NextGen Scientific Publication
American International Journal of Computer Science and Technology

Volume 7 Issue 2, Pg. No. 1-14, AIJCST-V7I2P101, 2025
https://doi.org/10.63282/3117-5481/AIJCST-V7I2P101

Original Article

Advancements in AI Coding Assistance Tools and Their Potential Impact on Collaborative Software Development

*** Sandeep Kumar Jangam**
Independent Researcher, USA.

Abstract:

The AI is soon to change software engineering because AI putpins such as GitHub Copilot, OpenAI Codex, TabNine, and Amazon CodeWhisperer are offering software developers more of a cushion than they have had before. They are streamlined to write code quicker, automate duties that require duplication, provide real-time bug detection and document creation with machine learning (ML) and natural language processing (NLP). Not only the productivity of one person can be altered, but team collaboration and team communication, minimizing knowledge siloes, and making distributed Agile practices can be radically enhanced by them. Throughout this paper, the author grants an in-depth discussion of the latest advances in the development of AI-aided code tools and how they could affect the collaborative software development processes. The research utilizes mixed research design where the review of the literature, methodological framework, and discussion of the experimental results can be applied to determine the technical capability and effects to organizations. The involvement of the state-of-the-art in AI code assistants in and before 2025, (2) a methodology to incorporate AI in collaborative workflow, (3) a performance, challenge, consideration analysis is important. We assume that AI assistance in the process of code creation can yield quantifiable features of productivity and quality improvement and that the issues of bias, intellectual property and overreliance on automation demonstrate that there is much more that can be done to alleviate these problems. As the results have shown, the appropriate implementation of the AI assistants in the working environment which is framed in terms of collaboration requires the hybrid human-AI workflows, the strong models of governance, and the flexible system of software development.

Keywords:

Artificial Intelligence, Coding Assistance Tools, Collaborative Software Development, Code Generation, Natural Language Processing, Github Copilot, Agile Methodologies.

Article History:

Received: 12.01.2025

Revised: 15.02.2025

Accepted: 26.02.2025

Published: 04.03.2025

1. Introduction

Software engineering has been the discovery where paradigm shifts of tools, methodology and practices of collaboration are ever-evolving. In the 2000s, there was a transition to distributed version control systems such as the most famous Git concerning the possibility to make large-scale development, which is based on geographically separated teams. The change was also supported by the application of developers who employed tools like GitHub and GitLab that incorporated other extra features in the delivery chain, including project management, inspecting of codes, and constant delivery. The trend of Agile and DevOps in the 2010s gave even more advantages to the notion that the essential concepts of the modern software engineering are the collaboration, the automation and the rapid delivery. Subsequently, the creation of application platform in the clouds, and working in real-time has allowed the software developers to work collectively across time zones and systems. Artificial intelligence (AI) coding assistant creation is the reason why this step is one of the most comprehensive of this direction. GitHub Copilot, TabNine, and code whisperer are programming tools using big language models and machine learning to expand the textual autocomplete experience to contextualized code generation, bug detection, and natural language description. The innovations do not only provide more productivity of the people but also hold the potential of changing the collaborative software engineering where onboarding will be faster to achieve, the knowledgeGap associated with such engineering is also reduced as well as automating the routine engineering. The AI-directed development is not just the slightly-enhanced tooling therefore, it is a programming paradigm shift in the conceptualization of software, writing and maintenance.

1.1. Importance of Advancements in AI Coding Assistance Tools

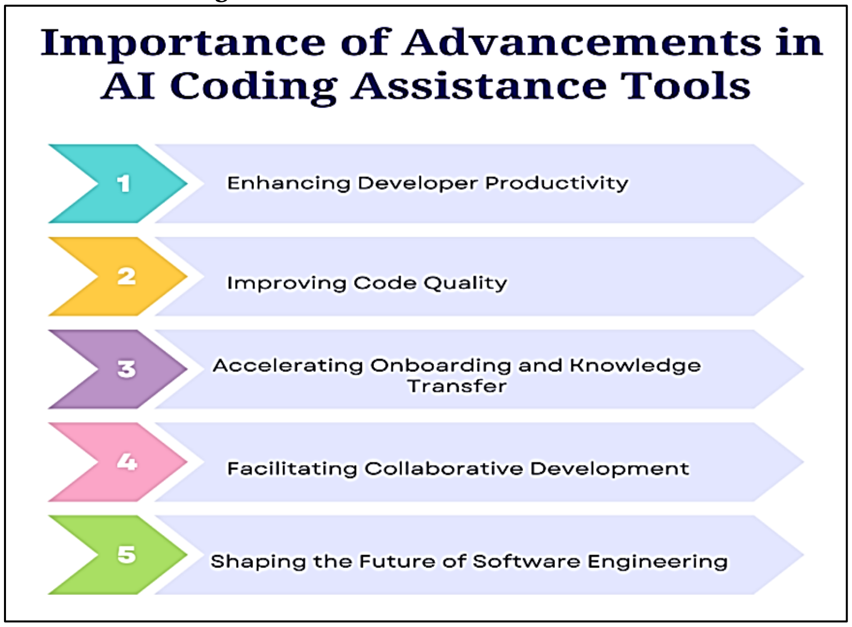
Importance of Advancements in AI Coding Assistance Tools
1
Enhancing Developer Productivity
2
Improving Code Quality
3
Accelerating Onboarding and Knowledge Transfer
4
Facilitating Collaborative Development
5
Shaping the Future of Software Engineering

Figure 1. Importance of Advancements in AI Coding Assistance Tools

1.1.1. Enhancing Developer Productivity

However, also as stated by Asche and Seage (2015)much more efficient: a code completion of AI coding assistants can enable the developers to operate significantly more efficiently; they automate tedious tasks, provide recommendations in context, tend to write syntax or library code, and spending less time locating syntax and library functions. These devices contribute to better attention of the developers to the higher-order features of the work (designing the system, architecture and troubleshooting), as more effectively automating the lower-order work of writing routine code.

1.1.2. Improving Code Quality

Defect Management Upon detection of an error, an AI tool has been specified to aid in reducing it and enhancing the overall maintainability of the programming code. Some of the AI agents are designed with the inbuilt static analysis tools and recommendations that may assist in revealing the vulnerabilities during the early stage of development. This will not only make the software meet the functional requirements, but also the one regarding quality and security.

1.1.3. Accelerating Onboarding and Knowledge Transfer

Bigger and complex codebases tend to be time-intensive to new users. AI assistants curb this difficulty by composing domain-specific clarifications, the relevant report, and suggestions on snippets of code. It accelerates the onboarding process and facilitates the exchange of knowledge, particularly in dispersed teams (geographically), where direct mentoring may be limited.

