable language for large-scale applications.Other languages, such as Ruby, PHP, and Go, are also supported by AI tools, however their accuracy and advanced capabilities may not match those of the more widely used languages. With the use of AI, this technology facilitates code generation, suggestion generation, and the automation of hard coding jobs for developers working in these languages. Nikolaos Nikolaidis et al (2024) ChatGPT and Copilot were tested on 60 Python problems, with ChatGPT outperforming Copilot in terms of accurate solutions[15]. Autumn Clark et al (2024) analyzes the greatness and consistency of Python code generated by using ChatGPT, using the DevGPT dataset[4]. Md Fazle Rabbi et al.(2024) The researchers had analyzed the quality and security of Python code written with ChatGPT's help (1,756 snippets)[17]. Marchel Christopher Wuisang et al. (2023) discussed the Python issues and fixed them using a framework. Ionut Daniel Fagadau et al. (2024) They took a look at analyzed 124,800 prompts for 200 Java methods, evaluating the generated code based on correctness, complexity, length, and similarity to the intended code[6]. Vincenzo Corso et al.(2024) discussed the comparison between 4 different tools using the 100 Java methods from Github Projects[5]. Santiago Aillon et al (2023) explores the use of ChatGPT3.5 to develop a mobile app using Flutter[1]. Yunhe Feng† et al (2023) They found that ChatGPT is used

for various programming languages, on the whole Python and JavaScript, and for tasks like debugging, interview prep, and academic assignments[7]. Muhan Guo(2024) explores ChatGPT's potential to generate complicated Java code for web development, especially a user login system[8]. Zhijie Liu et al.(2024) The researchers evaluated code in 5 programming languages (C, C++, Java, Python, JavaScript) for correctness, complexity, and security[12]. Haoquan Zhou et al (2023) explores the use of GitHub Copilot in data analysis. Researchers chose three exploratory data analysis problems from the actual world that can be resolved with Python code[22]. WEI WANG et al.(2024) The examination involved 109 contributors who used ChatGPT to solve coding puzzles and carry out software development tasks[19]. Alessio Bucaioni et al. The researchers conducted experiments where they gave ChatGPT diverse programming issues in C++ and Java. They discovered that ChatGPT may want to solve easy and medium problems , but struggled with harder ones[3]. The table depicts the count of languages which were reviewed from the 24 research papers which were selected.

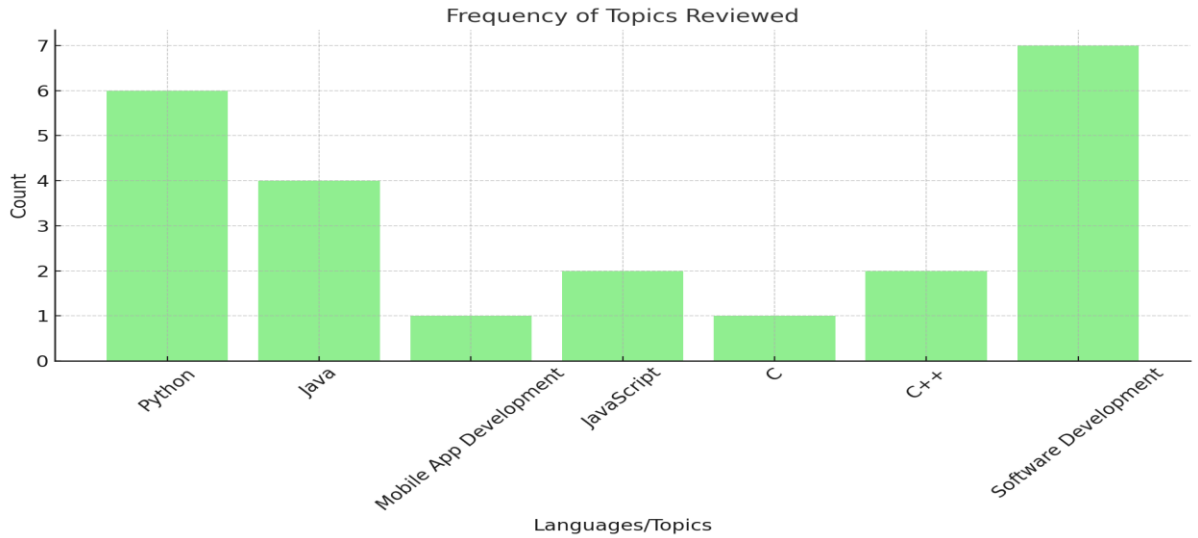

Fig 4 : Frequency of Programming Languages Reviewed.

