Торайғыров университетінің
ҒЫЛЫМИ ЖУРНАЛЫ

НАУЧНЫЙ ЖУРНАЛ
Торайгыров университа

Торайғыров университетінің ХАБАРШЫСЫ

Педагогикалық сериясы
1997 жылдан бастап шығады

ВЕСТНИК ТОРАЙГЫРОВ УНИВЕРСИТЕТА

Педагогическая серия
Издается с 1997 года

ISSN 2710-2661

№ 3 (2026)
Павлодар

НАУЧНЫЙ ЖУРНАЛ
Торайгыров университета

Педагогическая серия
выходит 4 раза в год

СВИДЕТЕЛЬСТВО
о постановке на переучет периодического печатного издания,
информационного агентства и сетевого издания
№ KZ03VPY00029269
выдано
Министерством информации и коммуникаций
Республики Казахстан

Тематическая направленность
публикация материалов в области педагогики,
психологии и методики преподавания

Подписной индекс – 76137

https://doi.org/10.48081/BTOUP3.2026/ISSN2710-2661

SRSTI 14.35.07

https://doi.org/10.48081/BTOUP3.2026/130-143

S. Meruert[1], *A. A. Turdybek[2]

[1,2]L. N. Gumilyov Eurasian National University
Republic of Kazakhstan, Astana.
[1]ORCID: https://orcid.org/0000-0002-2801-432X
[2]ORCID: https://orcid.org/0009-0004-1145-921X
*e-mail: serik_meruerts@mail.ru

VIBE CODING IN HIGHER EDUCATION: NOVICE MOTIVATION, DEBUGGING GAPS, AND PEDAGOGICAL SCAFFOLDING

Amidst these international changes we are witnessing in higher education, AI and vibe coding are progressively being adopted as a part of pedagogy. Vibe coding (a concept that appeared in 2025) challenges educators to think about its role in shaping the future of education and the further development of AI-driven instructional methods. Using international research and practical cases, this paper analyzes both the strengths and weaknesses of vibe coding. Particular attention is given to studies implemented at universities abroad; then, strategies can be put forward for adapting such a system to Kazakhstan's higher education system. This research utilizes qualitative and comparative analysis, with higher education institutions as the main setting for investigations. Vibe coding technologies in Kazakhstan are becoming an important driver of educational change, offering flexible tools, creative learning environments, and student engagement through digital culture. They help teachers modernize lessons, solve traditional problems and support equal access to knowledge, making the system more adaptive and sustainable for future generations together.

Keywords: vibe coding, generative AI, programming education, computational thinking, debugging, higher education, AI-enhanced learning.

Introduction

Artificial intelligence (AI) is transforming the way universities design instruction, deliver learning experiences, and measure student performance. Apart

from such well-known applications as automated feedback, adaptive tutoring, and learning analytics, AI in various cases has even begun to impact how students create artifacts – i.e., digital documents (including software, digital content, and technical documentation) – in daily courses. Within this broader context, vibe coding has recently arisen as a leading teaching and practicing method. First defined for use in 2025, vibe coding is traditionally defined as an AI-supported development process where students produce, test and revise code through natural-language interaction with AI tools, as opposed to writing in a manual manner. Conceptually, it can be understood as a scaffolded prototyping model that emphasizes iterative implementation: students create intent, receive AI-generated outputs, test action before making decisions about the testing (e.g. by testing), then refine prompts or specifications to reach a solution that works. This new trend has become interesting because it seems to lower the cost of software creation and for newbies to programming there may be motivation rewards. Nevertheless, same characteristics that make vibe coding attractive (e.g. being fast, convenient and cognitive load is minimized at the initiation time of construction) could also pose pedagogical risks. When students are able to write code but do not know how it works, and why, the method can inadvertently incentivize shallow involvement, poor debugging, and too much use of AI. For institutions of higher education whose purpose is to develop durable computational thinking, professional reasoning, and applicable problem-solving skills, this tension is not trivial. So the question we face is not whether vibe coding can produce working prototypes, but if it is conducive to meaningful learning under why and what types of instructional design are necessary to avoid harmful learning experiences. Beyond home countries and unresolved methodological questions. Early studies out of Kazakhstan offer encouraging but incomplete signals. One such study (2025) examined vibe coding and the learning processes of 38 medical students in Gazi University in Turkey [1]. Results indicated gains in academic achievement and costs and time to prepare learning material for experimental groups were reduced. Such claims count in that they suggest an incremental improvement in efficiency: that if AI-supported workflows shorten time taken to prepare, while improving (or even erasing) learning outcomes, universities "may be able to shift instructional time into activities that have a higher value on learning such as mentoring, clinical reasoning practice, and project-based learning".

