**International Journal of Research Publication and Reviews**

Journal homepage: www.ijrpr.com ISSN 2582-7421

# FROM PROMPTS TO PROJECTS: EVALUATING AI'S ROLE IN FULL-STACK WEB DEVELOPMENT FOR STUDENTS

***Parth Mahadik[a], Atharva Shirishkar[b], Shonal Vaz[c], Yogita Khandagle[d]***

[a]Department of Information Technology, Vidylankar Polytechnic, Mumbai, India

ABSTRACT :

Artificial Intelligence (AI) has become an essential part of our lives, and it is experiencing significant growth across various applications. As a student it is crucial to keep up with modern technology and learn how to utilize it effectively. In this paper, we present a brief study on the role of AI in a student's learning and development journey, particularly in full-stack web development. We explore effective ways to use AI , where it excels, and its limitations, with a focus on prompt engineering. Additionally, we discuss how AI benefits students such as enabling faster development, providing exposure to best coding practices, and more as well as the challenges they might face, including the risk of plagiarism and difficulties in understanding complex concepts without AI assistance. Finally, we examine the extent to which students should rely on AI and how frequently they should use it

**Keywords**: Artificial Intelligence, AI, Prompt Engineering, Full-Stack Web Development, Student Learning, Web Applications

## 1. Introduction :

With the increasing accessibility of AI-driven coding assistants, students now have advanced tools to enhance their learning experience. AI has become an integral part of various applications, with software development experiencing one of the most significant transformations. Many students prefer AI-driven assistance over traditional documentation when developing programs, as it offers faster solutions, clearer explanations, and optimized executions.

However, while instant access to solutions can speed up development, it may also hinder deep learning and problem-solving skills. Traditionally, students learn best by writing, executing, and debugging code manually, which reinforces a deeper understanding of programming concepts. Yet, AI assistance provides distinct advantages, enhancing learning efficiency when used appropriately.

Beyond simply using AI in development, how AI is utilized is a critical aspect that needs exploration. In this paper, we examine "StallSpot," a project developed with AI assistance, as a case study to evaluate the impact of AI-driven tools on learning, efficiency, and dependency. Various AI models were integrated for different tasks:

- OpenAI's ChatGPT for structuring discussions and brainstorming development strategies.
- Anthropic's Claude for code generation and debugging.
- Vercel's V0 for frontend component generation.

This study seeks to answer a key question: Does using AI accelerate learning, or does it create dependency?

This paper presents an analysis of AI-assisted development based on our personal experience using tools like GPT, Claude, and Bolt.new while building StallSpot. While these tools significantly influenced our workflow, their effectiveness may vary depending on individual proficiency, project complexity, and the nature of AI interactions.

## AI-ASSISTED FULL-STACK DEVELOPMENT :

### *1.1. Challenges Faced*

Before AI, beginners struggled with structuring projects, debugging, and understanding complex frameworks. AI tools now provide real-time solutions, allowing students to learn more efficiently. Before AI, beginners struggled with structuring projects, debugging, and understanding complex frameworks. AI tools now provide real-time solutions, allowing students to learn more efficiently. Common challenges included:

- Difficulty in writing optimized React and Next.js code
- Debugging complex application logic
- Understanding API integration and backend development

### *1.2. AI-Driven Development*

The following insights are derived from our hands-on experience in developing StallSpot using various AI models. The effectiveness of these tools may differ for others depending on how they interact with AI, structure their prompts, and validate AI-generated outputs.

AI-powered platforms were extensively used in StallSpot to streamline the development process. The following tools played a significant role:

### *1.3. ChatGPT – Discussions and Planning*

ChatGPT was primarily used for **structuring the project and making key decisions** rather than direct code generation. We leveraged it for:

- **Project architecture discussions:** Choosing the appropriate programming languages, database, APIs, and extensions.
- **Best practices:** Identifying good coding standards to ensure production-quality software.
- **Feature planning:** Deciding the flow of the website, the number of user roles, and their access levels.

However, ChatGPT was **not used for coding** due to its limitations, such as **generating excessive content, struggling with project-specific implementation, and quickly hitting token limits**.

### *1.4. Claude Sonet – Code Generation and Structuring*

Claude Sonet emerged as the most effective AI for **writing, debugging, and structuring code** in StallSpot. It was particularly useful because:

- **Higher accuracy in code generation** – It produced functional code with minimal errors.
- **Longer conversation memory** – Allowed us to work on extensive project parts without losing context.
- **Context-aware suggestions** – It understood project structure when given a **Markdown file of the folder structure**.

#### *2.4.1 Prompt Engineering for Claude Sonet*

To get the best results, we followed a structured prompting strategy:

1. **Preventing assumptions:** AI models tend to generate generic code that may not align with the actual project schema. We explicitly instructed Claude **not to assume unknown details**.
2. **Asking clarifying questions:** Before generating code, we encouraged Claude to ask relevant questions, ensuring **accurate and customized solutions**.
3. **Step-by-step guidance:** Instead of providing full code blocks, we instructed Claude to generate **stepwise implementation instructions**, making it easier to integrate the code while understanding it.
4. **File path references:** By specifying file paths in responses, Claude reduced confusion when integrating new files into an expanding project.
5. **Using Markdown files:** Regularly updating Claude with the latest folder structure helped it recognize changes and **avoid outdated references**.