RQ1.2 From where the researchers used the datasets?

Researchers design AI models for code generation using datasets from several sources. Platforms like LeetCode, a website where programmers tackle coding problems, have become popular sources. These challenges and their resolutions offer useful information to inform AI about typical programming issues and solutions[12,15]. DevGPT, a dataset that compiles code examples, explanations, and programming-related content from various coding websites, forums, and public repositories, is an additional source. Open-source code from websites like GitHub, where programmers freely share their projects and code, is also used by researchers. AI models may learn coding styles, trends, and best practices across many languages and contexts with the use of this data. However, researchers concentrate on publicly available and open-source code to avoid copyright since ethical issues and data privacy are important[4,17]. Nikolaos Nikolaidis et al (2024) uses the 60 leetcode problems of the python programming language[15]. Autumn Clark et al (2024) analyze the python code samples with the help of DevGPT dataset[4]. Md Fazle Rabbi et al.(2024) also used DevGPT datasets to compare the python code generated by ChatGPT from scratch (ChatGPT-generated) to code modified from user input (ChatGPT-changed)[17].Ionut Daniel Fagadau et al. (2024) examine the effects of the identical prompt features when they appear in various prompt types using both GitHub and LeetCode[6]. Vincenzo Corso et al.(2024) study compares 4 AI-based total code assistants with the help of github projects[5]. Zhijie Liu et al.(2024) analyzes the exception of code generated by way of ChatGPT, a large language model. The researchers examined the datasets from the leetcode problems and CWE (Common Weakness Enumeration) scenarios (CWE's code scenarios) as provided in[12] . Burak Yetistiren et al. (2022) investigates the quality of code generated by means of GitHub Copilot, a brand new AI device for programmers. The researchers used the HumanEval dataset which contains 164 problems[21].Alessio Bucaioni et al. the dataset was created by utilising the vast library of programming problems on LeetCode. The runtime, memory consumption, and acceptance rates of human-generated programming problem solutions are all statistically gathered using LeetCode[3].

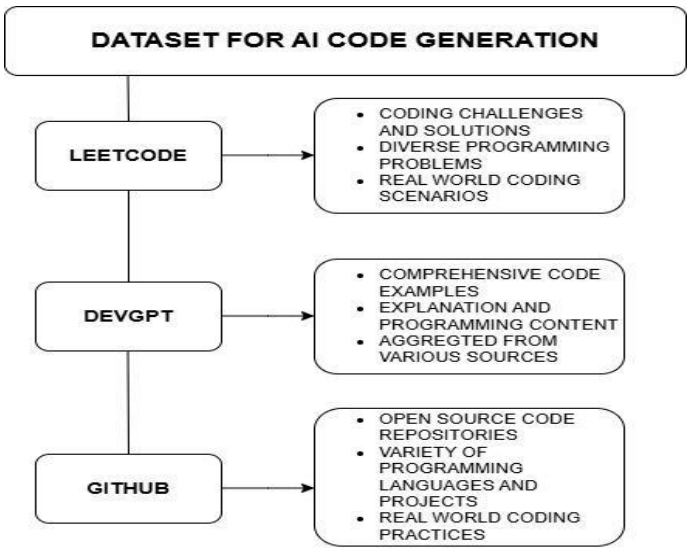

Fig 5 : Dataset

RQ2 What are the different parameters to check the quality of code generated by AI?

A number of crucial metrics are necessary to assess the caliber of AI-generated code. Correctness guarantees that the code performs its intended function without errors. Efficiency examines how effectively the code executes, ensuring that it is fast and resource-efficient. For code to be readable, it must be transparent, simple to understand and well-named and commented. Security checks make sure the code is secure and free of errors that hackers may take advantage of. The reliability is about ensuring that the code can be easily updated or resolved in the future, whereas scalability examines if the system can handle more information or users without experiencing issues. These characteristics assist in determining the code's dependability and suitability for usefulness[16].

