УДК 004.056.5
DOI: 10.26102/2310-6018/2026.59.8.014

Integrating information security and systems engineering into a multilingual sign language learning game

A. Ashrafi✉, Y.N. Philippovich, V.S. Mokhnachev, A.A. Makhmud

Moscow Polytechnic University, Moscow, the Russian Federation

Abstract. The development of accessible and secure digital tools is critical for inclusive education. While our prior work demonstrated the pedagogical efficacy of a multilingual sign language learning game-evidenced by a 35 % improvement in sign recognition and a System Usability Scale (SUS) score of 90/100-the transition to a production-ready, internet-connected platform introduces significant security and operational risks. To address this, this paper presents a unified framework that integrates four core disciplines-web application security analysis, software lifecycle automation (CI/CD), secure digital document workflows, and security information and event management (SIEM)-to systematically harden the platform and ensure its long-term resilience. We implement an integrated approach where continuous security testing, protected data workflows, and threat-informed development work in concert to ensure platform integrity. Results demonstrate the elimination of critical vulnerabilities, an 87.5 % reduction in high-severity issues within three months, and a 65.4 % improvement in mean time to remediate (MTTR). The secure certificate system achieved a 99.6 % issuance success rate and 100 % forgery detection, with public verification in under one second. This lab-validated methodology provides a practical blueprint for transforming innovative educational tools into resilient, production-ready systems.

Keywords: sign language, educational game, web application security, CI/CD automation, secure document flow, SIEM, data privacy, inclusive educational technologies.

For citation: Ashrafi A., Philippovich Y.N., Mokhnachev V.S., Makhmud A.A. Integrating information security and systems engineering into a multilingual sign language learning game. *Modeling, Optimization and Information Technology.* 2026;14(8). URL: https://moitvivt.ru/ru/journal/article?id=2396 DOI: 10.26102/2310-6018/2026.59.8.014

Интеграция информационной безопасности и системной инженерии в мультиязычную обучающую игру по жестовым языкам

А. Ашрафи✉, Ю.Н. Филиппович, В.С. Мохначев, А.А. Махмуд

Московский политехнический университет, Москва, Российская Федерация

Резюме. Разработка доступных и безопасных цифровых инструментов имеет очень важное значение для инклюзивного образования. Хотя в нашей предыдущей работе была продемонстрирована педагогическая эффективность мультиязычной обучающей игры по жестовым языкам, о чем свидетельствует улучшение распознавания жестов на 35 % и оценка по шкале юзабилити системы (SUS) 90 из 100, переход на готовую к промышленному использованию платформу, подключенную к интернету, создает значительные риски с точки зрения безопасности и эксплуатации. Для решения этой проблемы в данной работе представлена унифицированная методология, интегрирующая четыре ключевых направления: анализ безопасности веб-приложений, автоматизация жизненного цикла ПО (CI/CD), организация защищенных цифровых документооборотов и управление информацией и событиями безопасности (SIEM). Данная интеграция позволяет систематически усиливать защиту платформы и обеспечивать ее долгосрочную отказоустойчивость. В работе реализуется комплексный подход, при котором непрерывное тестирование безопасности, защищенные

потоки данных и разработка с учетом угроз (threat-informed development) действуют согласованно для обеспечения целостности платформы. Результаты демонстрируют полную ликвидацию критических уязвимостей, снижение количества проблем высокой степени серьезности на 87,5 % в течение трех месяцев, а также сокращение среднего времени на устранение уязвимостей (MTTR) на 65,4 %. Система выдачи цифровых сертификатов достигла 99,6 % успешности выпуска и обеспечила 100 % обнаружение попыток подделки при времени публичной верификации менее одной секунды. Данная методология, валидированная в лабораторных условиях, представляет собой практическое руководство по трансформации инновационных образовательных инструментов в отказоустойчивые системы, готовые к промышленной эксплуатации.

Ключевые слова: язык жестов, образовательная игра, безопасность веб-приложений, автоматизация CI/CD, защищенный документооборот, SIEM, конфиденциальность данных, инклюзивные образовательные технологии.

Для цитирования: Ашрафи А., Филиппович Ю.Н., Мохначев В.С., Махмуд А.А. Интеграция информационной безопасности и системной инженерии в мультиязычную обучающую игру по жестовым языкам. *Моделирование, оптимизация и информационные технологии.* 2026;14(8). (На англ.). URL: https://moitvivt.ru/ru/journal/article?id=2396 DOI: 10.26102/2310-6018/2026.59.8.014

Introduction

The challenge of creating engaging, effective, and accessible digital tools for sign language education is both a pedagogical and a technological challenge. Our prior work presented a fun and engaging educational game built with HTML5, CSS3, and JavaScript, where players take on a first-person journey through Bengali, Russian, and American Sign Language (ASL), as shown in Figure 1. Using a "Hunt-Revelation-Reinforcement" cycle [1] in culturally authentic environments, the game showed strong potential (Figure 2). This approach engaged learners and supported steady progress.

