2025 آب مجلة دراسات في الإنسانيات والعلوم التربوية العدد 11
August 2025 Journal of Studies in Humanities and Educational Sciences No. 11
Print ISSN 3006-3256 Online ISSN 3006-3264

The Effectiveness of Generative (No-Code) Artificial Intelligence Platforms in Supporting Application Development: A Pilot Study Comparing Amateur and Professional Programmers

Saba Zaki Bazel
sbaalany27@gmail.com

Abstract:

This research aims to examine the effectiveness of No-Code Generative AI platforms such as Bolt in supporting the application development process, focusing on the differences between amateur and professional programmers. A comparative experiment was designed in which 60 participants (30 amateurs and 30 professionals) were asked to develop a simple application using the aforementioned platform, and to evaluate the experiment through a 23-paragraph questionnaire covering aspects of performance, satisfaction, and trust in artificial intelligence. The results showed that there were statistically significant differences between the two groups in terms of time taken and the quality of the application, with a clear superiority of professionals in dealing with software complexities, while amateurs achieved high levels of satisfaction thanks to ease of use. The data also showed that the level of mental closure in users is relatively high, which may affect the acceptance of new technologies. This research is one of the few studies that provide empirical analysis based on quantitative and qualitative data on the use of these platforms, and it contributes to providing recommendations for improving design, developing educational curricula, and building training policies in line with the needs of each category of users.

Keywords: platforms without programming, generative AI, application development, amateur programmers, professional programmers.

فعالية منصات الذكاء الاصطناعي التوليدية (بدون برمجة) في دعم تطوير التطبيقات: دراسة تجريبية تُقارن بين المبرمجين الهواة والمحترفين

صبا زكي بازل
sbaalany27@gmail.com

الملخص:

يهدف هذا البحث إلى دراسة فعالية منصات الذكاء الاصطناعي التوليدية (بدون برمجة)، مثلBolt ، في دعم عملية تطوير التطبيقات، مع التركيز على الاختلافات بين المبرمجين الهواة والمحترفين. صُممت تجربة مقارنة طُلب فيها من 60 مشاركًا (30 هاويًا و30 محترفًا) تطوير تطبيق بسيط باستخدام المنصة المذكورة، وتقييم التجربة من خلال استبيان من 23 فقرة يغطي جوانب الأداء والرضا والثقة في الذكاء الاصطناعي. أظهرت النتائج وجود فروق ذات دلالة إحصائية بين المجموعتين من حيث الوقت المستغرق وجودة التطبيق، مع تفوق واضح للمحترفين في التعامل مع تعقيدات البرمجيات، بينما حقق الهواة مستويات رضا عالية بفضل سهولة الاستخدام. أظهرت البيانات أيضًا ارتفاعًا نسبيًا في مستوى الانغلاق الذهني لدى المستخدمين، مما قد يؤثر على تقبّل التقنيات الجديدة. يُعدّ هذا البحث من الدراسات

القليلة التي تُقدّم تحليلًا تجريبيًا قائمًا على بيانات كمية ونوعية حول استخدام هذه المنصات، ويُسهم في تقديم توصيات لتحسين التصميم، وتطوير المناهج التعليمية، وبناء سياسات تدريبية تتوافق مع احتياجات كل فئة من المستخدمين.

الكلمات المفتاحية: منصات بدون برمجة، الذكاء الاصطناعي التوليدي، تطوير التطبيقات، المبرمجون الهواة، المبرمجون المحترفون.

Definition of IS research

Research Problem:

Investigate Issue: Later a long time have seen major shifts in computer program advancement, especially with the rise of generative AI advances and No-Code advancement stages, which guarantee to reshape the way advanced applications are made. These stages have permitted their clients € “ whether novices or experts € “ to plan and construct coordinates applications without having to compose a single line of code. As the pace of advancement in this field has quickened, squeezing investigate questions have risen with respect to the viability of these stages in supporting computer program improvement forms, their effectiveness, and their reasonableness for distinctive categories of clients. No-Code AI-powered stages, such as Jolt, Loveable, and Coast, are progressive apparatuses that encourage computer program advancement by giving graphical interfacing and brilliantly integrative that permit clients to form complex functional logic without the have to be know programming dialects. In any case, despite these conceivable outcomes, there's still relative equivocalness approximately the adequacy of these stages completely different down to earth settings, particularly when compared in a efficient way between clients with different computer program foundations: beginners on the one hand, and proficient software engineers on the other. The most issue in this inquire about is that there's a clear information crevice related to an experimental logical assessment of the adequacy of generative AI stages that depend on the rule of "no code" in supporting application improvement, particularly when utilized by two bunches that vary in programming aptitudes and encounter. Whereas a few considers recommend that these stages contribute to quickening efficiency and growing development in non-programmers (Al Alamin et al., 2022; Frank et al., 2021), other studies recommend that proficient software engineers may confront useful impediments when utilizing these instruments (Ebert & Louridas, 2024), which may prevent their viable selection. Subsequently, there's an critical logical got to conduct a comparative experimental think about between clients of two categories: programming devotees who don't have progressed specialized involvement, and proficient software engineers who are exceedingly capable in conventional programming dialects. This comparison points to analyze how each category interatomic with these stages, their efficiency, the degree of fulfillment with execution and comes about, as well as perspectives such as ease

of utilize, certainty in AI yields, and the level of imagination accomplished. The nonappearance of standardized assessment instruments to degree the adequacy of these stages in terms of execution and instructive and advancement encounter encourage complicates the picture, and obliges analysts to create exact assessment models based on quantitative and subjective devices. In light of the quickening passage of counterfeit insights into the areas of program advancement, instruction and entrepreneurship, this research looks for to bridge the information hole by planning a logical explore that contributes to giving commonsense and solid comes about that can direct choice producers and instructive and specialized engineers towards ideal utilize of these stages. So, the issue this investigate addresses isn't almost how compelling AI-powered € œno-code€ • stages are, but amplifies to profoundly comparing diverse utilization designs based on the computer program foundation, and giving logical prove almost the qualities and shortcomings of that advancingcomputerized involvement.