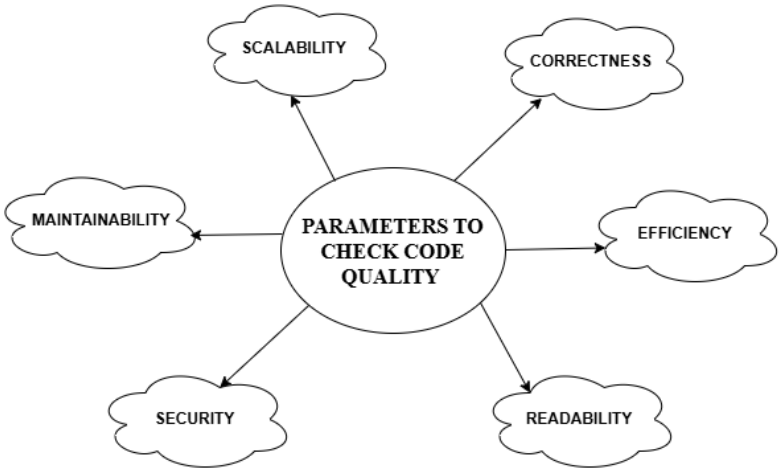

Fig 6 : Parameters.

RQ2.1 What are the different metrics used to check the quality of AI generated code?

AI-generated code is evaluated using a number of metrics to evaluate various parts of the code. Among these is Cyclomatic Complexity, which counts the number of possible ways through a code to determine its complexity. A higher number suggests the code is more complicated and more difficult to read or update[15]. Time Complexity examines how the code's runtime varies with increasing input size. It assists in figuring out whether the code is effective or whether it can become problematic while dealing with bigger datasets. In order to make sure the code doesn't utilize excessive amounts of resources, space complexity quantifies how much memory the code requires as the input size grows[12]. Halstead Complexity is an additional statistic that assesses the complexity of the code by counting the number of operators and operands. This measure assists in determining how hard it is to update or maintain the code. These metrics assist developers in evaluating the AI-generated code to make sure it is scalable, effective, and clear[4]. Nikolaos Nikolaidis et al (2024) The metrics that researchers used in their research to improve the code quality and the performance of the code are: The number of linearly distinct paths in a program is measured by its cyclomatic complexity. This is closely related to how many decision

points there are in the program. It is believed that low cyclomatic complexity is a sign of high-quality code. Token Count: A program's overall number of operators and operands. Code Lines: The function's total amount of code lines. Time Complexity: The runtime of a program is the amount of time it takes to execute. Space Complexity: Memory use refers to how much memory a software uses to store data while it is running[15]. Autumn Clark et al (2024) in this research the researchers have discussed the measures to improve the code quality of python code samples. They have used the Halstead Complexity. Eight indicators are used in the Halstead Complexity Measure to assess the level of complexity of a particular piece of code. Volume, difficulty, length, calculated length, effort, programming time, number of delivered errors, and vocabulary are these eight criteria[4]. Zhijie Liu et al.(2024) analyzes the exception of code generated by way of ChatGPT, a large language model.The researchers evaluated code in 5 programming languages (C, C++, Java, Python, JavaScript) for correctness, complexity, and security. They have discussed the code snippets in various languages, the distribution of cyclomatic and cognitive complexity levels differs[12]. Rabimba Karanjai et al (2023) We used the Code Validity, Correctness, and Security metrics to evaluate the resulting code. However, we recognise that more criteria, such as Readability, Cyclomatic Complexity, and Reusability, might be included to assess the produced code more thoroughly[10]. Burak Yetistiren et al. (2022) investigates the quality of code generated by means of GitHub Copilot, a brand new AI device for programmers. For time complexity, Copilot's code equaled human efficiency in 87.2% of accurate solutions and 76.2% of partially correct ones. Most of the space complexity was comparable to human efficiency[21].

Table 4 : Different Metrics used to check Quality of AI generated code

Metric	**Formula**	**Purpose**	**Higher Values Indicate**	**When It's Important**
Cyclomatic Complexity	$V(G)=E-N+2P$ Where: E = number of edges N = number of nodes P = number of connected components (usually 1 for a single program)	Assesses code readability, maintainability, and testability.	More complex code, harder to understand and maintain.	When you need to evaluate how complicated the control flow of your code is.
Time Complexity	Big-O notation, e.g O(1), O(n), $O(n^2)$, O(log n), etc., where n is the input size.etc.	Assesses efficiency in terms of execution speed.	Longer runtime, less efficient.	When you're concerned with performance, especially with large datasets.
Space Complexity	Big-O notation, e.g O(1), O(n), $O(n^2)$, etc.	Assesses efficiency in terms of memory usage.	More memory usage, less efficiency.	When you're concerned with resource usage, particularly in limited environments.
Halstead Complexity	N = N1 + N2 n = n1+n2 $V=N \cdot \log2(n)$ Where: N1 = number of operators N2 = number of operands n1 = number of distinct operators n2 = number of distinct operands	Assesses the difficulty of understanding, modifying, or maintaining the code.	More complex and harder to maintain code.	When you're evaluating how easily code can be understood, modified, or maintained.

