American Journal of Applied Research and AI (AJARAI)

VOLUME 1 ISSUE 1 (2026)

PUBLISHED BY
E-PALLI PUBLISHERS, DELAWARE, USA

American Journal of
Applied Research and AI (AJARAI))

Volume 1 Issue 1, Year 2026
https://journals.e-palli.com/home/index.php/ajarai

Vibe Coding in Nepal: Opportunities and Challenges in Leveraging AI tools for Software Development

Sujan Poudel[1], Shyam Maharjan[1*], Khilanath Luitel[1]

Article Information

Received: November 09, 2025
Accepted: February 16, 2026
Published: April 07, 2026

Keywords

AI Tools, Code Quality, Nepal, Software Development, Vibe Coding

ABSTRACT

AI tools enable users to generate executable code from natural-language descriptions via a technique known as vibe coding. Developers can quickly create software using vibe coding, which lowers the barrier to entry for non-developers and democratizes the creation of software. Vibe coding is likely to greatly enhance the ability of entrepreneurs, small businesses, and individuals in Nepal who have limited opportunities to pursue formal programming education to quickly design functional applications. However, because it employs a fast-paced development approach, there are concerns about the quality, security, and long-term maintainability of implementation of code by users without a background in software technology. The benefits and challenges of vibe coding in Nepal will be examined in this study to include an analysis of user experiences and its overall impact on software development practices and the growing technology ecosystem in Nepal while considering the inherent risks associated with the adoption of vibe coding to ensure a balance between accessibility and security.

INTRODUCTION

New tools based on artificial intelligence (AI), especially large language models (LLM), are also radically changing software development. These tools allow novice and experienced developers to describe how a program should work using only natural language, with the AI writing the executable code to perform that functionality. LLMs have significantly reduced the time it takes for developers to complete their work and broken down barriers preventing those without programming experience from working on software; they also encourage the development of programs through agreeable coding, vibe coding for short. Vibe coding is fundamentally different than conventional coding, where coding is done through writing structured and detailed lines of code; vibe coding is done by thinking of the desired results and experimenting to find the correct code. The advantage of this method is that it is quick to build software and does not require a thorough understanding of the code. All that is needed to find out if you have built the software correctly is to think of what you want and let the AI help you figure out the coding needed to accomplish it (Sapkota *et al.,* 2025).

With countries like Nepal, where formal programming training is hard to come by, yet there is a high demand for software developers, Vibe coding has great potential. It enables individuals with no coding experience (e.g., entrepreneurs and small business owners) to create functional applications quickly and easily without extensive coding knowledge or skills. At a time when we are all becoming more dependent on technology, having access to locally developed solutions (technological) is critical. The ability to create software quickly and relatively easily will empower many people from a wide range of backgrounds/ skill sets in Nepal to innovate, prototype and actualize their ideas, and ultimately contribute to the country's digital transformation (Chow & Ng, 2025; Ray, 2025).

However, the large-scale implementation of vibe coding has caused significant concerns about the quality, security, and sustainability of the automatically- generated source code. Many people, especially those without formal programming backgrounds, use AI-generated source code but do not have enough knowledge to understand it and, as a result, they often fail to follow the necessary steps (i.e., testing and debugging). In Nepal, where the software quality assurance (QA) infrastructure may not be as well-developed as in other areas, this will create weaknesses, technical debt, and unstable software (De Silva *et al.,* 2025; Krings *et al.,* 2025). This article focuses on the implementation of vibe coding in Nepali context by discussing its advantages and disadvantages. The goal of this project is to determine how vibe coding can be used responsibly in Nepal's expanding technology sector, resulting in a product that is available to all individuals and secure.

LITERATURE REVIEW

Ray (2025) reported that vibe coding enables the entire life cycle of computer programming using high-level natural language interaction involving multi-agent pipelines, Retrieval-Augmented Generation, and dynamic context management. From the literature review, four interaction styles that affect the role of collaboration between developers and AI in computer programming are identified, including the idea that vibe coding enhances prototyping, ease of use, and collaboration, but risks exist in the form of hallucinations, technical debt, security and governance issues, and knowledge decay.