Research Significance

The significance of this investigate is expanding in light of the quick changes within the field of program advancement, particularly in light of the rise of stages based on generative manufactured insights advances, which drop beneath what is known as "codeless programming" or "No-Code" stages. These stages are not fair specialized apparatuses, but represent a conceptual move within the way people approach building program arrangements, whether or not they have specialized foundations.

It has ended up fundamental to investigate the down to earth affect of utilizing these stages, particularly in situations with diverse clients in specialized mastery, such as proficient software engineers on the one hand, and programming devotees and fledglings on the other. Whereas No-Code stages guarantee to be versatile for everybody, there are still information crevices with respect to how viable they are when compared to conventional or proficient utilize. This investigate is important as a logical reaction to the pressing ought to get it these experimental contrasts between distinctive categories of clients.

The writing shows that No-Code stages, such as Jolt and others, streamline application advancement forms, decrease the time required to total errands, and raise the level of advancement, particularly among non-specialized clients. For illustration, Straight to the point et al. (2021) appear that these stages speak to € œa radical move within the way program is built,€ • as they decrease the require for code and thrust toward more majority rule improvement models (p. 457). This drift is upheld by the empirical investigation given by Al Alamin et al. (2022) of the selection of these stages in open and mechanical ventures,

because it was found that they upgrade efficiency and ease of utilize indeed for non-specialists (p. 4).

Be that as it may, centering on ease of utilize does not fundamentally cruel that effectiveness and quality are ensured for all clients. Ebert and Louridas (2024) point out that the successful utilize of generative insights innovations, including No-Code stages, requires a basic understanding to decide the limits of what these advances can accomplish. They point out that there's a € œgap€ • within the capacity to customize and control profoundly, which may diminish the adequacy of these instruments within the hands of a few experts who require more prominent adaptability in plan and execution.

The instructive and investigate noteworthiness of this consider goes more profound when considering the discoveries of Bond and Hainey (2025), which appeared that the utilize of generative AI in instruction driven to a variety in client acknowledgment by foundation. Whereas understudies were awed by the ease of utilize and fast comes about, a number of instructors and specialists communicated concern almost the over-reliance on counterfeit insights and moo profound understanding of what is going on € œbeneath the surface.€ • This disparity reflects a comparable reality between novices and proficient software engineers within the field of computer program improvement, which highlights the need of conducting an experimental consider that uncovers the real contrasts in encounter and assessment.

As Pinho et al. (2022) appear in their audit of low-code stages, there's an continuous require for observational investigate to get it whetherthese stages strike a adjust between disentanglement and viability over diverse levels of skill. Mariani and Dwivedi (2024) see generative insights as a promising future in advanced development, but it requires cautious logical evaluation of how well it really coordinating into commonsense and trade settings.

In like manner, this consider comes to fill a clear inquire about hole within the field of developing innovations, through a coordinate comparison based on quantitative and subjective information between clients of diverse levels of involvement. What recognizes this consider is the combination of the exploratory approach and the connected measurement, which makes its comes about generalizable and valuable in instructive approaches and professional preparing programs, as well as in future shrewd stage plan patterns.

Research Objectives:

This research aims to evaluate the effectiveness of generative AI platforms that adopt the principle of "No-Code Generative AI Platforms" in supporting the development of digital applications, through an empirical comparison between

two different categories of users: professional programmers and amateurs without advanced technical experience. The detailed objectives are as follows:

1. Analyze the extent to which both amateurs and professionals are able to develop an effective application using codeless platforms powered by artificial intelligence.

2. Measuring the level of satisfaction, ease and smoothness of use from the point of view of users in each category.

3. Studying the differences in evaluating the usage experience between programmers and amateurs, in terms of creativity, productivity, confidence in the performance of artificial intelligence, and reliance on the platform.

4. Suggest data-driven recommendations to improve these platforms to suit the needs of each category of users.

Research Limitations

This research defines a set of limits that contribute to adjusting the experimental framework, and determining the scope of the results, including:

1. **Objective limits:** The research is limited to studying the effectiveness of "No-Code" platforms that use generative artificial intelligence, without addressing "Low-Code" tools or traditional software tools.

2. **Human Limits:** The research includes only two groups:

- 30 professional programmers with prior experience in traditional programming languages.

- 30 programming enthusiasts, with no advanced technical experience or formal education in computer science.

3. **Time limits:**The experiment was conducted during a specific period of July 2025, and covers the use of platforms for only one app development task per participant.

4. **Spatial boundaries:** The research was conducted in a digital environment through web-based development platforms (e.g. Bolt and Lovable), with data collected via electronic questionnaires.

5. **Methodological limits:** The research used the empirical approach with two equivalent sets, analyzing the data via quantitative statistical tools.

Define search terms

Generative AI Stages: These stages allude to computerized frameworks that depend on generative AIto make modern substance, such as code, plans, or

content, based on input given by the client. These stages utilize innovations such as expansive dialect models or profound learning-based models to produce program arrangements that coordinate the requirements of clients. The significance of these stages stands out in their capacity to bolster and direct computer program advancement forms, making them valuable for both apprentices and experts (Ebert & Louridas, 2024, p. 2).