By using these strategies, **Claude significantly improved our development workflow**, though it had occasional drawbacks, such as **slower responses with longer prompts and occasional out-of-context suggestions**.

### *1.5. Bolt.new and V0 – Frontend and Backend Integration*

Bolt.new and V0 were primarily used for **frontend component generation**, particularly in maintaining a **uniform design using the ShadCN UI library**. However, V0 also contributed to **backend development**, particularly for setting up the **payment gateway and aligning with Next.js 15 best practices**. Since Claude's knowledge was limited to April 2024, V0 proved useful for generating **updated code** and resolving issues where Claude failed due to outdated references.

### *1.6. GitHub Copilot, Cody, and Sourcegraph Models – Code Optimization*

GitHub Copilot, Cody, and Sourcegraph's AI models played a crucial role in **identifying and fixing minor coding issues**. Their **IDE integrations** allowed them to:

- Detect **unused variables, type mismatches, and schema-related errors**.
- Suggest corrections based on the **entire project context**, eliminating the need to manually provide background information.
- Provide **inline code completions and improvements**, making debugging more efficient.

These tools were particularly beneficial because they **analyzed the entire source code directly**, unlike GPT and Claude, which required manual context input.

### *1.7. AI in StallSpot's Development Process*

We used AI tools for almost every aspect of **StallSpot**, from structuring the database to implementing key features. Rather than manually writing code from scratch, we **described the problem statement, defined the requirements, and outlined the website's flow**, allowing AI to generate initial implementations. This approach saved significant time but also required **constant validation and debugging**.

## 2. IMPACT ON LOGICAL THINKING & CODE UNDERSTANDING :

Using AI-driven tools has significantly impacted logical thinking. While crafting precise prompts and defining problem statements relies heavily on language skills and clarity, understanding coding logic remains essential. AI can generate functional solutions, but it assumes prior knowledge of coding practices, such as environment variables, API keys, and security principles. Without foundational software knowledge, AI-generated solutions may become ineffective or difficult to implement.

The biggest challenge arises in mathematical computations, algorithmic logic, and formula-based implementations. While AI can explain complex algorithms, translating that knowledge into working code without AI assistance remains difficult. We often found that while we understood the theory behind implementations, independently coding them proved challenging. This highlights a gap where AI improves conceptual understanding but does not always reinforce hands-on problem-solving skills.

Additionally, while AI streamlines debugging by detecting errors and suggesting corrections, manual debugging remains crucial for developing strong analytical skills. AI-generated solutions may not always align with project requirements, requiring students to critically evaluate and modify suggestions. Over-reliance on AI can lead to difficulty in troubleshooting errors independently, highlighting the need for a balanced approach.

While AI-generated solutions often follow best practices, students may implement them without fully understanding the logic behind them. This is particularly evident in code optimization, where AI suggests performance improvements without explaining the underlying reasoning. Without careful review, students may struggle to apply these optimizations in different contexts, reducing their ability to write efficient code independently.

These observations are based on our personal experience using AI in the StallSpot project. The impact of AI on logical thinking and code understanding may vary depending on an individual's prior knowledge, learning style, and level of engagement with AI-generated solutions.

## 4. BENEFITS AND DRAWBACKS :

### *4.1 Benefits of AI in Full-Stack Development*

AI has provided several advantages in the development of **StallSpot**, making the process more efficient and structured. Some of the most significant benefits include:

- **Time Efficiency & Increased Productivity:** AI significantly reduced development time by generating boilerplate code, suggesting best practices, and assisting with debugging. This allowed us to focus on high-level logic rather than syntax-related issues.
- **Error Reduction & Debugging Support:** AI tools not only detected errors but also explained them, helping us quickly identify and fix issues. Debugging became faster, and reasoning-based explanations made it easier to understand where problems originated.
- **Exposure to Advanced Concepts:** AI introduced us to new methodologies, frameworks, and coding standards that we might not have explored otherwise. This enhanced our overall knowledge of web development.
- **Step-by-Step Guidance for Implementation:** AI provided structured steps to integrate code into our project, making implementation smoother and ensuring best practices were followed.
- **Improved Problem-Solving Abilities**: By breaking down complex problems, AI allowed us to explore multiple solutions and better understand how different approaches could be implemented in our project.

### *4.2 Drawbacks & Challenges of Using AI*

While AI has been highly beneficial, it also comes with limitations and challenges that must be considered:

- **Contextual Errors & Repetitive Outputs:** AI sometimes provided the same output despite changes in context, requiring repeated prompt modifications to get the desired result.
- **Assumptions & Missing Details:** AI often assumed the presence of certain variables, APIs, or environment keys, which were not actually set up in our project. This created unexpected errors when implementing AI-generated code.
- **Limited Understanding of Complex Mathematics & Algorithms:** While AI explained concepts well, implementing complex mathematical formulas and algorithmic logic without AI assistance was significantly harder. Understanding was not enough; implementing from scratch remained difficult.
- **Over-Reliance Risk:** Using AI for every problem can reduce independent thinking and problem-solving skills, especially when students do not attempt solutions manually before consulting AI.