[1] Nepal Kasthamandap College, Nepal
[*] Corresponding author's email: samymhr31@gmail.com

Sapkota *et al.* (2025) presented a comparative study that introduces vibe coding and agentic coding as complementary paradigms in the application of AI technology in software development. According to the comparative study, there is an indication that vibe coding is most appropriate for idea generation, experimenting, early prototyping, and educational purposes. This is because the paradigm supports user-friendly processes that involve humans. Agentic coding, on the other hand, supports a high level of autonomy when applied in enterprise automation, large-scale refactoring, and CI/CD. Mitchell and Shaaban (2025) indicated that despite vibe coding's adoption being quite rapid in resolution, it has been hindered by issues such as technical debt, security weaknesses, and code churn. This research makes it clear that such problems emanate from the inability of LLM models to handle conflicts in constraints provided by humans as well as developers' poor ability to resolve such conflicts. On the other hand, it clearly indicates through its findings the ineffectiveness of current integrations with regard to reliability and formal methods for verification-based feedback.

Sarkar and Drosos (2025) explored that vibe coding follows iterative goal-satisfaction cycles in which developers alternate between prompting AI, evaluating generated code, testing, and selective manual editing. The findings show that prompts blend high-level intent with technical detail, while debugging remains a hybrid process combining AI assistance and manual practices. Importantly, the study finds that vibe coding does not replace programming expertise but redistributes it toward context management, rapid code evaluation, and situational trust calibrated through continuous verification.

Fortes-Ferreira *et al.* (2025) analyzed that vibe coding is an effective approach that can reduce the barriers of entry to coding for new programmers because it is based on natural language interaction with an LLM and enables creative software development techniques. It is evident from this study that by use of iteration, there is an improvement in functionality and graphical representation such that new programmers with no prior experience can develop sophisticated applications. However, the study indicates that there are still pitfalls, such as illogically written code, frequent reprocessing of instructions, and the use of programming knowledge for debugging functions.

Hemdev (2025) showed that vibe coding, as a conversational and AI-driven method of software development, greatly speeds up prototyping by about 60–80 percent while amplifying creativity across real-world applications. The results indicate that EconoFacts, BusinAI, and SimuStocks are actual tools gaining from rapid iteration and the exploration of ideas through LLM-guided code generation. However, the study also reveals that active human supervision is indispensable to guarantee code quality, security, and long-term maintainability. The overall results of the findings indicate that vibe coding constitutes a very strong facilitator in its initial development stages but is utterly unreliable without explicit developer activity.

De Silva *et al.* (2025) examined that vibe coding offers an instant speed benefit in prototyping industrial systems by trusting the entire coding and implementation task to AI agents via a conversation. The result clearly explains that vibe coding decreases cognitive overhead and time in industrial prototyping compared to traditional prototyping, but the risk revealed by the study encompasses code quality, difficulty in debugging, and AI dependence, as these aspects call for well-defined best practices. Overall, the study revealed that vibe coding helps in industrial prototyping tasks by speeding them up, but proper risk management and human control are required for successful implementation.

Bamil (2025) demonstrated that vibe coding is an AI-native paradigm for programming, whereby programmers articulate high-level functional intention along with "vibe" descriptors for intelligent agents to produce executable code. The results reveal that it is feasible to have a reference architecture involving intent parsing, semantic embeddings, agent code production, and interaction feedback to enable this process. The results of this study reveal benefits of productivity gain and democratization to reduce entry barriers for software production. However, this literature review has found some key issues to emerge, such as alignment, reproducibility, bias, maintainability, and security, which if left unattended, can lead to vulnerabilities and bottlenecks in this process. The results of this research reveal that though vibe coding enhances collaboration and possibilities of human AI partnership in software development, it calls for careful design principles to make it sustainable for adoption by society.

Chow and Ng (2025) showed that vibe coding can successfully fill a gap in healthcare training as a way for experts in learning simulation development without coding. The case study has revealed that by utilizing no-code AI platforms, there has been successful rapid prototyping within educational applications, shifting the emphasis towards pedagogical goals rather than technical restrictions. Educational applications such as the "Differential Diagnosis " or "The Insulin & Blood Sugar Simulation" have successfully highlighted improvements in accessibility, scalability, and cost-effectiveness, contributing simultaneously towards the adaptation of clinical reasoning skills. Overall, evidence suggests that vibe coding enables educators in actively designing personalized versions of healthcare simulation that potentially influence successful educational outcomes.

Meske *et al.* (2025) observed that vibe coding entails a paradigm shift in software development by adjusting the balance of epistemic labor between humans and generative AI in cooperative natural language interaction. The data reveals that expertise of software developers shift from direct technical executory action to orchestration, articulation of intent, and cooperation with AI systems. The key findings have showed important opportunities,

such as democracy in software development, rapid development, and better leverage. But at the same time, important risks have been uncovered, such as "opaque source code generated by AI systems," accountability, and bias at an ecosystem level. The key findings have revealed that it is important to understand paradigm shift in intent mediation in relation to vibe coding.