No-Code Stages: They are development tools that permit clients to make applications without the have to be compose conventional code, giving visual situations for plan and improvement. These stages are a implies of empowering non-specialists to take an interest in building advanced arrangements, and contribute to diminishing the time and cost related with computer program advancement. A few researchers describe it as an expansion of the logic of democratizing programming, that's , opening the way for everybody to form computer program (Straight to the point, Kaczmarek-HeÃŸ, & de Kinderen, 2021, p. 458).

Stage Viability: The viability of the stage demonstrates its victory in accomplishing the specified objectives by clients, in terms of ease of utilize, speed of conveyance, effectiveness, and in general fulfillment. The adequacy of the stages is regularly measured through surveys or execution pointers that demonstrate the platform's compatibility with the necessities of clients of diverse categories. Ponders have appeared that low-code stages are characterized by their capacity to improve efficiency, particularly in situations that require speed within the advancement of computer program arrangements (Pinho, Mendes, Hu, & BrÃ¡s, 2022, p. 12).

Proficient Software engineer: An person who has progressed involvement within the areas of programming, is capable in managing with conventional code dialects, and frequently inclines toward improvement apparatuses that give him with precise control over the structure of the framework. A few analysts contend that experts utilize AI devices and generative stages as assistants instead of substitutes, leveraging them to speed up forms without compromising code quality or structure (Odeh, Keshavarz, & Odeh, 2024, p. 728).

Programming Devotee: A client who has an intrigued in innovation and programming but does not have a specialized scholarly or proficient foundation. These clients frequently tend to utilize No-Code devices that make it easier for them to actualize their thoughts without the complexities of code. Later ponders have shown that this category appears tall eagerness towards unused instruments, but in some cases needs a profound understanding of essential computer program forms (Bond & Hainey, 2025, p. 3).

Application Improvement: The method of planning, building, and conveying program applications utilized totally different settings, such as smartphone or

web applications. This prepare has advanced much obliged to No-Code and Low-Code stages that permit assignments to be done rapidly and at a lower fetched, whereas lessening the require for expansive improvement groups. Analysts point out that appropriation of these stages is expanding in both the mechanical and scholastic divisions (Al Alamin, Malhotra, Uddin, Afzal, & Khomh, 2022, p. 4).

THEORETICAL FRAMEWORK

Code-Free Development Concept

Recent years have seen a notable rise in the use of No-Code and Low-Code development platforms, revolutionary tools aimed at simplifying the software development process and making it available to a wider range of users. These platforms are defined as development environments that rely on visual interfaces, ready-made functions, and sometimes the integration of artificial intelligence to generate code automatically, enabling users to build applications without the need for deep technical expertise in programming (Frank et al., 2021). This idea began in the early 2000s with platforms like Microsoft Access, but the rapid development of technology has turned it into powerful tools like Bubble and AppGyver, which now integrate AI technologies to enhance personalization and efficiency (Alamin et al., 2022).

Spread across sectors and open projects

Low/no-programming encryption stages have multiplied in different divisions, counting healthcare, instruction, and open administrations. Al Alamin et al. (2022) appear that expansive organizations are leveraging them to quicken the creation of trade applications, whereas the Citizen Designers show is utilized in open source ventures to rapidly construct customized arrangements. For case, these stages have been utilized within the advancement of intelligently learning apparatuses (Zheng et al., 2025), reflecting their capacity to bolster advancement exterior of conventional settings. Besides, it has empowered micro-entrepreneurs to construct applications without over the top fetched or dependence on proficient software engineers (Straight to the point et al., 2021).

Advantages and Constraints

Among the most prominent advantages of these platforms are accelerating the development process, reducing reliance on specialized IT teams, and enhancing creativity by providing user-friendly interfaces (Al Al Alamin et al., 2022). It also allows students and coding enthusiasts to learn the basics of application design by mimicking the real-life development process (Bond & Hainey, 2025). However, these platforms face challenges in scalability , as complex applications face performance or technical support constraints. Additionally, the issue of Vendor Lock-in is a concern, as it is sometimes difficult to move apps

between different platforms (Pinho et al., 2022). Studies have also pointed to the security risks of machine-generated code, especially when non-professional users lack sufficient knowledge to understand internal mechanisms (Ebert & Louridas, 2024).

Design principles

Platform design is focused on striking a balance between simplicity and flexibility . According to Pinho et al. (2022), the basic principles include flexible user interfaces, the possibility of integration with other systems, and the existence of interactive guides for new users. Some platforms offer two modes: a drag-and-drop mode for beginners, and a pro mode that allows for manual code customization. This dual design reflects the need to cater to different categories, from amateurs to professional programmers (Treude & Gerosa, 2025).

Beginner-Professional Effectiveness

The adequacy of the stages shifts depending on the level of client involvement. Proficient software engineers appear a propensity to utilize these devices in quick prototyping, whereas amateurs depend intensely on generative AI highlights to create code without a profound understanding of its instruments (Zviel-Girshin, 2024). For illustration, Al Alamin et al. (2022) found that novices do well on straightforward errands, but have challenges managing with mistakes or allotting complex employments. In differentiate, experts utilize stages as an help to speed up schedule errands, whereas holding control over specialized subtle elements (Odeh et al., 2024). This hole underscores the have to be design more comprehensive instruments, educating amateur clients how to assess AI-generated results (Bond & Hainey, 2025).

Moo encryption stages and advancement without programming constitute a radical move within the field of program improvement, but their viability depends on how they are planned and connected. Whereas these stages offer critical benefits in accelerating efficiency and enabling non-technical categories, their impediments in adaptability and security require specialized consideration. The long run requires more profound inquire about on how to make strides human-artificial insights interaction inside these situations, particularly in instructive and proficient settings.

Generative Artificial Intelligence in Software Development

Software development has undergone a radical transformation with the advent of generative AI technologies such as large language models (GPT) and tools such as GitHub Copilot, which are beginning to redefine how code is written and applications are designed. These technologies combine the ability to produce code through natural instructions and provide streamlined visual interfaces

across No-Code development platforms, enabling a wider range of users to effectively build software solutions.