- **Occasional Confusion in Debugging:** Although AI generally improved debugging, there were instances (about 2 in 10 cases) where it provided misleading explanations, requiring additional manual verification.

## 5. How Frequently Should Students Use AI?

Based on our experience, **AI should be used strategically rather than frequently**. We recommend:

- **Try solving problems independently first.** Students should attempt a solution at least **3-4 times** before seeking AI assistance.
- **Use AI when you have a strong foundational understanding** of the programming language. Without basic knowledge, AI-generated solutions may be difficult to integrate effectively.
- **Use AI for learning best practices, but not as a replacement for manual coding.** AI should enhance learning, not replace critical thinking.
- **Balance AI assistance with hands-on debugging.** While AI is great for quick fixes, manually identifying and solving errors strengthens problem-solving skills.

These recommendations are based on our direct experience with AI in StallSpot and may vary depending on individual learning styles and project complexity.

## 6. Conclusion.

The integration of AI in full-stack web development has proven to be a transformative experience. In our project **StallSpot**, AI tools played a significant role in enhancing productivity, reducing errors, and providing exposure to advanced web development concepts. Through this experience, we have observed that AI can serve as a powerful learning aid when used correctly.

AI met our initial expectations by streamlining the development process and improving our overall efficiency. While no unexpected challenges were encountered, we found that AI influenced our development skills in a unique way—it made us **better at designing and structuring projects** while simultaneously creating **some level of dependency in coding tasks**. This raises a key consideration: **AI should not replace manual coding efforts but rather be used as a supportive tool for improving skills and understanding.**

If given the opportunity to start another project, we would adopt a different approach. AI would be used **as an answer key** rather than a direct code generator. Instead of relying on AI-generated solutions upfront, we would first attempt to **solve problems independently** before consulting AI. This method would allow us to **enhance our skill set, become more self-sufficient developers, and improve our job readiness**.

Ultimately, we conclude that **AI can enhance the learning process if used with discipline and strategy**. It should be treated **as a guide, not as an unlimited source of ready-made answers**. Students must establish strict usage boundaries to prevent AI from negatively impacting their logical reasoning and problem-solving abilities. However, it is undeniable that AI has become an **essential part of modern development workflows**, and learning how to integrate it efficiently is now a necessary skill.

Moving forward, future research could explore how AI can be optimized for **personalized learning experiences**, helping students develop a **balance between automation and manual problem-solving**.

***Acknowledgements***

We would like to express our sincere thanks to our guide Er. Shonal Vaz and our Head of the Information Technology Department Er. Yogita Khandagle and all the staff in the faculty of Information Technology Department for their valuable assistance.

REFERENCES :

[1] A. Smith and J. Brown, "The Role of AI in Software Development," IEEE Transactions on Software Engineering, vol. 47, no. 3, pp. 450-465, Mar. 2023.

[2] M. Patel, "Enhancing Student Learning Through AI: A Case Study," in Proceedings of the IEEE International Conference on Education Technology (ICET), 2022, pp. 112-118.

[3] OpenAI, "ChatGPT Documentation," 2024. [Online]. Available: https://openai.com/docs

[4] Anthropic, "Claude AI Documentation," 2024. [Online]. Available: https://www.anthropic.com

[5] Vercel, "V0: AI-Powered Web Development," 2024. [Online]. Available: https://vercel.com/docs

[6] GitHub, "GitHub Copilot Documentation," 2024. [Online]. Available: https://docs.github.com/copilot

[7] Sourcegraph, "Cody: AI Code Search and Assistance," 2024. [Online]. Available: https://sourcegraph.com/cody

[8] "Artificial Intelligence and the Future of Teaching and Learning," U.S. Department of Education, 2023. [Online]. Available: https://www.ed.gov/sites/ed/files/documents/ai-report/ai-report.pdf

[9] R. Kumar, "The Impact of AI and Automation on Software Development," in IEEE Chicago, 2023. [Online]. Available: https://ieeechicago.org/the-impact-of-ai-and-automation-on-software-development-a-deep-dive/

[10] B. Wilson, "The Future of Software Engineering in an AI-Driven World," in arXiv, 2024. [Online]. Available: https://arxiv.org/abs/2406.07737

[11] S. Gupta, "AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions," Applied Sciences, vol. 15, no. 3, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/15/3/1344

[12] T. Anders, “Artificial Intelligence in Higher Education: The State of the Field,” International Journal of Educational Technology, 2023. [Online]. Available: https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-023-00392-8
[13] M. Lopez, “A Review of Artificial Intelligence in Education,” in Proceedings of the International Conference on AI and Education, 2023. [Online]. Available: https://libraetd.lib.virginia.edu/downloads/6d56zx83j