# AI-Driven Microlearning for Proactive DevSecOps: Closing the Security Skill Gap

Giampiero Granatella[1,2,*], Maura Cerioli[1], Giovanni Lagorio[1] and Angelo Lupo[2]

[1]DIBRIS - Università degli studi di Genova, Viale Causa, 13 - 16145 Genova, Italia
[2]ManyDesigns S.r.L. – Palazzo Lercari Parodi – Via G. Garibaldi 3 – 16124 Genova, Italia

**Abstract**

DevSecOps, defined as the integration of security practices into the entire software development lifecycle (SDLC), ensures a focus on both *security* and *reliability* from inception to deployment. Although numerous tools support the diverse stages of DevSecOps, managing the vast array of generated data, particularly regarding organisational capabilities and expertise, remains a challenge.

In this research, we introduce a novel *Competency Management Support Tool (CMST)* designed to analyse software development metrics and artefacts, thus providing deep insights into existing team skill sets. Using the data analysed by the CMST, we subsequently developed and defined a specialised educational component that *uses Artificial Intelligence (AI)* and *Microlearning* principles.

This integrated tool chain effectively identifies skill gaps and security vulnerabilities within a development team by meticulously analysing historical and ongoing development data. Upon identifying these deficiencies, the system automatically designs and prepares customised microlearning sessions. The primary goal of this AI-driven approach is to enhance security awareness and technical expertise proactively, thereby preventing the recurrence of security-related issues in future software iterations and fundamentally strengthening the organisation's DevSecOps maturity.

**Keywords**

human resources, artificial intelligence, micro learning, secdevops

## 1. Introduction

The landscape of modern software engineering is characterised by unprecedented speed and continuous delivery, driven by agile methodologies and the adoption of DevOps practices. Although this velocity is crucial for market competitiveness, it simultaneously introduces inherent risks by accelerating the pace at which vulnerabilities can be introduced and deployed. Consequently, the traditional security model, characterised by end-of-cycle audits and security gatekeepers, has become an insurmountable bottleneck.

To address this challenge, *DevSecOps* has emerged not merely as a set of tools, but as a critical practice and a fundamental cultural transformation. DevSecOps mandates the integration of security testing and awareness into every single phase of the Software Development Lifecycle (SDLC). It promotes a philosophy of *shared responsibility*, ensuring that security becomes the joint concern of developers, security specialists, and operations teams, thereby proactively creating efficient and secure software.

The SDLC itself remains the structured backbone for producing high-quality applications. This life cycle guides teams through the essential stages: requirement Analysis, Planning, Architectural Design, Software Development, Testing, and Deployment. Supporting this process is a sophisticated ecosystem of tooling, including Version Control Systems (VCS) like Git, continuous integration and continuous deployment (CI/CD) platforms such as GitLab and Azure DevOps, and Static Application Security Testing (SAST) solutions like SonarQube.

*Joint National Conference on Cybersecurity (ITASEC SERICS 2026), February 09-13, 2026, Cagliari, IT*
*Corresponding author.
giampiero.granatella@edu.unige.it (G. Granatella); maura.cerioli@unige.it (M. Cerioli); giovanni.lagorio@unige.it (G. Lagorio); angelo.lupo@manydesigns.com (A. Lupo)
0009-0001-9785-1391 (G. Granatella); 0000-0002-8781-8782 (M. Cerioli); 0000-0002-6632-1523 (G. Lagorio); 0009-0000-6033-6997 (A. Lupo)

Crucially, the artefacts and logs generated by these tools—ranging from code commit frequency and project management metrics to vulnerability reports—implicitly track the *skills and competencies* that individual developers and teams acquire during their work. This wealth of information represents an untapped resource for organisational learning.

However, a significant gap persists: the DevSecOps tooling excels at identifying *symptoms* (specific security flaws), but fails to connect these symptoms directly to the underlying *skill deficit* of the engineer or team responsible. This inability to translate raw data into actionable educational insights leads to the recurrence of common security errors, diminishing the overall resilience promised by DevSecOps.

This paper addresses this critical oversight by introducing an integrated system designed to close the competency loop. Our approach begins with the development of a *Knowledge Graph* that ingests and processes data primarily from static code analysers and versioning systems. This graph provides a semantic layer of understanding, revealing patterns such as the most common security issues encountered by a team, the specific projects and contexts (domains, programming languages) in which these issues occurred, and their frequency.