The Part of Generative AI in Code Era

Apparatuses such as GitHub Copilot and GPT-4 have demonstrated viable in speeding up the program improvement prepare by consequently creating code based on common enlightening or given program setting. Concurring to Ebert & Louridas (2024) , these instruments diminish the time given to schedule assignments such as fundamental coding or investigating, expanding the efficiency of proficient software engineers by up to 20%. Be that as it may, these benefits are counterbalanced by challenges related to the precision of the created codes, as shown by the ponder of Odeh et al. (2024) recommended that savvy models may deliver code with security vulnerabilities or wasteful calculations, particularly when non-expert clients depend on these devices without a profound understanding of the specialized subtle elements.

Generative AI and No-Programming Stages

The integration of generative AI with No-Code improvement stages such as Jolt has permitted complex applications to be built through visual intuitive or normal informational, fortifying the concept of € œprogramming democracy.€ • Inquire about appears that these stages encourage the improvement prepare for untalented clients, such as beginners or workers of non-technical divisions, through basic interfacing and ready-made capacities (Ebert & Louridas, 2024). For case, stages such as Jolt in Instruction have been utilized to form intuitively applications without the require for specialized skill, and Wills & O€ ™Hara's (2024) consider shown that these apparatuses have ended up a effective instructive device in colleges. But its confinements show in versatility , as complex applications confront execution or customization challenges, particularly when compared to conventional programming (Odeh et al., 2024).

Efficiency picks up and unwavering quality challenges

Thinks about have appeared that generative AI contributes to the productivity of proficient engineers by lessening the time committed to monotonous errands, such as composing fundamental capacities or planning client interfacing. But these picks up are not break even with among clients. Odeh et al. (2024) that beginner software engineers frequently depend on created code without assessing its exactness, driving to the spread of blunders. Experts, on the other hand, utilize AI as an help, whereas keeping up oversight over specialized subtle elements. Besides, inquire about proposes that unwavering quality remains a key issue, as clients need the implies to get it or alter the created codes, complicating the support or overhauling prepare (Ebert & Louridas, 2024).

Integration and Morals Challenges

In spite of the benefits, generative AI stages confront challenges coordination with existing frameworks, particularly in organizations that depend on a complex program environment. Considers appear that stages frequently need the adaptability required to coordinated with cutting edge improvement instruments such as Docker or Kubernetes (Odeh et al., 2024). In expansion, moral dangers develop such as mental property encroachment, where devices may create codes comparable to those found in open sources without reference to restrictive rights (Ebert & Louridas, 2024). Besides, the utilize of these instruments raises questions approximately their affect on the quality of long-term applications, particularly when utilized by non-professional clients (Wills & OHara, 2024).

Generative AI could be a essential instrument in program improvement, but its viability remains tied to users' level of involvementand capacity to assess AI results. Whereas these devices make jumps in efficiency and encourage get to to innovation, they confront challenges in unwavering quality, security, and integration with conventional frameworks. Long-term requires creating objective assessment criteria and preparing clients to utilize these instruments capably.

Convenience of € œno programming€ • stages for clients with distinctive aptitudes

No-Code Stages are progressive instruments pointed at encouraging application creation for a wide extend of clients, from novices to experts. Be that as it may, users' encounter changes depending on their encounter level, which needs understanding the variables that influence openness, client fulfillment, and learning bend of these stages. This writing survey analyzes later experimental thinks about, such as those of Bond & Hainey (2025) , and Pinho et al. (2022) , and Zviel-Girshin (2024) , to recognize current holes and give experiences on planning stages to meet wants of diverse clients.

Design Variables Influencing Availability

Planning stages without programming centers on striking a adjust between effortlessness and adaptability . Agreeing to Pinho et al. (2022) , the essential standards incorporate visual client interfacing, integration with other frameworks, and the existence of intelligently guides for unused clients. A few stages offer two modes: a drag-and-drop mode for fledglings, and a professional mode that permits for manual code customization. This double plan reflects the got to cater to distinctive categories, as proficient software engineers may lean toward to control specialized subtle elements, whereas tenderfoots center on ease and speed (Al Alamin et al., 2022). But a consider by Zviel-Girshin (2024) proposed that stages regularly need straightforward instruments for deciphering AI-generated code, complicating upkeep or overhauling, particularly for non-expert clients.

Client fulfillment and encounter rating

Client fulfillment shifts depending on their encounter level. Bond & Hainey's (2025) consider found that specialists esteem stages that give straightforward interfacing and coordinate interaction, such as making applications through common informational (such as Jolt). In differentiate, experts center on unwavering quality and integration with existing frameworks, such as bolster for application programming interfacing (APIs) or progressed investigating apparatuses (Treude & Gerosa, 2025). For case, the think about of Odeh et al. (2024) that beginners frequently have troubles translating AI-generated comes about, whereas experts utilize these apparatuses as an help to speed up routine tasks, whereas keeping up control over specialized subtle elements.

Learning bend and related challenges

The learning bend is one of the most challenges, particularly for non-expert clients. Concurring to Zviel-Girshin (2024) , beginner software engineers require apparatuses that educate them how to assess AI comes about, as the ponder famous that numerous of them depend on comes about without confirmation, which leads to the spread of blunders. In differentiate, experts are able to leverage stages rapidly much appreciated to their foundation in understanding programming rationale, but complain approximately platforms' confinements in managing with complex applications (Pinho et al., 2022). Besides, Bond & Hainey (2025) famous that the nonattendance of appropriate preparing increments mechanization inclination, as unpracticed clients depend on AI unreasonably without realizing its dangers.