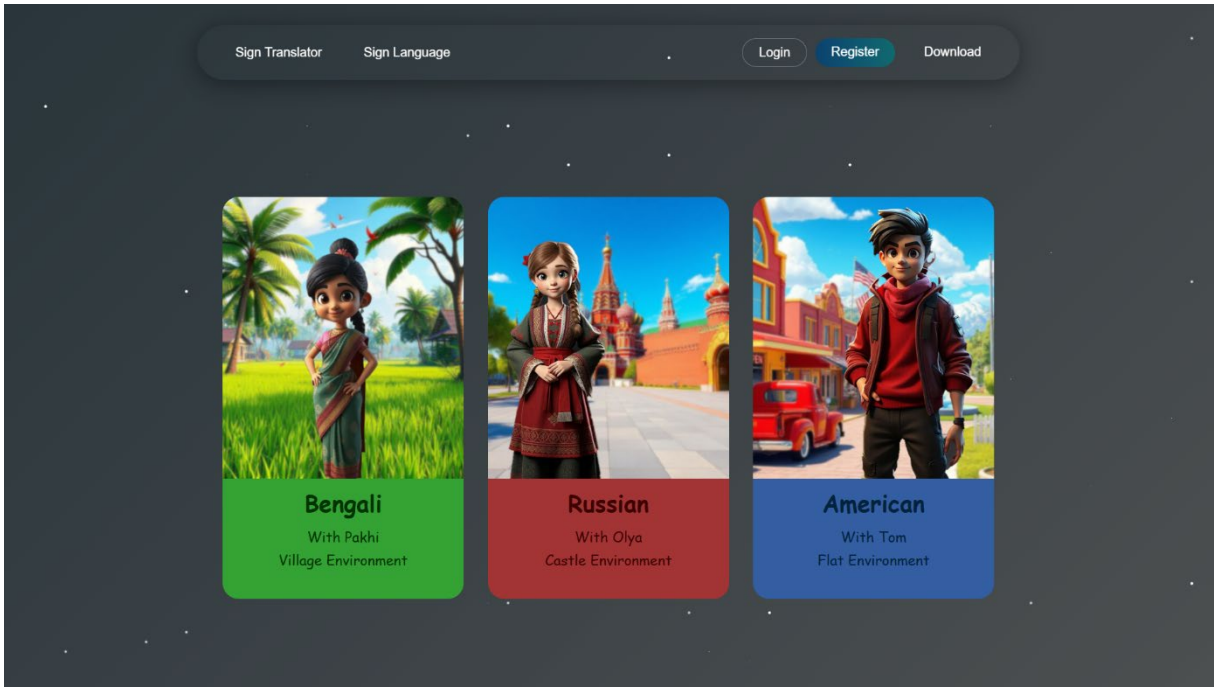

Figure 1 – User interface for selecting a target sign language
Рисунок 1 – Пользовательский интерфейс для выбора целевого жестового языка

Alpha testing with 50 participants showed a statistically significant 35 % improvement in sign recognition, with 85 % of volunteers maintaining high motivation and the platform achieving an excellent System Usability Score (SUS) of 90/100 [2]. These findings validated the core hypothesis that a game-based approach can foster high engagement and steady learning progress.

The first version of the game was simple and worked well with a limited number of users on similar devices. However, the next steps for the game-like saving each player's progress, giving them achievements, and awarding official certificates-require a significant architectural change. This demand makes it clear we need to move from a simple app to an internet-connected system with a central server. This shift is necessary for the game to grow and offer these new features.

This change also creates new risks. Once the platform saves personal user data and issues digital certificates, it becomes more attractive to attackers. So, as the game evolves, implementing much stronger security is a must.

Figure 2 – User interface for answer selection and validation
Рисунок 2 – Пользовательский интерфейс для выбора и проверки ответа

Our previous work on Bengali Sign Language lexicography [3] established an information-theoretic metric for automated dictionary development, demonstrating the importance of systematic, data-driven approaches for low-resource sign languages. The present study extends this methodological rigor to the security and systems engineering domain, ensuring that the educational game platform is not only pedagogically effective but also resilient and trustworthy for its users.

This paper constructs a unified framework, drawing from four key information security and systems engineering disciplines. It is structured as follows: Section 2 provides an overview of the four integrated security and engineering disciplines. Also it details the proposed framework and methodology. The next section describes the results and validation of our

approach. Then the discussion section, and finally, it concludes with the outlined directions for future work.

Materials and methods

To systematically address the risks and operational challenges, we integrate four complementary disciplines into the game's development lifecycle.

Web Application Security Analysis (From Vulnerabilities to Protection): This discipline involves the proactive identification and mitigation of weaknesses in the game's code and architecture. For a web application, common threats include:

– Cross-Site Scripting (XSS) [4];
– Cross-Site Request Forgery (CSRF) [5];
– Insecure Data Handling [6].

To solve such issues, we model the educational game application A as a state machine, defined by the tuple (C, Γ, S_t), where C is the codebase, Γ the system architecture, and S_t the state at time t. A threat agent TA can compromise A by exploiting a vulnerability V from the set of all potential weaknesses ν.

Cross-Site Scripting (XSS): XSS [7] constitutes an injection attack where unsanitized user input is rendered within the application's output. Formally, let Render (S, I) be the function that renders application state S with embedded input I.

If I contains a malicious script σ, and Render executes it in the user's browser, that's a security problem. The main way to prevent this is by carefully encoding the input before displaying it. We use a sanitization function, Encode, which transforms I into a safe version, I'. When we render with $Encode(I)$, any harmful scripts σ are turned into harmless data, preventing the attack defined in formula (1), which is given below:

$$\forall \sigma \in I, Render(S, Encode(I)) \vDash executes(\sigma). \quad (1)$$

Cross-Site Request Forgery (CSRF): CSRF forces an unauthorized state transition in A via a request crafted from an external domain. A vulnerability is present when a valid user session Φ can be leveraged to execute a privileged action a ∈ A via a request R where the request origin is not trusted, defined in formula (2), which is given below:

$$Origin(R) \notin TrustedDomain(A). \quad (2)$$

The standard mitigation is the use of a synchronized anti-CSRF token. The server embeds a token τ = f(Φ), where f is a cryptographically secure function and K_{secret} a server-side secret. Any state-changing request R must include τ, which the server verifies as a formula:

$$VerifyToken\, R \equiv (Token(R) - f(Session_{\Phi}, K_{secret})). \quad (3)$$

This ensures the request is intentional and originates from the legitimate application state.

Automation of Software Lifecycle Processes (CI/CD): Continuous Integration and Continuous Deployment (CI/CD) [8] automate the process of moving code from development to production. For our research, a CI/CD pipeline automatically tests, integrates, and deploys changes reliably. This includes:

– Automated Testing: Unit tests for game logic, integration tests for APIs and content loading, and end-to-end (E2E) tests simulating user interactions.
– Security Testing: It uses special tools called Static Application Security Testing. The short name for these tools is SAST [9]. These tools are added into the pipeline. They automatically scan the code to find security vulnerabilities.
– Automated Deployment: It uses scripts to move the software.

– Infrastructure as Code (IaC): Server configuration is managed in a special way. This includes setting up the web server. It also includes writing firewall rules. All of this is done using definition files. These files are written in a format that a computer can read directly. This method gives two big benefits. The first benefit is consistency. The second benefit is repeatability.