At the same time, however, several aspects of the evidence are ambiguous. An approximate initial cost of USD 0.5 for first-year medical students [1], which was described in the referenced figure, has suggested efficiency in cost. So far, however, it is not clear if that amount represents:

– a marginal cost of producing AI material per learner,
– the cost per student of accessing a platform subscription,
– a one-time initial expense of onboarding, or
– aggregate group-level savings values.

These distinctions matter for what they tell researchers about the interpretation and replicability of this outcome. If the value represents access to a platform rather than content generation, then cost benefits that come from this platform may entirely differ depending on all different institutions and their different licensing models and infrastructure. Furthermore, the results of the study are not readily transferable without full statistical verification. In studies that report "better performance" with only so little detail on variance, confidence intervals, effect sizes or robustness checks, that readers can glean to whether an observed gain can be considered to be stable and/or be explained by some combination of cohort effects, teacher differences, novelty effects or uncontrolled confounding. That is, although the direction of the findings appears promising, methodological constraints limit the extent to which conclusions be made based on them. However, a thread that remains consistent from first attempts is that students tend to consider AI-supported platforms particularly effective for learning challenging topics [1]. The point is, from a pedagogical perspective, perceived usefulness is not (or at least is not) equivalent to validated learning gain, but that it is meaningful in that perceived usefulness can impact on engagement, persistence, and readiness to try anything too challenging. It is on affective and motivational dimensions, e.g. simply "easier" does not necessarily mean "deeper". Second line of research has pointed out emotional aspect of learning to code with AI support. Empirical studies have shown emotional benefits of vibe coding, explaining how it brings down psychological pressure compared to other programming settings and how it can make learning feel easier, or even more manageable, for some students [2]. This is significant, for programming education often presents an emotional wall (i.e. anxiety over making errors, frustration about debugging, fear of failure) at an early stage in the learning process. Reduction of these barriers by AI scaffolding is likely to increase the likelihood of novices persisting with these structures long enough to become competent. Yet affective gains must be taken very seriously. Reduced stress, and faster progress, can support motivation, but are also capable of creating a "false fluency" effect: Students sense competence because outputs are visible; whereas their internal understanding remains shallow. For higher education, the practical implication is that vibe coding should be evaluated not only on satisfaction or perceived easiness, but also with a look at conceptual understanding, reasoning quality, debugging proficiency as well as transfer into

new contexts. Without this wider measurement framework, it's difficult to separate real improvement in learning from 'just-in-time convenience.