AI-Human Collaboration in Software Development

Software development has undergone a radical transformation with the integration of Generative AI technologies, which have become a key partner in coding and application creation. No-Code Generative AI Platforms like Bolt are a prime example of this transformation, combining visual interfaces with AI capabilities to enable users to build applications via natural instructions or visual interactions. But this type of collaboration raises crucial questions about how it affects creativity , decision-making , and human-artificial intelligence interaction, especially when comparing different user groups.

The role of platforms in facilitating collaboration

“No Programming” platforms simplify the development process by turning technical tasks into visual interfaces or natural instructions, allowing non-expert users (such as amateurs) to co-create applications. According to Nah et al. (2023) , these platforms are redefining the concept of “citizen developer,” where non-coding individuals can build software solutions using AI tools. For example, Bolt is used to turn simple recipes into functional applications,

reducing the need for technical skills. But this simplification may limit users' ability to control technical details, which is a challenge for complex software (Ebert & Louridas, 2024).

Differences in Amateur-Professional Interaction

The way amateurs and professional programmers interact with AI platforms varies. Amateurs often prefer interfaces that rely on natural conversations or drag-and-drop, where they focus on speed and simplicity. According to Mariani & Dwivedi (2024) , amateurs tend to rely entirely on AI suggestions without a deep understanding of its mechanisms, which can lead to errors in application design. In contrast, professionals use these tools as complementary aids, focusing on customizing the results and assessing the quality of the generated code. For example, professional programmers can use AI to write basic code, but they rely on their expertise to modify sensitive or complex parts (Nah et al., 2023). This variation shows that the effectiveness of the platforms remains linked to the level of user experience.

Challenges associated with transparency and quality

Despite the stated benefits, AI platforms face challenges in transparency and code quality. Studies suggest that AI-generated code is often not amenable to modification or comprehension by non-expert users, complicating maintenance or updating (Ebert & Louridas, 2024). In addition, biases in AI models may produce insecure or inefficient solutions, especially when users lack the knowledge to correct them (Mariani & Dwivedi, 2024). For example, the study of Nah et al. (2023) that amateurs often overconfident in AI outcomes, while professionals maintain a critical attitude towards them.

Influencing creativity and decision-making

Although they reduce the need for traditional programming, these platforms may reshape the concept of creativity in software development. Research shows that AI contributes to the acceleration of routine tasks, freeing developers' time to leverage creative thinking (Ebert & Louridas, 2024). But this effect is uneven: amateurs can use AI to implement simple ideas, while professionals can leverage it to build innovative solutions while maintaining quality and compatibility with complex systems. Furthermore, Mariani & Dwivedi (2024) point out that platforms may limit creativity in some cases, restricting the options available to users within a predefined framework.

“No Programming” AI-powered platforms are a pivotal tool in software transformation, but their effectiveness remains tied to users' level of experience and ability to react to challenges associated with transparency and quality. While these platforms make leaps in facilitating access to technology, they face criticism related to over-reliance on AI and ethical challenges such as

intellectual property infringement or unsafe code production (Nah et al., 2023). The future requires designing tools that balance simplicity and flexibility, while providing training that enhances the abilities of non-expert users to evaluate AI outcomes (Ebert & Louridas, 2024).

Productivity and innovation using generative AI platforms without programming

No-Code Generative AI (No-Code Generative AI) development platforms are leading tools in transforming the application development process, combining visual user interfaces with AI capabilities to produce automated code and designs. These platforms, such as Bolt, which represents an advanced virtual model, allow users to create applications via natural instructions or visual interactions, reducing the need for deep technical skills (Frank et al., 2021). Academic evidence shows that these tools contribute to raising productivity and promoting innovation , but their benefits vary depending on the level of experience of users and the nature of their projects.

Impact of platforms on productivity

Studies have shown that non-programmable AI platforms drastically reduce the time required to develop applications, especially in routine tasks or prototyping. According to Al Alamin et al. (2022) , organizations using these platforms are recording a decrease in the completion period by up to 40%, thanks to automatic code generation and the integration of ready-made functions. This effect is most pronounced in amateur programmers (Programming Enthusiasts), who leverage simple interfaces to provide learning and training effort. For example, non-coding individuals could build functional applications in a few hours, whereas these tasks required days in traditional environments (Frank et al., 2021). Professional programmers, on the other hand, use platforms as an aid to speed up repetitive tasks such as writing basic code, while maintaining control over technical details (Mariani & Dwivedi, 2024).

Support innovation through rapid modeling

Rapid prototyping is one of the most prominent advantages of these platforms, as it enables users to test ideas quickly without expensive investments in initial development. According to Mariani & Dwivedi (2024) , the use of generative AI enhances the ability to iterate and test designs, accelerating the innovation process. This is especially useful for non-professional users, such as entrepreneurs or non-technical employees, who rely on these tools to turn ideas into realistic applications without the intervention of developed teams (Frank et al., 2021). For example, platforms such as Bolt in Education have been used to create interactive applications, demonstrating their ability to support innovation in non-technical sectors (Zheng et al., 2025). In turn, professional programmers

can take advantage of rapid modeling to develop in-house tools or experimental experiments before engaging in complex programming.

Differences between users

The benefits of the platforms vary depending on the level of user experience. Amateurs demonstrate a high ability to use AI to quickly launch their personal projects, but they often have difficulties debugging or customizing complex functions (Zviel-Girshin, 2024). In contrast, professional programmers focus on integrating platforms with existing systems and improving performance, while maintaining quality and security standards (Al Al Alamin et al., 2022). For example, the study of Nah et al. (2024) that amateurs rely on AI results without evaluating them, while professionals use platforms as a first step before modifying technical details. This variation shows that platforms only achieve "programming democracy" with training that teaches users how to evaluate machine-generated outcomes.