RQ2.2 What kind of errors or the issues occur in AI generated code?

AI-generated code should be thoroughly examined for a variety of errors and issues. Logical errors are frequent; the code may execute without crashing, but because of logical or run errors, it may not accomplish the intended goal. Runtime errors occur when unusual events, such as dividing by zero or accessing incorrect data, cause the code to break while it is being executed[23]. Inefficient code can lead to time limit problems if it takes too long to execute, particularly when dealing with bigger data sets. This frequently indicates that the algorithm has to be improved. Syntax errors are bugs in the code that stop it from executing at all, such as missing brackets or misspelled keywords. Semantic errors occur when syntactically valid code is not suitable for the task at present,

producing unexpected results. These problems show how important it is to thoroughly test and debug AI-generated code to make sure it works as expected[15].Nikolaos Nikolaidis et al (2024) looked into the kind of errors that frequently occurred, the models' performance, and the quality of the code that was produced. The analysis showed that while both ChatGPT and Copilot are prone to syntactic and semantic errors, they can be useful tools for producing code solutions for simple issues[15]. Md Fazle Rabbi et al.(2024) Examine the code samples created and modified by ChatGPT to identify important code quality problems of four different kinds: refactoring (R), conventions (C), errors (E), and warnings (W). In both ChatGPT-generated and ChatGPT-modified code, errors occur most frequently, whereas refactoring recommendations occur least frequently[17].Marshall Christhoper Wuisang et al. (2023) Automated program repair seeks to identify or locate issues, identify or cure errors, and automatically apply fixes to software flaws in order to improve software reliability. One factor that makes ChatGPT so good at spotting and fixing complex programming problems is its capacity to comprehend and produce code explanations that are human-like. Santiago Aillon et al (2023) Human involvement may be necessary to fix problems and guarantee that the code satisfies the criteria because the produced code quality may not always be dependable. Syntax, run-time, and logic errors were among the several faults introduced by the command. Although a number of logical mistakes were made, fixing the syntax issues was not too difficult[1]. Zhijie Liu et al.(2024) Wrong Answers, Compile defects, Wrong Details, Multi-hunk Errors, Incompatible Parameters, Logic Errors, and Misunderstandings are the categories into which the article divides code defects, emphasising problems such as misalignment, logical errors, and incorrect answers[12]. Rabimba Karanjai et al (2023) describes the Non-deterministic code generation, unclear input, incomplete evaluation metrics, dataset limitations, and security flaws are some of the mistakes that affect the reproducibility, code quality, generalisability, and dependability of AI-generated code[10].

