BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

AI-augmented Software Development Lifecycle: Improving Planning, Coding, Testing and Deployment

Atharva Kale
Department of Artificial Intelligence and Data Science
Ajeenkya DY Patil School of Engineering
Pune, Maharashtra, India
Academic Year: 2026–2027
PRN:72426592C

Prof. Rupali Wagh
Department of Artificial Intelligence and Data Science
Ajeenkya DY Patil School of Engineering
Pune, Maharashtra, India

Abstract

Software is created in stages: planning, design, coding, testing, deployment, and maintenance, with developers wasting time in the process. They would wait for releases, write nearly similar code many times, extensively debug manually, and stay on testing processes for a long time. The role of GitHub Copilot, ChatGPT and testing solutions will be discussed on a few studies that prove that such systems can help with planning, coding, testing and deployment. Additionally, there is a basic framework that shows how a developer reviews the output of the software in detail and a few examples of how to accomplish subsequent verification. The framework has not been designed and developed, thus there are no results to be discussed. There are still some open questions in terms of security, code quality, privacy and explainability.

Keywords : AI Code Assistants, Artificial Intelligence, CI/CD, DevOps Automation, Generative AI, GitHub Copilot, Large Language Models, Software Development Lifecycle, Soft- ware Testing

I. INTRODUCTION

No one creates software in a single go. The process of devel- oping software consists of planning, designing, programming, testing, releasing and maintaining the software. The phases mentioned above form the Software Development Lifecycle (SDLC). During the planning phase, the development team builds the idea of software functionality. After that, software architecture is decided. When software developers write the program, the testers check its correctness and finally, the users receive it during the release phase. When software is already available on the market, its maintenance phase begins. All phases are interdependent. A lack of planning may lead to a bug in the final release. Besides, this process

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

requires lots of human resources. Teams spend much time creating an idea of the software, implementing similar codes regularly, fixing bugs, doing tests and waiting for release. Artificial Intelligence (AI) is becoming more common to ease the burden of software engineering [1], [2], [3]. Programs such as GitHub Copilot and ChatGPT can assist the software engineer in code writing, bug discovery and test case genera- tion. There have been a number of studies addressing the use of AI in software engineering [1], [2], [4], [5]. The point is that while the tools are known as AI-augmented, they should still only be used to assist engineers, who are responsible for the final outcome and should check the outputs of the tools. Studies have been dealing with the reliability of the tools used and the necessity of human oversight in the process of software development [6], [7]. As a result, the role of engineers and developers changes. Routine code generation takes less time than before, while more time is dedicated to design and verification of the outputs. In this paper we set the following objectives: to review the existing studies on AI in software engineering; to outline how AI is used at every stage of software development; to create a simple framework for using AI that involves supervision of the engineer; to suggest a number of ways to evaluate the proposed framework in future. Issue and Impetus. A significant portion of the SDLC involves redundancy. Company teams write plans, produce code that is not new, identify and correct various deficiencies, run tests and deliver software, and repeat these processes every time. Every operation requires time; for this reason, it is precisely in these areas that AI is used most frequently. The work concentrates on figuring out where exactly such assistance is useful at different phases of the SDLC and how the role of developers changes with the emergence of the technology. Goals. The goals are to do the following: (i) comprehend the role of AI in different phases of the SDLC, starting from planning and ending with maintenance; (ii) explain how AI can help improve planning, coding, testing, and delivery; (iii) identify in what way AI can assist in speeding up the work of developers, making it more effective, and making the same amount of work in less estimated time; (iv) elaborate on good methods of coding, bug detection, automated testing, and monitoring of software after it is delivered; and (v) enhance the quality of software development process through automation. In this section, research questions are stated. There are four research questions formulated in guiding this particular study. The first question is whether AI tools can be used in planning, writing, testing and deploying. The second question is how people are expected to control the process when AI writes code, generates tests or assists with deployment. The third question is what needs to be measured in order to establish whether AI helps speed development, enhances software and improves productivity. The fourth research question refers to the issues of security, quality of code, privacy and explain- ability that need to be addressed. The term explainability refers to the ability to comprehend why a particular tool gave a particular response. In addition, the scope of the paper is described. The study is concerned with the major aspects and areas where AI tools can be useful in software development, such as planning, establishing the functionalities of the expected software product, software design, coding and debugging, making tests and creating test cases, and reviewing the developed software. The topics of security, privacy, and ethics involved in utiliz- ing AI tools have been discussed. The impact of AI on human work has been explained along with possibility of future Ai-enabled machines doing work independently. Information given here is based on findings from other research papers. The framework we are proposing

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

is just a theoretical outline; it can't be implemented or tested in practice, so we don't expect its measurable impact on any project.