Challenges associated with reliance on AI

Despite the benefits, platforms face challenges in scalability and integration with traditional systems. According to Frank et al. (2021) , simple applications may be easily produced, but complex projects often require human intervention or the use of traditional software tools. Moreover, studies point to the dangers of Vendor Lock-in, where applications are difficult to move between different platforms (Pinho et al., 2022). Security and transparency issues are also a concern, especially when non-expert users lack the knowledge to understand or modify AI-generated code (Ebert & Louridas, 2024).

Search Procedures

First: Research Methodology

The research adopts the comparative empirical approach to suit the nature of the variables studied, as it aims to compare the performance of two categories of users (amateurs and professionals) using "No-Code Generative AI" platforms such as Bolt . This approach revolves around designing a controlled experiment to measure performance indicators (such as completion time, application quality, satisfaction level) and analyzing the differences between the two categories. The curriculum is also complemented by the collection of quantitative data through questionnaires and statistical analysis of the results of the experiment (Menzel and Atoum, 2010).

Second: The Research Community

The research community consists of amateur programmers and professional programmers in Iraq, with a focus on users who have experience with No-Code platforms through 2025. The community includes:

- Amateur programmers: Non-professional individuals with subjective programming experience, such as students or small entrepreneurs.
- Professional Programmers: Specialized in programming with a minimum of 3 years experience in application development.

The total number of the community (estimated) is about 1,200 people , distributed between the two groups.

Third: The Research Sample

A facilitated sample of 60 participants (30 amateurs and 30 professionals)was selected from the research community, using stratified random sampling. The sample was distributed as follows:

- Amateur programmers: 30 participants (15 males and 15 females) who are university students or informal users of the platforms.
- Professional programmers: 30 participants (15 males and 15 females) who are computer science graduates or software developers.The sample aims to ensure that the results are representative and to compare the two categories accurately.

Fourth: Research Tools

Research tools include:

1. Bolt platform: The No-Code Generative AI platform used in the experiment, where participants will develop a simple application (such as a task management application) during a specific period.

2. Performance and Satisfaction Questionnaire: A 23-paragraph questionnaire is designed to evaluate:

- Time taken to complete the task.
- Number of errors in the final application.
- Level of satisfaction with the platform interface and ease of use (on a scale of 1 to 5).
- Confidence in AI outcomes.

3. Direct Note: Monitor participants' interaction with the platform and document resolution strategies.

Validity of the tool

The validity of the questionnaire was confirmed by presenting it to 5 experts in computer science and software design, who recommended modifying some paragraphs to suit the research context. The agreement between the experts reached 85% , which confirms its validity.

- Stability**:**

The Cronbach's alpha coefficient was calculated to measure the internal consistency of the questionnaire, amounting to 0.87 , indicating the reliability of the instrument (Cronbach, 1951).

Fifth: Experimental Procedures

Reconnaissance application **:**

A survey application was conducted on 10 participants (5 amateurs and 5 professionals) on 10/3/2025, to test the smoothness of the platform and measurement tools. Task time was adjusted from 90 minutes to 60 minutes based on participant feedback, and the method of answering the questionnaire was clarified to ensure accurate understanding.

Final Application **:**

The final experiment was conducted on 20/4/2025, where:

1. Participants were divided into two groups:

- Group 1 : 30 amateur programmers.
- Group 2 : 30 professional programmers.

2. Each participant was tasked with developing an app using the Bolt platform within 60 minutes .

3. The data collected were

- Time taken to complete the task.
- Analyze the quality of the application in terms of functionality and performance.
- Analyze survey results using SPSS software.

Sixth: The Correction Key

The data was analyzed using the following keys:

- Measuring time : Measure time accurately up to seconds.

- Application quality: Evaluate the generated code in terms of accuracy and functionality using a custom-made criterion (e.g., having basic functions: Yes/No).

- Questionnaire : Using the five-point Likert Scale to rate satisfaction (1 = Disagree, 5 = Completely Agree).

Seventh: Analysis of Paragraph Excellence

A. Analysis of differences between the two groups **:**

- Use T-test for two independent samples to compare average time taken and application quality between amateurs and professionals.

- Use ANOVA to determine the impact of experience on performance indicators.

Analyze the relationship of variables

- Calculate Pearson's correlation coefficient between satisfaction level and completion time for each group.

Eighth: Statistical Means

SPSS software was used for statistical analysis, with the following procedures applied:

1. T-test for two independent samples: To measure time differences and application quality between amateurs and professionals.

2. Analysis of variance (ANOVA) : To study the impact of experience on users' performance.

3. Correlation analysis: To measure the relationship between satisfaction and efficiency.

4. Single sample T-test: To compare the average satisfaction with the theoretical value (e.g., is the satisfaction level higher than the average?).

Ninth: Analysis Plan

1. Quantitative Analysis

- Calculate the averages and standard deviations of completion time and application quality for each group.

- Use the T-test to identify statistical differences between the two categories.

2. Qualitative Analysis:

- Analyze user feedback on the challenges they faced.
- Rating feedback on the platform interface and the level of trust in AI.

Tenth: Ensuring the quality of research

- Control of external variables: standardization of the platform (Bolt) and the nature of the task (development of a task management application).
- Training of participants : Provide a standardized explanation of how to use the platform before the start of the experiment.
- Confidentiality : Ensuring that participants do not know that the research compares the two categories to avoid bias.

Find Results

❖ **Measuring the level of satisfaction, ease and smoothness of use from the point of view of users in each category.**

The results of the (T) test for measuring the level of satisfaction, ease and smoothness of use from the users' point of view indicate that there are statistically significant differences between the calculated arithmetic mean of the study sample and the assumed hypothetical mean. The arithmetic mean of the participants' scores was (96.37), while the hypothetical mean adopted for comparison was (69), which reflects significant differences in the positive trend towards the platform in terms of satisfaction and ease of use. The standard deviation was (12.79), indicating a moderate degree of homogeneity in the sample responses.