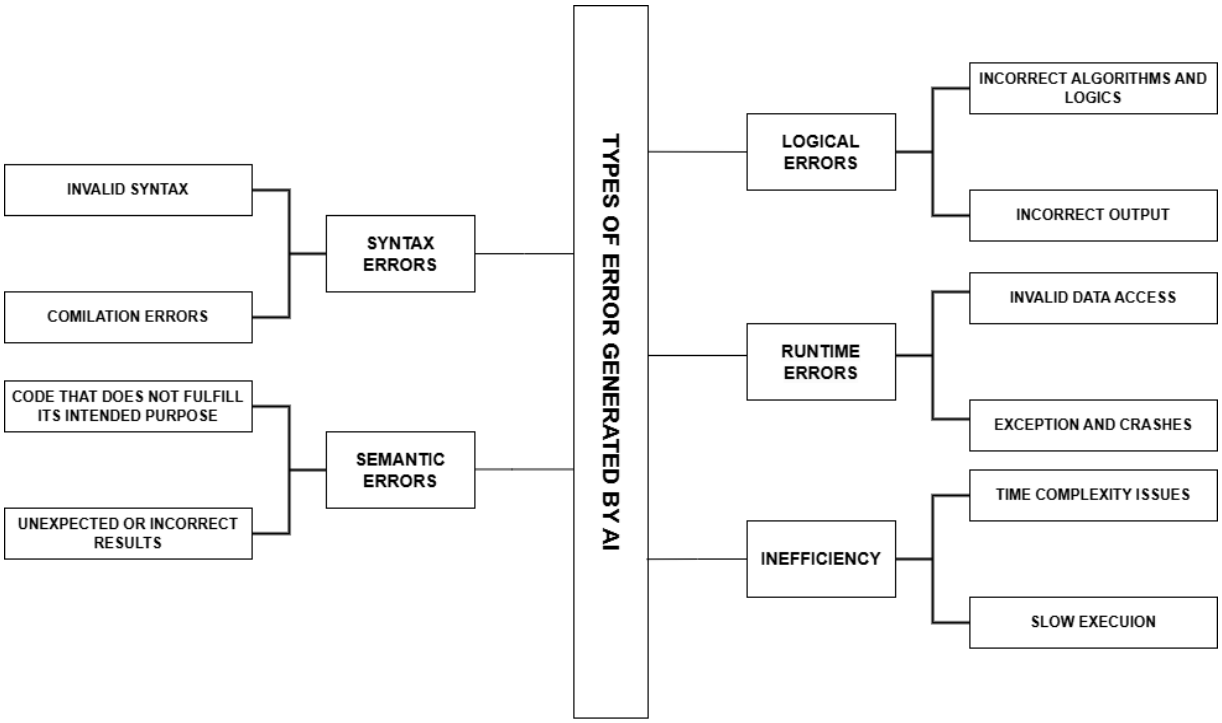

Fig 7 : Types of Errors

RQ2.3 Is there any tool which can be used to fix the errors in AI generated code?

Yes, QuixBug is a tool for fixing bugs in AI-generated code. It functions by automatically identifying errors and offering solutions. QuixBug is particularly helpful since it may identify problems or feasible problems in code produced by AI models, such as ChatGPT or GitHub Copilot. This includes errors in syntax, logic, and even performance. The code becomes more dependable and effective when developers use QuixBug to quickly find and fix bugs. In addition to saving developers time and effort, the tool helps improve the quality of AI-generated code before it is utilized in real applications by reducing the need for human debugging[20]. Marchel Christhoper Wuisang et al. (2023) assesses the use of the QuixBugs framework and how well GPT-3.5 automatically fixes Python issues. The results demonstrate that, with 30 out of 40 bugs successfully fixed, GPT-3.5 works better than other programs like Codex and preferred application repair. The version works well because it can comprehend and produce code that looks human. Nevertheless, there are drawbacks, such as the potential for incorrect or illogical responses, sensitivity to input phrases, and QuixBugs' limited functionality. Future

research can assess it against various models and find out how well it performs in other languages. For GPT-3.5 to be fully utilized for efficient bug fixes, these obstacles must be removed[20].

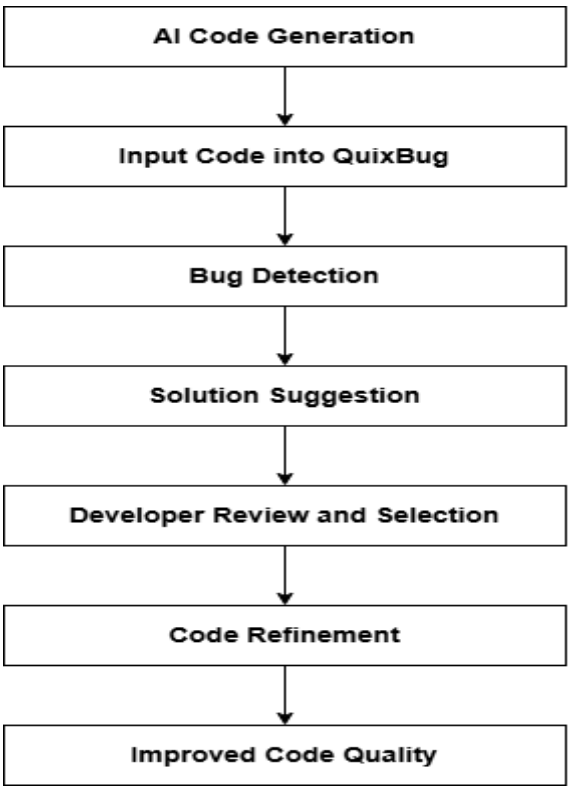

Fig 8: Working of QuixBugs.