II. CONCEPTS

A. Software Development Life Cycle

At the initial stage, the team decides what the software will do, and then they think of how to do it. After coding and testing the software, it is released to users and maintained afterwards. Each of these stages is connected with each other. If a mistake happens on an early stage, it will produce a bug or a failure at a later stage. Thus, it is important to analyze whether it is possible to use AI to improve each of these stages.

B. Generative Artificial Intelligence and LLMs

Generative AI takes in a large data set of items, including written works and computer code, and creates new items based on what it has learned. A Large Language Model (LLM) is an example of gen- erative AI and consists of an enormous corpus of text and programming code. Users can give LLMs simple commands, and the LLM will respond with programs, explanation and test code. Because of the capabilities of LLMs, they can assist in software development by providing outline and suggestions of solutions. Some examples of LLMs include ChatGPT and GitHub Copilot. Unfortunately, LLMs do not always give cor- rect answers, and this is why the proposed approach includes the verification of LLM responses.

C. AI-Augmented Development

In such a case as the AI-enhanced SDLC, the resources aid in each and every phase, while it is the developer who makes every decision. The software is applied for the purposes

of writing code and identifying bugs and test cases, whereas developers are the ones to analyze what results these resources provide. An easy way to express this situation through formula would be: Let Ti be the time taken by a stage without employing any tools, Si be the time saving generated by the introduction of the tool, and Vi be the additional time necessary for verification of what was produced by the tool. The formula showing the time of a stage using the tool would be written in the following way:

$$T'i = Ti - Si + Vi. \quad (1)$$

The tools contribute to the reduction of a stage's duration when time savings amount to more than the verification time, that is when Si is greater than Vi. The current formula merely illustrates the idea without bringing in measurable results. The idea is backed up by one study showing that implementation of these tools shifts a developer's work focus toward verification.

III. LITERATURE REVIEW

We conducted a literature analysis that included five dif- ferent studies. They included research on the activities of developers, the design of a software development life cy- cle (SDLC) framework, an analysis of software engineering methodologies, an overview of the research on LLMs, and an experimental study of developers. The summary of the results of the studies is shown in Table I. The

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

studies by Vella and Blincoe are based on the point of view of developers. The former conducted research on AI coding assists used by programmers. Their findings show that developers write codes faster because they can spend more effort on code verification. The researchers, however, pointed out the danger of overusing such tools, which leads to heavy cognitive burden on users [8]. Another study conducted by Peng et al. showed empirically that those programmers who worked with the help of GitHub Copilot coped with their coding tasks faster than others [12]. Both works focus on coding activities, while the second study is only concerned with coding tasks. The remaining three studies go beyond coding. Saha and Patra suggest an SDLC framework where AI agents assist in planning, coding, testing, and deployment. They acknowledge that human oversight and security measures are still necessary [9]. Durrani et al. studied AI utilization at various stages of software engineering. They identified areas of improvement with respect to planning and requirements and development efficiency but found that results varied by stage [10]. Hou et al. analyzed the research on LLMs in software engineering. They note that LLMs are utilized in requirements, coding, testing, and maintenance but there are still problems with reliability and security [11]. Summary. Overall, although coding speed is where AI makes the largest contribution, it is useful at other stages too. All five studies have one main message—a human is required to control, secure, and validate the results obtained through the utilization of different services. Additionally, the

TABLE I SUMMARY OF REVIEWED STUDIES

No. Study Type of Study What the Study Concluded Caution Noted 1 Vella and Blincoe [8] Study of professional developers using AI coding assistants

Possible over-reliance on the tools and increased cognitive load. 2 Saha and Patra [9] Proposed SDLC framework with AI agents

Developers write code more rapidly and devote more of their effort to verification.

Still requires human supervision and security controls. 3 Durrani et al. [10] Analysis of AI across the stages of software engineering

AI agents assist with planning, coding, test- ing and deployment.

Results varied across the different stages of the SDLC. 4 Hou et al. [11] Review of research on LLMs in software engineering

Improvements in planning, requirements and development efficiency.

Reliability and security remain is- sues. 5 Peng et al. [12] Controlled experiment with de- velopers using GitHub Copilot

LLMs assist with requirements, coding, test- ing and maintenance.

Developers with access to the tool com- pleted their coding tasks more rapidly.

