

Empirical Evaluation of AI-Assisted Code Generation on Full-Stack Web Development Performance

Author: Anirudh Negi, Chirag Yadav | Co-author: Ms. Bhawana Sharma

School of Engineering and Technology

Sushant University

Gurugram, Haryana, India

Abstract: AI technology has advanced rapidly, which has changed how software development works today because developers now use AI-based code generation tools like ChatGPT better than before. This research conducts an empirical study to assess how full-stack web application development differs between two coding methods: traditional manual coding and AI-powered development. The researchers use a MERN-based event booking platform to develop identical features through two different methods, which will help them achieve consistent results. The study demonstrates that AI tools help developers work more efficiently, yet human specialists remain essential to the development process. AI functions as a collaborative tool that helps developers throughout their work. The findings add to current debates about how AI affects software engineering processes and what this means for future software development methods.

Keywords: *AI-assisted development, Full-stack web development, Code generation, Developer productivity, Software engineering, MERN stack, Comparative analysis, Prompt engineering, Human-AI collaboration, Web application development.*

I. INTRODUCTION

The field of software development has undergone major changes because of artificial intelligence's rapid technological progress which creates new methods for application development and implementation and system maintenance. The development of AI-assisted code generation systems enables software developers to automate their repetitive work tasks while producing operational code which speeds up their software development processes. The ChatGPT system uses large language models that have been trained on extensive programming data to generate code recommendations and solutions that respond to current context.

Full-stack web development requires developers to master all essential technologies which include frontend frameworks and backend architectures together with database management and system integration. Developers had to create system logic through their own design work while they wrote code and fixed system faults and enhanced system efficiency. The process requires extensive time to complete because it needs deep mental concentration which becomes more difficult when handling intricate application specifications. AI-assisted tools bring intelligence to this workflow which enables developers to finish their work faster while they develop their projects.

The increasing use of artificial intelligence for software development creates critical challenges that evaluate its effects on developer work efficiency and software quality and software development processes. While AI tools can generate code quickly and assist in problem-solving, their outputs are not always contextually accurate or optimized for specific project requirements. The developer must assess AI-generated code because their proficiency

determines how they should evaluate and improve and combine code from AI sources. The development process transforms into a collaborative system where humans work together with machines because humans use their intelligence to evaluate machine-generated suggestions.

The development of AI systems requires prompt engineering which requires engineers to create structured input queries that help AI systems produce specific output results. The AI-generated code achieves its best performance when developers use clear and precise prompts to guide their work. Software developers now need to learn a new skill which allows them to communicate their project needs to AI systems just as they must learn to write code.

The testing of AI-assisted tools requires research to verify their performance in real-world development work which involves building functional applications instead of testing single code components. Full-stack applications involve the integration of multiple components such as authentication systems, APIs, user interfaces, and database interactions, making them ideal for evaluating the effectiveness of AI-assisted development in a holistic manner.

The main purpose of this study is to assess how AI-based development affects development time and code quality and debugging work and developer performance.

The study investigates these elements to demonstrate how AI integration into software development processes brings both benefits and drawbacks. The research aims to enhance knowledge about the ways artificial intelligence will determine upcoming

ing trends in software engineering. The study investigates how artificial intelligence functions as a productivity tool which helps developers while they still need to use their expertise for essential tasks.

II. LITERATURE REVIEW

Recent advancements in artificial intelligence have led to increased research interest in the application of AI-assisted tools within software engineering. Multiple studies have researched how AI coding tools affect developer productivity. Mark Chen et al. [1] investigated large language models trained on code and demonstrated that such models can significantly accelerate programming tasks by generating functional code from natural language descriptions. Their findings demonstrate that AI tools work best for automated processing of repetitive coding tasks which leads to faster software development.

Eirini Kalliamvakou and her co-authors conducted an empirical research project about AI-assisted development which they found to enhance developer productivity. Developers who used AI tools completed their work with increased speed and im-

proved efficiency according to the research findings. The study demonstrated that developers who effectively used AI systems achieved better productivity results.

The study conducted by Hannah Pearce and her co-authors assessed AI-generated code because they wanted to determine its reliability through their research. The study showed that AI-generated code produces correct syntax but creates logical defects and security risks that require thorough examination. The process of implementing AI-based solutions in operational environments requires human experts to confirm their effectiveness.

The research conducted by Premkumar Devanbu and his colleagues in reference [4] showed how artificial intelligence technology functions as a vital element of software engineering processes which requires people to collaborate with artificial intelligence systems. The researchers found that development teams should consider artificial intelligence technologies as tools which provide assistance to developers who need to make decisions and debug programs and design systems.