Pimenova *et al.* (2025) studied that vibe coding is a conversational, co-creative programming paradigm strongly associated with flow, experimentation, and enjoyment. In this large-scale qualitative analysis, the research shows how developers use vibe coding to balance delegation and co-creation, while trust in AI dynamically controls this movement along a continuum of control. The findings identify recurring challenges concerning specification clarity, reliability, debugging, latency, code review burden, and collaboration. However, the study also documents best practices that come from the community and mitigate these risks, sustaining productive developer experiences. Overall, the findings suggest that vibe coding's effectiveness depends on managing AI trust, supporting flow, and embedding practical safeguards within future AI development tools.

Krings *et al.* (2025) showed that vibe coding embodies an intuitive and affective-oriented paradigm for programming, which reconfigures the ways in which developers engage with generative AI tools. In their qualitative research inspired by itemized interviews conducted with experienced developers, it has been demonstrated that vibe coding stimulates creativity, rapid prototyping, and novel collaboration modes by focusing on improvisational interaction instead of guided instruction. In addition to this, it has been found in their research that vibe coding redefines the role of developers by dissolving the traditional boundary between developers and non-developers. Nevertheless, their research has identified major issues regarding reproducibility, scalability, and inclusiveness.

Gadde (2025) demonstrated that the application of generative AI within software development through a process such as vibe-coding combined with no-code and low-code development holds immense potential for software development democratization. This discussion clearly reveals through this study that generative AI bridges the gap between expressing a need through natural language and developing functional software by reducing the otherwise significant entry barrier within software development. Yet, the same findings unveil challenges being witnessed by the software development sector through generative AI.

According to Poudel & Maharjan (2025), there is an observed rise in the use of artificial intelligence among Nepalese students, who show high levels of awareness and usage, especially in urban settings; yet, significant gaps remain between the use of AI among urban and rural students due to a lack of digital infrastructure, lower digital literacy, and a lack of institutional support. This study, conducted using a mixed-methods approach, concluded that despite students' views on AI's positive impact on learning efficiency, academic productivity, and equity in education, its successful implementation requires teacher readiness, integration with the curriculum, and culturally suitable AI tools. This information is particularly pertinent to the Nepalese vibe coding scenario, where AI-based software development also requires digital literacy, equity in access, and responsible implementation, indicating that unless specific infrastructure development and capacity-building efforts are made, the benefits of AI-based programming practices may also be inequitably distributed.

MATERIALS AND METHODS

This research adopted a qualitative methodology to examine the phenomenon of vibe coding in Nepal, with a focus on the motivations, experiences, and problems encountered by users of AI-powered coding tools. The data was collected using in-depth interviews with 41 participants, who were chosen using convenience-based sampling. The participants were chosen based on their familiarity with AI-powered tools. A semi-structured interview schedule was developed to examine different facets of vibe coding, such as the motivations for its usage, the scenarios in which it is most effective, and the problems encountered, such as the difficulties in understanding and manipulating the AI-generated code, as well as the problems encountered in debugging, security, and code quality. Probing was used to examine the responses in more detail, to gain insights into the underlying problems and to gain a deeper understanding of the experiences of the users.

The data was analyzed using thematic analysis. This involved familiarizing the research team with the data, developing initial codes, and then determining important themes like the rate of development, ease of use, and difficulties associated with code quality and security. The ethical standards were maintained throughout the research, and informed consent was obtained from the participants. The research was conducted using a combination of qualitative and quantitative research methods. This enabled the researchers to gain a complete understanding of the potential and limitations of vibe coding, especially in the Nepalese context, where there is limited access to programming education but a growing need for software development.

RESULTS AND DISCUSSION

The data from the survey shows a clear preference for the speed of development, with 82.9% of the respondents choosing this option. This is an indication that most of the users of AI tools such as vibe coding are mainly driven by the need to speed up their development process, thus taking less time to develop functional code. Ease of use was also a major consideration, with 63.4% of the respondents choosing this option as a reason for using AI code generation tools. This is an indication that the ease of use and friendliness of these tools make them very attractive to users, regardless of their level of expertise.