TABLE II RESEARCH DIRECTIONS OF THE REVIEWED STUDIES AND THEIR IMPLICATION FOR THE AI-AUGMENTED SDLC

Research Di- rection

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

Studies Implication

Developer perspective

Coding becomes faster and de- veloper effort shifts toward ver- ification; risk of over-reliance and increased cognitive load. SDLC frame- work

Vella and Blin- coe [8]; Peng et al. [12]

AI agents can assist planning, coding, testing and deployment, but human supervision and se- curity controls are still needed. Analysis across stages

Saha and Patra [9]

Improvements in planning, re- quirements and development efficiency, with results varying across stages. Review of LLM research

Durrani et al. [10]

Hou et al. [11] LLMs assist with requirements, coding, testing and mainte- nance; reliability and security remain issues.

amount of evidence obtained from application of these tools at different stages of SDLC is lesser compared to that of coding. Therefore, this is what the framework that the authors have developed is aimed for—bringing AI into the process.

IV. COMPARING RESEARCH DIRECTIONS AND THEIR POTENTIAL INFLUENCE ON AI-INTEGRATED SDLC

Table II organizes the studies we analyzed by research direction and in what way each direction impacts the AI- integrated SDLC. It is important to note that the grouping described solely serves formal purposes, meaning that studies are by no means evaluated in any specific order. The theory we developed does not substitute any developer's work. It builds on existing tools, like development tools, testing tools, and CI/CD pipeline tools. The main point is that whatever is produced by AI can be regarded only as a recommendation, thus ensuring that all outputs produced by AI will certainly be validated before their application. We did not examine to what extent this influences the quality of results, which is left for future studies.

V. PROPOSED METHODOLOGY

This section describes in detail ways how AI can help at every stage of the SDLC. Citations included are quotes from

Focuses almost exclusively on coding tasks.

our related studies, while all statements without references are presented with our own ideas. Planning and studying the requirements. Comprehensive planning is time-consuming when done manually. The use of AI in the planning process helps to save time. The found research indicates that it improves the process of planning and requirement analysis [10]. In our method, the generated information is a draft prepared by the AI regarding the whole project, and its contents are checked by people since they should monitor the activities performed by the tools [9]. Software design. AI is helpful for engineers to choose

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

various ways in which the software can be put together. The found studies did not pay attention to the issue of design, therefore we will leave it as a topic for future research since the choice of the design format stays with the programmer. Code generation and debugging. AI coding assistants make coding and bug finding quicker [3]. Those developers who used these tools for their work were able to complete coding tasks ahead of schedule [12]. On the other hand, they may rely too much on the tools and experience cognitive over- load [8]. That means that produced code should be checked twice before usage. The process of creating and performing tests. AI tools can create test cases and QA software can apply them. This allows you to minimize manual work and detect problems early. Nonetheless, developers should still analyze the tests because a successful test doesn't prove whether the requirement is met or not. If a test fails, it returns to the programming stage for correction. Code analysis. We introduce AI-enabled code analysis as an additional check of code quality. It greatly supports the analyzer by showing potential problems. However, it is still the analyzer who is entitled to accept or reject the proposed changes. Continuous integration and deployment. AI-aided CI/CD is used in DevOps automation. Monitoring and upkeep. Once the launch takes place, AI- based monitoring can facilitate early detection of the problems so that prompt resolution is possible. Some investigations have found that it is possible to use LLMs for maintenance-related aspects. Checks and green light. A designer checks all results of AI work before accepting the output, including the code and tests

Planning Design Coding Testing Deployment Monitoring

Verification and approval of AI output by a developer after each stage

Fig. 1. Proposed workflow: AI assistance in every stage, with human verification after each stage and feedback for correction.

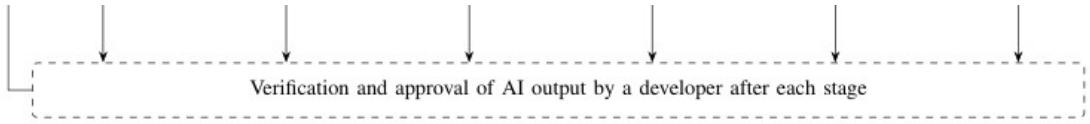


Fig. 1. Proposed workflow: AI assistance in every stage, with human verification after each stage and feedback for correction.

TABLE III AI ASSISTANCE COVERED AND CONCERNS IDENTIFIED IN THE REVIEWED STUDIES, BY STAGE

Stage AI Assistance Covered Concerns Iden- tified Planning and re- quirements

Human supervision required [9]; results vary across stages [10] Software design Not examined independently by the reviewed studies

Improved planning and re- quirements [10]

Deferred to fu- ture research

Coding and de- bugging

Over-reliance and increased cognitive load [8] Testing Part of AI agent framework [9]; LLMs assist with test- ing [11]

Faster code writing and bug identification [3]; tasks completed more rapidly [12]

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

Reliability and security [11]

Deployment Part of AI agent framework [9]

Human supervi- sion and security controls [9] Maintenance LLMs assist with mainte- nance [11]

Reliability and security [11]

made, and deployment actions taken. Incorrect or incomplete output is sent for improvement. A human gives the go ahead since the AI does not understand the needs of the project. In Table III presented below, one can find information about AI assistance in terms of each stage and problems mentioned in the research papers mentioned.