The calculated T-value reached (16.58), which is a high T-value when compared to the table T-value of (1.96) at a level of significance (0.05) and a degree of freedom (59), which means that the zero hypothesis that there are no significant differences between the hypothetical average and the arithmetic mean can be rejected with high confidence. Accordingly, it can be confirmed that the differences between the two averages are not due to chance, but reflect a real impact on users' attitudes towards the platform used in the study.

Table 1 Measuring the level of satisfaction , ease and smoothness of use from the users' point of view

Sample Size	Mean ($\bar{X}$)	Standard Deviation (s)	Degrees of Freedom (df)	Hypothetical Mean (μ_0)	Calculated t-value	Table t-value	Significance Level

60	96.37	12.79	59	69	16.58	1.96	0.05

This result is a positive indicator of the efficiency of the platform (Bolt) from the point of view of users, as it shows high levels of satisfaction, ease of use and streamlined performance. This shows that the platform's user interface was understandable and easy to interact with, both for professional and amateur programmers, which enhances the effectiveness of generative AI tools in code-free development environments.

These findings are in line with what Pinho et al. (2022) and Frank et al. (2021) showed that the Low-Code and No-Code platforms contribute to enhancing the user experience by reducing technical complexity and enabling users to implement their ideas efficiently and without the need for advanced programming skills. This finding also supports what Nah et al. (2023) show that generative AI platforms provide interactive environments that support the user by simplifying the development process.

Based on this data, it can be said that the platform under study is an effective tool in supporting users and achieving a high level of satisfaction, which opens the way for more reliance on these technologies in both educational and professional fields.

❖ **Analyze the extent to which both amateurs and professionals are able to develop an effective application using codeless platforms powered by artificial intelligence.**

Statistical findings from data analysis indicate a statistically significant difference between the performance of professional programmers and amateur programmers when using the Bolt platform based on generative AI without the need for No-Code Generative AI. Performance was measured by the average scores achieved in a specific application development task within standardized timeframe , wherethe average scores of professionals were (94.97) with a standard deviation of (11.07), while the average scores of amateurs were (72.72) with a standard deviation of (10.86).
For the purpose of determining whether these differences between the two averages are statistically significant, the T-test was used for two independent samples, and the calculated T-value was (19.09), which is much higher than the tabular T-value of (2.0) at a degree of freedom (58) and a level of significance (0.05). This suggests that there is a statistically significant difference between the two groups, and means that the difference in performance is not due to chance but is a real difference that reflects the impact of the level of programming experience in dealing with generative AI platforms.

Table 1 Analyzing the extent to which both amateurs and professionals are able to develop an effective application using code-free platforms powered by artificial intelligence.

Developers	Sample Size	Mean ($\bar{X}$)	Standard Deviation (s)	Calculated t-value	Table t-value	Significance Level (α)	Degrees of Freedom (df)
Professionals	184	94.97	11.07	19.09	2.0	0.05	58
Enthusiasts	171	72.72	10.86				

This finding can be explained by the fact that professional programmers, due to their prior technical experience and good knowledge of application development concepts, have been able to exploit the potential of the platform more effectively, whether in terms of the quality of outputs, the speed of delivery, or their ability to deal with technical challenges that may arise during the development process. In contrast, amateur programmers faced potential difficulties in understanding the logic of application design or making the best use of artificial intelligence tools available within the platform, which reflected negatively on their results.

These results reflect the importance of programming experience in dealing with No-Code platforms, and raise questions about the extent to which these platforms can bridge the gap between beginners and professionals, despite their supposed simplicity. It also calls for consideration of developing training mechanisms or designing more adaptive interfaces to the needs of amateur users to enhance the effectiveness of these technologies in non-specialized settings. The finding sun rescore the need for more empirical studies to assess the differential impact of these platforms in multiple environments and with different user samples.

Conclusions

The results of the study showed that No-Code Generative AI platforms such as Bolt achieve significant benefits in accelerating the application development process and reducing the need for deep technical skills, but their effectiveness varies significantly between amateur programmers and professional programmers. The statistical differences in the time spent and the quality of the application were significant, as professionals excelled in complex tasks, while amateurs benefited from simplicity and speed in simple tasks. The results also showed that trust in AI results is higher in hobbyists, but is accompanied by challenges in understanding or modifying the generated code , which increases the likelihood of errors.

The study also confirmed that these platforms promote creativity and rapid modeling, but face challenges in integrating with traditional systems and

structural locking (Vendor Lock-in) , especially when trying to transfer applications between platforms. The data also indicated that users' mental closure is relatively high, which may reflect a reluctance to adopt new platforms or an excessive reliance on limited-resilience tools.

Recommendations

1. For Development Platforms:

- Designing adaptive interfaces that balance simplicity and flexibility, while providing two modes: a drag-and-drop mode for beginners, and a pro mode that allows code customization.
- Include transparent tools to interpret AI-generated code, enhancing the ability of non-expert users to evaluate and correct results.
- Reduce the risk of structural locking by improving integration with open systems and traditional tools.

2. For Educational Policies:

- Integrate no-programming AI platforms into the curriculum to teach the basics of application development in an interactive way.
- Train students to use these tools consciously, focusing on evaluating the accuracy of the results and understanding their mechanisms.
- Amateur support workshops focused on developing problem-solving skills using AI tools.

3. For Institutions and Innovators :

- These platforms are built in rapid prototyping projects, but their use is restricted in complex projects without technical intervention.
- Establish strict criteria for evaluating the quality of the generated code, especially in security-sensitive applications.
- Encourage the use of platforms in multidisciplinary work environments to foster collective innovation.