Table 1: Motivation to choose Vibe coding

Option	Frequency (Responses)	Percentage (%)
Speed of development	34	82.9%
Ease of use	26	63.4%
Creativity in building projects	18	43.9%
Learning and improving coding	5	12.2%
To experiment with code ideas	5	12.2%
Doing on your own pace	1	2.4%

Learning and improving coding skills (12.2%) and trying out code ideas (12.2%) are also major considerations for users of AI code generation tools such as vibe coding. This is an indication that although these tools offer users a chance to learn and improve their coding skills, they are not the main reason why most of the users are attracted to the tools. Creativity in building projects is also a major consideration, with 43.9% of the respondents choosing this option. This is an indication that vibe coding offers users the freedom to new ideas.

Table 2: Contexts Where Vibe Coding is Most Useful

Option	Frequency (Responses)	Percentage (%)
Rapid prototyping	28	68.3%
Personal projects	11	26.8%
Professional development	19	46.3%
Collaborating with teams	14	34.1%

The most useful context for vibe coding is rapid prototyping, as found by 68.3% of the respondents. This shows that users appreciate the speed at which they can implement and test their ideas, which is a major advantage of vibe coding. Professional development is another useful context, as found by 46.3% of the responses. This shows that vibe coding is considered to be a useful tool for enhancing one's skills in the workplace, possibly for improving coding skills or making work-related projects.
Team collaboration was another useful context, as found by 34.1% of the users. This shows that the team collaboration aspects of vibe coding, such as allowing multiple users to collaborate on code easily, are considered to be a major advantage. Interestingly, Personal Projects was the next useful context, as found by 26.8% of the responses. This shows that vibe coding is used by some programmers for personal projects, although it is not the most useful context for most users.
Finally, Other (please specify) received no responses, which shows that the contexts listed above encompass the major ways in which vibe coding is used.

Table 3: Experience When Using Vibe Coding

Option	Frequency (Responses)	Percentage (%)
Easy and intuitive	19	46.3%
Somewhat difficult, but manageable	15	36.6%
Challenging, with frequent issues	6	14.6%
Frustrating, leads to abandoned projects	1	2.4%

A large number of participants (46.3%) found their experience with vibe coding easy and intuitive, which shows that most people find the tool easy to use and intuitive. This also shows that vibe coding is a preferable option for people who want a seamless and efficient development process. The next most common experience was a little difficult, but manageable (36.6%), which shows that some people may face difficulties in using vibe coding, but they can still overcome the difficulties and use the tool effectively. This also shows that vibe coding may have a learning curve or may sometimes be complex, but the users find it manageable.
Difficult, with frequent problems was the next option selected by 14.6% of the participants, which shows that some people may face frequent problems while using vibe coding, which may affect their development process. This also shows that vibe coding may not be the best option for everyone, and the tool's effectiveness may depend on the user's experience or the project complexity.
Finally, only 1 participant (2.4%) found the tool frustrating and leading to abandoned projects, which shows that for some people, vibe coding may not be an effective tool, and it may cause them to abandon projects due to difficulties in using the tool.

Table 4: What Worked Well During Vibe Coding

Option	Frequency (Responses)	Percentage (%)
Speed of generating code	17	41.5%
Creativity and exploration of new ideas	13	31.7%
Low barrier to entry for non-developers	11	26.8%

The most prominent problem that the users encountered during vibe coding is the understanding or changing of the AI code, which was chosen by 63.4% of the users. This indicates that although vibe coding is helpful, the users are struggling with the results of the AI and are having trouble interpreting or adjusting the code to fit their needs. The problem of the code output being inaccurate or buggy was another problem that drew the attention of 48.8% of the users. This shows that although the code generated by vibe coding is extremely fast and

efficient, there are times when the code does not behave as it should, which can be a reason for concern for the users who require accurate and correct code output.
Troubleshooting/debugging the code generated by AI ranked 53.7%, which shows that most people face problems while trying to debug/troubleshoot the code generated by AI. Debugging the code generated by AI can itself be a problem, as it is not always an easy and simple task. Security or performance issues were also a major concern for 46.3% of the users, which shows that there are concerns about the reliability and security of code generated using AI tools, especially when it comes to applications where security is a major concern.
Lastly, security and sometimes hallucinations was the least selected challenge, with only 1 response (2.4%). This shows that while hallucinations (code generated by AI that doesn't make sense) may happen, it is not a common problem for most users.