Leveraging this comprehensive knowledge base, we then harness *Artificial Intelligence (AI)* to design and deliver targeted educational interventions through *Microlearning*. By providing AI with reliable data sources, we can automatically curate small, concise, and focused web-based training sessions. This approach maximises knowledge retention and directly enhances the specific capabilities of the development team, ensuring that skill development is perfectly aligned with observed deficiencies in security practices in the real-world.

The remainder of this paper is structured as follows: Section 2 reviews related work in AI-driven competency management and security training within the DevSecOps context. Section 3 details the architecture of our Competency Management Support Tool (CMST) and the construction of the underlying Knowledge Graph and presents the design and implementation of the AI-driven Microlearning module. Finally, Section 4 concludes and details some further developments.

## 2. Related Work

The demands of modern software consumers are rapidly accelerating, constantly pushing IT organisations towards significantly shorter development and release cycles to meet constantly evolving requirements [1]. However, this relentless pressure for speed must be rigorously balanced against crucial non-functional requirements, most notably *security* and *regulatory compliance* [2]. To navigate this dynamic tension, organisations have widely adopted widespread automation and increased deployment frequencies as key strategic solutions [3]. These fundamental changes, significantly facilitated by modern cloud computing paradigms, naturally culminated in the widespread institutionalisation of *DevOps* [4]. Formally defined as an organisational approach, DevOps fundamentally emphasises *empathy and cross-functional collaboration* between development, security, and operations teams to accelerate the delivery of changes and ensure the operation of resilient systems [5, 6]. The systematic adoption of DevOps yields tangible and documented benefits, including faster time-to-market, improved product quality, and demonstrably increased organisational effectiveness. This success is quantitatively reflected in recent industry reports, indicating that approximately 36% of the development teams daily release in production and 46% once a week [7]. The term *DevSecOps* denotes the deliberate extension of the DevOps paradigm to formally incorporate security principles (Sec) [8], marking a significant movement toward embedding security practices directly into the automated delivery pipeline [9].

To spot possible skill gaps, we use *static analysis* to explore code files in search of possible security weaknesses. Static analysis is the process of automatically analysing source code using methods such as rule checking, lexical analysis, and pattern matching, [10] empirically investigates the adoption and efficacy of Static Code Analysis tools within Open Source projects' Continuous Integration processes. For static analysis to be effective and attractive to developers, it must resolve key limitations. Specifically, tools must consolidate high-volume output from various scanners into a single comprehensive view; natively support scanning across multiple repositories; offer standardised integration into existing

third-party dashboards; and provide multilingual support, as most modern applications are polyglot [11].

Building on the necessity of integrating security practices early and acknowledging the limitations of current tooling highlighted by developer fatigue, the core idea of this research is to use microlearning—defined as the delivery of concise, focused educational content in short bursts—to address demonstrated security skill gaps. Defined as just-in-time learning, microlearning is a teaching technique that breaks subjects into small, manageable pieces, using both traditional and digital methods. Its key characteristics—including being accessible anytime, anywhere, and incorporating cost-effectiveness and gamification—make personalised on-demand education possible [12]. The validity and benefits of microlearning for the enhancement of professional competence have been empirically validated in numerous research studies [13, 14].

Our approach synthesises specific security issues derived from Static Application Security Testing (SAST) tools, transforms this raw output into targeted learning objectives, and presents them to development teams. Using artificial intelligence (AI) to personalise and optimise these learning sessions, this methodology effectively closes the competency loop within the Software Development Lifecycle (SDLC), proactively improving the security expertise of both the team and individual developers. This type of approach is common in many research efforts that combine the power of LLMs with microlearning according to a competency model [15, 16, 17, 18].

However, the convergence of content generation driven by cybersecurity, microlearning, and Large Language models (LLM) remains a significantly understudied area [19]. Specifically, the integration of security awareness training with microlearning methodologies and the subsequent use of advanced generative models—such as LLM architecture—for the automated and contextualised creation of instructional materials represents an unexplored research gap. Based on our comprehensive review of the existing literature, this work is positioned as one of the first to address this critical intersection formally.

## 3. Building a dynamic and Adaptive micro learning module for Personalized Skill Gap Remediation

Our research is fundamentally based on a robust *Knowledge Base* that serves as the core of a comprehensive *Competency Management Support Tool (CMST)*. This proprietary system is delivered to users as a Web Application, providing a centralised platform to manage, visualise, and analyse the organisation's entire competency profile, treating corporate expertise as the sum of individual and team skill sets. The CMST is meticulously designed to provide actionable intelligence throughout the corporate community, catering to the needs of diverse stakeholders, including *Human Resources (HR) personnel*, *senior management*, *project managers*, and *individual developers*.