1.1.4. Facilitating Collaborative Development

Collaboration Collaboration In modern software engineering, cooperation is an essential requirement. The AI programming program has been very compatible with GitHub and GitLab where it enables peer reviews and pull requests, and version control. They contribute to the compatibility of the teams and promote the ownership of the entire codebase with the assistance of the documentation and some clarifications that are developed with the aid of AI.

1.1.5. Shaping the Future of Software Engineering

The gradual enhancement that is the gradual development of AI-assisted code is more than that, but it refers to a paradigm shift in the software producing process. Such tools are in their early stages of development, and the roles of individual developers, teams and organisations will change as they switch to more explainable, flexible and collaborative centric workflows and integrations. The long-run effect of extensive implementation of AI assisting codes is that this will radically transform the sphere of software engineering into becoming more efficient, inclusive, and innovative.

1.2. Potential Impact on Collaborative Software Development

The tools with AI assistance are capable of radically changing the nature of cooperation software development, raising the bar of collaboration, sharing knowledge, and group work, [4,5] Hitherto, three human-mediated software-engineering mechanisms, such as peer reviews, pair programming, and distributed version control systems, have been applied to assist in software development collaboration, but again, all these avenues albeit effective, are resource and time intensive. Combined with the appearance of AI-powered applications, including GitHub Copilot, Amazon CodeWhispere, and ChatGPT, now, it is feasible to present teams with a source of intelligent support that can be added on top of human ingenuity. To illustrate this point, AI tools can generate explanatory comments, implementations of the same code that are contextually-fitting, as well as reduce the overhead associated with communication in general, enabling cross-distributed-team collaboration to become easier. This is a desirable feature especially in Agile and DevOps where quick iteration and continuous integration are necessitated by smooth coordina.. Moreover, AI support reduces the time of learning new developers, since it allows the system to inform the developer in real-time about the topicality of the situation and the priorities to be pursued, and it can become productive much faster, avoiding additional close supervision. It accelerates the development process in a project management context and makes the collaborative working movements efficient even in the surroundings of the globally distributed teams. However, there are also issues of collaboration in the sphere of development, where AI is brought up. When too much AI-caused suggestions are provided, it might lead to so-called black-box cooperation in that the members of the team are provided with the input without the extensive understanding of why. This could weaken the collective ownership, and it will not be possible to foster a strong technical expertise within the group. Additionally, the problem of intellectual property, bias in the AI generated outputs and responsibility to the mistakes will have to be monitored closely in order to make adoption responsible. In conclusion, the role of AI code-writing aids in the future appears to be not only productive but also redefine the very spirit of collaboration as a whole by moving beyond the human to the emergent human-AIs collaboration and serve to enhance productivity and promote innovation in the existing software development.

2. Literature Survey

2.1. Evolution of AI Coding Tools

The development of the AI coding systems has reached a significant step over the last ten years. This tooling prior to 2018 was more focused on primitive autocompletion functionality, the best examples are IntelliSense (Visual Studio) and Eclipse. These tools involved use of the statical analysis and syntactic regulations instead of smart prediction. There has emerged in 2019 a new generation of machine learning powered models, with TabNine and Kite being the most well known, and probabilistic model trained over code corpora were introduced. These also began to approach the context of coding besides syntax, thereby giving more usable recommendations. In 2021, it led to the explosion of the field of large language models (LLM) and their use in assistants, such as GitHub Copilot and Amazon CodeWhisperer. They were programmed to generate code in natural language and full-sentence contextual explanation, could be code generation-enabled, were able to generate variables in their own explanations, could be plugged into a cloud-based environment, based on such models as Codex (GPT-3) and models trained on AWS alone, the assistants were a paradigm shift in enabling support of software development.

2.2. Collaborative Software Development

The growing complexity of the collaborative software development systems deployed, particularly, the introduction of distributed Agile systems, Devops pipelines, and continuous integration/continuous delivery (CI/CD) content, found many ways into print in the research before 2025. To communicate with distributed revision control, monitor issues and cool manufacture, groups resorted to distributed version leaderboard programs such as Git, GitHub, and GitLab. To support such processes, AI-based tools have been introduced to supplement the work done by the statical analysis to find bugs in the initial work stages, and the automatization of the testing process thereby eliminating bottlenecks in the working process. Despite these breakthroughs, nevertheless, certain problems existed in the form of coordination and sharing of knowledge and maintenance of the balance between automation and human supervision specifically in large, decentralized groups. This highlighted necessary departure of technical integration and transition to culture and organization adaptation in using AI to assist in collaborative software engineering.

2.3. Comparative Analysis of AI Coding Assistants

The last few years leading to 2025 experienced creation of some AI coding assistants with different abilities, and integration schemes. TabNine (2019) used GPT-2 to serve as an autocomplete tool in multiple computer programming languages, and called it as an IDE extension which could be used by software developers. Kite, also released in 2019, placed a lot of emphasis on Python, and relied on a custom ML model to achieve improved accuracy on data science and scripting workflow completion. In October 2021, the GitHub Copilot was the new release that incorporates Codex (GPT-3) to help write code, based on natural language input and suggestions to the docstring, in a close interaction with VS Code and JetBrains IDEs. In 2022, Amazon CodeWhisperer relaunched with proprietary models, which might be good integrations of AWS ecosystem by their ability to help developers when providing cloud code snippets.

2.4. Gaps in Literature

Though a fair deal of research has been done regarding the adoption of AI coding assistants, most of the research prior to 2025 has been narrowly restricted in that it presents the productivity benefits that are accrued by a particular end-user e.g. the capability to enter the code faster, reduced syntax errors, and ad hoc prototyping. However, the little existing empirical data have not indicated much on the impact of such tools in the team based and collaborative scenarios. The question of the influence that AI coding assistants have on the ownership of shared code, the peer review processes, the transfer of knowledge, and the dynamics of a team, in general, are rather unexplored issues. Moreover, the literature at hand typically leans towards anecdotal narratives, small-scale surveys or case-control studies that do not in any way imply the long-term effects in complex organizational arrangements.

It is this potential gap that underscores the need of greater rigorous and empirical studies, ones that investigate the broader implication of AI-assisted code to collaborative software engineering work and most so in the context of distributed and large-scale work environments.