A divergent outcome by learner background: conclusions from comparative research. As a particularly educational point of view is provided by the recent research at the University of California San Diego and the University of Toronto that provided a clear comparison between how students varying in programming ability interacted with AI-enabled tools [3]. In such a study students were split between novice (CS1) students who had just begun programming and a small number of advanced fourth-year software engineering students. The approach differentiated between conventional development workflows to the advanced group, and the AI-powered environment (Replit) to the novices of the advanced group, inspired by vibe coding features. The results underscore a central pedagogical problem: the advantages of vibe coding aren't equally spread among skill levels. Both groups generated applications with readily apparent drawbacks, but beginner students' AI code contained far more errors, and they struggled to interpret, evaluate, and optimize the output. Most critically, novices hadn't mastered the technical reasoning necessary to diagnose problems, figure out what was wrong with a program, and make adjustments to the system to make the solution more effective. Advanced scholars were more prepared to give context-rich prompts for code development, give reasoning of expected behavior of code development processes, and structured debugging cycles. This indicates that on one hand, vibe coding can help to lower the bar for building a prototype, and the successful use of that code to the best of its ability relies on the knowledge of foundational skills (computational thinking, problem decomposition, debugging literacy) [3]. The other important finding was that interaction with AI was primarily by testing and debugging, and not by cutting code manually [3]. This is pedagogically instructive: as students would, this could be seen as treating AI as a generator, and then trying to validate or «steer» the output through iterations of validation. In this process, the ability to test, understand failures, and refine requirements is far greater than the ability to write every line of code from memory. Sophisticated students who understand the program's behavior are thus better positioned for taking benefits from the process; novices, in which they get stuck in exploratory and trying cycles with not a clue, might become habitual and become reliant on it instead. The implication for higher education: why we must be explicit in the pedagogy of integration. Combined, these preliminary studies point to a balanced interpretation. Though vibe coding can speed prototyping and lend a more supportive environment to learning, it also brings about risks that institutions of higher education will need to counter through explicit instructional design. Universities that promote vibe coding as a "shortcut" without developing

reasoning, debugging, and computational thought may end up with portfolios where students have functional demos but no ability to explain or to sustain the code they made – an outcome not consistent with the expected behavior in professional development of software. For this reason, teaching integration would require direct training on (a) code reasoning, (b) debugging, (c) testing design, and (d) critical analysis of AI-generated outputs [3]. Practically this translates into associating vibe coding with the learning outcomes that involve explaining and verifying the logic, not just working prototypes. It also demands tests that reward students for getting at program rationale, the analysis of errors, and the trade-offs of design – areas where novices tend to need structured guidance. Vibe coding constructs: A functional classification for educational programs. In order to facilitate systematic inquiry and curriculum development, vibe coding can be explained in terms of its key modalities. Based on the literature about interactivity and programming type, vibe coding elements are classified into visual programming, audio-based programming, and text-generative programming [4]. Although the modalities differ in interface, they all share a common purpose: teaching learners how to quickly build software artifacts with the help of AI-mediated assistance. A common vibe coding process in web development contexts constitutes a four-step cycle:

But the same process also shows the limitations. If this process evolves to an infinite cycle of prompt-generate-test without any awareness of conceptual reflection, students may not generate lasting mental models of how a program behaves. This can, over time, become a hindrance to being able to transfer information to novel problems, write code without help, and maintain systems in more complex environments. Hence, comparisons of vibe coding with conventional coding (Figure 2) must not be construed as either-or. Rather, they are best conceptualized as a continuum of scaffolding: from the most high-supported AI-assisted generation to the least supported manual, instructionally driven development of these approaches across the continuum of learners' competence. The context of Kazakhstan and the importance of national-level research. Currently, no evidence base for the purpose of providing a way for universities to evaluate vibe coding in local higher education context in Kazakhstan exists (in spite of the international efforts). This gap is critical. The efficacy of implementation can still hinge upon things like the language we use in prompts and with instructional software (in and around them), the alignment of curricular material, faculty preparedness, institutional assessment, and access to stable AI infrastructure. Furthermore, cultural expectations about independent work, academic integrity, and the role of technology in learning may influence how students adopt or resist AI-mediated workflows. Absence of evidence of this situation can lead to adoption decisions that are driven purely by global trends

rather than educational impact. As such, Kazakhstan will need to have a more methodical study of vibe coding at both the opportunities and constraints level. Such research needs to address not only outcomes but learning processes as well: how students generate prompts, how they confirm results, how they debug, and how they explain their solutions. Just as important is understanding the conditions and how they can inhibit over-reliance: what prep instruction is desired, what instructional supports are efficacious, and what measure.

Materials and Methods

The study utilized a qualitative, descriptive–exploratory design to investigate the influence of Vibe Coding and its possible impact on students' programming development and emotional engagement in learning within AI-augmented environments. Instead of testing one intervention in a single classroom, the research conducted a secondary-data approach and performed an integrative examination of peer-reviewed research articles, conference proceedings, and documented classroom implementations of AI-assisted programming education, computational thinking (CT), and learner experiences in coding contexts.