The CMST's functionality extends beyond mere reporting. It supports the full lifecycle of competency data management: it allows for the submission and management of *Curriculum Vitae (CVs)*, from which it automatically extracts professional experiences and technical competencies. Furthermore, it integrates features to analyse *software projects* and their *artifacts* and facilitates the precise definition of *project teams*. This capability allows the CMST to move beyond static assessments, offering a dynamic and verifiably grounded model of the organisation's collective and individual skill set.

Crucially, the application includes an advanced *textual interface* that allows users to query the underlying Knowledge Base. This interface enforces strict visibility constraints, ensuring that stakeholders only access information relevant to their roles and designated permissions, such as the aggregated competencies of their assigned teams or departments.

Although CMST offers broad functionalities for enterprise-wide skill management, the primary objective of this document is to detail a specific integrated module: the component dedicated to analysing *development team competency gaps*. Based on this diagnostic information, the module proceeds to automatically generate *microlearning content* that is specifically tailored to remediate these identified deficiencies. This targeted approach represents our core contribution to enhancing developer skill sets within the modern DevSecOps environment.

### 3.1. Information Extraction to build a knowledge base

The various elements that make up our knowledge graph are mainly sourced from various systems within the SecDevOps ecosystem. Initially, we started by analysing the *GitHub repositories* of the company's projects, which are a rich source of information. Indeed, they provide data on the programming languages used within each project, the teams involved, and the libraries and frameworks utilised. This process gradually builds a network of heterogeneous nodes and relationship types, creating paths that connect individuals to specific competencies such as programming languages, libraries, or frameworks. For example, being part of a team that works on a project that uses a specific language guarantees at least basic knowledge of it [20].

However, a model that only includes such coarse-grained data is insufficient to discriminate members proficient in the language from those with only a passing acquaintance. Using a more sophisticated analysis of *individual commits*, we were able to refine the model by weighting individual contributions to projects. For example, we can understand when team members are major contributors to files in a given language and attribute them a higher degree of such competence.

Although extremely useful and rich, the knowledge we can extract from repositories alone is limited. For example, consider the concept of *knowledge gaps*. From the analysis of a project repository, we can understand the competencies people show through their contributions to the project, but do not get a clue about their weak points. To add this dimension, we include the output of *static code analysis systems*, which provides insights into potential project vulnerabilities and security guideline violations. This creates paths related to security issues and problems.

### 3.2. Targeted Remediation for Security Gaps

Using the fine-grained detailed information contained within the knowledge graph, the focus of this research shifts specifically to utilising the data for *proactive competence improvement*, with a particular emphasis on addressing deficiencies related to *software security*. The structure of the graph allows for a diagnostic analysis that determines

- The *security errors most frequently made* are classified and mapped against standardised classifications such as *OWASP*, *CWE*, and other relevant frameworks.
- The specific *projects and code sections* where these vulnerabilities were detected.

By contextualising the identified errors, the system can further deduce critical project-level factors, including the *programming language* used and the specific *development team* responsible for the codebase.

### 3.3. Agile-Inspired, Team-Centric Gap Attribution

A critical methodological decision involves the attribution of identified competency deficiencies. In adherence to the principles of Agile development, which hold the development team as the collective unit responsible for the task's quality and conclusion, we have determined that *competency deficiencies are grouped and attributed to the entire team* involved in the code development. When a vulnerability is flagged by static analysis, it signifies a systemic failure in the team's collective development and review process, making the knowledge gap a shared responsibility. This team-level approach fosters a culture of collaborative learning and shared accountability, moving beyond individual blame.

These highly contextualised and accurate deficiency data—linked to real-world code, language, and teams, are the cornerstone of our custom microlearning solution. This allows for the generation of a *personalized microlearning module* that *features practical examples directly extracted from and related to the individual's or team's deficiencies*. This unprecedented level of personalization ensures maximum relevance, driving engagement, and significantly enhancing the effectiveness of the remediation process for identified security and skill gaps.

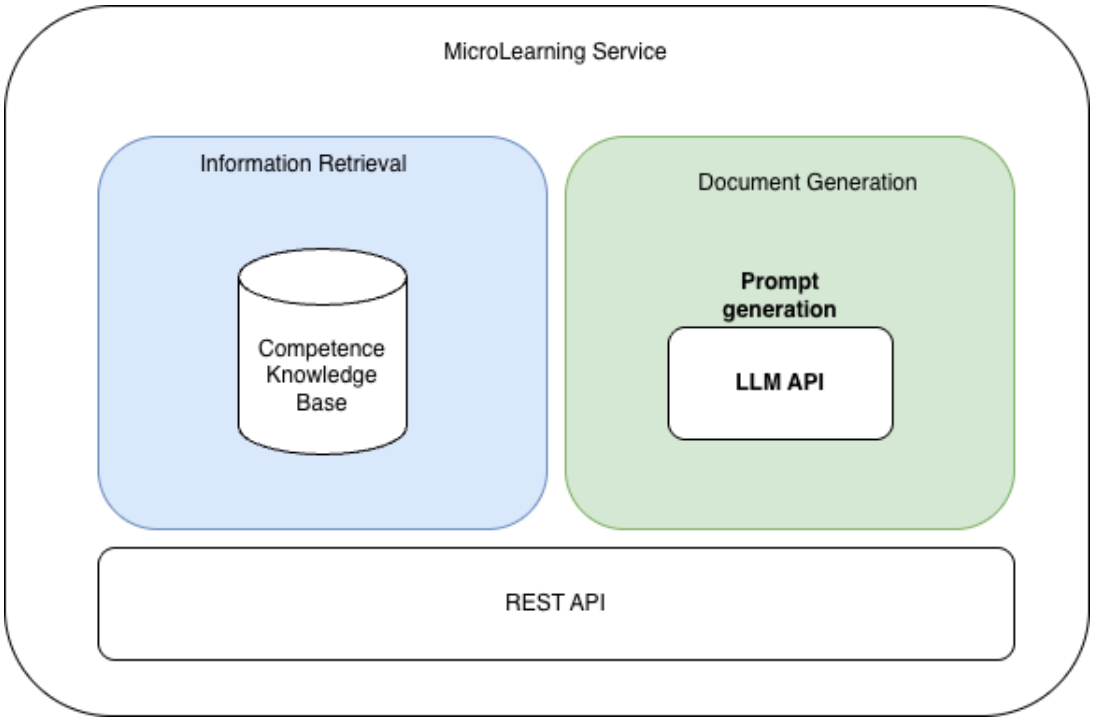


**Figure 1:** General architecture of Microlearing module

### 3.4. Automated Microlearning Generation

The final stage of our methodology involves the synthesis of the educational material, a process realised through a specialised *Retrieval-Augmented Generation (RAG)* architecture. RAG serves a critical function in enterprise applications, addressing the primary limitation that Large Language Models (LLMs), typically trained on public data, inherently lack access to the private, proprietary, and highly specialised information essential for organisational training. By connecting the LLM to our custom knowledge source in real-time, RAG ensures that the generated output is accurate and contextually relevant to the organisation's unique requirements.

In our system, the RAG architecture is uniquely anchored to the *Competency Knowledge Base*. This Knowledge Base, primarily a knowledge graph, functions as the proprietary external source, providing highly structured and diagnostic information that guides the generation process.

This tailored RAG approach is vital as it elevates the LLM from a general-purpose text generator to a personalised instructional designer, ensuring that the generated training directly addresses the specific competency gaps identified within the DevSecOps pipeline.

The proposed *MicroLearning Service* architecture, as conceptualised in 1, is fundamentally divided into two core functional components, all exposed and managed via a unified `REST API`. The first component, *Information Retrieval*, serves as the data backbone, leveraging a central *Competence Knowledge Base* to retrieve relevant domain-specific information related to user competency profiles and defined learning objectives. This structured data is then piped to the second component, *Document Generation*. Within this module, the *Prompt generation* layer plays a critical role by dynamically structuring the retrieved information into a suitable, context-rich input for the underlying LLM API (Large Language Model Application Programming Interface). The **Claude 3.5 Sonnet LLM** processes this input to produce high-quality, targeted microlearning content, such as quick guides or interactive summaries. This modular approach ensures both the scalability and the adaptability required to provide personalised security awareness training at the point of need.