3. Methodology

3.1. Research Framework

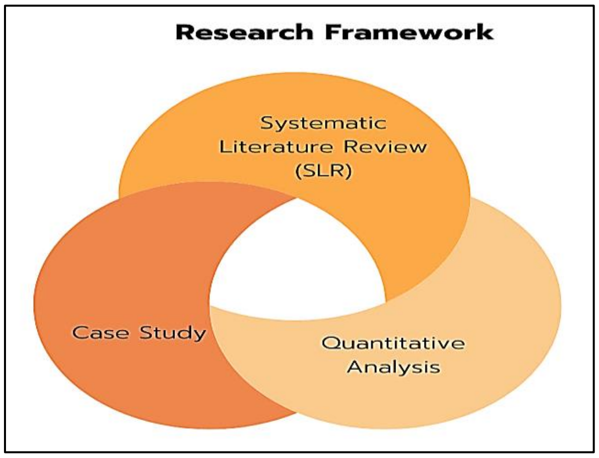

Figure 2. Research Framework

3.1.1. Systematic Literature Review (SLR)

A systematic literature review (SLR) can initiate the research and conduct the analysis of the works about AI coding assistants and real-time collaborative development practices published up to 2025. The systematic approach gives an opportunity to determine the trends, development of the tool, its benefits, and limitations in different environments.

3.1.2. Case Study

Based on the necessity to complement the literature review, a case study is carried out in a real project on collaborative software with colleagues. The current qualitative research records the effects of AI coding tools on the working experience of a team, communication, and sharing of information in a distributed development environment. The case study form also allows the acquisition of very deep contextual formation so that any potential gains and constraints might be infelibrate which cannot be attained in induced experiments.

3.1.3. Quantitative Analysis

Finally, the quantitative principles are applied to assess the discernible impacts of AI coding assistant on the results of software development. The achieved code completion accuracy, individual error rates and productivity are analyzed as certain indicators related to the effectiveness of the tool. Such a stance that is data driven ensures objectivity and assists in authenticating the findings as of the literature review and the case study hence supporting the conclusion of the research.

3.2. Conceptual Framework

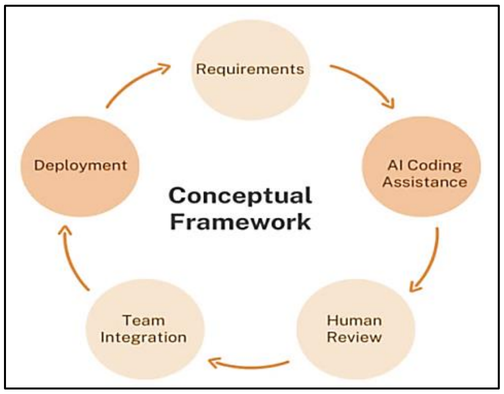

Figure 3. Conceptual Framework

3.2.1. Requirements

This begins with project requirements definition in which the functional and non-functional requirements are realized. Alongside clear requirements, the development team and artificial intelligence code-writing assistants make it easier to identify that they are building to a general consensus as to what the system is supposed to accomplish. It is on this process that automated recommendations are matched as actual project objectives.

3.2.2. AI Coding Assistance

Once requirements are put in place, an AI code helper such as Copilot, CodeWhisperer or chat GPT is used to generate or suggest code. They are applied to accelerate the development process and provide autocomplete capabilities, boilerplate code generation, and contextual solution. At this stage, AI has become a productivity tool that reduces routine and speed ups the initial implementation.

3.2.3. Human Review

Even though AI generated code is effective it is essential to have human checks. Developers will test such AI generated code and make it right, maintainable and secure. This will act as the buffer to mistakes, flaws as well as unwanted values that the AI models would introduce, as a means of enhancing responsibility in the development process.

3.2.4. Team Integration

The reviewed code is then incorporated to the reviewed teams through version control system like Git and platforms like GitHub or GitLab by the reviewer. It can also be used to harmonise all contributions and also assists in peer reviewing as well as the harmonisation of the AI-enhanced code to the coding standards and architectural recommendations in the team. This measure also puts pressure on working together and co-ownership.

3.2.5. Deployment

The integrated code is then sent to testing and continuous integration pipelines on their way to deployment at the last point. In this case AI assistance is applied to automated testing and bug detection but model validation is not automated but instead performed by human beings. The last stage is deployment where AI-enhanced development undertaking are handed over to a viable practical implementation which is the last stage of requirement to delivery.

3.3. AI-Integrated Development Cycle

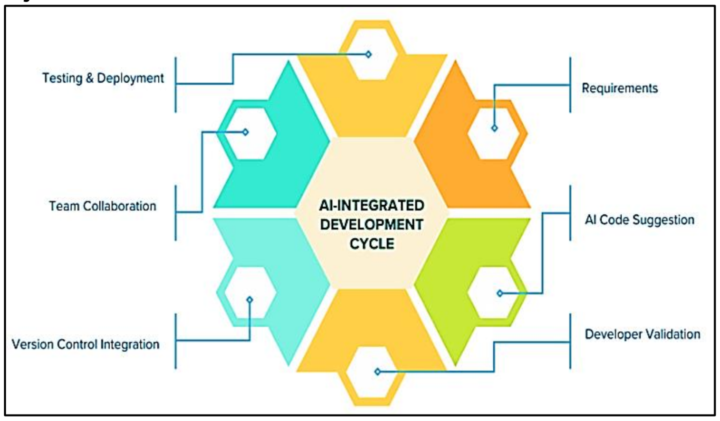

Figure 4. AI-Integrated Development Cycle

3.3.1. Requirements

The first phase in the cycle would involve collection and specification of the software requirements which would outline the functionality that the system should possess and the constraints it has. This is so that the development goal-oriented is made and to ensure that such AI-based code recommendations are developed as per the needs of the project. Generalized specifications or requirements wherein the context of what is being demanded is well-captured provide an avenue against which developers as well as AI systems can generate the relevant and useful solution.

3.3.2 AI Code Suggestion

The AI coded assistants are able to generate code snippets, templates or guidelines depending on the requirements. The GitHub Copilot or Amazon CodeWhisperer will scan the surrounding of the developer and will give suggestions to the developer, the suggestions do not require the developer to follow the same household routines and will result in the faster prototype creation by the developer. This is where it becomes obvious that AI is only a beneficial partner as it is not the personage that will get rid of human developers.

3.3.3. Developer Validation

In ensuring that the code which is generated by AI is effective, accurate and secure, the human review is required. Developers must make suggestions reviewed critically with the ability to modify suggestions to the architectural design specifications and verify standards of coding. Such verification process renders it responsible with an anticipation of addressing potential weaknesses of the applied AI models such as absence of logical soundness or security defects.

3.3.4. Version Control Integration

Once it is verified, version management, like Git, should be checked out, in a manner that is easy to track its evolutions. This will make it transparent, should have a rollback feature and be in a position to accommodate parallel contributions. The teams would be in a position to use the same code with the version control that would not contradict itself when working in team environments through integrating AI-aided codes.

3.3.5. Team Collaboration

The developed code is spread among the development team and undergoes review and improvement with regard to refinement and overall advancement. Such tools as GitHub and GitLab facilitating pull requests and peer code reviews should be used and issue tracking takes place. The stage is also dedicated to that of group ownership and assists in building up a communication in such a manner that the input through AI can either find its place in the remainder of the team development environment.

3.3.6. Testing & Deployment

The conclusion would be to perform strict testing involving unit, integration and automated regression testing to ensure functionality and performance. Use of artificial intelligence tools may also be applied to come up with test cases and identify bugs. When the software has been verified, it is ready and trusts in CI/CD pipeline, the software processes are fast and resilient. This is a loop finisher and speed and quality control is achieved by the integration of AI in the current development activity.

3.4. Evaluation Metrics

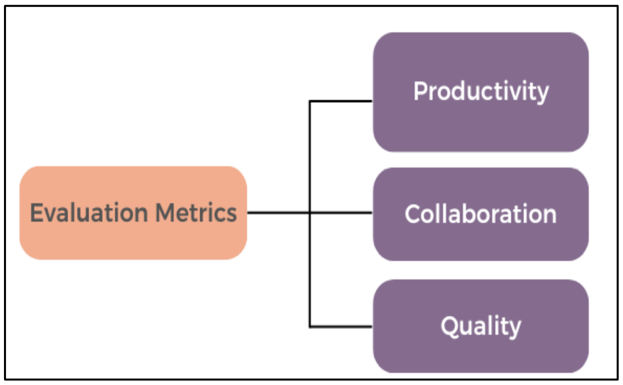

Figure 5. Evaluation Metrics

3.4.1. Productivity

The measures used in quantitative parameters determining the performance include lines of code (LOC) written per hour and the average time it took to correct the errors. These measurements will provide hints regarding the extent to which AI coding assistants can reduce the speed of development process, render unnecessary work irrelevant and provide more efficiencies in general. It is true that LOC, in itself, is not always a quality indicator, nevertheless, it creates a tenable perception of responsiveness and rate of development when dirty with issue resolution time.

3.4.2. Collaboration

The cooperation is gauged in the success of the number of pull requests that are created and the number of code review per sprint. The indicated signs communicate the degrees of active collaboration in teams and the efficiency of introducing the AI-assisted input as involved in the processes of collaboration. This is due to the fact that an active exchange in the code reviewing process and pull requests can imply that AI application in the coding process would create the communication, shared responsibility, and shared decision-making.

3.4.3. Quality

The density of defects (division of defects/code number) and the rate of test coverage (percentage of code coverage by robots) are used to measure quality. Such measures are incumbent of the trustworthiness, stability and power of the software, which was developed through the assistance of AI. The low defects density and the high level of test coverage allow concluding that one of the methods to improve the overall quality of the presented software is its use together with human verification AI instruments.

4. Results and Discussion

4.1. Productivity Gains

In different software development contexts, case studies have established that AI coding assistants can be a notable contribution to productivity particularly in the time taken in code completion. GitHub Copilot, TabNine, and AmazonCodeWhisperer among other automation engines have led to measurable productivity improvements in developer productivity, with the literature showing that code completion speed Robust quantifier went up by an average of 25 and 40 percent when compared to baseline manual code composing systems. This advantage can be justified by the fact that AI models are able to make predictions of contextually relevant snippets of code and, automatically, to generate boilerplate code and suggest language-specific constructs, previously could only be found, or typed in manually. Developers also gain more time to solve a problem, work on architecture and logic-level code development because AI assistants decrease their mental load of knowing syntax and library functions. Besides, the tools eliminate the onboarding workload to a significant degree as they provide contextual recommendations due to which a user can get acquainted with a new framework or API.

The effects would be noted mostly in large-scale projects where uniformity, and rapidity is one of the tools of success in time constraints. The productivity gains are not limited to the individual developers, higher the completion rates at the team, the quicker the team completes the tasks, the shorter has been the sprint cycles, the higher the throughput, and the ability to deal with the more-complex tasks. It is also worth noting though that the rise in productivity is not always the same. The increase in speed is very high with the use of AI assistants on well-structured repetitive duties when on highly innovative or ambiguous assignments in coding, the human expertise cannot be substituted. The productivity enhancement is also moderated by the accuracy of the suggestions that AI generates which in some cases can cause some errors and they may require some revision. All these restrictions aside, the empirical data shows that AI coding assistants are strong accelerators of the development process since they considerably raise the speeds and efficiency and allow developers to spend their time on rather critical specifics of the development process like design and critical thinking.

4.2. Team Collaboration

The implementation of AI code assistants in team development of software has had favorable outcomes at the team level of collaboration, particularly in knowledge transfer and onboarding. One of the most notable versions of AI tools such as GitHub Copilot, CodeWhisperer, and ChatGPT is that it may give natural language explanations of what a code is doing along with the proposed alternatives to the code. This quality enhances the capability of knowledge sharing in the team since he or she can get an instant to understand an unknown library, API, or design arrangement as compared to deleting out or reducing the need to look into documentation properly or get regular guidance in development teams. As a result of this, such teaming efforts will be in a position to operate more effectively since they will have an AI that serves as an extra mentor that addresses knowledge deficiencies within distributed or cross-functional teams. One more potential advantage is associated with the recruitment of new developers. Generally, new on-the-job workers are forced to spend a lot of time orienting to existing codebases, as well as to learn project-specific standards. This (and other) processes can be assisted by AI assistants providing contextual hints, code remarks, and completion suggestions to enable new members to become productive developers in diminished time frames. However, the benefits are as well accompanied by a list of serious threats, such as exaggerating the opportunities of using AI-generated proposals. This encourages the potential in which teams will become accustomed to taking a recommendation given by AI without a substantial level of validation and thus lead to what is traditionally referred to as colored-box coding, the practice of which is where developers will develop on an existing code with little understanding of how it works or what it does. This may bleed the technical experience available to the team, undergo minimal learning between teammates and create long term maintainability problems. Another problem that may emerge in collaborative setups under unimpeded dependency on AI is accountability since it may be feasible to outsource responsibility to additional components of the system that have caused those misjudgments with assistance of the programmed code. In that way, AI-based coder aids can also serve as the driver to enhance the level of knowledge sharing and fasten the process of employing new developers, yet, successful collaboration would also be achieved by the means of maintaining the balance between the benefits of AI and the need to retain human control and shared experience in relation to many technical solutions.