Table 5: Challenges During Vibe Coding

Option	Frequency (Responses)	Percentage (%)
Inaccurate or buggy code outputs	20	48.8%
Difficulty understanding or modifying AI-generated code	26	63.4%
Problems with debugging or troubleshooting AI-generated code	22	53.7%
Security or performance issues	19	46.3%
Security and sometimes hallucinations	1	2.4%

The most prominent problem faced by users in vibe coding was the understanding or modification of AI code, which was chosen by 63.4% of the respondents. This indicates that despite the benefits of vibe coding, users are struggling with the output of the AI and are having difficulty interpreting or modifying the AI code to suit their requirements. Inaccurate or buggy code output was another prominent problem for 48.8% of the respondents. This indicates that despite the efficiency of vibe coding, there are instances when the code generated does not work as expected, leading to frustration for developers who require accurate and precise results.
Debugging or troubleshooting AI code was ranked 53.7%, indicating that many users had issues when attempting to correct errors or bugs in the code generated by the AI. Debugging AI code may be a challenging task, as it may not always be transparent or clear. Security or performance problems were also a problem for 46.3% of the users, indicating that despite the efficiency of AI code generation tools, there are issues with the reliability and integrity of the code generated, especially when it comes to applications requiring high security.
Finally, security and sometimes hallucinations was the least selected challenge, with only 1 response (2.4%). This indicates that while hallucinations (AI-generated content that doesn't make sense) may occur, it is not a widespread concern for most users.
The majority (70.7%) of those surveyed believed that the code written with AI had high quality, was modular and reusable. This suggests that most users believe that the code they receive from AI will last long enough to be of value. The second most chosen response was function, but with problems - which 17.1% of all respondents chose. This strongly indicates that code as produced is working, but not perfectly, and that there are some issues or bugs that still need to be worked out.
12.2% of responses indicated that the code produced is prone to errors and frailty. This shows that some of the users believe that the code produced by AI is unstable and would require significant changes to operate correctly.
However, Zero respondents chose unusable without making considerable changes; indicating that code produced by AI can generally be used without any considerable changes.

Table 6: Rating the Quality of AI-Generated Code

Option	Frequency (Responses)	Percentage (%)
High quality, clean, and maintainable	29	70.7%
Functional but flawed	7	17.1%
Error-prone and fragile	5	12.2%
Unusable without significant modifications	0	0%

Table 7: Perception of AI-Generated Code for Production Applications

Option	Frequency (Responses)	Percentage (%)
Yes, it's production-ready	25	61%
No, it requires review and modifications before use	15	36.6%
I wouldn't use it for production without thorough testing	1	2.4%

According to a survey conducted among the respondents, 61% stated that they trust AI generated code is ready for use in production by being reliableenough when deployed to a real world application with no significant modifications. However, 36.6% of the respondents stated that before an AI generated code could be used, it needs to be reviewed and modified, suggesting that the produced AI code simply does not meet the developer's production quality yet is functional and needs refinement or Adjustments to achieve production standards.

Only a small minority of respondents (2.4%) indicated that they would not consider using AI generated Code in a production environment without extensive testing; therefore, their use of AI generated code in mission critical applications is associated with being very cautious and they are likely to require extensive testing and validation of the code produced by AI to confirm its accuracy of operation prior to deployment.

Table 8: Confidence in AI-Generated Code

Option	Frequency (Responses)	Percentage (%)
Yes, I trust it works as expected	23	56.1%
No, I review and test it carefully before using	13	31.7%
Sometimes, depending on the complexity of the task	5	12.2%

The majority (56.1%), of all respondents believe that the AI-generated code works as expected, even if they don't understand how or why it works that way. Thus, a large number of users believe that the AI-generated code has enough reliability so they can proceed to use it, without having to understand all of the details.

There is still a very large (31.8%) of all respondents that have stated that they carefully review and test the created coding before using it; all of these respondents seem to show that they prefer to use the AI-generated code for those projects, by ensuring that the code will work correctly and to the full requirements, even though they do not understand all of the coding being used.

A small portion (12.1%) of respondents indicated that they are only confident when given a simple coding task, and will be cautious when given a more difficult and critical coding assignment, therefore, suggesting that they trust only to a certain level for the AI-generated code.