Table 1: Summary of Existing Research on AI-Assisted Development

Author	Focus Area	Key Findings	Limitations
Mark Chen et al.	AI Code Generation (Codex)	AI models can generate from natural language & improve development speed	Limited contextual understanding in complex applications
Erini Kalliamvakou et al.	Developer Productivity	AI tools significantly reduce development time and improve efficiency	Effectiveness depends on developer experience & usage
Hannah Pearce et al.	Code Security & Reliability	AI-generated code is syntactically correct but may introduce security vulnerabilities	Requires human validation & security review
Premkumar Devanbu et al.	Human-AI Collaboration	AI tools act as supportive assistants rather than replacements for developers	Over-reliance may impact critical thinking & problem solving

The studies show positive outcomes but their results show multiple limitations which need to be addressed. The majority of current studies investigate single programming tasks which include code completion and debugging and algorithm implementation. The studies offer useful information yet they fail to represent all aspects of software development which occurs in full-stack environments that need complete system integration.

The study faces another restriction because it fails to give adequate attention to prompt engineering

which serves as the key factor that determines how well AI development work will perform. The AI system produces its best results when developers give precise details about their needs to the system. Existing research studies have not investigated this particular aspect of the topic.

Researchers report productivity improvements for their work yet only one study compares how development teams operate between manual methods and AI-powered systems within a specific project.

The development lifecycle research remains challenging because AI tools demonstrate their effects throughout development instead of showing their impact on separate tasks.

RESEARCH

GAP:

Although prior studies highlight the benefits of AI-assisted coding in improving productivity and reducing development effort, there is a lack of comprehensive evaluation in the context of full-stack web application development. Existing research mainly investigates isolated tasks which researchers conduct within controlled environments because they want to study AI tools performance in complete real-world projects. The research demonstrates this existing research gap by examining manual and AI-assisted development processes in full-stack application development.

III. METHODOLOGY

A. Research Design

The research examination uses a comparative research design to assess the success of AI-driven development methods when they are compared to traditional coding methods in full-stack web application development. The study aims to investigate the impact of various development methods on three specific outcomes which include developer productivity and code quality and debugging time and overall development experience.

The researchers executed their investigation in a controlled setting which included two developers

who possessed identical educational qualifications and complete-stack development expertise. The two developers used different methods to complete their work one developer used manual coding while the second developer worked with AI-based development tools. The two developers worked on the same functional requirements which ensured both consistency and a valid comparison between their work.

The study shows how development methods create effects which go beyond individual performance measurements. The application used in this study serves as an experimental environment to evaluate these approaches under realistic conditions.

B. Experimental Application Context

The complete web application environment served as the testing environment for our practical assessment. The application operates as a real-world event management and booking system which uses multiple frontend and backend components to create its operational functions.

The system was developed using the MERN stack which includes MongoDB for data storage and Express.js and Node.js for backend services and React for frontend interface design. The application includes essential modules such as user authentication and event listing and booking management and administrative controls.

Table 2: Functional Modules & Implementation Scope

Module	Description	Implementation Scope	Purpose in Study
Authentication System	OTP-based user login & verification	Frontend + Backend	Evaluates API integration & security handling
Event Listing	Displaying available events with details	Frontend + Backend	Tests data fetching & UI rendering
Booking System	Users can request & manage bookings	Frontend + Backend	Assesses business logic implementation
Admin Dashboard	Admin can manage events & bookings	Frontend + Backend	Evaluates role-based access & control logic
Database Integration	Storage & retrieval of user & event data	Frontend + Backend	Tests database operations & data consistency

C. Development Approach

I used two different approaches for developing the same set of features in order to subsequently compare them in a meaningful way.

1. Manual Development Approach

The manual development approach required developers to create all system functions through conventional programming techniques while refraining from using any artificial intelligence development tools. The process required developers to create complete code from the beginning while they developed application functionality and integrated APIs and worked on fixing system faults through manual testing.

The developer had to solve problems through direct work while making decisions and optimizing processes. The development process required developers to handle all tasks which included building components and controlling data flow and maintaining system state.

2. AI-Assisted Development Approach

The team used several AI coding tools that included ChatGPT, GitHub Copilot, and Claude to support their implementation work through the AI-assisted development method. The tools created code solutions while delivering implementation recommendations and debugging support and detailed explanations of complicated logical systems.

The development workflow involved providing structured prompts to AI systems describing the required functionality. For example, tasks such as creating API endpoints, implementing authentication flows, or designing UI components were initiated through natural language prompts. The generated

Table 3: Tools such as VS Code, Node.js, React, etc.

outputs were then reviewed, modified, and integrated into the application.

A key component of this approach was prompt engineering, where the effectiveness of the generated code depended on the clarity and specificity of the input provided. In many cases, iterative refinement was required to improve accuracy and align the output with project requirements.

Additionally, AI tools were used for:

- Generating boilerplate code for backend routes and controllers
- Assisting in frontend component creation and state handling
- Debugging errors by analysing logs and suggesting fixes
- Providing explanations for unfamiliar implementations

While this approach reduced manual effort in writing code, it introduced the need for validation, correction, and contextual adaptation of AI-generated outputs.

D. Experimental Setup

The research setup was established to create identical testing conditions which would measure the results of two different development methods. The two implementation methods used identical system requirements because both methods operated with the same technological framework and development equipment and system setup.

The two methods of implementation used identical functional modules to create a standard testing method. Developers used basic development tools which included code editors and runtime environments and browsers for their testing work. debugging.

Component	Tool/Technology Used	Purpose
Code Editor	Visual Studio Code	Writing, Editing & managing source code
Frontend Framework	React.js	Building user interface & client-side logic
Backend Framework	Node.js with Express.js	Handling server-side logic & API development
Database	MongoDB	Storing & managing application data
Runtime Environment	Node.js	Executing backend services
Browser	Google Chrome	Testing & debugging frontend functionality
AI Tools (AI- Assisted Approach)	ChatGPT, GitHub Copilot, Claude	Code generation, debugging & assistance

E. Evaluation Metrics

To evaluate the effectiveness of each development approach, a set of metrics was defined to capture both quantitative and qualitative aspects of the development process.

The primary evaluation metrics include:

- **Development Time:** Time required to complete specific features

Table 4: Metric | Description | Measurement Method

Metric	Description	Measurement Method
Development Time	Total time to implement each module or feature	Measured in min/hrs during development
Code Quality	Structure, readability & maintainability of the code	Evaluated based on clarity, modularity & organization
Debugging Effort	Effort required to identify & fix errors	Number of issues encountered & time spent on resolving them
Developer Experience	Ease of development & cognitive effort involved	Observational analysis based on development workflow
Dependency on Tools	Level of reliance on external assistance during development	Observed based on frequency of AI tool usage vs Manual coding

- **Code Quality:** Readability, structure, and maintainability of code

- **Debugging Effort:** Effort required to identify and resolve issues

- **Developer Experience:** Ease of development and cognitive load

These metrics provide a comprehensive basis for comparing the efficiency and practicality of manual and AI-assisted development approaches.

F. Data Collection Method

The researchers gathered data through direct observation and tracking activities during their development work. The team documented two types of information which included the implementation time for each feature and the problems they faced during their development work. The researchers recorded their qualitative observations to document how different approaches affected workflow efficiency and

implementation simplicity and operational difficulties. The evaluation process uses this method to assess both measurable results and experiential elements.

G. Analysis Approach

The research team used a comparative method to examine the gathered data which showed the main differences between manual development work and AI-assisted development work. The analysis fo-

cuses on practical insights rather than complex statistical methods, which ensures that the findings maintain their relevance to actual development work.

The study results use two evaluation methods which include development time measurements and developer experience assessments to create a complete evaluation of the study. The results of the study explain the advantages and disadvantages and the compromises which each development method presents.

IV. RESULTS & ANALYSIS

A. Overview of Results

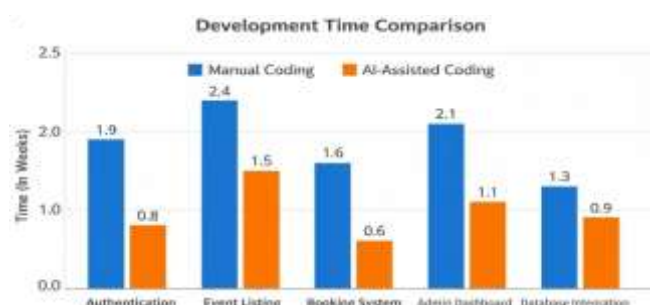
The section shows a comparative study which evaluates manual development methods against AI development methods through the efficiency metrics established in the research methodology. The results are derived from observations recorded during the implementation of identical functional modules under both approaches.

The analysis focuses on key parameters including development time period code quality debugging effort and developer experience. The findings aim to highlight the strengths and limitations of each approach in a practical full-stack development context.

B. Development Time Analysis

The most visible distinction between the two methods appeared through their different development durations. The use of AI tools helped speed up task completion which included API development and UI component creation and fundamental logic development because the tools provided access to pre-written code and coding recommendations. The manual method took extra time because it needed complete planning and development work plus debugging for every single feature.

Figure 1: Bar Graph comparing time taken for each module in manual vs AI-assisted development



Overall, the AI-assisted approach demonstrated improved efficiency, particularly for repetitive and boilerplate tasks. However, additional time was sometimes required to refine AI-generated code to match specific project requirements.

C. Code Quality Evaluation

The two coding methods produced different results which affected the structure and stability of their generated code. The AI-assisted approach produced well-structured code which maintained modular design throughout standard implementations. The AI system created structured templates which enabled users to develop their code according to established programming standards.

Manual development allowed developers to manage both logic design and their own specific customization needs. The manually written code showed better alignment with the developer's architectural design and achieved improved module-to-module context integration.

The AI-generated code presented two major issues which required resolution because it contained unnecessary logical elements and it used basic solutions. AI tools assist in achieving better code consistency results yet they lack the ability to replace the process of critical evaluation.

D. Debugging Effort Analysis

The two testing methods showed different levels of difficulty when debugging their respective tests. The manual approach enabled developers to find and solve errors during the development process because developers understood all aspects of the codebase. The AI-assisted approach created new problems through its introduction of faulty logic and dependency errors and unfinished system components. The team used AI tools for debugging purposes yet they needed extra testing to confirm their findings.

The developers handled common errors through AI-assisted debugging which helped them to find solutions for those errors. The developer needed to authenticate the solution's accuracy before using it.

E. Developer Experience and Workflow

The two development methods produced different results during their development process. The manual approach required developers to maintain mental effort throughout their work because they needed to solve problems and create code structures and find errors. The AI-assisted approach reduced cognitive load during code writing because it delivered prebuilt solutions and ongoing recommendations. The system enabled faster implementation work, but its performance depended on the standard of incoming requests and the system-created results. Prompt engineering played a crucial role in this process. The use of well-structured prompts produced accurate and useful code output, but the

use of vague inputs led to incomplete or incorrect results that needed additional work to improve.

F. Comparative Summary

To provide a consolidated view of the findings, a comparative summary of both approaches is presented below.

Table 5: Manual vs AI across all metrics

Parameter	Manual Development	AI-Assisted Development
Development Time	High (approximately 2 weeks)	Low (approximately 1 weeks)
Code Quality	High – Better logical accuracy & customization	Moderate to High – Good structure but requires validation
Debugging Effort	Moderate – Easier due to full understanding of code	Moderate – Requires additional verification of generated code
Development Speed	Slower due to manual implementation	Faster due to automated code generation
Learning Depth	High – Strong understanding of logic & system design	Medium – May reduce deep understanding due to reliance in AI
Dependency on Tools	Low – Independent development process	High – Relies on AI tools for code generation & assistance
Consistency	Moderate – Depends on coding practises	High – Uniform structure & formatting
Flexibility	High – Fully customizable solutions	Moderate – Limited by AI-generated patterns

G. Key Observations

From the analysis, several key observations can be derived:

- AI-assisted development significantly improves development speed, especially for standard and repetitive tasks
- Manual development provides better control and deeper understanding of application logic
- AI-generated code requires validation and refinement to ensure correctness
- Prompt engineering plays a critical role in determining the effectiveness of AI-assisted development
- AI tools act as productivity enhancers rather than complete replacements for developers

V. DISCUSSION

A. Interpretation of Findings

The study results demonstrate that manual development methods and AI-assisted development methods produce different outcomes through their effects on efficiency and control and user experience during development. The AI-assisted method decreased development time because AI tools can

quickly create code and automate common tasks and they deliver immediate fixes for standard programming issues. The process needs less time for basic setup work which results in faster development.

The manual development method produced better logical consistency because it met specific application needs better than other methods. The developer acquired system architecture and data flow knowledge through active involvement in designing and building all system components. The process of manual development enables users to control all aspects of final results because it allows them to customize outputs and verify logical correctness of their work.

B. Trade-offs Between Efficiency and Control

The study found that development efficiency increases when developers have more control over their work. AI-assisted development enables teams to complete their work more quickly which makes it an ideal tool for developing prototypes and working on their first development stages. Developers lose control over the code that AI generates because the system operates at high speeds.

Manual development produces slower results but delivers better reliability because developers possess complete knowledge of the codebase. The

trade-off shows that both methods lack general superiority because their success depends on specific situations. Developers need to find an appropriate balance between fast work progress and precise outcomes when selecting between these two methods.

C. Role of Prompt Engineering

AI development results depend on how well the AI system receives prompt input from developers. The study observed that well-defined and structured prompts resulted in more accurate and relevant code generation whereas ambiguous or incomplete prompts led to outputs that required additional correction and refinement. The study establishes prompt engineering as an essential competency which modern development teams need to succeed in their work.

Developers who can communicate requirements to AI systems will increase their development efficiency because AI systems will produce better solutions.

D. Human-AI Collaboration

The findings from this research study support the view that AI tools work best when they assist developers instead of taking over their tasks. The AI systems operate successfully because they handle basic tasks but require human skills to make choices and test systems and confirm accurate logical operations.

The AI-assisted approach enables developers to stop writing code and begin supervising and validating all produced results. The collaborative work process boosts productivity while human decision-making remains essential for software development work.

E. Practical Implications

The results demonstrate that AI-powered tools deliver major advantages for organizations because they decrease the time required for product development while making it easier to execute their work. The tools show their greatest value when developers need to handle repetitive coding work or when they face new technological challenges.

The research demonstrates that when programmers depend too much on AI tools their ability to study vital programming skills will decrease. The dependency of developers on generated outputs will harm their ability to solve problems and their knowledge of system design.

The most successful method combines human programming work with AI technology support. The developers need this method because it enables

them to use AI tool benefits while keeping control of essential implementation features.

F. Limitations of the Study

The research offers beneficial findings but its deficiencies impede proper evaluation of its outcomes. The research used a small participant group because it included only two software developers for its testing. The research team worked to create identical academic backgrounds and development conditions yet their effort failed because individual developers had distinct coding and problem-solving methods. The evaluation used one application context but this method failed to capture all aspects of software systems used in large-scaled and industrial environments. The outcomes of the study will change because different project types and development environments will produce different results. The researchers used qualitative data to assess code quality and developer experience yet they did not use any standardized quantitative assessment methods. The method presents practical solutions but it creates evaluation subjectivity through its use. The success of AI-supported development work depends on which development tools users select and which prompt quality they employ. The research did not investigate how different tool capabilities and prompt design can produce distinct results which need to be studied.

VI. CONCLUSION

The research performed a comparison study between manual development methods and AI-supported development methods for building complete web applications. The research team tested both development methods by creating the same functional modules to study how AI tools affected development speed and code quality and debugging activities and developer satisfaction.

The study results demonstrate that AI-powered development speeds up projects because it automates standard tasks and produces complete development code. The system proves most effective during fast development and prototype creation for new products. The research demonstrates that AI-generated code needs testing and development work to achieve both logical accuracy and product-specific requirements.

The manual development process enables developers to control every aspect of their code which leads to results that meet their specific needs. The process demands extra resources for execution yet it provides developers with complete system comprehension which helps them handle intricate situations better.

The research results show that organizations must choose between two opposing goals which create a trade-off between operational efficiency and managerial authority. Human experts remain necessary because AI tools can boost work efficiency. The success of AI-assisted development work depends on three factors which include effective prompt engineering methods and developer supervision and the ability to assess output results.

The study finds that AI technology functions as a supportive tool which enhances software development work without taking over developer responsibilities. Developers can achieve optimal results by using both manual coding methods and AI coding support which together create the most efficient solution.

The research findings show how artificial intelligence technology changes software development methods. The research demonstrates how AI-assisted coding helps software development but also shows its limits which need to be addressed through a proper distribution between automated processes and human work in contemporary development environments.

IV. REFERENCES

[1] M. Chen, J. Tworek, H. Jun, et al., "Evaluating Large Language Models Trained on Code," *arXiv preprint arXiv:2107.03374*, 2021.

[2] E. Kalliamvakou, C. Bird, T. Zimmermann, et al., "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot," *IEEE Software*, vol. 41, no. 3, pp. 34–42, 2024.

[3] H. Pearce, B. Ahmad, B. Tan, et al., "Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions," in *IEEE Symposium on Security and Privacy*, 2022, pp. 754–768.

[4] P. Devanbu, T. Zimmermann, and C. Bird, "Beliefs and Evidence about Software Engineering," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 108–119.

[5] S. Barke, M. B. James, and N. Polikarpova, "Grounded Copilot: How Programmers Interact with Code-Generating Models," *Proceedings of the ACM on Programming Languages*, vol. 7, no. OOPSLA1, pp. 85–111, 2023.

[6] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models," in *CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–23.

[7] N. Perry, M. Srivastava, D. Kumar, and D. Boneh, "Do Users Write More Insecure Code with AI Assistants?" in *ACM Conference on Computer and Communications Security*, 2023, pp. 2785–2799.

[8] N. Nguyen and S. Nadi, "An Empirical Evaluation of GitHub Copilot's Code Suggestions," in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 1–5.