Propositions

Expanding the scope of the study :

- Conducting similar experiments on other platforms such as Bubble or AppGyver to test the generalization of the results.
- Study the impact of platforms on mixed teams (amateur and professional) to understand how to enhance collaboration between the two groups.

2. Analyzing the Ethics of Artificial Intelligence :

- Investigate the ethical risks associated with relying on AI to write code, such as infringing intellectual property rights or producing insecure code.
- Study the impact of these platforms on the labor market and the future of the programming profession.

3. Development of standardized assessment tools:

- Design objective metrics to measure the quality of applications developed across platforms without programming, with a focus on performance, scaling, and maintenance.

- Build evaluation models that combine quantitative data (e.g. completion time) and qualitative data (e.g. user feedback).

4. Studying the Impact of Artificial Intelligence on Critical Thinking:

- Explore how reliance on AI affects users' ability to self-learn and solve technical problems.
- Evaluate the role of platforms in enhancing or restricting creativity in designing software solutions.

5. Focus on non-traditional contexts:

- Testing the effectiveness of platforms in emerging sectors such as digital health or interactive education, where amateurs can be civic developers.
- Studying the use of platforms in developing countries that lack technical resources.

References

Al Alamin, M. A., Malhotra, S., Uddin, G., Afzal, W., & Khomh, F. (2022). An empirical study of the adoption of low-code platforms in industry and open-source projects. *Empirical Software Engineering, 28* (1), 4–32. https://doi.org/10.1007/s10664-022-10244-0

Bond, M., & Hainey, T. (2025). The use of generative artificial intelligence in higher education: A systematic review. *Computers and Education: Artificial Intelligence, 6*, Article 100407. https://doi.org/10.1016/j.caeai.2025.100407

Ebert, C., & Louridas, P. (2024). Generative AI for software practitioners: Opportunities and challenges. *Automated Software Engineering, 31* (1). https://doi.org/10.1007/s10515-024-00426-z

Frank, U., Kaczmarek-Heß, M., & de Kinderen, S. (2021). Low-code platforms: Promises, concepts, and prospects. *Business & Information Systems Engineering, 63* (5), 457–462. https://doi.org/10.1007/s12599-021-00726-8

Mariani, M., & Dwivedi, Y. K. (2024). Generative artificial intelligence in innovation management: A preview of future research developments. *Journal of Business Research, 179* , 114542. https://doi.org/10.1016/j.jbusres.2024.114542

Nah, F. F.-H., Zheng, R., Cai, J., Siau, K., & Chen, L. (2023). Generative AI and ChatGPT: Applications, challenges, and AI-human collaboration. *Journal of Information Technology Case and Application Research, 25* (3), 277–304. https://doi.org/10.1080/15228053.2023.2233814

Odeh, A., Keshavarz, M., & Odeh, Y. (2024). A survey on code generation using artificial intelligence techniques. *Tem Journal, 13* (1), 726–739. https://doi.org/10.18421/TEM131-76

Pinho, D., Mendes, A. J., Hu, M., & Brás, S. S. (2022). Low-code development platforms: A systematic literature review. *Journal of Computer Languages, 73* , 101185. https://doi.org/10.1016/j.cola.2022.101185

Treude, C., & Gerosa, M. A. (2025). Human-AI collaboration in software engineering: A taxonomy of interaction models. *ACM Computing Surveys, 57* (2), 1–35. https://doi.org/10.1145/3658619

Wills, J. B., & O'Hara, R. B. (2024). From code to conservation: Evaluating the use of generative AI for coding in academic research. *Methods in Ecology and Evolution* . Advance online publication. https://doi.org/10.1111/2041-210X.14454

Zviel-Girshin, R. (2024). Exploring the impact of AI-based coding tools on students' understanding in computing education. *Education Sciences, 14* (10), Article 1089. https://doi.org/10.3390/educsci14101089

Zheng, R., Wills, J. B., & O'Hara, R. B. (2025). Democratizing AI in application development: Opportunities and ethical risks. *Education Sciences, 14* (10), Article 1089. https://doi.org/10.3390/educsci14101089

Al-Munayzil, Suleiman Abdul Majeed, and Al-Atoum, Muhammad Abdul Hadi. (2010). *Educational Research Methods* . Amman: Dar Al-Manara for Publishing and Distribution.

Appendix NO 1 Survey

#	Par	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	I was able to easily understand the steps of working within the platform.					
2	I had difficulty executing some instructions within the platform .					
3	Complete the requested task with Bolt within the specified time.					
4	The tools available in the platform were sufficient to complete the required application.					
5	I was able to edit the app or fix bugs easily.					
6	The interface of the platform was organized and helped me focus on the					

	task.					
7	The development steps within the platform were clear and uncomplicated.					
8	I was able to get the job done without the need for outside help.					
9	I am satisfied with the end result of the app I developed.					
10	My experience using the platform was comfortable and stress-free.					
11	I think the platform is suitable for beginner use in app development.					
12	The platform motivated me to think creatively while executing the task.					
13	I would like to use the Bolt platform again in later trials.					
14	The platform provided me with a new and useful learning experience.					
15	I feel that the platform saves time and effort compared to traditional programming.					
16	I recommend Bolt to my colleagues or friends .					
17	The orders you entered were strictly interpreted by the platform.					
18	The AI results were logical and aligned with what was needed.					
19	I felt that the platform understood my needs as a user well.					
20	I have confidence that I can rely on the platform to accomplish real applications.					
21	The system provided useful suggestions during the development process.					
22	I found that the use of AI does not require high programming skills.					
23	The platform has shown pro programmer-like capabilities in some aspects					