Table 9: QA Practices Applied When Using AI-Generated Code

Option	Frequency (Responses)	Percentage (%)
Skipped QA (accepting the code without validation)	4	9.8%
Manual testing or edits to ensure functionality	26	63.4%
Delegating QA to AI or automated tools	11	26.8%
Run-and-see validation (testing by running the code and observing results)	26	63.4%
Reprompting or revising the code instead of debugging	15	36.6%

Of the participants surveyed, 63.4% chose manual testing as their top-quality assurance method, making it the most popular QA practice. This implies that developers predominantly want to manually check AI-created code through running trials themselves or changing code as needed to verify it works properly. Another notable QA practice is run-and-see verification, which was also selected by 63.4% of individuals polled. In a run-and-see method, developers execute an application's code and glance at the results in real-time to help pinpoint any problems also verify that the application generated the intended outcome. 36.6% of the respondents prefer to instruct rather than debug code created by AI; therefore, some developers choose to change code generated by AI directly instead of debugging it. The flexibility of the used AI allows developers to make code changes more quickly than if they were debugging. 26.8% of the surveyed developers prefer delegating the quality of assurance to AI or to some automated tool, therefore it follows that some people use AI or automation tools for validating or quickly validating the quality of developed applications.

Lastly, skipping QA (accepting the code without validation) was selected by only 9.8% of users, showing that a small minority are willing to accept the AI-generated code without further validation, possibly due to confidence in the tool's reliability.

Table 10: Performing Traditional QA Practices When Vibe Coding

Option	Frequency (Responses)	Percentage (%)
Always	23	56.1%
Sometimes	18	43.9%

Majority (56.1%) of the respondents said that they always conduct traditional QA practices such as unit testing and code reviews when vibe coding. This shows that the respondents value quality in their code, no matter how it is generated, as long as it is of high quality.

Sometimes was chosen by 43.9% of the respondents.

This shows that although the respondents appreciate the importance of traditional QA practices, they may not always follow them. Such respondents may follow QA practices depending on the task they are undertaking.

Table 11: Issues Encountered from Skipping Traditional QA Practices

Option	Frequency (Responses)	Percentage (%)
Yes, leading to bugs, security issues, or performance problems	18	43.9%
No, I haven't noticed any significant issues	21	51.2%
Occasionally, but they were manageable	2	4.9%

More than half (51.2%) of all surveyed individuals reported having relatively no issues, as far as they were concerned, by utilizing vibe coding as an alternate method of coding versus following traditional methods of quality assurance testing. This implies that the majority of these individuals believe that the code produced from vibe coding is satisfactory enough for their purposes, even if no formal review of that code was conducted or much testing was performed on it before receiving its final product delivery.

Conversely, 43.9% reported how skipping the quality assurance testing phase has caused them to experience bugs in their application(s) due to the lack of appropriate quality assurance testing and reviews. This indicates that while increasing the amount of time sped through the entire development process, utilizing vibe coding also has the potential to add additional problems into any type of programming that can affect the overall performance, stability, and security of the final product produced.

The very small percentage (4.9% of individuals) indicated that they only sometimes encountered problems due to skipping quality assurance testing, and thus, the problems noted were easy for them to resolve. Therefore, the challenges associated with skipping quality assurance testing were not very significant and were resolved very simply.

Table 12: Managing Risks with AI-Generated Code Quality

Option	Frequency (Responses)	Percentage (%)
I actively review and test the code	20	48.8%
I trust the AI tool's output and skip further validation	3	7.3%
I rely on external tools or third-party reviews to assess code quality	18	43.9%

Participants expressed the most frequent use of actively reviewing and testing AI-generated code to manage risk as the most popular response (48.8%). Users appear to prefer to manage their risk through hands-on methods of verifying that code meets their standards through error checking, performance testing, security testing, etc…

Participants also demonstrated some confidence in the AI-generated code; as 7.3% of participants stated they trusted AI output and therefore do not feel the need for further validation before using it indicates that some users are comfortable with AI-generated code and may even feel additional verification is unnecessary.

The use of other resources such as external tools or having code quality review completed by third parties was cited as an option by 43.9% of users, indicating that some users are also utilizing additional resources to validate that AI-generated code meets their quality expectations, this may include the use of automated code quality tools or code quality review by third-party experts.

Discussion

The research suggests that coding via vibe is highly advantageous for increasing the pace of software development (particularly) in communities lacking formal programming training, such as in Nepal. Similarly, there are previous studies stating that the vibe coding tool creates faster access and democratizes the ability to develop software, e.g. Bamil (2025) and Fortes-Ferreira *et al.* (2025) indicated that vibe coding lowers access barriers for non-developers by producing usable software through natural language commands. Because the population of Nepal does not have a high portion of individuals capable of writing code, tools that use vibe coding allow entrepreneurs and small business owners to prototype and innovate (without extensive programming skills), which our study participants confirmed. A majority (82.9%) of participants identified speed of development as a primary reason for using the vibe coding tool, providing further evidence for the utility of these tools to accelerate development cycles.