Technologies for Secure Document Workflow. The game's certificate system is a form of digital credential. A secure workflow ensures these documents are authentic, tamper-proof, and verifiable. This involves:

– Cryptographic Hashing;
– Digital Signatures;
– Verification Mechanisms.

Security Information and Event Management (SIEM). As the game scales, centralized monitoring becomes critical. SIEM[1] involves:

– Log Aggregation;
– Event Correlation and Analysis;
– Incident Detection and Response.

Proposed Framework: This study uses a practical method. It integrates four core disciplines – web application security, automated software lifecycle management, secure digital workflows, and security monitoring-into a unified development and operational framework.

The methodology is structured into six sequential phases, each combining theoretical analysis with hands-on implementation and validation.

To visualize how these four disciplines collectively support the transformation, Table 1 maps each discipline to its primary contribution across the six methodological phases.

Table 1 – Mapping of security and engineering disciplines to methodology phases
Таблица 1 – Сопоставление направлений безопасности и инженерии с этапами методологии

Methodology Phase	Web Application Security	CI/CD Automation	Secure Document Workflow	SIEM & Monitoring
Phase 1: Threat Modeling & Design	Threat Identification	Pipeline Planning	Credential Requirements	Logging Requirements
Phase 2: CI/CD Security Pipeline	SAST/DAST Integration	Automated Gates		
Phase 3: Cryptographic Certificates		Build Automation	Signing & Verification	Audit Logging
Phase 4: SIEM Deployment	Alert Rules			Correlation & Detection
Phase 5: Penetration Testing	Validation	Remediation Pipeline	Forgery Testing	Detection Validation
Phase 6: Continuous Operation	Vulnerability Patches	Automated Updates	Credential Revocation	Ongoing Monitoring

Threat Modeling and Secure Design: It means proactively identifying how attackers could target the platform – such as stealing user progress data, forging achievement certificates, or disrupting learning sessions – and building protective measures directly into the game's

[1] Griffith S., Morris Th.H. Network intrusion detection system. In: *Encyclopedia of Cryptography, Security, and Privacy*. Cham: Springer; 2025. P. 1646–1648. https://doi.org/10.1007/978-3-030-71522-9_1491

architecture from the start. This involves analyzing each component – user accounts, certificate issuance, progress storage, and game APIs – through frameworks like STRIDE [10] to anticipate threats like spoofing (fake logins), tampering (altered scores), or data leaks (defined in Figure 3).

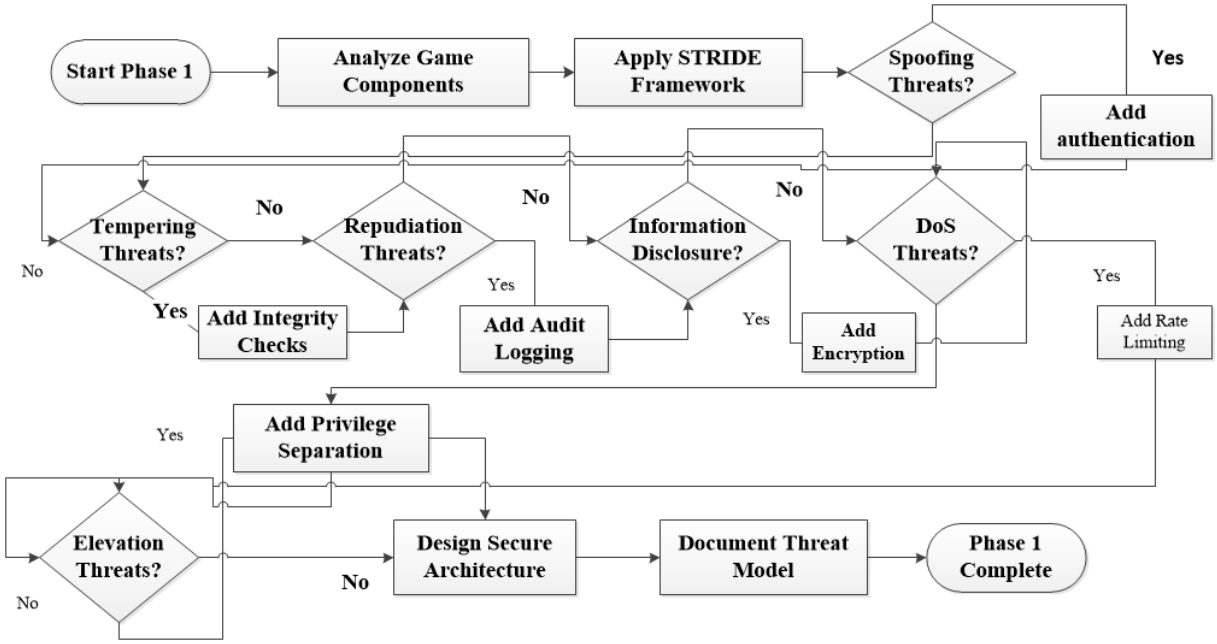

Figure 3 – Steps of threat modeling and secure design
Рисунок 3 – Этапы моделирования угроз и проектирования безопасной среды

Automated CI/CD Security Pipeline: It implements a fail-fast, multi-stage verification process where every code commit triggers automated static security scanning (SAST), defined in Figure 4, followed by comprehensive testing suites (unit, integration, and end-to-end tests), and-upon successful deployment to a staging environment-dynamic security analysis (DAST) [11].

Any detected vulnerability or test failure immediately halts the pipeline, notifies developers, and requires remediation before progression, ensuring only secure, functional code advances. Successful builds proceed through an optional manual security [12] review before final deployment via Infrastructure as Code (IaC), guaranteeing consistent, reproducible, and secure production releases while embedding security validation throughout the entire development lifecycle.

Cryptographic Certificate System: It establishes a secure, verifiable digital credential workflow where user achievements are cryptographically signed using a securely stored private key, embedded within certificates containing QR codes for third-party validation, with all issuances logged to SIEM for audit and fraud detection (shown in Figure 5).

The Security Monitoring [13] and SIEM Deployment: It establishes centralized log aggregation with configurable correlation rules for detecting threats like brute-force attacks and certificate fraud, connected to automated alert channels, and validated through simulated attack scenarios to ensure proactive incident detection (defined in Figure 6).

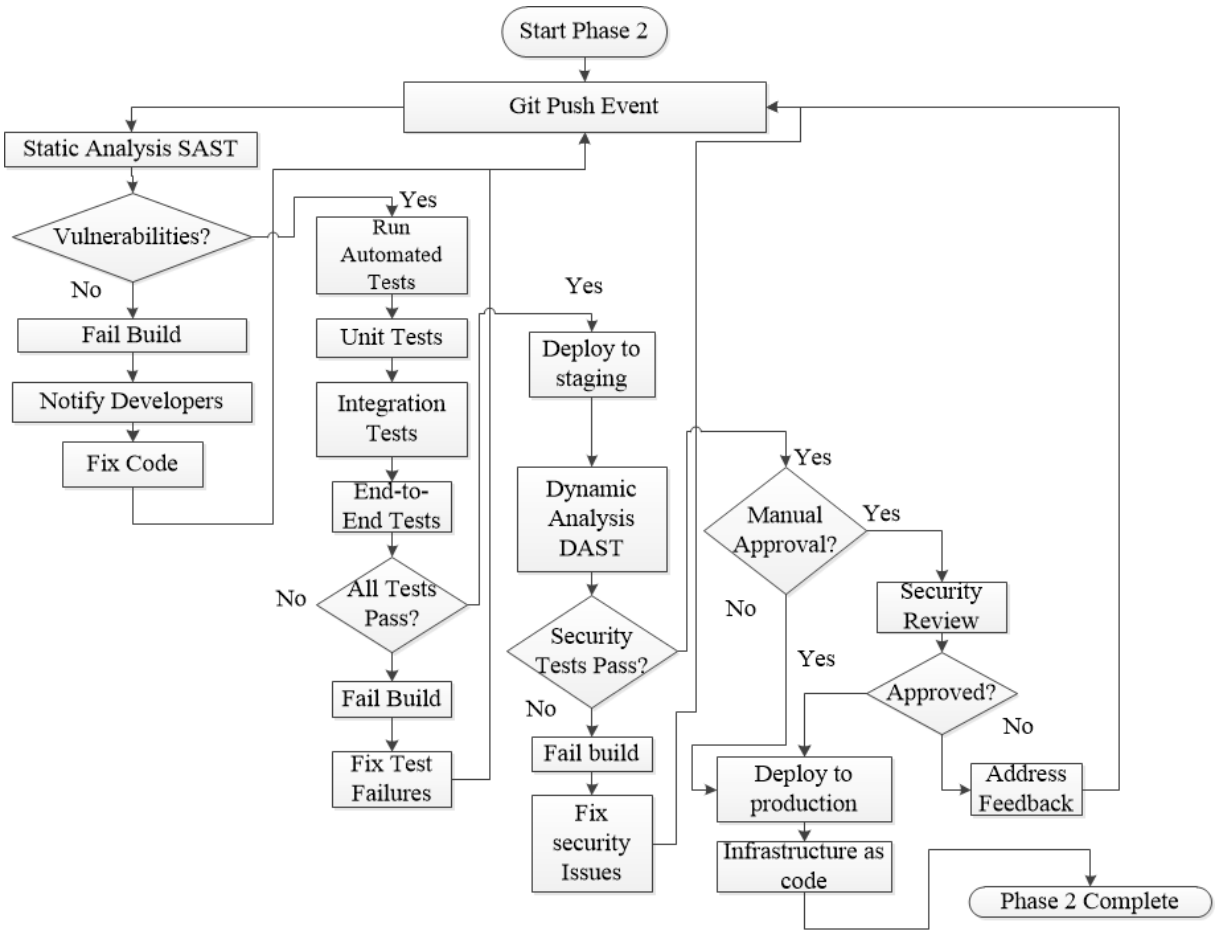

Figure 4 – Steps of phase 2 (Automated CI/CD Security Pipeline)
Рисунок 4 – Этапы фазы 2 (Автоматизированный пайплайн безопасности CI/CD)

Validation and Penetration Testing: It conducts systematic security assessments to identify and prioritize vulnerabilities, validates fixes through retesting, engages real users in acceptance testing, and formalizes response procedures through documented incident playbooks, ensuring the system is both technically secure and trusted by its users (defined in Figure 7).

Continuous Operation and Improvement: This phase establishes a cyclical, time-bound operational protocol to ensure the sustained security, availability, and evolution of the web application game.

The process has three main parts. Part one is real-time monitoring. Part two involves long-term strategic cycles. Part three is a formal decommissioning procedure. These parts are connected. The methodology is designed to fulfill the core operational security tenets of prevention, detection, and response, while ensuring adaptability to a changing threat landscape.

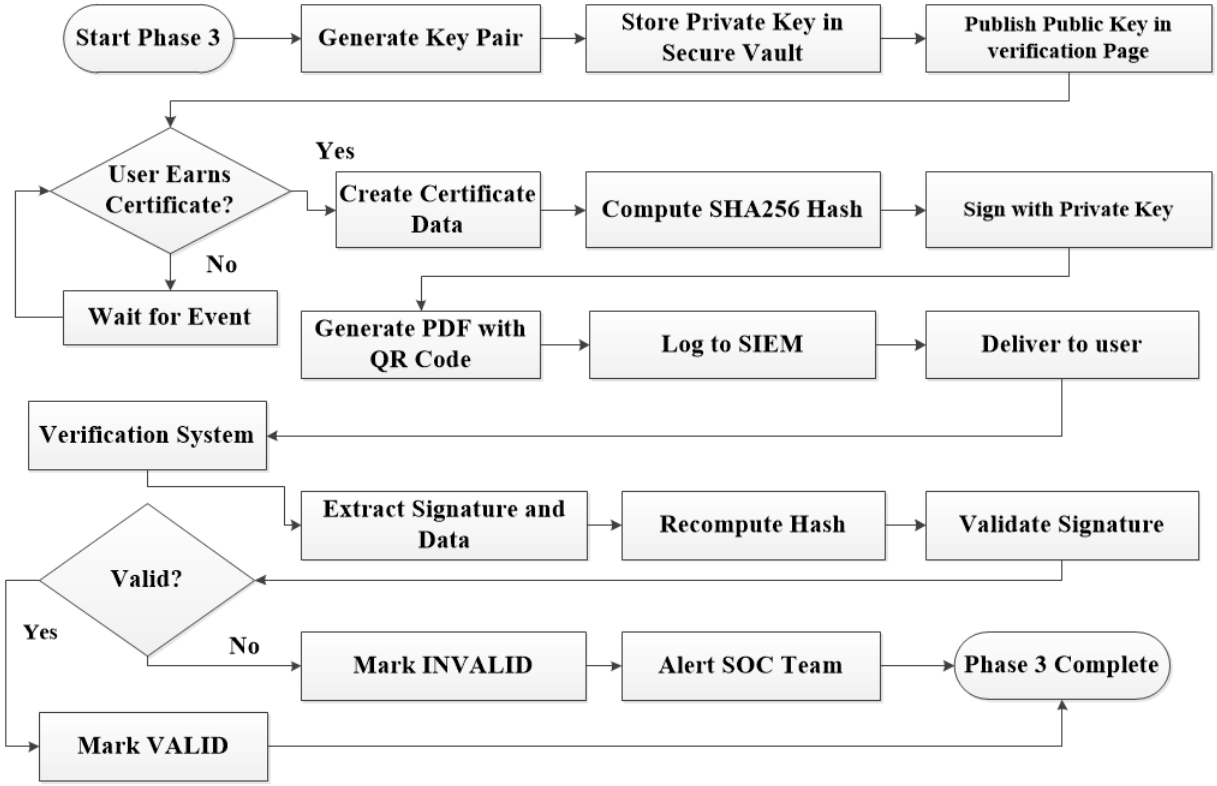

Figure 5 – Steps of cryptographic certificate system
Рисунок 5 – Этапы системы криптографических сертификатов

Results

The four basic areas had to be fully tied together in order to move from a proof-of-concept that had been vulnerable to a secure system of production. This section describes the actual implementation, which has been verified by system engineering labs and penetration testing.

In the initial implementation, a click event on an answer option would directly trigger the handleAnswer(isCorrect) function, which manipulated client-side state by updating the inventory object and persisting progress directly to localStorage-a design fundamentally vulnerable to user manipulation.

The revised implementation enforces a secure sequence of operations: the same user interaction now initiates a POST request to a dedicated endpoint, such as /api/validate-answer, carrying a structured payload including levelId, questionId, the selectedOption, and a secure sessionToken. The server subsequently validates the session, confirms the user's authorization to attempt the question, and verifies the answer against the correct value stored in a trusted database. Only upon receiving a 200 OK response does the client proceed to update the UI, add the reward to the inventory, and proceed with narrative feedback via showNextMessage(). This end-to-end, server-validated workflow ensures the integrity of the learning assessment and effectively prevents players from manipulating game results. As a result, we have implemented our proposed framework and collected the results for further progression. Below is the description of our implemented automated pipelines.

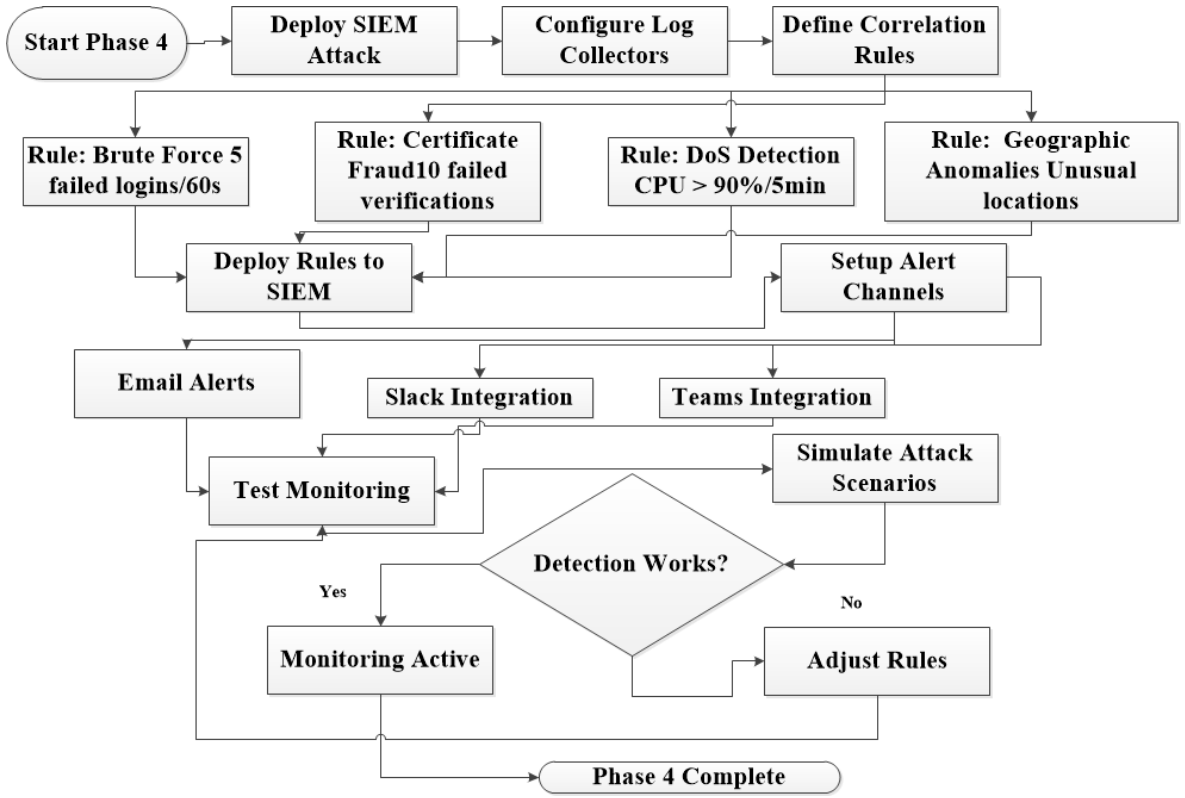

Figure 6 – Steps to complete phase the security monitoring and SIEM deployment
Рисунок 6 – Этапы завершения этапа мониторинга безопасности и развертывания SIEM-системы