VI. PROPOSED SYSTEM ARCHITECTURE

The system is divided into six layers according to the design: layer for developer interface, layer for AI help, layer for development toolchain, layer for verification, layer for monitoring, and layer for data storage. Each component can be analyzed separately and it will be known who made what decision. Figure 2 presents the diagram of the layers. The architectural model is only theoretical. It is not built yet. The developer interface layer is where developers and teams submit their tasks here and get back plans, code, test cases, and deployment information. The AI assistance layer is the one that contains different AI coding assistants (such as GitHub Copilot and ChatGPT) and automated testing tools. These components provide code suggestions, bug detection, and test case generation functions.

Developer Interface Layer

AI Assistance Layer

Development Toolchain Layer

Verification Layer

Monitoring and Maintenance Layer

Data Layer

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

Fig. 2. Proposed conceptual system architecture.

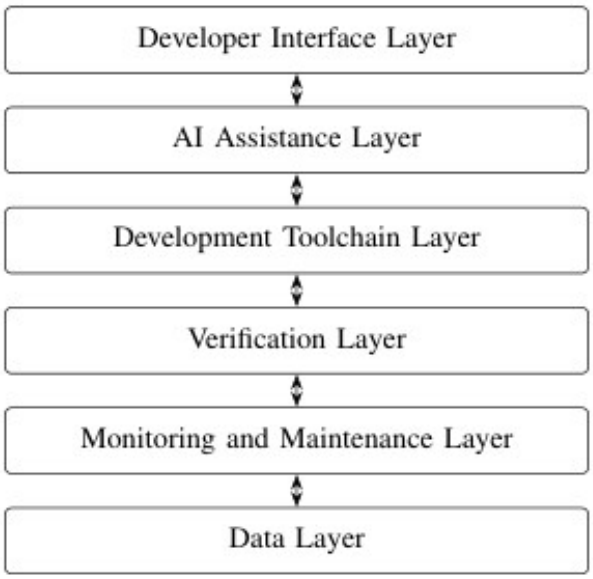


Fig. 2. Proposed conceptual system architecture.

The output is kept separated from the code created and accepted by the developers. The development toolchain layer is the layer where CI/CD and DevOps pipeline are located, which allow carrying out the software building, testing, and releasing. Before being accepted, the AI output undergoes a review of the code, testing results, and checks for implementation. If a control does not pass, the programmer receives a notification about the issue. Monitoring and maintenance module. The released pro- gram is monitored and any problems are reported that would need to be fixed. Data module. The documentation related to the project, program code, testing results, and information about the ac- cepted or rejected AI recommendations is stored here. As discussed in the security and privacy section, only authorized people may have access to it. Workflow. The workflow is sequential. It refers to the stages of planning and designing processes, followed by coding, testing, deployment, and monitoring.

VII. RELEVANCE TO CURRENT TRENDS

Generative AI. Applications like ChatGPT and GitHub Copilot are powered by large language models (LLMs). The research analyzed indicates that LLMs play a positive role in requirements specification, programming, testing, and mainte- nance [11]. The pivotal aspect of assistance provided by LLMs is not how much the AI generates but rather how good the generated output is and how much effort it takes to check it.

DevOps automation. CI/CD stands for continuous inte- gration and continuous delivery; it implies the automated processes of building, testing, and releasing software products. AI-enabled deployment contributes to that automation. De- ployment methodologies utilizing AI agents must also ensure security [9]. Cloud computing. Cloud computing plays an important role in deployment and monitoring. It is not the focus of the analyzed studies, thus we only mention it in the context. Human-AI collaboration. AI

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

generates drafts and sugges- tions while humans make decisions and assume responsibility for the results. One of the proofs of this statement is that developers check the output much more carefully [8]. Agentic software engineering. AI agent-based systems [9] point the way to a software engineering stage: much of the work is undertaken automatically. This is perceived as a future development in our study.