The study also discovered that while vibe coding is a quick way to create prototypes and generate new ideas while at the same time having problems that can affect the quality, safety and how long the AI code will remain useful. This is in line with previous research findings including Mitchell and Shaaban (2025) and Meske *et al.* (2025) identifying issues with technical debt, security issues related to the AI models not being able to generate functioning code without needing human supervision to do this. Over 60% of the survey participants indicated they had difficulty understanding and modifying AI produced code. Others mention issues with errors in the code and safety issues. This agrees with the findings of Ray (2025) who stated that even though vibe coding helps create codes rapidly it also carries risks such as the presence of bugs and security issues that would be impossible for non-programmers to

fix. Seventy-eight percent of those responding felt they were getting codes to function, but roughly 36% felt they still required human review and improvement prior to being able to use it in the work place; thus requiring human designers and developers to complete every project before they could be classified as complete.

Regarding usability, the study results indicate that 46.3% of respondents say that using vibe-based coding tools was easy and intuitive, which is in line with the results reported by Fortes-Ferreira *et al.*(2025), who state that using vibe-based coding removes entry barriers for new programmers. However, a consistent negative comment was that many of the participants did not fully understand the AI produced code, and thus found debugging to be challenging, as 53.7% of responses indicated difficulty debugging code. This finding corresponds with commented by Mitchell and Shaaban (2025) and Sarkar and Drosos (2025), where they state that both the debugging process and manual modifications become necessary to validate that your code is functioning correctly.

Although vibe coding offers a paradigm shift in software development, not only in emerging markets such as Nepal but also across the globe, the study reiterates the importance of finding a middle ground between the ease of use of vibe coding and the mitigation of risks associated with it. As has been pointed out in the literature review, future implementation of vibe coding in software development needs to incorporate measures such as quality assurance software and training the users to ensure the quality and security of the software developed.

Key Observations

This research provided some compelling evidence of how the increasing popularity of vibe coding is having a positive impact on software developers and non-programmers in Nepal by speeding up the development cycle. The majority of participants (82.9%) indicated that the primary reason they were adopting vibe coding was due to how much faster it allowed them to create usable code. This is consistent with previous literature, such as that by Bamil (2025), which indicates that the implementation of vibe coding technology increases the efficiency of the prototyping process and provides a means for users to quickly test and iterate ideas before production. Furthermore, participants reported that AI tools were easy to use; thus, providing an opportunity for individuals with limited coding knowledge to get involved in the software development process in Nepal, ultimately encouraging innovation and entrepreneurship within the country. This finding corresponds to that reported by Fortes-Ferreira *et al.* (2025), who found that vibe coding reduced the barriers to entry for new programmers, making it easier to develop creatively without a large amount of programming knowledge.

However, the study also presents important challenges that are inherent with the extensive adoption of vibe coding, especially in relation to code quality, security, and maintainability. While 70.7% of the respondents considered the quality and maintainability of AI-generated code to be high, nearly 36.6% of the respondents considered the code to need substantial modifications before it could be applied in a production setting. Problems such as the lack of understanding of AI-generated code (63.4%), debugging (53.7%), and security concerns (46.3%) were prevalent, suggesting that although vibe coding is an efficient way of developing software, it does not render human involvement unnecessary in the quality assurance process. The results seem to confirm the concerns expressed by Mitchell and Shaaban (2025) and Sarkar and Drosos (2025) in the literature review, who pointed to the dangers of technical debt, security risks, and the need for manual code modification when developing software using AI technology.

Future Work and Open research Questions

Research that is still on the horizon will reveal how to improve the consistency and security of AI code that are produced through the use of vibe coding products for people who do not have software development experience. This research study showed that there are many vibe coding tools available to people in Nepal and that they are being used rapidly and effectively. However, it also showed that there are major issues with the code quality generated by these tools, as well as with the ability of people to debug them, and with the overall security of the software; these issues require future investigation. Future research efforts could include developing and testing a set of integrated quality assurance tools for vibe coding, so that users can automatically discover defect(code bugs), being aware of security problems, and that the AI produced code will be hastily and efficiently allocated to secure future software productions, providing long-term maintainability of the code for all future coding.

Another area that holds great promise for future research is the creation of educational tools and resources that could help improve the users' understanding of the code generated by AI. Although vibe coding makes software development more accessible to everyone, it is possible that a better understanding of the code could help users better cope with problems. Researching ways to make more structured learning or feedback loops a part of the vibe coding experience could greatly benefit users in terms of learning to use AI tools effectively. Future research could also focus on the applicability of vibe coding in other developing nations, where access to programming education is still a problem.