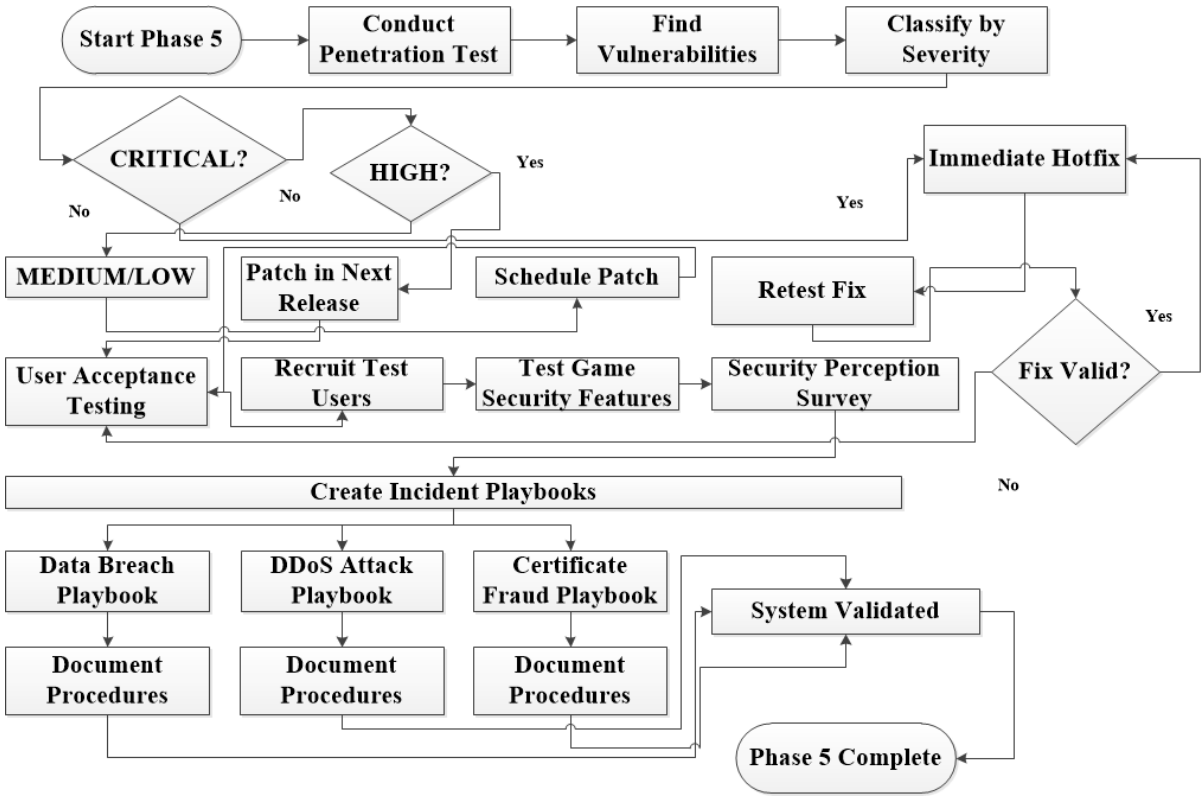

Figure 7 – Steps for validation and penetration testing
Рисунок 7 – Этапы валидации и тестирования на проникновение

The implementation of the CI/CD security pipeline and proactive threat modeling led to a dramatic reduction in vulnerabilities. As shown in Table 2.

Critical vulnerabilities (shown in Figure 8) were eliminated, and high-severity issues were reduced by 87.5 % by Month 3.

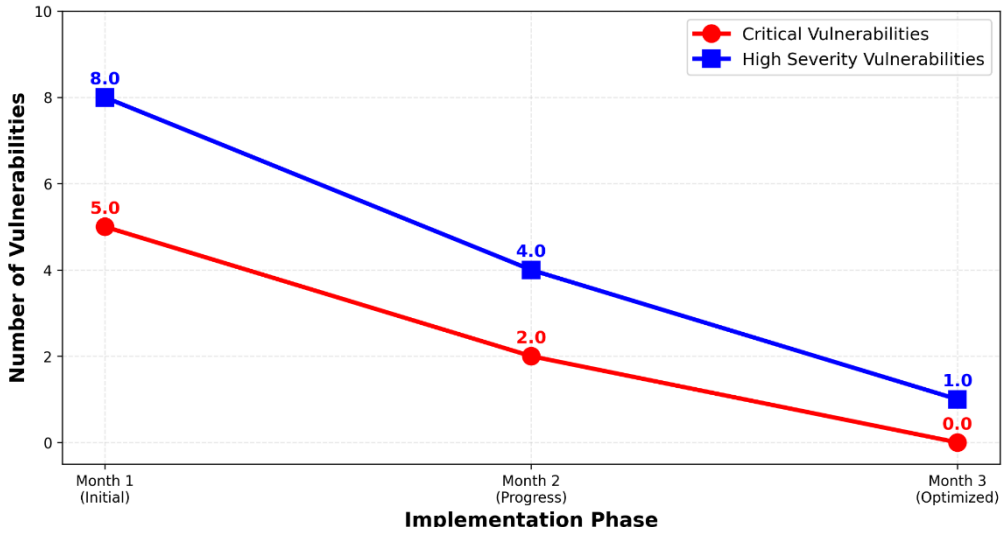

Figure 8 – Vulnerability reduction trend over three-month implementation
Рисунок 8 – Тенденция снижения уязвимости за три месяца внедрения

The Mean Time to Remediate (MTTR) improved by 65.4 %, demonstrating the efficiency of the "shift-left" security approach.

Table 2 – Vulnerability reduction over three-month implementation
Таблица 2 – Снижение уязвимости за три месяца внедрения

Metric	Month 1	Month 2	Month 3	Trend
Critical Vulnerabilities	5	2	0	↓ 100 %
High Severity Vulnerabilities	8	4	1	↓ 87.5 %
SAST Detection Rate	82 %	88 %	92 %	↑ 12.2 %
Mean Time to Remediate	5.2 days	3.1 days	1.8 days	↓ 65.4 %

Operational Efficiency Gains: Automation through the CI/CD pipeline yielded significant efficiency improvements (Table 3).

Table 3 – CI/CD pipeline efficiency metrics
Таблица 3 – Показатели эффективности CI/CD пайплайна

Metric	Baseline (Pre-Impl.)	Month 3	Improvement
Deployment Frequency	Weekly	Daily	↑ 300 %
Lead Time for Changes	3.5 days	1.3 days	↓ 62.9 %
Build Success Rate	84 %	98.2 %	↑ 16.9 %
Security Scan Automation	Manual	100 %	Full Automation

Efficacy of Security Monitoring (SIEM): The integrated SIEM and monitoring stack proved highly effective. The Mean Time to Detect (MTTD) incidents improved by 64.4 %, and the False Positive Rate was reduced by 64.7 % through refined correlation rules (Table 4). Simulated attacks, including brute-force and injection attempts, were detected with a 97.2 % success rate.

Table 4 – Security monitoring and incident response performance
Таблица 4 – Показатели эффективности мониторинга безопасности и реагирования на инциденты

Metric	Month 1	Month 3	Improvement
Mean Time to Detect (MTTD)	8.7 min	3.1 min	↓ 64.4 %
False Positive Rate	34 %	12 %	↓ 64.7 %
Incident Detection Rate	91.3 %	97.2 %	↑ 6.5 %

User Experience and Performance Impact. Despite the added security controls, user experience metrics improved or remained stable (Table 5). The System Usability Scale (SUS) score increased from 81 to 84, and the perceived security rating (on a 5-point scale) increased by 25 %. Performance overhead was minimal, with API response latency increasing by less than 9 % under load – a worthwhile trade-off for the security benefits.

Table 5 – User experience and performance impact
Таблица 5 – Пользовательский опыт и влияние на производительность

Metric	Pre-Implementation	Month 3	Change
System Usability (SUS)	81/100	84/100	↑ 3.7 %
Perceived Security (1–5)	3.2	4.0	↑ 25 %
API Latency Increase	Baseline	+8.7 %	Acceptable

Validation of the Cryptographic Certificate System. The secure certificate system met all reliability targets. It achieved a 99.6 % issuance success rate and a 100 % forgery detection rate. Verification via the public portal took an average of 0.23 seconds, proving the feasibility of a cryptographically secure, user-friendly credential system.

Discussion

The three-month results confirm the central thesis: a systematic integration of security engineering is essential for transitioning a pedagogically successful prototype into a resilient service. The framework successfully reduced critical risks while improving development agility and user trust.

However, these security gains came with trade-offs. The security remediation process faced significant challenges. One of the biggest challenges was changing the codebase from being built around trusting the client side to a server-centric approach. This required a lot of major rewrites and took a lot of effort.

The transition to a server-authoritative model established the backend API as a critical path dependency, introducing a single point of failure; any service degradation or outage now directly cascades to client functionality, trading security vulnerabilities for availability risks.

Development complexity increased substantially through the introduction of asynchronous I/O-bound operations for previously synchronous game logic. Each user

interaction now requires structured HTTP requests. Also, server-side validation and state synchronization impose non-trivial latency and complicate client-side error handling and recovery flows.

The developer workflow was adjusted to focus more on pre-validating data and ensuring secure data transmission. This meant following strict development protocols and carefully adhering to API agreements, rather than rushing to deliver quick client-side features.

Based on current trends, six-month projections suggest a move toward an enterprise-grade security posture, including maintaining zero critical vulnerabilities through the continued use of advanced SAST and DAST tools.

– MTTD [14] of under 2 minutes and MTTR under 15 minutes via enhanced SIEM automation.

– Scalability to support over 3,000 active users and 30 daily certificate issuances.

– Increased Adoption, with projected use by 8+ educational institutions.

Limitation: The current validation is based on simulated attacks and controlled environments. Future work includes launching a controlled bug bounty program and conducting stress testing at a larger scale with real-world user traffic.

Conclusion

This research presents a framework for building security and systems engineering principles directly into educational game development. Using a multilingual sign language learning game as our case study, we show that integrating security from the start is not only technically possible but essential for turning an educational game into a real-world product.

The main contribution of this work is bringing together four areas that don't always work simultaneously.

Our lab-tested approach shows that when these pieces connect (web application security, automated CI/CD pipelines, secure digital document workflows, and centralized monitoring through SIEM), they create a much stronger security system. Threat modeling helps configure automated security checks in the development pipeline, and those results then improve what the SIEM system looks for. The results from three months of implementation back our proposed framework. We saw critical vulnerabilities drop by 100 % and high-severity ones by 87.5 %, while the time to fix issues improved by 65.4 %. And we achieved this without a lot of changes in user experience or system performance-usability scores actually went up by 3.7 %, and performance overhead stayed under 9 %.

Our years-long experience shows that for educational technology serving vulnerable groups-like the hearing-impaired community-building trust through verifiable security is essential for long-term adoption. The cryptographic certificate system (99.6 % issuance success, 100 % forgery detection, verification in under a second) offers a model for issuing trustworthy credentials in educational settings.

In short, it is possible to state that this paper provides a practical, repeatable blueprint for securing software that makes a social impact. For further progression, we plan to study how this framework holds up at larger scales and how it might apply to other areas of inclusive educational technology.

REFERENCES / СПИСОК ИСТОЧНИКОВ

1. Ashrafi A., Philippovich Y.N., Mokhnachev V., et al. Development of the Multilingual Sign Language Learning Game for Interactive Education. *International Journal of Open Information Technologies.* 2026;14(2):128–133.
2. Ашрафи А., Мохначев В.С., Махмуд А.А. и др. Разработка политики информационной безопасности для образовательной игры. В сборнике: *III*

Международный конгресс «Русский инженер». Искусственный интеллект в автоматизированных системах управления и обработки данных (ИИАСУ'25): Сборник статей IV Всероссийской научной конференции, 29–30 октября 2025 года, Москва, Россия. Москва: Издательство МГТУ им. Н.Э. Баумана; 2026. С. 483–491.
Ashrafi A., Mokhnachev V.S., Makhmud A.A., et al. Developing an information security policy for an educational game. In: *The III International Congress "Russian Engineer". Artificial Intelligence in Automated Control and Data Processing Systems (AIACS'25): Collection of Articles of the IV All-Russian Scientific Conference, 29–30 October 2025, Moscow, Russia*. Moscow: Publishing House of Bauman Moscow State Technical University; 2026. P. 483–491. (In Russ.).

3. Ашрафи А., Мохначев В.С. Информационно-теоретическая метрика для автоматического лексикографического отбора в бенгальском жестовом языке. *Моделирование, оптимизация и информационные технологии.* 2026;14(6). (На англ.). https://doi.org/10.26102/2310-6018/2026.57.6.011
Ashrafi A., Mokhnachev V.S. An information-theoretic metric for automated lexicographic selection in Bengali Sign Language. *Modeling, Optimization and Information Technology.* 2026;14(6). https://doi.org/10.26102/2310-6018/2026.57.6.011
4. Biswas J., Hasan M., Saiful M., et al. A review on mitigating security risks: Effective strategies to prevent cross-site request forgery vulnerabilities. In: *Cyber Intelligence and Information Retrieval, 14–15 December 2023, Kolkata, India*. Singapore: Springer; 2025. P. 309–317. https://doi.org/10.1007/978-981-97-7603-0_27
5. De Ryck Ph., Desmet L., Joosen W., et al. Automatic and precise client-side protection against CSRF attacks. In: *Computer Security – ESORICS 2011: 16th European Symposium on Research in Computer Security, 12–14 September 2011, Leuven, Belgium*. Berlin, Heidelberg: Springer; 2011. P. 100–116. https://doi.org/10.1007/978-3-642-23822-2_6
6. Simplice I., Fidel O., Kennedy Ch.G., et al. Enhancing information system security: A vulnerability assessment of a web application using OWASP top 10 list. In: *Proceedings of 3rd International Conference on Smart Computing and Cyber Security: Strategic Foresight, Security Challenges and Innovation (SMARTCYBER 2023), 05–06 December 2023, South Korea*. Singapore: Springer; 2024. P. 385–397. https://doi.org/10.1007/978-981-97-0573-3_31
7. Khazal I.F., Ghaib A.A., Shareef A., et al. Cross Site Scripting Attacks (XSS): A Review. In: *Software Engineering: Emerging Trends and Practices in System Development: Proceedings of 14th Computer Science On-line Conference 2025: Volume 7, 01–03 April 2025, Czech Republic*. Cham: Springer; 2025. P. 342–359. https://doi.org/10.1007/978-3-032-04581-2_24
8. Li K., Liu H., Zhang L., et al. Automatic inspection of static application security testing (SAST) reports via large language model reasoning. In: *Artificial Intelligence Logic and Applications: 4th International Conference, AILA 2024, 10–11 August 2024, Lanzhou, China*. Singapore: Springer; 2025. P. 128–142. https://doi.org/10.1007/978-981-96-0354-1_11
9. Hüther L., Sohr K., Berger B.J., et al. Machine Learning for SAST: A Lightweight and Adaptable Approach. In: *Computer Security – ESORICS 2023: 28th European Symposium on Research in Computer Security: Part IV, 25–29 September 2023, The Hague, The Netherlands*. Cham: Springer; 2024. P. 85–104. https://doi.org/10.1007/978-3-031-51482-1_5
10. Melis A., Giovine A., Rinieri L. Time-Sensitive Networking Digital Twin for STRIDE-based security testing. *EURASIP Journal on Information Security.* 2025;2025:26. https://doi.org/10.1186/s13635-025-00213-7

11. Shahrivar P., Millar S. Detecting Web Application DAST Attacks in Large-Scale Event Data. In: *Artificial Intelligence for Security: Enhancing Protection in a Changing World*. Cham: Springer; 2024. P. 325–343. https://doi.org/10.1007/978-3-031-57452-8_14
12. Machap K., Ang X.Y. Case study for implementation of host-based intrusion detection system in cyber security. In: *Evolution in Signal Processing and Telecommunication Networks: Proceedings of Ninth International Conference on Microelectronics Electromagnetics and Telecommunications (ICMEET 2024): Volume 2, 19–20 December 2024, India*. Singapore: Springer; 2026. P. 591–595. https://doi.org/10.1007/978-981-96-7241-7_47
13. Carruthers A., Ahmed S. Security Monitoring. In: *Maturing the Snowflake Data Cloud: A Templated Approach to Delivering and Governing Snowflake in Large Enterprises*. Berkeley: Apress; 2023. P. 105–144. https://doi.org/10.1007/978-1-4842-9340-9_3
14. Kumar U.D., Crocker J., Knezevic J., et al. Analysis of Reliability, Maintenance and Supportability Data. In: *Reliability, Maintenance and Logistic Support: A Life Cycle Approach*. New York: Springer; 2000. P. 423–471. https://doi.org/10.1007/978-1-4615-4655-9_12

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Арифа Ашрафи, ассистент, Московский политехнический университет, Москва, Российская Федерация.
e-mail: arifaa13@gmail.com
ORCID: 0000-0002-5753-6135

Arifa Ashrafi, Assistant, Moscow Polytechnic University, Moscow, the Russian Federation.

Филиппович Юрий Николаевич, кандидат технических наук, профессор кафедры «Инфокогнитивные технологии», Московский политехнический университет, Москва, Российская Федерация.
e-mail: y_philippovich@mail.ru
ORCID: 0000-0001-9419-2282

Yuriy N. Philippovich, Candidate of Engineering Sciences, Professor at the Department of Infocognitive Technologies, Moscow Polytechnical University, Moscow, the Russian Federation.

Мохначев Виктор Сергеевич, ассистент, Московский политехнический университет, Москва, Российская Федерация.
e-mail: gagashaggy@inbox.ru
ORCID: 0000-0001-6520-0386

Viktor S. Mokhnachev, Assistant, Moscow Polytechnic University, Moscow, the Russian Federation.

Махмуд Али Айманович, студент, Московский политехнический университет, Москва, Российская Федерация.
e-mail: makhmud_ali22@mail.ru
ORCID: 0009-0008-1402-8617

Ali A. Makhmud, Student, Moscow Polytechnic University, Moscow, the Russian Federation.

Статья поступила в редакцию 09.06.2026; одобрена после рецензирования 31.07.2026; принята к публикации 19.08.2026.

The article was submitted 09.06.2026; approved after reviewing 31.07.2026; accepted for publication 19.08.2026.