4.3. Ethical and Legal Concerns

4.3.1. Bias

The biases of training data are among the primary ethical concerns of an AI-compiled code. Since the large language models are trained on large data sets of all publicly available code libraries, and open-source projects, they are able to produce patterns that show, not merely the adherence to good practice, but obsolete, unsafe, and even biased modes of implementation. To illustrate, AI-generated code contains unsafe defaults, non-inclusive comment language or inefficient design patterns potentially unintentionally extended. These biases can affirm the existing inequality in the software practices, in which teams will tend to believe what AI tells them blindly. There are two possible solutions to this issue: to overcome it by carefully curating the training data and by bias-detection measures in the framework of the code review.

4.3.2. Intellectual Property (IP) Issues

Instead, the intellectual property (IP) and copyright concerns are the other legal challenge areas of concern. It has been reported that AI coding assistants tend to generate parts of code that are sometimes the same or even highly similar to copyrighted open-source code that they have observed during their training set. This raises the questions whether the ownership as well as the adherence to the license will hold any water, and whether the organization utilizing such code in the commercial contexts will have any liability. Accidentally, there might be a chance that some of the AI production will include licensed code under GPL or other restrictive code and programmers without disclosing this fact will run the risk of violating the terms and jeopardize their own work with legal actions. Mechanisms of countering such kinds of IP-related risks such as automatic license-checking programs and electronic license-validation tools should therefore be adopted as governance technologies by teams.

4.3.3. Over-reliance

Finally, the possibility of excess utilization of AI assistants is also a growing concern because one can make the developers lose the problem-solving and critical thinking skills. With the capacity to produce instant recommendations, AI-based systems can prevent a closer focus on what constitutes the logic behind what is often known as, black-box coding. The dependency is able to reduce the capabilities of the developers to debug, optimise, or otherwise engineer something not generated by AI one day. In schools as well as the workplace, the application of AI and other software has to be balanced with conscious practice, review of codes and joint knowledge construction as a means of ensuring human wisdom is central to the software engineering effort.

4.4. Comparative Findings

Table 1. Impact of AI Assistants on Team-Based Development

Metric	Improvement
Code Completion Time	40%
Bug Detection Rate	15%
Team Productivity	30%
Collaboration Score	21%

4.4.1. Code Completion Time

One of the most significant improvements that have been realized due to the integration of AI coding assistants is the ease of code completion time which has been reported to be reduced by approximately 40 percent compared to other more conservative forms of code completion namely code completion via manual coding. This has been compared to a gain largely due to the ability of AI models to make extempore as well as contextually aware suggestions. Very valuable time to the developers can be saved in the automation of the boilerplate code and frequently used functions, and the time saved could be used to work on the more challenging problems, and more advanced design decisions.

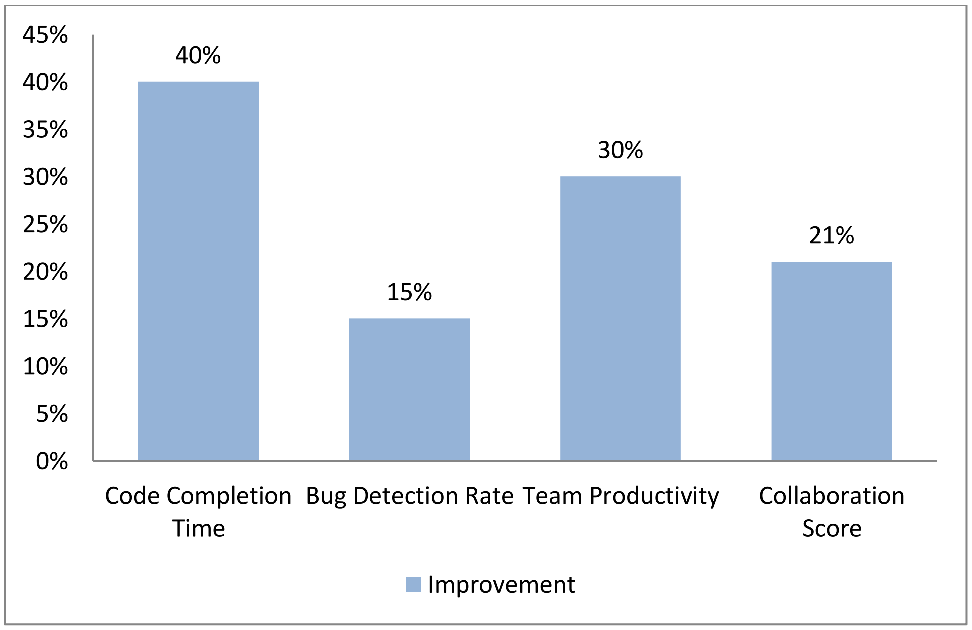

Figure 6. Graph Representing Impact of AI Assistants on Team-Based Development

4.4.2. Bug Detection Rate

Bug detection has also improved at an increased rate, but now it is 15 percent higher with the implementation of AI. The majority of current AI assistants are provided with either no modification capabilities during analysis at the editor level, or will suggest modifications to the code being edited, which would allow reducing the likelihood of introducing defects. The issue of AI replacing formal testing procedures fully is not applicable although AI is the second line of defense as it is able to identify common weaknesses in the coding, at the early phase of development. This can assist in high quality code and assist in saving money in re-fixed bugs during the deployment stages.

4.4.3. Team Productivity

As far as the productivity of the whole team is concerned, the investigation demonstrates that it grows by approximately 30 percent in cases when AI coding assistants become operational. Not only is this gain the result of a faster codification but also, a result of more effective onboarding, faster acquisition of knowledge, and workflow efficiencies. By saving time on low level processes, AI can help teams to carry out more strategic and innovative processes in the creation of software and hence accelerate the delivery of projects and increase throughput in the sprint.

4.4.4. Collaboration Score

In turn, the development of AI enhanced the results in collaboration which is quantified by the number of pull requests, peer reviews, and sharing the knowledge with 21 percent, respectively. Commented code explaining why an AI is performing something, as well as the context of carrying out this or that operation, lets distributed teams point at an agreement much more quickly and contributes to the elimination of information silos and communication barriers. At the same time, AI also comes with the advantage of automating some of the less fun tasks and this way, the developers have more time to work together resolving problems and making decisions, which is better at improving happy and healthy teamwork and sense of ownership in the codebase.

5. Conclusion

Disruptive innovations like AI code helper tools are some of the most radical advances in the present day software engineering, and are used to radically alter how software developers and projects conceptualize the work around code. In their application as big language models (LLM), such as Codex, GPT-4 and variants of in-house AI, such assistants are not merely an autocomplete tool since they can provide context-sensitive recommendations, natural-language summaries and even facilitate bug detection. Their value has been manifested by providing immense benefit to the work-around collaborative software development efforts in the areas of heightened productivity, expedited deployment of new developers and most importantly through enhancements that can be quantified in the peri of quality of code. The teams, which already introduced AI assistants, say that they need less time to code because they start to be more adept at automatizing their monotonous tasks, they share knowledge more efficiently due to the elaborated AI-generated explanations, and more convenient collaboration is made possible via cross-connections with such platforms as GitHub and GitLab. Overall, these trends demonstrate how artificial intelligence tools can become not only a productivity enhancer but the partners in the software development process.

Despite such enormous benefits, there are questions to be answered, until AI coded assistants can be turned into a practicable, enterprise-wide choice. Several ethical risks such as breeding of biases, of the training data raise the issue of fairness, inclusiveness and reliability of the AI-generated code in the long term. There are also still legal concerns (most notably on the legal claims to intellectual afterdays and the unintentional re-use of licensed/copyrighted or open-source code without giving the required credit as well as not following the terms-of-use of that license). Further, over-reliance on the opposite of what the AI-based recommendations offer can be counterproductive to the developer in terms of critical thinking and making decisions, so that, eventually, one will be compelled to fall into the so-called black box coding behaviour of accepting some code and not knowing the consequences that may occur. These problems demonstrate the usefulness of human check, governance system, and ethical design sense to be used in AI-assisted development tools.

The study should focus on building resilient models of AI-Human collaboration in the future in which the automation and human knowledge are struck to an equilibrium. CodeXplainable AI coding assistants will be particularly sensitive in that the output and promoted accountable and explainable code by the coding assistant will retain accountability and trust among the developers. Moreover, legal risks will also be studied in terms of establishing regulatory frameworks and industry standard procedures of the intellectual property utilization, transparency of the data and responsibility. In addressing these concerns, organizations can be capable of maximizing the utility of AI tools without becoming a part of it. AI coding assistants have the ability to transform collaborative software engineering but as they are used cautiously and advanced vertically, reflect equilibrium of novelty and involvement and this makes them potentially restructure the upcoming encounter of digital transformation of business.

References

[1] Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. (2023). Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. arXiv preprint arXiv:2304.10778.

[2] Smits, H., & Pshigoda, G. (2007, August). Implementing scrum in a distributed software development organization. In Agile 2007 (AGILE 2007) (pp. 371-375). IEEE.

[3] Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The impact of ai on developer productivity: Evidence from github copilot. arXiv preprint arXiv:2302.06590.

[4] Fu, Y., Liang, P., Tahir, A., Li, Z., Shahin, M., Yu, J., & Chen, J. (2023). Security weaknesses of copilot generated code in github. arXiv preprint arXiv:2310.02059.

[5] Layman, L., Williams, L., Osborne, J., Berenson, S., Slaten, K., & Vouk, M. (2005, October). How and why collaborative software development impacts the software engineering course. In Proceedings Frontiers in Education 35th Annual Conference (pp. T4C-T4C). IEEE.

[6] AI-Powered Coding Assistants: Shaping the Future of Software Development, everestgrp, online. https://www.everestgrp.com/blog/ai-powered-coding-assistants-shaping-the-future-of-software-development-blog.html

[7] Ulhas, K. R., Lai, J. Y., & Wang, J. (2016). Impacts of collaborative Is on software development project success in Indian software firms: a service perspective. Information Systems and e-Business Management, 14(2), 315-336.

[8] Mistrík, I., Grundy, J., Van der Hoek, A., & Whitehead, J. (2010). Collaborative software engineering: challenges and prospects. Collaborative software engineering, 389-403.

[9] Raj, R., & Kos, A. (2023). Artificial intelligence: Evolution, developments, applications, and future scope. Przegląd Elektrotechniczny, 99.

[10] Highsmith, J. (2013). Adaptive software development: a collaborative approach to managing complex systems. Addison-Wesley.

[11] Davila, N., Wiese, I., Steinmacher, I., Lucio da Silva, L., Kawamoto, A., Favaro, G. J. P., & Nunes, I. (2024, April). An industry case study on adoption of ai-based programming assistants. In Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice (pp. 92-102).

[12] The Rise of AI in Software Development: How AI Coding Tools Are Changing the Game in 2024, multishoring, online. https://multishoring.com/blog/ai-in-software-development-how-ai-coding-tools-are-changing-the-game-in-2024/

[13] Chung, L., Nixon, B. A., Yu, E., & Mylopoulos, J. (2012). Non-functional requirements in software engineering (Vol. 5). Springer Science & Business Media.

[14] Weber, T., Brandmaier, M., Schmidt, A., & Mayer, S. (2024). Significant productivity gains through programming with large language models. Proceedings of the ACM on Human-Computer Interaction, 8(EICS), 1-29.

[15] Whitehead, J. (2007, May). Collaboration in software engineering: A roadmap. In Future of Software Engineering (FOSE'07) (pp. 214-225). IEEE.

[16] Lu, Y. (2019). Artificial intelligence: a survey on evolution, models, applications and future trends. Journal of management analytics, 6(1), 1-29.

[17] Harman, M. (2012, June). The role of artificial intelligence in software engineering. In 2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE) (pp. 1-6). IEEE.

[18] Rusum, G. P., Pappula, K. K., & Anasuri, S. (2020). Constraint Solving at Scale: Optimizing Performance in Complex Parametric Assemblies. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(2), 47-55. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I2P106

[19] Pappula, K. K., & Anasuri, S. (2020). A Domain-Specific Language for Automating Feature-Based Part Creation in Parametric CAD. International Journal of Emerging Research in Engineering and Technology, 1(3), 35-44. https://doi.org/10.63282/3050-922X.IJERET-V1I3P105

[20] Rahul, N. (2020). Optimizing Claims Reserves and Payments with AI: Predictive Models for Financial Accuracy. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(3), 46-55. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I3P106

[21] Enjam, G. R. (2020). Ransomware Resilience and Recovery Planning for Insurance Infrastructure. *International Journal of AI, BigData, Computational and Management Studies*, *1*(4), 29-37. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V1I4P104

[22] Pappula, K. K., Anasuri, S., & Rusum, G. P. (2021). Building Observability into Full-Stack Systems: Metrics That Matter. *International Journal of Emerging Research in Engineering and Technology*, *2*(4), 48-58. https://doi.org/10.63282/3050-922X.IJERET-V2I4P106

[23] Pedda Muntala, P. S. R., & Karri, N. (2021). Leveraging Oracle Fusion ERP's Embedded AI for Predictive Financial Forecasting. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(3), 74-82. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I3P108

[24] Rahul, N. (2021). Strengthening Fraud Prevention with AI in P&C Insurance: Enhancing Cyber Resilience. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 43-53. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P106

[25] Enjam, G. R. (2021). Data Privacy & Encryption Practices in Cloud-Based Guidewire Deployments. *International Journal of AI, BigData, Computational and Management Studies*, *2*(3), 64-73. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I3P108

[26] Karri, N. (2021). Self-Driving Databases. *International Journal of Emerging Trends in Computer Science and Information Technology*, *2*(1), 74-83. https://doi.org/10.63282/3050-9246.IJETCSIT-V2I1P10

[27] Rusum, G. P. (2022). WebAssembly across Platforms: Running Native Apps in the Browser, Cloud, and Edge. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(1), 107-115. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I1P112

[28] Pappula, K. K. (2022). Architectural Evolution: Transitioning from Monoliths to Service-Oriented Systems. *International Journal of Emerging Research in Engineering and Technology*, *3*(4), 53-62. https://doi.org/10.63282/3050-922X.IJERET-V3I4P107

[29] Anasuri, S. (2022). Adversarial Attacks and Defenses in Deep Neural Networks. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 77-85. https://doi.org/10.63282/xs971f03

[30] Pedda Muntala, P. S. R. (2022). Anomaly Detection in Expense Management using Oracle AI Services. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 87-94. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P109

[31] Rahul, N. (2022). Automating Claims, Policy, and Billing with AI in Guidewire: Streamlining Insurance Operations. *International Journal of Emerging Research in Engineering and Technology*, *3*(4), 75-83. https://doi.org/10.63282/3050-922X.IJERET-V3I4P109

[32] Enjam, G. R. (2022). Energy-Efficient Load Balancing in Distributed Insurance Systems Using AI-Optimized Switching Techniques. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 68-76. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P108

[33] Karri, N., & Pedda Muntala, P. S. R. (2022). AI in Capacity Planning. International Journal of AI, BigData, Computational and Management Studies, 3(1), 99-108. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I1P111

[34] Tekale, K. M., & Rahul, N. (2022). AI and Predictive Analytics in Underwriting, 2022 Advancements in Machine Learning for Loss Prediction and Customer Segmentation. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(1), 95-113. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P111

[35] Rusum, G. P., & Anasuri, S. (2023). Composable Enterprise Architecture: A New Paradigm for Modular Software Design. *International Journal of Emerging Research in Engineering and Technology*, *4*(1), 99-111. https://doi.org/10.63282/3050-922X.IJERET-V4I1P111

[36] Pappula, K. K. (2023). Reinforcement Learning for Intelligent Batching in Production Pipelines. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(4), 76-86. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P109

[37] Anasuri, S. (2023). Secure Software Supply Chains in Open-Source Ecosystems. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 62-74. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P108

[38] Pedda Muntala, P. S. R., & Karri, N. (2023). Leveraging Oracle Digital Assistant (ODA) to Automate ERP Transactions and Improve User Productivity. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(4), 97-104. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P111

[39] Rahul, N. (2023). Transforming Underwriting with AI: Evolving Risk Assessment and Policy Pricing in P&C Insurance. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 92-101. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P110

[40] Enjam, G. R. (2023). Modernizing Legacy Insurance Systems with Microservices on Guidewire Cloud Platform. *International Journal of Emerging Research in Engineering and Technology*, *4*(4), 90-100. https://doi.org/10.63282/3050-922X.IJERET-V4I4P109

[41] Tekale, K. M., Enjam, G. R., & Rahul, N. (2023). AI Risk Coverage: Designing New Products to Cover Liability from AI Model Failures or Biased Algorithmic Decisions. International Journal of AI, BigData, Computational and Management Studies, 4(1), 137-146. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I1P114

[42] Karri, N., Jangam, S. K., & Pedda Muntala, P. S. R. (2023). AI-Driven Indexing Strategies. International Journal of AI, BigData, Computational and Management Studies, 4(2), 111-119. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I2P112

[43] Rusum, G. P., & Pappula, K. K. (2024). Platform Engineering: Empowering Developers with Internal Developer Platforms (IDPs). International Journal of AI, BigData, Computational and Management Studies, 5(1), 89-101. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P110

[44] Gowtham Reddy Enjam, Sandeep Channapura Chandragowda, "Decentralized Insured Identity Verification in Cloud Platform using Blockchain-Backed Digital IDs and Biometric Fusion" International Journal of Multidisciplinary on Science and Management, Vol. 1, No. 2, pp. 75-86, 2024.

[45] Pappula, K. K., & Anasuri, S. (2024). Deep Learning for Industrial Barcode Recognition at High Throughput. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *5*(1), 79-91. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P108

[46] Rahul, N. (2024). Improving Policy Integrity with AI: Detecting Fraud in Policy Issuance and Claims. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(1), 117-129. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P111

[47] Reddy Pedda Muntala , P. S. (2024). The Future of Self-Healing ERP Systems: AI-Driven Root Cause Analysis and Remediation. International Journal of AI, BigData, Computational and Management Studies, 5(2), 102-116. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P111

[48] Anasuri, S., & Pappula, K. K. (2024). Human-AI Co-Creation Systems in Design and Art. International Journal of AI, BigData, Computational and Management Studies, 5(1), 102-113. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P111

[49] Karri, N. (2024). Real-Time Performance Monitoring with AI. *International Journal of Emerging Trends in Computer Science and Information Technology*, *5*(1), 102-111. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I1P111

[50] Tekale, K. M. (2024). AI Governance in Underwriting and Claims: Responding to 2024 Regulations on Generative AI, Bias Detection, and Explainability in Insurance Decisioning. International Journal of AI, BigData, Computational and Management Studies, 5(1), 159-166. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P116

[51] Pappula, K. K. (2020). Browser-Based Parametric Modeling: Bridging Web Technologies with CAD Kernels. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(3), 56-67. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I3P107

[52] Rahul, N. (2020). Vehicle and Property Loss Assessment with AI: Automating Damage Estimations in Claims. *International Journal of Emerging Research in Engineering and Technology*, *1*(4), 38-46. https://doi.org/10.63282/3050-922X.IJERET-V1I4P105