RQ3 Is there any similarity between the AI generated code and code generated by the developer?

Developer-generated code and AI-generated code are quite comparable, especially in terms of quantity and complexity. In terms of McCabe complexity and code lines, AI assistants' code is often comparable to that of human coders. Yet, AI-generated code still differs greatly from developers' implementations in terms of accuracy and similarity, frequently requiring modifications. Even though AI code may pass test cases (plausible code), human examination does not always confirm that it is accurate. Overall, AI-generated code still has to be improved to closely match developer standards, although Tabnine's generated code had the highest degree of closeness to developers' code[5].

RQ3.1 Is code generated by AI helpful to everyone?

Although many individuals receive advantages from AI-generated code, its usefulness varies depending on the profession at hand and the user's experience. AI may serve as a tutor for novices, teaching them syntax, how to organize code, and how to solve typical programming issues. By providing immediate advice, it can help make coding a bit easier. Experienced developers may concentrate on more difficult aspects of a project by using AI technologies to generate basic code, speed up repetitive processes, and offer quick improvements[17]. AI may also be used to find errors in code and provide solutions.It might not be as helpful, though, for jobs requiring in-depth knowledge, such as developing highly specialized systems or complex algorithms, where human imagination and sense of smell are still important. Furthermore, if developers rely too much on AI, they may become dependent on recommendations, which could restrict their ability to improve their own problem-solving abilities. In conclusion, AI-generated code may be a very useful tool, but it works best when paired with human expertise and careful review[15].

RQ3.2 Which AI tool is better for coding?

There are several AI tools which are used to generate code.Copilot smoothly integrates with well-known code editors and makes intelligent recommendations as you type, it is frequently regarded as one of the greatest AI tools for coding. It speeds up manual tasks, generates code snippets, and assists in code completion. Because it can explain code, assist with debugging, and provide answers to complex programming queries, ChatGPT is widely regarded for a more conversational and problem-solving approach. Depending on whether you want a

more interactive problem-solving assistance (ChatGPT) or textual recommendations (Copilot), you may choose between them[4].

Table 5 : Comparison of Different AI Tools

AI Tool	Code Quality	Correctness	Performance	Error Rates	Reliability
ChatGPT	Good for general code, but can be verbose and may need optimization	High for common tasks, but variable on complex code	Moderate; may generate non-optimized code	Occasional errors, especially in nuanced code	Reliable for broad tasks, less so for specifics
GitHub Copilot	High; code is concise and aligns with developer conventions	High for standard patterns; excellent in supported languages	Generally good; suggests efficient code, but context matters	Lower error rate, especially in IDE	Very reliable within IDE environments and supported languages
DevGPT	Good, focused on task-specific code; may lack adaptability in complex scenarios	Moderate; strong for repetitive or task-specific code	Adequate, but some outputs lack efficiency	Low to moderate; fewer errors for narrow tasks	Reliable within supported workflows, less adaptable
Tabnine	Good for boilerplate and repetitive code	High for standard, repetitive patterns	Moderate; doesn't prioritize optimization	Low error rate for predictable code	Reliable for repetitive tasks; less versatile for complex code
Google Bard	Moderate; often verbose or simplistic code	Moderate; accuracy drops in complex tasks	Moderate; needs refinement for efficiency	Moderate error rate on complex code	Reliable for simpler tasks; struggles with nuanced code
Google PaLM 2	High; code often refined, suitable for complex problems	High; strong in logic-heavy tasks	Good; more capable of efficient solutions than others	Low to moderate; fewer errors on complex code	Very reliable across diverse coding tasks

V.Discussion

The dialogue surrounding the combination of AI tools in software program improvement highlights a transformative shift in coding practices and productivity. As AI technology like ChatGPT, GitHub Copilot, and others gain traction, they demonstrate the capacity to seriously enhance code quality, automate repetitive tasks, and assist developers in generating code snippets correctly. However, challenges remain, including concerns about the reliability, security, and performance of the AI-generated code. Issues which include verbosity, redundancy, and adherence to best practices need to be addressed to make certain developers can trust these tools in difficult situations. Furthermore, the dearth of deep contextual knowledge in AI models underscores the necessity for human oversight in code validation and optimization. The effectiveness of AI tools varies across specific programming languages and alertness contexts, indicating a need for tailor-made approaches. Ultimately, even as AI tools give amazing opportunities for streamlining software improvement, a balanced attitude that acknowledges their boundaries is crucial. By specializing in collaborative frameworks where human know-how and AI abilities complement each other, the software program engineering can leverage those innovations to power significant improvements in productiveness and code integrity