VIII. PROPOSED EVALUATION FRAMEWORK

The framework still needs practical evaluation. The criteria below are our own suggestions for future research, they are not results. Total stage time change means the stage's time difference after AI assisted application that takes into account verification time (1):

$\Delta Ti = Ti - T'i\ Ti \times 100.$ (2)

Positive changes means that the stage is completed faster after verification. Bug identification rate means the percentage of the detected versus existing bugs (3):

$$BDR = \frac{Dfound}{Dtotal} \times 100 \quad (3)$$

Here Dfound refers to all known bugs found and Dtotal is the total number of known bugs. Non-bug reports are not included in this process separately. Acceptance rate means the ratio of AI recommendations that developers accepted after reviewing them (4):

$$AR = \frac{Naccepted}{Nsuggested} \times 100 \quad (4)$$

It is important to read the above statement alongside the verification process as a high result means that the developers have too much faith in artificial intelligence. Other possible methods would be the amount of the code covered with tests generated by AI, the frequency of the releases and their lateness, the quality of code judged by the reviewers and post-release defects and the time spent for the verification of AI-created work by developers. Experimental design. It would be possible to compare three ways of conducting development: (1) without employing artificial intelligence, (2) with the aid of artificial intelligence while skipping formal verification and (3) with the aid of artificial intelligence with a formal verification step. Projects, activities and people need to be included in one comparison

group. Independent experts should assess the quality of code, tests and defects. All utilized means and methods should be included in the report. We have no claims for improvements since no improvement is measured.

IX. GAPS IN RESEARCH

• Security: The need in security measures for AI-assisted development and AI-agent-based frameworks [9], [11].

• Code quality and reliability: The quality of AI- generated output is untrustworthy and such output must be verified [11].

• Privacy: The use of AI tools in development presents privacy issues, and the articles we have reviewed do not address them.

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

• Explainability: If developers are to supervise a tool properly, they must understand the output it produces.

• Dependence and Mental Load: The use of AI may cause an increase in both.

• Evidence at Different Stages: There are different evi- dences in the various SDLC stages.

X. LIMITATIONS AND FUTURE WORK

Limitations of Study. The study is based on evidence from other studies, and thus the framework is not verified and can neither be confirmed to help increase productivity, improve quality, or reduce cost. Only five studies were reviewed; this is not nearly all that was published on the topic, and most of the studies examined primarily coding but not much else— there is either no or very little data on preprocessing, training, evaluation, or tuning. Future work. The next step is to create a small prototype of the framework. This prototype will include AI programming assistants, automated testing tools and CI/CD pipeline. The latter will include a verification step. The prototype has to be tested both on real projects and on the projects created for the testing. Prompts should be recorded so that different people can do the testing independently. Both problems and corrective measures should be recorded properly. Other perspectives include improved security and privacy control, explainable AI output and software engineering projects in which tools will do everything themselves under the human control.

XI. CONCLUSION

AI-enhanced SDLC uses instruments such as GitHub Copi- lot, ChatGPT, testing instruments, etc. to assist developers in planning, coding, testing and deployment. The analysis shows that coding goes faster and AI helps in other phases of SDLC as well. It has been indicated that developers devote a greater effort to verification, with checking, safety and assurance as concerns. Research directions are compared, and a framework is proposed which requires that every output of AI to have checks conducted by a programmer. It is worth noting that this framework has not been tested, and the true usefulness of the value can be established only in future studies. There are still some open issues regarding security, quality of code, privacy and explainability.

REFERENCES

[1] Google Research. (2025). AI-Assisted Software Engineering and Code Generation. Google Research.

[2] Open AI. (2025) ChatGPT and Large Language Models.

[3] GitHub. (2025). Documentation for GitHub Copilot. GitHub.

[4] IEEE. (2024) Scientific Works on AI in Software Engineering. IEEE Xplore.

[5] ACM. (2024). AI-Assisted Software Development. ACM Digital Library.

[6] Sommerville, I. (2015). Software Engineering (10th ed.). Pearson.

[7] Pressman, R. S., & Maxim, B. R. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill.

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

[8] Vella, A., & Blincoe, K. (2026). The Impact of AI Coding Assistants on Software Engineering: A Longitudinal Study. arXiv.

[9] Saha, A. N., & Patra, D. (2026). AI-First Software Development Life cycle: An Agent-Driven Framework for Autonomous Planning, Coding, Testing, and Deployment. ESP Journal of Engineering & Technology Advancements.

[10] Durrani, U. K., et al. (2024). A Decade of Progress: A Systematic Literature Review on the Integration of AI in Software Engineering Phases and Activities (2013–2023). IEEE Access.

[11] Hou, X., et al. (2024). Large Language Models for Software Engineer- ing: A Systematic Literature Review. ACM Transactions on Software Engineering and Methodology.

[12] Peng, S., et al. (2023). The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv