ARTIFICIAL INTELLIGENCE IN FRONTEND DEVELOPMENT: REVIEW, CHALLENGES, AND FUTURE DIRECTIONS

Katarina Stojiljković*[1],
[0009-0001-3718-8647]

Dejan Bulaja[2],
[0009-0007-6992-979X]

Tamara Živković[2],
[0000-0003-2969-1709]

Miodrag Živković[2]
[0000-0002-4351-068X]

[1]Student,
Singidunum University,
Belgrade, Serbia

[2]Singidunum University,
Belgrade, Serbia

Abstract:

Artificial intelligence has become an increasingly influential component of modern software development, with a growing impact on frontend engineering practices. This paper presents a structured review of the application of AI in frontend development, with a particular focus on analyzing existing benchmarking studies of widely used AI tools. The study examines how these tools are applied in common frontend tasks, including component generation, styling, debugging, and design-to-code transformation.

By synthesizing findings from recent research, the paper identifies key performance patterns across different AI systems, highlighting their strengths in improving development speed and productivity, as well as their limitations in terms of reliability, security, and handling of complex frontend logic. The analysis also reveals a lack of standardized evaluation frameworks tailored specifically to frontend development, as most existing studies rely on general-purpose metrics that do not fully capture user interface and user experience requirements.

Based on these observations, the paper outlines several directions for future research, including the development of frontend-specific evaluation criteria, improvements in contextual understanding for complex tasks, and enhanced integration between design and development processes. The findings suggest that, while AI tools provide valuable support in frontend workflows, they currently function most effectively as assistive technologies rather than fully autonomous solutions. This review contributes to a clearer understanding of the current capabilities and limitations of AI in frontend development and highlights opportunities for further advancement in this rapidly evolving field.

Keywords:

Artificial Intelligence, Frontend Development, Generative AI, UI Development, AI-assisted Coding.

INTRODUCTION

Artificial intelligence has rapidly evolved from a specialized research domain into a widely adopted technology that is reshaping modern software development. The rapid growth of large language models and generative AI tools has significantly changed how developers approach coding tasks, allowing faster implementation, automated suggestions, and new ways of interacting with development environments. These tools are increasingly integrated into everyday workflows, influencing not only backend systems and data processing, but also the way user-facing applications are designed and built.

Correspondence:
Katarina Stojiljković

e-mail:
katarina.stojiljkovic.25@sngimail.rs

DOI: 10.15308/Sinteza-2026-165-170

Frontend development represents a particularly interesting area within this transformation. Unlike other domains of software engineering, frontend development is closely tied to visual design, user experience, and interactivity. It requires attention to detail, pixel perfect work and alignment between design and implementation. With that, the integration of AI into frontend development introduces both opportunities and challenges that differ from those found in more logic-oriented or backend-focused tasks. AI tools are now capable of generating user interface components, assisting with styling, and even converting design concepts into code. However, their effectiveness in handling the complexity and precision of frontend requirements remains an open question.

Although the application of artificial intelligence in software development has been widely studied, existing research is often generalized and does not specifically address frontend-specific scenarios. There is a lack of consolidated analysis regarding how different AI tools perform when applied to common frontend tasks, as well as how their capabilities compare in terms of accuracy, efficiency, and reliability.

The aim of this paper is to provide a structured overview of artificial intelligence in frontend development, with a focus on analyzing and synthesizing existing benchmarking results. By examining current research and reported evaluations of widely used AI tools, the paper seeks to identify their strengths, limitations, and areas where further investigation is needed. Through this approach, the study contributes to a better understanding of the current state of AI-assisted frontend development and outlines directions for future research in this evolving field.

2. OVERVIEW OF AI IN FRONTEND DEVELOPMENT

Artificial intelligence has become a large part of modern frontend development, mostly with the integration of various tools that support developers in the development process. These tools are typically based on large language models and can understand natural language instructions, generate code, and assist with different tasks. Key ones are IDE-integrated solutions like GitHub Copilot and Cursor, specialized agents like Windsurf (with Cascade) and Replit Agent, alongside general chat assistants like Claude 3.5 Sonnet, which are designed to boost developer productivity and reduce the time required for the implementation.

In the context of frontend development, AI tools can be used across a range of tasks that go beyond the simple code generation. One of the most common use cases is the automatic creation of UI components based on textual descriptions or partial code input. These tools can be used to quickly create components, significantly speeding up the initial stages of development. Additionally, AI systems are increasingly used for styling and layout generation, including the creation of CSS rules or utility-based styling approaches, which are often repetitive and time-consuming when done manually [1]. Recent research has also explored optimization techniques in frontend systems, highlighting the growing complexity and performance considerations in modern web applications [2].

Another important area of application is the transformation of design concepts into functional code. AI tools can understand design descriptions and images to some extent, and generate corresponding frontend components and pages, effectively bridging the gap between design and development [3]. This includes generating layouts, handling responsiveness, and applying basic user interaction logic. AI is also used to assist in debugging and code optimization, where it can identify potential issues, suggest fixes, and improve overall code structure.

Despite these advancements, the integration of AI into frontend workflows is still evolving, and its capabilities are not uniformly reliable across all tasks. While AI tools perform well in structured and repetitive scenarios, their effectiveness in handling complex user interactions, advanced state management, and nuanced user experience requirements remains limited. As a result, understanding the current landscape of available tools and their practical applications is essential for evaluating their role in frontend development, which serves as a foundation for the benchmarking analysis presented in the following section.

3. BENCHMARKING AI TOOLS IN FRONTEND DEVELOPMENT

The evaluation of AI tools in frontend development has recently become an active area of research, with several studies proposing benchmarks to assess their performance in tasks such as code generation, UI reconstruction, and debugging. One of the most notable contributions is the Design2Code benchmark [4], which evaluates the ability of multimodal models to convert visual webpage designs into functional frontend code.

This benchmark is based on a dataset of 484 real-world webpages and combines automatic metrics with human evaluation to assess both visual fidelity and code correctness.

The results reported in these studies indicate that modern AI models achieve relatively strong performance in generating basic user interfaces, particularly for simpler layouts and structured designs. However, performance decreases significantly as the complexity of the interface increases. Specifically, models struggle with accurately capturing detailed visual elements, maintaining layout consistency, and preserving hierarchical relationships between components. These limitations highlight the gap between visual similarity and functional correctness, which is a recurring theme across benchmarking efforts.

3.1. COMPARISON OF MODEL PERFORMANCE

Recent benchmark leaderboards further illustrate the variation in performance across different models. For example, results from the Design2Code benchmark [4] show that leading models can achieve high overall scores, but with noticeable differences between them:

These results from Table 1 demonstrate that while top-performing models are approaching high levels of accuracy, there is still a significant performance gap between leading and mid-tier systems. Additionally, even high scores do not necessarily imply production readiness, as many evaluations rely on aggregate metrics that may not fully capture usability or maintainability concerns.

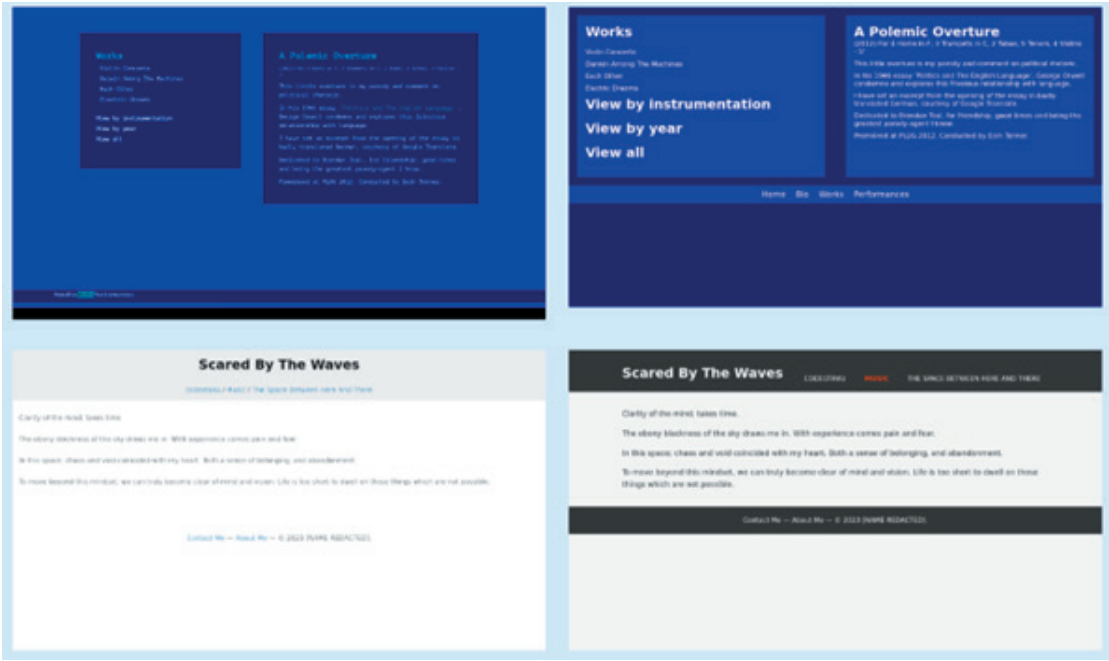

Figure 1. Example of Generation Examples

Table 1. Results for leading models

Model	Design-to-Code Accuracy
GLM-5V-Turbo	94.8%
Kimi K2.5	91.3%
Claude Opus 4.6	77.3%

Table 2. Evaluation Metrics

Metric	Description
Visual similarity	How closely generated UI matches the reference design
Structural accuracy	Correct hierarchy and layout of components
Code correctness	Functional validity of generated code

3.2. EVALUATION CRITERIA AND METRICS

Benchmarking studies typically rely on a combination of automated and human-centred evaluation metrics. In the context of frontend development, metrics are shown in the Table 2:

The Design2Code benchmark [4] uses metrics such as CLIP-based visual similarity and block-level structural evaluation, combined with human judgment to validate results. Importantly, studies report that automated metrics do not always align with human perception, suggesting that current evaluation approaches may not fully capture frontend quality.

3.3. MULTI-TASK AND FRAMEWORK-BASED BENCHMARKING

More recent research has attempted to address these limitations by expanding benchmarking approaches. The DesignBench framework [5] introduces a more comprehensive evaluation setup that includes multiple frontend frameworks (such as React, Vue.js, and Angular) and multiple task types, including code generation, editing, and repair. This reflects a more realistic representation of frontend development workflows, which are inherently iterative rather than static.

Findings from the study indicate that model performance varies significantly depending on the task type. While generation tasks generally achieve higher success rates, editing and repair tasks expose more weaknesses, particularly in maintaining consistency and understanding context across multiple iterations. These results suggest that current AI tools are better suited for initial code generation than for maintaining or evolving complex frontend systems.

4. CHALLENGES & LIMITATIONS

The analysis of existing studies on AI-assisted development reveals a set of limitations that exist across different tools and evaluation scenarios. One of the most frequently reported findings is the issue of reliability, particularly in relation to generated code correctness. While many tools demonstrate strong performance in structured and repetitive tasks, multiple studies indicate that they often produce outputs that are partially incorrect, incomplete, or based on non-existent functions and APIs [6], [7]. This pattern suggests that, despite high levels of syntactic accuracy, semantic correctness remains a significant challenge, requiring ongoing human verification. This distinction between syntactic and semantic correctness is well documented in evaluations of large language models for code generation, where models frequently generate plausible but functionally incorrect solutions [8].

Security-related concerns are also highlighted in several benchmarking studies. Research shows that AI-generated solutions do not consistently follow established security practices, occasionally introducing vulnerabilities or overlooking critical validation steps [9]. This is particularly evident in scenarios involving form handling, authentication logic, or data processing. Large-scale empirical analyses confirm the presence of widespread vulnerabilities in AI-generated code, spanning numerous Common Weakness Enumeration categories and often reflecting insecure patterns learned from training data [10]. These findings suggest that current AI tools lack a robust understanding of secure coding principles, reinforcing the need for cautious integration into development workflows.

Another important conclusion drawn from the reviewed literature is the inconsistency of AI-generated frontend code in terms of structure and adherence to design standards. Studies that evaluate UI-related tasks, such as component generation and styling, point out that while tools can produce visually functional results, they often fail to maintain consistency with design systems or established conventions. Research in design-to-code generation highlights issues related to layout structure, component hierarchy, and maintainability, even when visual similarity to the intended design is achieved [11].

Additionally, accessibility considerations are rarely handled adequately, as AI-generated interfaces often do not conform to established accessibility standards or fully support assistive technologies [12]. These findings indicate that AI tools do not yet fully account for the broader requirements of production-level frontend applications.

Finally, several studies point to the importance of human oversight in AI-assisted development. Although productivity gains are frequently reported, they are often accompanied by the need for manual corrections and iterative prompting. Empirical evaluations show that developers using AI tools still need to review, adjust, and validate generated outputs to ensure correctness and reliability [8], [13]. This indicates that AI tools function most effectively as supportive systems rather than autonomous agents.

Overall, the findings across the analyzed literature highlight a gap between the current capabilities of AI tools and the requirements of fully reliable, production-ready frontend development, which provides a foundation for identifying future research directions.

5. FUTURE WORK

The analysis of current research and benchmarking results indicates several directions for future work in the application of artificial intelligence to frontend development.

One of the identified research gaps relates to the evaluation of AI tools in frontend development. While recent studies have introduced more specialized benchmarking frameworks tailored to frontend tasks, such as multi-task and multi-framework evaluation approaches, these efforts remain limited in scope. Existing benchmarks often focus primarily on specific aspects such as UI code generation [4], [5], and do not fully capture the complexity of real-world frontend development workflows, which include iterative design, editing, and maintenance processes.

Another important direction involves improving the ability of AI systems to handle complex frontend scenarios. Existing studies indicate that current tools perform well in isolated and well-defined tasks but show reduced effectiveness when applied to more advanced challenges such as state management, asynchronous operations, and large-scale component interaction. Future work could explore approaches that enhance contextual understanding, enabling AI tools to operate more effectively within complete application architectures rather than at the level of individual components.

The reviewed literature also suggests the need for stronger integration between design and development processes. While initial progress has been made in generating code from design descriptions, the results remain inconsistent and often require significant manual adjustments. Further research could focus on improving design-to-code systems, enabling them to accurately translate visual and functional requirements into maintainable frontend implementations.

In addition, there is growing interest in the concept of intent-based development [14], where developers define high-level goals or user requirements, and AI systems generate corresponding interface solutions. Although still in early stages, this approach has the potential to significantly transform frontend workflows. Future studies could investigate how such systems can be made more reliable, interpretable, and adaptable to real-world development constraints.

Finally, the findings highlight the importance of redefining the role of developers in AI-assisted environments. Rather than replacing human expertise, future work should explore models of effective collaboration between the developer and AI, including best practices for interaction, validation, and control. Establishing clear guidelines for when and how AI tools should be used could help maximize their benefits while minimizing associated risks. Overall, these directions point toward a more structured and mature integration of AI into frontend development, where current limitations are addressed through targeted research and practical innovation.

6. CONCLUSION

This paper has examined the current state of AI in frontend development through a structured review of existing research and benchmarking studies. The findings indicate that AI tools have already established a meaningful role in supporting frontend workflows, particularly in tasks such as code generation, creating components, and basic debugging. Across multiple studies, these tools demonstrate clear benefits in terms of development speed and productivity, confirming their growing importance in modern software engineering practices.

At the same time, the analysis shows that the performance of AI tools is not consistent across all frontend scenarios. While strong results are reported for simpler and well-defined tasks, limitations become evident as complexity increases. Issues related to correctness, security, and adherence to design and accessibility standards are frequently observed, highlighting the gap between current capabilities and the requirements of production-ready applications. These findings reinforce the understanding that AI tools, in their current form, function more effectively as assistive technologies rather than independent development solutions.

By combining results from existing benchmarking studies, this paper has provided a clearer overview of how different AI tools perform within the context of frontend development. The review also highlights the lack of frontend-specific evaluation frameworks and the need for more targeted research in this domain. Based on these observations, several directions for future work have been identified, including the development of improved evaluation methods, better handling of complex frontend logic, and stronger integration between design and implementation processes.

In conclusion, artificial intelligence represents a significant and ongoing shift in frontend development, with the potential to reshape both workflows and developer roles. However, its effective adoption depends on a balanced approach that combines the strengths of AI systems with critical human oversight. Continued research and refinement will be essential in bridging the gap between current limitations and the vision of more autonomous and reliable AI-assisted development environments.

REFERENCES

[1] Y. Chen and L. Chen, "PSD2Code: Automated Front-End Code Generation from Design Files via Multimodal Large Language Models," Nov. 2025, doi: 10.48550/arXiv.2511.04012.

[2] Bulaja D, Stojiljković K, Živković M, Bačanin Džakula N, and Živković T, "Ant Colony Optimization Algorithm for Frontend Resource Prioritization," in *Sinteza 2025 - International Scientific Conference on Information Technology, Computer Science, and Data Science*, 2025, pp. 16–22. doi: 10.15308/Sinteza-2025-16-22.

[3] J. Budihartanto, G. D. Julianto, S. R. Manalu, and H. A. Shiddiqi, "Comparative Analysis of Gemini 2.0 Flash, Microsoft Copilot, and ChatGPT-4o's Accuracy in Translating UI Images into Front-End Web Code," *Procedia Comput. Sci.*, vol. 269, pp. 834–843, 2025, doi: 10.1016/j.procs.2025.09.026.

[4] C. Si, Y. Zhang, R. Li, Z. Yang, R. Liu, and D. Yang, "Design2Code: Benchmarking Multimodal Code Generation for Automated Front-End Engineering," Feb. 2025, doi: 10.48550/arXiv.2403.03163.

[5] J. Xiao et al., "DesignBench: A Comprehensive Benchmark for MLLM-based Front-end Code Generation," Mar. 2026, doi: 10.48550/arXiv.2506.06251.

[6] M. Chen et al., "Evaluating Large Language Models Trained on Code," Jul. 2021, doi: 10.48550/arXiv.2107.03374.

[7] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?," in *FAccT 2021 - Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, Association for Computing Machinery, Inc, Mar. 2021, pp. 610–623. doi: 10.1145/3442188.3445922.

[8] N. Nascimento, P. Alencar, and D. Cowan, "Comparing Software Developers with ChatGPT: An Empirical Investigation," May 2023, doi: 10.48550/arXiv.2305.11837.

[9] M. Schreiber and P. Tippe, "Security Vulnerabilities in AI-Generated Code: A Large-Scale Analysis of Public GitHub Repositories," Oct. 2025, doi: 10.1007/978-981-95-3537-8_9.

[10] B. Wang et al., "AI Code in the Wild: Measuring Security Risks and Ecosystem Shifts of AI-Generated Code in Modern Software," Dec. 2025, doi: 10.48550/arXiv.2512.18567.

[11] H. H. Zhang et al., "Widget2Code: From Visual Widgets to UI Code via Multimodal LLMs," Mar. 2026, doi: 10.48550/arXiv.2512.19918.

[12] G. Vera-Amaro and J. R. Rojano-Cáceres, "Towards accessible website design through artificial intelligence: A systematic literature review," *Inf. Softw. Technol.*, vol. 186, p. 107821, 2025, doi: 10.1016/j.infsof.2025.107821.

[13] M. S. I. Ovi, N. Anjum, T. H. Bithe, Md. M. Rahman, and Mst. S. A. Smrity, "Benchmarking ChatGPT, Codeium, and GitHub Copilot: A Comparative Study of AI-Driven Programming and Debugging Assistants," Sep. 2024, doi: 10.48550/arXiv.2409.19922.

[14] Z. Ding, "Towards Intent-based User Interfaces: Charting the Design Space of Intent-AI Interactions Across Task Types," May 2024, doi: 10.48550/arXiv.2404.18196.