VI.Future Work

Future work inside the field of AI tools for software improvement have to consciousness on several key areas to enhance their effectiveness and integration. This consists of growing comprehensive frameworks for assessing AI-generated code quality, undertaking longitudinal studies to understand the long-term influences of those tools on coding practices, and investigating user experience to optimize their interplay with AI technologies. Additionally, studies must aim to enhance human-AI collaboration via incorporating developer feedback into AI training procedures and ensuring diversity in training datasets to mitigate biases and vulnerabilities. Ethical issues surrounding AI's use in coding, inclusive of intellectual assets and responsibility, must also be addressed. Expanding the exploration of AI applications across diverse programming languages and domain names, continuously enhancing AI models for context-aware code generation, establishing best practices for powerful integration into development workflows, and analyzing the role of AI in newbie programmer training are crucial.

Collectively, those study guidelines will help maximize the importance of AI in coding whilst retaining high standards of quality and security.

VII.Conclusion

The research concludes by way of highlighting the pivotal function of AI tools in reworking software development methodologies. Technologies which include ChatGPT and GitHub Copilot significantly enhance coding performance and productivity through automation of repetitive tasks and the provision of code suggestions. However, developers need to continue to be privy to the intrinsic limitations and challenges connected to AI-generated outputs, including code quality, the risk of bugs, security vulnerabilities, and efficiency issues, which all necessitate human oversight and intervention. The paper underscores the significance of viewing AI tools as supportive assistants as opposed to direct replacements for human developers. It stresses the want for ongoing assessment and improvement of AI-generated code to make certain it meets high standards and aligns with project requirements. Additionally, the paper proposes instructions for future studies aimed at discovering the evolving abilities of AI in coding, with the goal of improving the trustworthiness and excellence of AI-produced answers. In essence, the powerful integration of AI into software program improvement relies on a collaborative framework that leverages each human expert and AI's capabilities. This partnership is important for riding innovation even as upholding stringent requirements for code quality and security. Striking this stability may be vital for progressing the field and absolutely tapping into the potential of AI technologies in programming practices.

References

[1]. Aillon, S., Garcia, A., Velandia, N., Zarate, D. and Wightman, P., 2023, November. Empirical evaluation of automated code generation for mobile applications by AI tools. In 2023 IEEE Colombian Caribbean Conference (C3) (pp. 1-6). IEEE.

[2]. Barenkamp, M., Rebstadt, J. & Thomas, O. Applications of AI in classical software engineering. AI Perspect 2, 1 (2020). https://doi.org/10.1186/s42467-020-00005-4

[3]. Bucaioni, A., Ekedahl, H., Helander, V. and Nguyen, P.T., 2024. Programming with ChatGPT: How far can we go?. Machine Learning with Applications, 15, p.100526.

[4]. Clark, A., Igbokwe, D., Ross, S. and Zibran, M.F., 2024, April. A quantitative analysis of quality and consistency in ai-generated code. In 2024 7th International Conference on Software and System Engineering (ICoSSE) (pp. 37-41). IEEE.

[5]. Corso, V., Mariani, L., Micucci, D. and Riganelli, O., 2024, April. Assessing AI-Based Code Assistants in Method Generation Tasks. In Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (pp. 380-381).

[6]. Fagadau, I.D., Mariani, L., Micucci, D. and Riganelli, O., 2024, April. Analyzing Prompt Influence on Automated Method Generation: An Empirical Study with Copilot. In Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension (pp. 24-34).

[7]. Feng, Y., Vanam, S., Cherukupally, M., Zheng, W., Qiu, M. and Chen, H., 2023, June. Investigating code generation performance of ChatGPT with crowdsourcing social data. In 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 876-885). IEEE.

[8]. Guo, M., 2024, April. Java Web Programming with ChatGPT. In 2024 5th International Conference on Mechatronics Technology and Intelligent Manufacturing (ICMTIM) (pp. 834-838). IEEE.