Research Corpus and Selection Logic

The empirical basis consisted of published and practice-oriented sources describing the use of AI tools in programming instruction – particularly for learning environments where Vibe Coding characteristics such as:

– prompt-based generation,

– rapid prototyping,

– iterative testing, and

– AI-guided debugging were present [4].

The corpus was created to facilitate comparative interpretation between:

1 conventional (non-AI-centered) programming teaching; and

2 studies in which Vibe Coding was central for instruction within the classroom.

We considered studies that:

– reported observable learning behaviors,

– discussed cognitive or affective outcomes (e.g., confidence, anxiety, motivation), and/or

– examined developmental aspects of CT-related skills (e.g., problem decomposition, debugging, logical reasoning), to receive pre-trial priority [2].

Subjects and Contexts of Practice

The target population covered undergraduate students in computing or software-engineering courses at universities where AI-based tools were being piloted or utilized directly within the curriculum. Although the current work

did not capture original experimental data from a single institution, comparable descriptions of learners' profiles were extracted, including:

– novice vs. advanced learners,

– types of tasks (e.g., web/app prototyping), and

– instructional scenarios (e.g., AI-supported environments such as Replit or similar platforms).

Types of Evidence and Data Sources

To construct a substantiated understanding of how Vibe Coding functions as a learning scaffold, the analysis relied on several forms of evidence identified in the literature:

1 Recordings of sessions with Vibe Coding workflows – allowing tracking of learners' emotional responses (e.g., reduced frustration), problem-solving behaviors (e.g., trial-and-test cycles), and task-performance indicators [3, 5].

2 Instructor notes – describing perceived impacts on learner engagement, productivity, and classroom-support characteristics [2, 4].

3 Written student feedback – covering perceived usefulness, motivational impact, and self-reported improvements in confidence and technical reasoning [1, 6].

Comparative Reference Cases

A key comparative aspect in the study was the difference in interaction between novice and advanced programmers when working with AI-generated code. The analysis drew upon reference cases published by the University of California San Diego and the University of Toronto, where learners were grouped by programming background and evaluated across AI-assisted and conventional development methods [3].

The comparative synthesis focused on three competence domains repeatedly discussed in the literature:

– computational thinking (CT) development,

– debugging capability, and

– logical/technical reasoning [3, 5, 6].

Analytical Procedure

Using an interpretive thematic analysis, the research sought to identify recurring patterns and derive categories of findings from the selected literature. Themes were organized into three major clusters:

– Positive outcomes: higher motivation, faster prototyping, reduced emotional strain [2, 4];

– Limitations: brittleness or inconsistency of generated code, novice dependency, and opaque reasoning [3, 5];

– Pedagogical implications: need for explicit instruction in debugging and CT development [2].

To ensure coherence, thematic coding was synchronized with documented feature sets of Vibe Coding and traditional coding – including ease of entry, accessibility, and depth of cognitive engagement [3, 4]. The final interpretation was produced by synthesizing converging and divergent findings into a coherent representation of Vibe Coding as an educational innovation and motivational scaffold in AI-based learning environments.

Methodological Rationale and Connection with the Topic

The selected qualitative design is appropriate because Vibe Coding remains a recently defined practice characterized by diverse implementations and evolving tool ecosystems. Under these circumstances, qualitative synthesis enables early identification of potential opportunities, limitations, and pedagogical requirements without overstating causal effects [1,3]. This approach helps reveal how Vibe Coding may accelerate early skill acquisition while reducing emotional barriers, yet also exposes risks to long-term learning outcomes – particularly in contexts with low CT or weak debugging foundations [5,6].

Methodological Note – Vibe Coding as a Learning Scaffold

Vibe Coding features significantly shorten the feedback loop between conceptualization, implementation, and visible output.For many learners, programming has traditionally been emotionally exhausting due to frequent errors and delayed progress.Because programming involves design thinking, creativity, and iterative reasoning, rather than mere syntax reproduction, Vibe Coding can serve as a temporary scaffold that allows students to begin projects, examine code structures, and achieve «quick wins» – especially when they are uncertain about where to start [6].However, these advantages have inherent limitations. When AI systems generate large portions of code, novices often struggle to assess correctness, identify subtle logical flaws, or explain why a solution works [7, 8].

Results and Discussion

The results of this report suggest that vibe coding as an AI-enabled course should be considered as an adjunct rather than substitute to traditional approaches to teaching programming. Specifically, AI assistance seems to alleviate cognitive barriers common for novice programmers. Instant feedback, code suggestions and a low-risk space to experiment allows students to test their ideas without the same fear of failure common in programming tasks. This affordance is likely to increase interest of learning, particularly in large enrolment cases in which individual help is limited. Nonetheless, a number of limits need to be considered. One of the central concerns is vibe coding might value speed and convenience over conceptual comprehension. We know that the students might focus on AI results over the logical reasoning, algorithmic design, or reasoning behind the design. This could perpetuate "black box" mentality, where AI replies are accepted as gospel

rather than questionable evidence[11, 12]. Thus, these key competencies such as debugging, problem decomposition, and rationale for the architecture may be a bit shallow in some cases. The latter issues include robustness of AI-generated code, and maintainability of the code. Solutions developed that are insufficiently basic can result in hidden errors and be susceptible to security, scalability, or maintainability issues. In addition, if learners are discouraged from achieving deeper competence and professional-level judgement since they evade the productive struggle, the continuous development of programming expertise might suffer [14]. From a pedagogical perspective, vibe coding can be worthwhile when added intentionally to instructional schemes that aim to increase the development of reasoning and reflection skills. In practice, it is most justifiable to add the notion of vibe coding to the curriculum after students have established baseline computational skills and programming logic, such that AI is not a crutch, but a collaborator that complements cognitive work. But under such a case, vibe coding becomes a creative resource and a positive catalyst for engagement, since it fits well with a structured approach to learning in which analytical skills, collaboration and ongoing professional development matters [13].

The results indicate that the vibe coding fosters students' motivation, in particular by promoting the confidence and confidence of early programming practice. When interacting with AI-based coding assistants, participants frequently reported finding more joy and feeling less anxiety. These tools facilitated quicker identification of syntax patterns, program structure, and execution flow for beginners, and many learners rapidly delivered prototype applications compared to those receiving traditional instruction only. But the approach showed a significant disparity between novice and advanced learners in how they had used and thrived with vibe coding. Novice students generally created functional but inefficient or messy code. They often found it difficult to elaborate on the logic behind AI suggestions and demonstrated a low level of proficiency with debugging principles. By contrast, advanced students generally approached vibe coding as a collaborative tool: they interacted in "talk" with the AI iteratively in context-rich ways, engaged in systematic problem-solving, and used AI output to refine architecture and design choices. Such behaviours correlate with greater computational literacy and technical maturity. Existing evidence from previous academic cases has highlighted a similar pattern between the two groups, where the majority of the use of AI (80 %) was from "beginners", as this is when the students generated code themselves, while only 40 % from the advanced students used AI for the same purpose [11, 16]. Less naive learners were more likely to use AI for hypothesis testing, efficiency gains, or small, targeted improvements. This pattern supports the final analysis that the educational potential of vibe coding will rely heavily on previous experience and

conceptual preparation of students. Furthermore, qualitative analysis of instructor responses indicated that while vibe coding can enhance students' creative interactions, targeted instruction on critical thinking and debugging still must be addressed by teachers. Without intentional guidance, heavy AI application can stifle mastery of core programming elements. Mor generally, the results highlight a need for curricular model balances (a blend of traditional content on the one hand and AI mediated contexts on the other) so that students cannot only generate code, but can also justify their design choices and understand.

Funding

This research has been funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP23489632, Didactic conditions for using the potential of big data science and machine learning in the training of future computer science).