Our methodology leverages *Microlearning*, a highly effective training paradigm that delivers educational content in small, focused, and easily digestible chunks. This approach is optimally suited for busy professionals, as it seamlessly integrates into constrained schedules, focused intently on a single

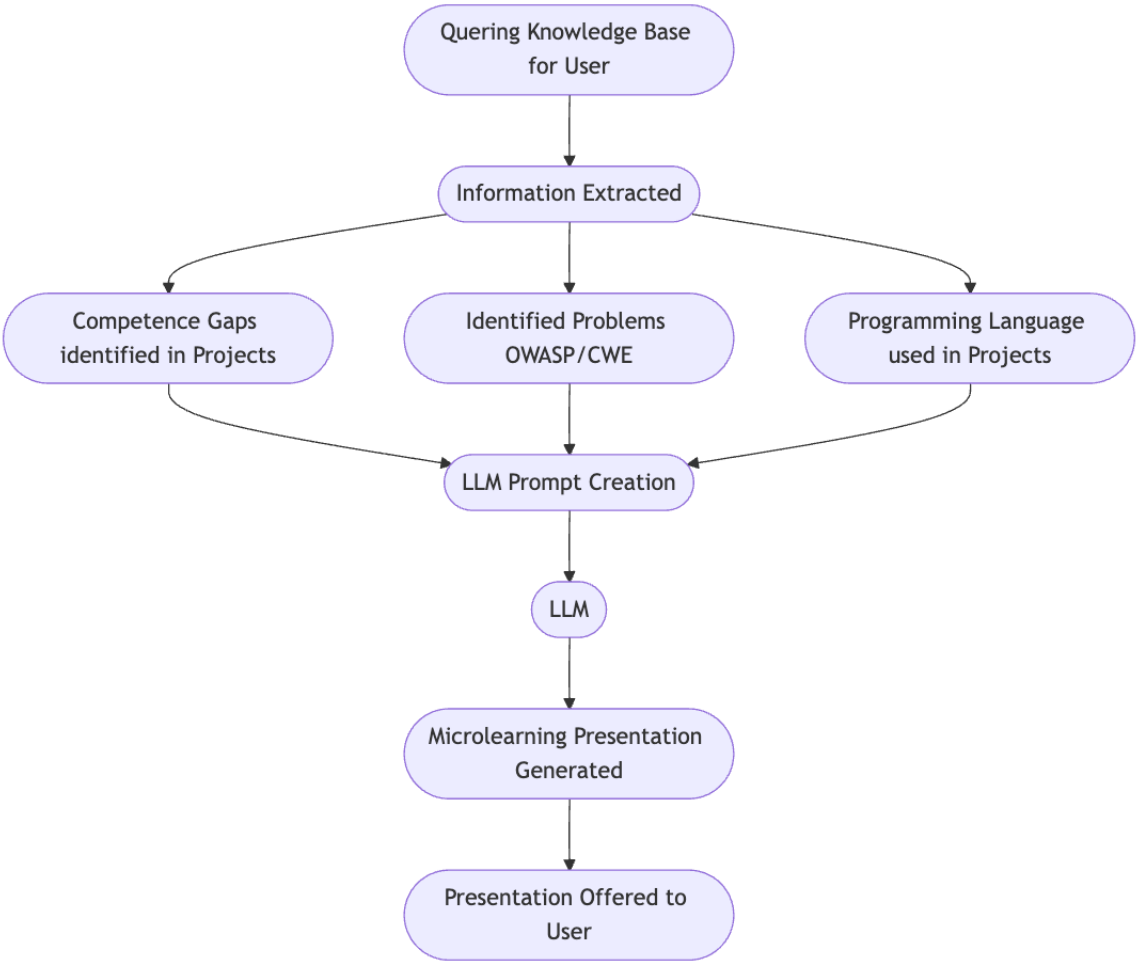


**Figure 2:** Microlearning module generator

learning objective per session.

Microlearning significantly improves knowledge retention and promotes continuous skill development, directly countering the information overload often associated with traditional training programmes. Our system capitalises on this methodology, using artificial intelligence to automatically generate personalised microlearning presentations. The content generation process is strictly anchored to security issues extracted from our software quality analysis pipeline, focusing on two industry-standard vulnerability classifications.

Using these established standards, we ensure that the generated training is not only personalised but also centred on the most critical and widely recognised security flaws. This direct link between the code committed by a developer and a specific known vulnerability provides the essential foundation for our targeted training.