CONCLUSION

Conclusion of this research is that vibe coding, which is made possible by the use of AI tools, is a huge opportunity for the democratization of software development in Nepal because it allows for rapid application development and the participation of non-developers in the development of digital solutions. The biggest benefit that users derive from vibe coding is its speed, simplicity, and efficiency

in rapid prototyping, which is very useful in innovative settings where access to programming knowledge is restricted. However, the results of this research also suggest that there are still some issues with understanding the code generated by AI, debugging, security risks, and maintainability, which suggest that human intervention and quality control processes are still required. Therefore, it is important to acknowledge that vibe coding is a supplementary process to software development that does not substitute programming knowledge, and its adoption in the developing tech setting of Nepal needs to be done in a responsible manner.

REFERENCES

Bamil, V. (2025). *Vibe Coding: Toward an AI-Native Paradigm for Semantic and Intent-Driven Programming*.arXiv. https://doi.org/10.48550/arXiv.2510.17842

Chow, M., & Ng, O. (2025). From technology adopters to creators: Leveraging AI-assisted vibe coding to transform clinical teaching and learning. *Medical Teacher, 47*(12), 1927–1929. https://doi.org/10.1080/0142159X.2025.2488353

De Silva, D., Mills, N., Issadeen, Z., Moraliyage, H., Jennings, A., & Manic, M. (2025). Generative AI Vibe Coding for Prototyping Industrial Systems. In *Proceedings of the 2025 IEEE 34th International Symposium on Industrial Electronics (ISIE),* 1–6. https://doi.org/10.1109/ISIE62713.2025.11124737

Fortes-Ferreira, M., Alam, M. S., & Bazilinskyy, P. (2025). Vibe coding in practice: Building a driving simulator without expert programming skills. In *Adjunct proceedings of the 17th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (pp. 60–66). https://doi.org/10.1145/3744335.3758482

Gadde, A. (2025). Democratizing Software Engineering through Generative AI and Vibe Coding: The Evolution of No-Code Development. *Journal of Computer Science and Technology Studies, 7*(4), 556–572. https://doi.org/10.32996/jcsts.2025.7.4.66

Hemdev, N. (2025). Vibecoding: A mixed-methods case study on conversational AI programming and application development. *International Journal on Science and Technology, 16*(3), 7765. https://doi.org/10.71097/IJSAT.v16.i3.7765

Krings, K., Bohn, N. S., & Ludwig, T. (2025). *(R)evolution of programming: Vibe coding as a post-coding paradigm* (Version 2). arXiv. https://doi.org/10.48550/arXiv.2510.12364

Meske, C., Hermanns, T., Von Der Weiden, E., Loser, K.-U., & Berger, T. (2025). *Vibe Coding as a Reconfiguration of Intent Mediation in Software Development: Definition, Implications, and Research Agenda. IEEE Access, 13,* 213242–213259. https://doi.org/10.1109/ACCESS.2025.3645466

Mitchell, J., & Shaaban, Y. (2025). Position: Vibe coding needs vibe reasoning: Improving vibe coding with formal verification. In *Proceedings of the 1st ACM SIGPLAN International Workshop on Language Models and Programming Languages* (pp. 84–90). https://doi.org/10.1145/3759425.3763390

Pimenova, V., Fakhoury, S., Bird, C., Storey, M.-A., & Endres, M. (2025). *Good vibrations? A qualitative study of co-creation, communication, flow, and trust in vibe coding* (Version 1). arXiv. https://doi.org/10.48550/arXiv.2509.12491

Poudel, S., & Maharjan, S. (2025). Artificial intelligence and education in Nepal: A mixed-methods study on student adoption and learning outcomes. *American Journal of Data Science and Artificial Intelligence, 1*(2), 18–25. https://doi.org/10.54536/ajdsai.v1i2.4763

Ray, P. P. (2025). *A review on vibe coding: Fundamentals, state-of-the-art, challenges and future directions.* TechRxiv. https://doi.org/10.36227/techrxiv.174681482.27435614/v1

Sapkota, R., Roumeliotis, K. I., & Karkee, M. (2025). *Vibe coding vs. agentic coding: Fundamentals and practical implications of agentic AI.* arXiv. https://doi.org/10.48550/arXiv.2505.19443

Sarkar, A., & Drosos, I. (2025). *Vibe coding: Programming through conversation with artificial intelligence.* arXiv. https://doi.org/10.48550/arXiv.2506.23253