Conclusion

Implementation of AI and vibe coding in higher education will bring opportunities and issues respectively. As is argued in the present study, vibe coding provides a practical and motivational option for novices by removing some of the hindrances to new entry and a great deal of the emotional stressors of traditional instructions. AI-enabled interactive platforms allow instant feedback, speeding up prototyping, and giving an encouraging sense of progress – important in large-scale educational systems where it's challenging to offer personalized coaching [16]. Nevertheless, the empirical evidence reviewed here is clear that vibe coding is an inadequate substitute for traditional programming instruction. AI-supported solutions could help beginners come up with working solutions, but without the conceptual abilities – the ability to reason computationally, formulate problem-solving strategies, and apply technical judgment in order to make sense of and validate and optimize AI code – to level up to more advanced students. Vibe coding lends itself to more complex and nuanced explorations, making it easier for more experienced learners to derive value from them through guided prompting and a well-orchestrated debugging approach. This emphasizes that the real difference lies in learners' foundational preparation and effective implementation. Without such preparation, over-trust in AI can slowly erode critical thinking and, eventually, slow long-term professional growth. There are also questions about software quality and collaboration. Work done mostly from AI-generated code is often brittle, complex to scale and not easily maintainable. Additionally, team collaboration can be detriment if learners lean on automated solutions without understanding logic behind them: common thinking, code check practices and co-designed decisions require transparency and understanding. So, vibe coding should be implemented in tandem with teaching explicit code

understanding, debugging, collaborative development. So, to conclude, vibe coding can be seen as a fantastic companion to classical programming education, not the ultimate replacement. Its best advantages have been increased engagement, prompt comments, and more safe experimentation – especially for newcomers. Yet, in order to produce competent and independent developers, further education needs to manage the tension between AI-driven workflows that support advanced workflows and instruction that lays those computational foundations, followed by a professional programming process. Vibe coding can be valuable in early software engineering education for prototyping of applications and development of applications when implemented and taught suitably [12]. In addition, the adoption of vibe coding into Kazakhstan's hig her education can be informed by structured scaffolding, ongoing assessment, and intentional development of technical reasoning so that students are not only producing working prototypes but also possessing the enduring competencies necessary for sustainable and lasting success in software development.

References

1 **Leksic, A.** Algospeak: How Social Media Is Transforming the Future of Language. – New York : Alfred A. Knopf, 2025. – 256 p. – ISBN 9780593804087.

2 **Avouris, N., Sgarbas, K., Caridakis, G., Sintoris, C.** Teaching Introduction to Programming in the Times of AI: A Course Redesign Case Study // arXiv preprint. – 2025. [Electronic resource] – URL: https://arxiv.org/abs/2508.06572 DOI: https://doi.org/10.48550/arXiv.2508.06572 (Accessed date: 02.01.2026).

3 **Chow, M., Ng, O.** From Technology Adopters to Creators: Leveraging AI-Assisted Vibe Coding to Transform Clinical Teaching and Learning // Medical Teacher. – 2025. – P. 1–3. – https://doi.org/10.1080/0142159X.2025.2488353.

4 **Elnaffar, S., Rashidi, F., Abualkishik, A. Z.** Teaching with AI: A Systematic Review of Chatbots, Generative Tools and Tutoring Systems in Programming Education // arXiv preprint 2510.03884. – 2025. – https://doi.org/10.30935/jdet/15860. 2025. – Vol. 5, No. 1. – Article ep2506. (Accessed date: 02.01.2026).

5 **Gadde, A.** Emerging Intuitive Programming Approaches and AI-Assisted Development Environments // Journal of Computer Science and Technology Studies. – 2025. – Vol. 7. – №. 4. – P. 556–572. – https://doi.org/1515211625-/12335860

6 **Geng, F., Shah, A., Li H., Mulla, N., Swanson, S., Raj, G., Zingaro, D., Porter, L.** Exploring Student AI Interactions in Vibe Coding // arXiv

preprint. – 2025. [Electronic resource] – URL: https://arxiv.org/abs/2507.22614 – https://doi.org/abs/2507.22614 (Accessed date: 02.01.2026).

7 **Hernandez, A. A., Albina, E. M., Caballero, A. R.** Generative Artificial Intelligence for Programming Education: Enhancing Input to Outcome Based Teaching and Learning Approaches // Proceedings of the 7th International Symposium on Computer Consumer and Control (IS3C). – 2025. – https://doi.org/10.1109/IS3C65361.2025.11131023.

8 **Kim, G., Yegge, S.** Vibe Coding: Building Production-Grade Software with GenAI Chat Agents and Beyond. – Portland : IT Revolution, 2025. – 320 p. – ISBN 9781966280026.

9 **Kıyak, Y. S., Emekli, E., Kara, T., Coşkun, Ö., Budakoğlu, I. İ.** AI Teaches Surgical Diagnostic Reasoning to Medical Students: Evidence from an Experiment Using a Fully Automated Low-Cost Feedback System // Journal of Surgical Education. – 2025. – Vol. 82, No. 10. – Article 103639. – https://doi.org/10.1016/j.jsurg.2025.103639.

10 **Ma, B., Li, G., Chen, L., Tang, C., Xie, Y., Shimada, A.** Scaffolding Metacognition in Programming Education: Understanding Student AI Interactions and Design Implications // arXiv preprint 2511.04144. – 2025. [Electronic resource] – URL: https://arxiv.org/abs/2511.04144 (Accessed date: 02.01.2026).

11 **Modley, E., Essex, M.** The Vibe Coding Playbook. – San Francisco : Pragmatic Bookshelf, 2025. – 288 p. – ISBN 9781680501234.

12 **Nikolopoulou, K.** Generative Artificial Intelligence and Sustainable Higher Education // Journal of Digital Educational Technology. – 2025. – Vol. 5. – №. 1. – P. 256. – https://doi.org/10.30935/jdet/15860

13 **Osmani, A.** Beyond Vibe Coding: From Coder to AI Era Developer. – Sebastopol : O'Reilly Media, 2025. – 254 p. – ISBN 9798341634725. – https://doi.org/104521/r0er/1512

Received 24.01.26.
Received in revised form 28.07.26.
Accepted for publication 03.08.26.

*С. Меруерт[1], *А. А. Тұрдыбек[2]*
[1,2]Л. Н. Гумилев атындағы Еуразия ұлттық университеті,
Қазақстан Республикасы, Астана қ.
24.01.26 ж. баспаға түсті.
28.07.26 ж. түзетулерімен түсті.
03.08.26 ж. басып шығаруға қабылданды.

ЖОҒАРЫ БІЛІМДЕГІ VIBE CODING: БАСТАУШЫЛАР МОТИВАЦИЯСЫ, ҚАТЕЛЕРДІ ТҮЗЕТУ ЖӘНЕ ПЕДАГОГИКАЛЫҚ СКАФФОЛДИНГ

Бұл мақалада жоғары білім беру контекстінде «вайб-кодинг» құбылысының (AI құралдары арқылы промптқа негізделген код генерациясы, жедел прототиптеу, итеративті тестілеу және AI-жетекшілігімен қателерді түзету) оқу үдерісіне ықпалы талданады. Зерттеу сапалық, сипаттамалық-ізденістік дизайнға негізделіп, AI-көмегімен бағдарламалау білімін оқыту, есептеуіш ойлау (computational thinking) және студент тәжірибесі бойынша жарияланған мақалалар мен оқу-тәжірибелік материалдарды интегративті шолу арқылы жүйелейді. Тақырыптық талдау нәтижелері вайб-кодингтің бастапқы кезеңде мотивацияны арттырып, оқу мазасыздығын азайтуы және синтаксис/құрылымды тез меңгеруге көмектесуі мүмкін екенін көрсетеді. Сонымен қатар жаңадан үйренушілерде «қара жәшік әсері», ягни AI нәтижелеріне шамадан тыс сенім, алгоритмдік түсінудің таяздығы, жүйелі дебаг пен архитектуралық ойлаудың әлсіздігі сияқты тәуекелдер байқалады. Сонымен бірге мақалада бұл тәсілді қазақстандық жоғары оқу орындарының тәжірибесіне бейімдеп енгізудің өзектілігі атап өтіледі, себебі цифрлық дағдыларды дамыту бүгінгі білім беру жүйесінің басты талаптарының біріне айналып отыр. Вайб-кодинг дұрыс ұйымдастырылған жағдайда студенттердің қызығушылығын оятып, өздігінен ізденуге, қателерден қорықпай тәжірибе жасауға үйретеді. Дегенмен AI құралдары тек көмекші рөл атқарып, оқытушының бағыт-бағдары мен терең түсіндіруімен қатар жүргенде ғана шынайы нәтиже береді.