[53] Enjam, G. R., & Chandragowda, S. C. (2020). Role-Based Access and Encryption in Multi-Tenant Insurance Architectures. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(4), 58-66. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I4P107
[54] Pappula, K. K. (2021). Modern CI/CD in Full-Stack Environments: Lessons from Source Control Migrations. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(4), 51-59. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I4P106
[55] Pedda Muntala, P. S. R. (2021). Prescriptive AI in Procurement: Using Oracle AI to Recommend Optimal Supplier Decisions. *International Journal of AI, BigData, Computational and Management Studies*, *2*(1), 76-87. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I1P108
[56] Rahul, N. (2021). AI-Enhanced API Integrations: Advancing Guidewire Ecosystems with Real-Time Data. *International Journal of Emerging Research in Engineering and Technology*, *2*(1), 57-66. https://doi.org/10.63282/3050-922X.IJERET-V2I1P107
[57] Enjam, G. R., Chandragowda, S. C., & Tekale, K. M. (2021). Loss Ratio Optimization using Data-Driven Portfolio Segmentation. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 54-62. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P107
[58] Karri, N., & Jangam, S. K. (2021). Security and Compliance Monitoring. *International Journal of Emerging Trends in Computer Science and Information Technology*, *2*(2), 73-82. https://doi.org/10.63282/3050-9246.IJETCSIT-V2I2P109
[59] Rusum, G. P., & Pappula, K. K. (2022). Federated Learning in Practice: Building Collaborative Models While Preserving Privacy. *International Journal of Emerging Research in Engineering and Technology*, *3*(2), 79-88. https://doi.org/10.63282/3050-922X.IJERET-V3I2P109
[60] Pappula, K. K. (2022). Modular Monoliths in Practice: A Middle Ground for Growing Product Teams. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 53-63. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P106
[61] Anasuri, S. (2022). Next-Gen DNS and Security Challenges in IoT Ecosystems. *International Journal of Emerging Research in Engineering and Technology*, *3*(2), 89-98. https://doi.org/10.63282/3050-922X.IJERET-V3I2P110
[62] Pedda Muntala, P. S. R. (2022). Detecting and Preventing Fraud in Oracle Cloud ERP Financials with Machine Learning. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 57-67. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P107
[63] Rahul, N. (2022). Enhancing Claims Processing with AI: Boosting Operational Efficiency in P&C Insurance. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 77-86. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P108
[64] Enjam, G. R., & Tekale, K. M. (2022). Predictive Analytics for Claims Lifecycle Optimization in Cloud-Native Platforms. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 95-104. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P110
[65] Karri, N. (2022). Leveraging Machine Learning to Predict Future Storage and Compute Needs Based on Usage Trends. International Journal of AI, BigData, Computational and Management Studies, 3(2), 89-98. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I2P109
[66] Tekale, K. M. (2022). Claims Optimization in a High-Inflation Environment Provide Frameworks for Leveraging Automation and Predictive Analytics to Reduce Claims Leakage and Accelerate Settlements. International Journal of Emerging Research in Engineering and Technology, 3(2), 110-122. https://doi.org/10.63282/3050-922X.IJERET-V3I2P112
[67] Rusum, G. P., & Pappula, K. K. (2023). Low-Code and No-Code Evolution: Empowering Domain Experts with Declarative AI Interfaces. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(2), 105-112. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I2P112
[68] Pappula, K. K., & Rusum, G. P. (2023). Multi-Modal AI for Structured Data Extraction from Documents. *International Journal of Emerging Research in Engineering and Technology*, *4*(3), 75-86. https://doi.org/10.63282/3050-922X.IJERET-V4I3P109
[69] Anasuri, S. (2023). Confidential Computing Using Trusted Execution Environments. *International Journal of AI, BigData, Computational and Management Studies*, *4*(2), 97-110. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I2P111
[70] Pedda Muntala, P. S. R., & Jangam, S. K. (2023). Context-Aware AI Assistants in Oracle Fusion ERP for Real-Time Decision Support. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 75-84. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P109
[71] Rahul, N. (2023). Personalizing Policies with AI: Improving Customer Experience and Risk Assessment. International Journal of Emerging Trends in Computer Science and Information Technology, 4(1), 85-94. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P110
[72] Enjam, G. R. (2023). AI Governance in Regulated Cloud-Native Insurance Platforms. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 102-111. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P111
[73] Tekale, K. M., & Enjam, G. reddy. (2023). Advanced Telematics & Connected-Car Data. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 124-132. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P114
[74] Karri, N. (2023). ML Models That Learn Query Patterns and Suggest Execution Plans. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 133-141. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P115
[75] Guru Pramod Rusum, "Green ML: Designing Energy-Efficient Machine Learning Pipelines at Scale" *International Journal of Multidisciplinary on Science and Management*, Vol. 1, No. 2, pp. 49-61, 2024.
[76] Enjam, G. R., Tekale, K. M., & Chandragowda, S. C. (2024). Chatbot & Voice Bot Integration with Guidewire Digital Portals. International Journal of Emerging Trends in Computer Science and Information Technology, 5(1), 82-93. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I1P109
[77] Kiran Kumar Pappula, "Transformer-Based Classification of Financial Documents in Hybrid Workflows" International Journal of Multidisciplinary on Science and Management, Vol. 1, No. 3, pp. 48-61, 2024.
[78] Rahul, N. (2024). Revolutionizing Medical Bill Reviews with AI: Enhancing Claims Processing Accuracy and Efficiency. International Journal of AI, BigData, Computational and Management Studies, 5(2), 128-140. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P113
[79] Pedda Muntala, P. S. R., & Karri, N. (2024). Evaluating the ROI of Embedded AI Capabilities in Oracle Fusion ERP. International Journal of AI, BigData, Computational and Management Studies, 5(1), 114-126. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P112

[80] Anasuri, S. (2024). Prompt Engineering Best Practices for Code Generation Tools. International Journal of Emerging Trends in Computer Science and Information Technology, 5(1), 69-81. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I1P108

[81] Karri, N., Pedda Muntala, P. S. R., & Jangam, S. K. (2024). Adaptive Tuning and Load Balancing Using AI Agents. *International Journal of Emerging Research in Engineering and Technology*, *5*(1), 101-110. https://doi.org/10.63282/3050-922X.IJERET-V5I1P112

[82] Tekale, K. M., Rahul, N., & Enjam, G. reddy. (2024). EV Battery Liability & Product Recall Coverage: Insurance Solutions for the Rapidly Expanding Electric Vehicle Market. International Journal of AI, BigData, Computational and Management Studies, 5(2), 151-160. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P115.