These initial steps, visually represented in 2, begin with the Knowledge Base extracting structured information about the identified security problem, the programming language involved, and the under-

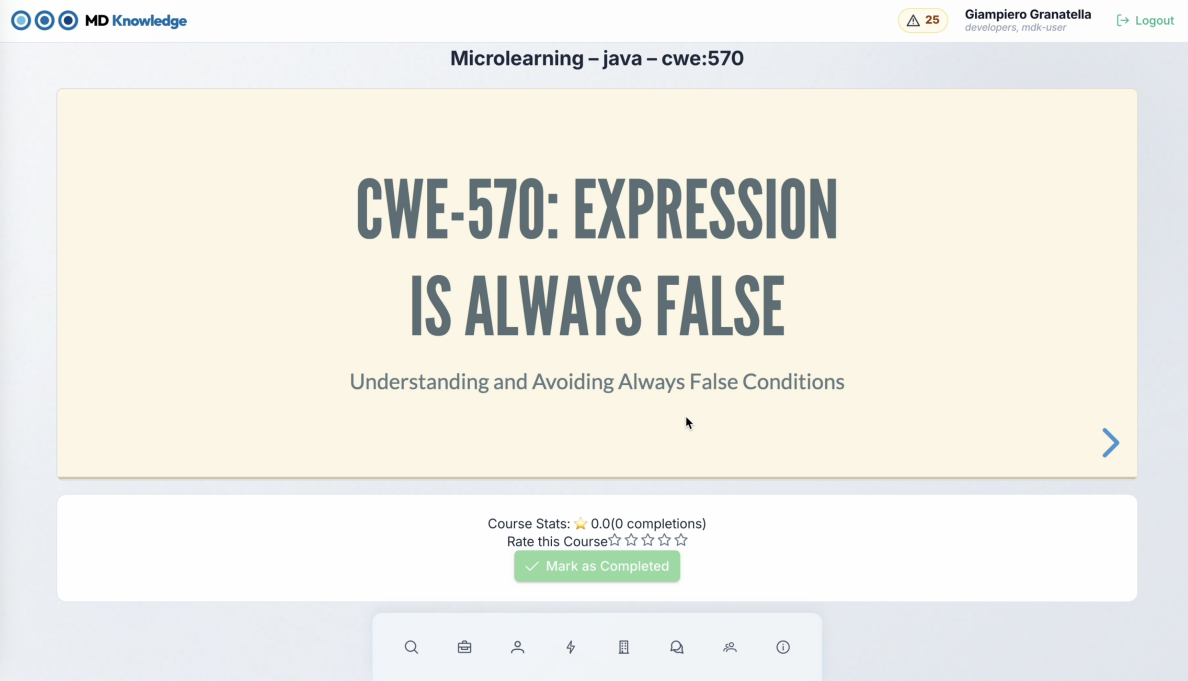


**Figure 3:** Microlearning course

lying competency gaps that are the root cause of the flaw.

With these precise contextualised data, we initiate a process designed to create a customised educational experience, as depicted in the lower part of 2. A *Large Language Model (LLM)* is queried using a highly specific prompt. This prompt includes the identified CWE or OWASP vulnerability ID, the exact programming language used in the problematic code, and an explicit request to generate a short, focused training presentation. Crucially, despite the prompt's domain-specific nature, no sensitive corporate data is transmitted to the LLM, ensuring full compliance with security and privacy standards.

The LLM is specifically tasked with producing content that clearly explains the vulnerability, provides concrete code examples demonstrating the error in the required programming language, and offers clear, actionable steps on how to resolve the issue.

The final output, shown in 3 and 4, is designed to be concise and immediately effective, typically structured as a set of approximately ten slides that cover the definition of the problem, the optimal solution, and relevant best practices. This automated generation process fundamentally shifts the training paradigm from generic, organisational-wide curricula to a dynamic, personalised learning path.

This methodology not only conserves valuable time and resources but also ensures that skill development is highly efficient, directly addressing specific competency gaps and significantly enhancing overall code quality.

### 3.5. Retrieval-Augmented Generation (RAG): Enhancing LLM Utility and Security

The adoption of *Retrieval-Augmented Generation (RAG)* provides significant benefits, primarily by mitigating well-known issues associated with large language models (LLMs) and enhancing *data security* and *content reliability*. By grounding the LLM's content generation on a proprietary knowledge base -such as web pages and internal documentation detailing a catalogue of known vulnerabilities and errors effectively overcomes a major challenge: *hallucinations*. This process forces the model to generate content based on verified domain-specific information, substantially improving the *accuracy* and *trustworthiness* of the output.