Кілтті сөздер: вайб-кодинг, генеративті AI, бағдарламалауды оқыту, есептеуіш ойлау, дебаг, жоғары білім, AI-қолдауымен оқу.

*С. Меруерт[1], *А. А. Тұрдыбек[2]*
[1,2]Евразийский национальный университет
имени Л. Н. Гумилева,
Республика Казахстан, г. Астана.
Поступило в редакцию 24.01.26.
Поступило с исправлениями 28.07.26.
Принято в печать 03.08.26.

VIBE-КОДИНГ В ВЫСШЕМ ОБРАЗОВАНИИ: МОТИВАЦИЯ НАЧИНАЮЩИХ, ПРОБЕЛЫ В ОТЛАДКЕ И ПЕДАГОГИЧЕСКИЙ СКАФФОЛДИНГ

В статье рассматривается феномен «вайб-кодинга» как AI-поддерживаемой практики обучения программированию, основанной на промпт-генерации кода, быстром прототипировании, итеративном тестировании и AI-ориентированном исправлении ошибок. Исследование выполнено в логике качественного описательно-исследовательского дизайна и опирается на интегративный анализ публикаций и учебно-практических кейсов по AI-ассистированному программированию, развитию вычислительного мышления и учебному опыту студентов. Тематический анализ показывает, что вайб-кодинг способен повышать мотивацию начинающих, снижать тревожность и ускорять освоение базовых представлений о структуре программ и синтаксисе за счет быстрых циклов обратной связи. Вместе с тем выявляются значимые ограничения: риск «эффекта черного ящика», смещение внимания к готовым AI-ответам, недостаточное развитие алгоритмического мышления, навыков декомпозиции задач и систематического отладки. Дополнительно в статье подчеркивается практическая значимость внедрения вайб-кодинга в образовательную среду Казахстана, где существует потребность в обновлении методов обучения и повышении цифровой грамотности учащихся. Отмечается, что грамотное сочетание AI-инструментов с традиционным преподаванием может способствовать формированию самостоятельности, критического мышления и устойчивого интереса к программированию, при условии педагогического сопровождения и методического контроля.

Ключевые слова: вайб-кодинг, генеративный ИИ, обучение программированию, вычислительное мышление, отладка, высшее образование, AI-усиленное обучение.

Теруге 03.08.2026 ж. жіберілді. Басуға 14.09.2026 ж. қол қойылды.
Электронды баспа
6,45 МБ
Шартты баспа табағы 49,9
Компьютерде беттеген З. Ж. Шокубаева
Корректорлар: А. Р. Омарова, Д. А. Қожас
Тапсырыс № 4597

Сдано в набор 03.08.2026 г. Подписано в печать14.09.2026 г.
Электронное издание
6,45 МБ, Усл.п.л. 49,9.
Компьютерная верстка З. Ж. Шокубаева
Корректоры: А. Р. Омарова, Д. А. Қожас
Заказ № 4597

«Toraighyrov University» баспасынан басылып шығарылған
Торайғыров университеті
140008, Павлодар қ., Ломов к., 64, 133 каб.

«Toraighyrov University» баспасы
Торайғыров университеті
140008, Павлодар қ., Ломов к., 64, 133 каб.
8 (7182) 67-36-69
e-mail: kereku@tou.edu.kz
www.pedagogic-vestnik.tou.edu.kz