[9]. J.N. Kok, E.J. Boers, W.A. Kosters, P. Van der Putten, M. Poel, Artificial intelligence: definition, trends, techniques, and cases. Artif. Intell., 1 (2009), pp. 270-299

[10]. Karanjai, R., Li, E., Xu, L. and Shi, W., 2023, October. Who is smarter? an empirical study of ai-based smart contract creation. In 2023 5th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS) (pp. 1-8). IEEE.

[11]. Kuhail, M.A., Mathew, S.S., Khalil, A., Berengueres, J. and Shah, S.J.H., 2024. "Will I be replaced?" Assessing ChatGPT's effect on software development and programmer perceptions of AI tools. Science of Computer Programming, 235, p.103111.

[12]. Liu, Z., Tang, Y., Luo, X., Zhou, Y. and Zhang, L.F., 2024. No need to lift a finger anymore? assessing the quality of code generation by chatgpt. IEEE Transactions on Software Engineering.

[13]. Marcello Mariani, Yogesh K. Dwivedi,"Generative artificial intelligence in innovation management: A preview of future research developments", Journal of Business Research, Volume 175, 2024, 114542, ISSN 0148-2963, https://doi.org/10.1016/j.jbusres.2024.114542.

[14]. Mendes, W., Souza, S. and De Souza, C., 2024, April. " You're on a bicycle with a little motor": Benefits and Challenges of Using AI Code Assistants. In Proceedings of the 2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering (pp. 144-152).

[15]. Nikolaidis, N., Flamos, K., Gulati, K., Feitosa, D., Ampatzoglou, A. and Chatzigeorgiou, A., 2024, March. A Comparison of the Effectiveness of ChatGPT and Co-Pilot for Generating Quality Python Code Solutions. In 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering-Companion (SANER-C) (pp. 93-101). IEEE.

[16]. Odeh, Ayman & Odeh, Nada & Mohammed, Abdul. (2024). A Comparative Review of AI Techniques for Automated Code Generation in Software Development: Advancements, Challenges, and Future Directions. TEM Journal. 726-739. 10.18421/TEM131-76.

[17]. Rabbi, M.F., Champa, A.I., Zibran, M.F. and Islam, M.R., 2024, April. AI writes, we analyze: The ChatGPT python code saga. In Proceedings of the 21st International Conference on Mining Software Repositories (pp. 177-181).

[18]. Taeb, Maryam & Chi, Hongmei & Bernadin, Shonda. (2024). Assessing the Effectiveness and Security Implications of AI Code Generators. Journal of The Colloquium for Information Systems Security Education. 11. 6. 10.53735/cisse.v11i1.180.

[19]. Wang, W., Ning, H., Zhang, G., Liu, L. and Wang, Y., 2024. Rocks Coding, Not Development: A Human-Centric, Experimental Evaluation of LLM-Supported SE Tasks. Proceedings of the ACM on Software Engineering, 1(FSE), pp.699-721.

[20]. Wuisang, M.C., Kurniawan, M., Santosa, K.A.W., Gunawan, A.A.S. and Saputra, K.E., 2023, September. An evaluation of the effectiveness of openai's chatGPT for automated python program bug fixing using quixbugs. In 2023 International Seminar on Application for Technology of Information and Communication (iSemantic) (pp. 295-300). IEEE.

[21]. Yetistiren, B., Ozsoy, I. and Tuzun, E., 2022, November. Assessing the quality of GitHub copilot's code generation. In Proceedings of the 18th international conference on predictive models and data analytics in software engineering (pp. 62-71).
[22]. Zhou, H. and Li, J., 2023, April. A case study on scaffolding exploratory data analysis for AI pair programmers. In Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems (pp. 1-7).
[23]. Le, K.T. and Andrzejak, A., 2024. Rethinking AI code generation: a one-shot correction approach based on user feedback. *Automated Software Engineering*, *31*(2), p.60.
[24]. Omari, S., Basnet, K. and Wardat, M., 2024. Investigating large language models capabilities for automatic code repair in Python. *Cluster Computing*, pp.1-15.
[25]. Mišić, M. and Dodović, M., 2024. An assessment of large language models for OpenMP-based code parallelization: a user perspective. *Journal of Big Data*, *11*(1), p.161.
[26]. Atkinson, C.F., 2023. ChatGPT and computational-based research: benefits, drawbacks, and machine learning applications. *Discover Artificial Intelligence*, *3*(1), p.42.