In terms of *privacy and security*, a key advantage is that all sensitive information pertaining to individuals and projects *remains internal* to the company. This private data is *never passed* to the external LLM, ensuring that it is not disseminated and strictly adheres to internal data governance

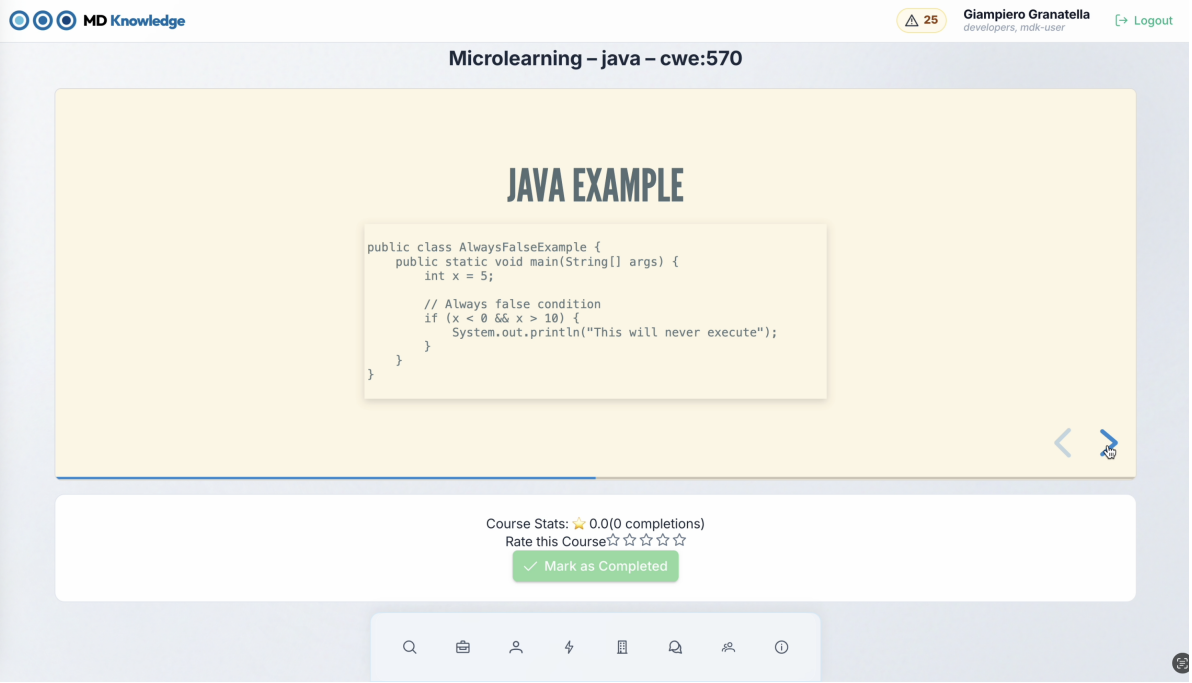


**Figure 4:** Microlearning course - 2

policies. Furthermore, the knowledge base enables the *replicability* of the generation process, for instance, the logical steps and rationale behind creating a training course are *deducible and traceable* from the established knowledge base. A final, crucial benefit is RAG's ability to *synthesize* and *close the loop on security* opened by static code analysers, providing contextual explanations and resolutions directly linked to the identified security issues.

## 4. Conclusion and further work

In this research, we have presented a practical and novel approach that effectively links the diagnostic information reported by static code analysers to personalised microlearning courses. This system is designed to significantly enhance developers' skills and competencies, specifically targeting recurring security flaws and competency gaps. Within the context of the DevSecOps lifecycle, our methodology successfully *closes the loop* of the code review process. By facilitating the rapid acquisition of knowledge directly relevant to identified mistakes, the system aims to prevent developers from repeating common errors, thereby establishing a continuous process of skill refinement and quality improvement.

### 4.1. Application and Evaluation

The research presented here has been practically implemented and tested within a leading Italian IT company. The size and complexity of the company provided a substantial testing ground, comprising *94 employees*, *428 distinct software projects*, and involving *38 programming languages*. This extensive environment allowed for a comprehensive evaluation of the system's operational viability. To formally evaluate the effectiveness of the entire Competency Management Support Tool (CMST) and its acceptance within the organisation, we employed the widely recognised *Technology Acceptance Model (TAM)*. TAM, a foundational theoretical framework in Information Systems research developed by Fred Davis in 1989, provides a robust method for explaining and predicting user adoption of new technologies. Our evaluation included specific questions targeting the microlearning module. The results demonstrated a high level of acceptance: *85% of the participants surveyed* found the microlearning module useful and positively impactful on their professional competencies. Overall, tool usage was highly rated, achieving

an average score of *4.8 on a 5-point Likert scale*, indicating a strong positive user perception.

### 4.2. Future developments

Beyond the initial exploratory survey on software adoption, we have defined key metrics that will be analysed in the long term. These long-term analyses are primarily focused on assessing the ultimate effectiveness of the intervention: specifically, whether participation in targeted microlearning modules on particular security themes leads to a sustained *reduction in errors* committed by both the individual developer and the development team regarding those specific themes. Data related to the consumption of the generated courses and the subsequent anomalies reported by static code analysers will be systematically cross-referenced. This rigorous data intersection will allow us to formally evaluate the long-term impact and overall efficacy of the proposed solution in driving demonstrable improvements in code quality and team competency.

## Declaration on Generative AI

During the preparation of this work, the author(s) used Gemini in order to: Grammar and spelling check. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

## References


[1] J. Humble, D. Farley, Continuous delivery: reliable software releases through build, test, and deployment automation, Pearson Education, 2010.

[2] B. Fitzgerald, K.-J. Stol, Continuous software engineering and beyond: trends and challenges, in: Proceedings of the 1st International Workshop on rapid continuous software engineering, 2014, pp. 1–9.

[3] J. Wettinger, U. Breitenbücher, F. Leymann, Devopslang–bridging the gap between development and operations, in: European Conference on Service-Oriented and Cloud Computing, Springer, 2014, pp. 108–122.

[4] S. K. Bang, S. Chung, Y. Choh, M. Dupuis, A grounded theory analysis of modern web applications: knowledge, skills, and abilities for devops, in: Proceedings of the 2nd annual conference on Research in information technology, 2013, pp. 61–62.

[5] A. Dyck, R. Penners, H. Lichter, Towards definitions for release engineering and DevOps, in: 2015 IEEE/ACM 3rd International Workshop on Release Engineering, IEEE, 2015, pp. 3–3.

[6] L. Prates, R. Pereira, DevSecOps practices and tools, International Journal of Information Security 24 (2025) 11.

[7] GitLab, The Intelligent Software Development Era, Online Report, 2025. URL: https://about.gitlab.com/developer-survey/, [Accessed 12 November 2025].

[8] Z. Ahmed, S. C. Francis, Integrating security with devsecops: Techniques and challenges, in: 2019 International Conference on Digitization (ICD), IEEE, 2019, pp. 178–182.

[9] H. Myrbakken, R. Colomo-Palacios, DevSecOps: a multivocal literature review, in: International Conference on Software Process Improvement and Capability Determination, Springer, 2017, pp. 17–29.

[10] F. Zampetti, S. Scalabrino, R. Oliveto, G. Canfora, M. Di Penta, How open source projects use static code analysis tools in continuous integration pipelines, in: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), IEEE, 2017, pp. 334–344.

[11] A. Walker, M. Coffey, P. Tisnovsky, T. Cerny, On limitations of modern static analysis tools, in: Information Science and Applications: ICISA 2019, Springer, 2019, pp. 577–586.

[12] A.-d. Taylor, W. Hung, The effects of microlearning: A scoping review, Educational technology research and development 70 (2022) 363–395.

[13] I. Oyeyipo, N. J. Isibor, V. Attipoe, D. C. Ayodeji, B. A. Mayienga, E. Alonge, O. Onwuzulike, Investigating the effectiveness of microlearning approaches in corporate training programs for skill enhancement, Gulf Journal of Advance Business Research 2 (2024) 493–505.
[14] N. Kannan, Assessing the effectiveness of microlearning in employee training programs, International Journal of Training and Development (IJTD) 2 (2024) 1–9.
[15] J. Zhang, R. E. West, Designing microlearning instruction for professionaldevelopment through a competency based approach, TechTrends 64 (2020) 310–318.
[16] V. Mattei, Development of an AI-enhanced microlearning platform for enterprise knowledge transfer, Ph.D. thesis, Politecnico di Torino, 2025.
[17] S. Almuqhim, J. Berri, AI-Driven Personalized Microlearning Framework for Enhanced E-Learning, Computer Applications in Engineering Education 33 (2025) e70040.
[18] K. Boumalek, A. El Mezouary, B. Hmedna, A. Bakki, Transforming Microlearning with Generative AI: Current Advances and Future Challenges, General Aspects of Applying Generative AI in Higher Education: Opportunities and Challenges (2024) 241–262.
[19] D. Le, C. Matsuda, S. Pena, I. Platou, T. Olsen, Effective cybersecurity training using microlearning and the drip concept: A case study of a large regional hospital, Drake Management Review2023 (2023).
[20] G. Granatella, M. Cerioli, G. Lagorio, A. Lupo, Towards Automated Skill and Competency Management in IT Companies, in: 2025 20th Annual System of Systems Engineering Conference (SoSE), IEEE, 2025, pp. 1–6.