Odidont: A GDPR Learning Tool for Developers

Jeroen Chu 5320348
Maria Cristescu 5704502
Antonio Lupu 5677211
Sophie (Schaaf) Langeveld 5345278
Diana Şutac 5756448

***Abstract*—The General Data Protection Regulation (GDPR) establishes rigorous standards for data privacy, yet many software developers find it difficult to translate these high-level legal requirements into practical technical implementations [1]. The abstract nature of these regulations and a general lack of developer-focused education often lead to compliance being overlooked during the initial design phases of a project [2].**

To address this, we propose Odidont, a learning platform specifically designed to bridge the gap between GDPR regulations and software engineering practice. The tool utilizes a microlearning approach to deliver concise content that fits into the modern developer's iterative flow [3]. By combining interactive lectures, quizzes, and a compliance questionnaire, Odidont creates a learning loop that enables engineers to assess their own systems for compliance gaps. Our design emphasizes technical contextualization and gamification to improve engagement and support the long-term retention of privacy principles [4]. Ultimately, Odidont demonstrates a path for making Privacy by Design an achievable and sustainable standard in software development.

I. Introduction

In the modern digital era, the widespread distribution of personal data across the internet introduces significant security risks [1]. The introduction of **General Data Protection Regulation** (GDPR) added more responsibilities on how this data should be handled, while also raising the concerns about privacy. Unfortunately, these rules are not always followed correctly, resulting in data loss from data breaches [5]. For example, Odido, a mobile phone and internet provider in the Netherlands, suffered a data breach in February 2026 in which the personal data of **over 6 million users** was leaked [6]. This included individuals who had not held a subscription with the provider for over 10 years [7]. This violated GDPR Article 5(1)(e) - which mandates that personal data should be kept for no longer than necessary, and Article 5(2) - because the provider promises to keep the data of old customers for 2 years only. Thus, more people were affected from this breach than if Odido would have followed the GDPR regulations.

This begs the question: **why do these leaks happen?** Are companies not complying with the GDPR, and if yes, why? According to the concept of "Privacy by Design", which mentions that privacy and GDPR compliance should be integrated within the software since its initial design stages, software developers should play a big role when it comes to how data is handled [8]. But research from [2] shows that the GDPR seems too complex and hard to understand, and the rules are too vague and not related to technology. This makes it hard for engineers to understand and implement them in their design. Another problem in this context is the lack of investment into teaching developers about privacy and data handling. Most of them are not familiarized with GDPR and its regulations, therefore they focus on other parts of the software [5]. It is important to research and invest in ways to teach developers these rules [1] and in tools that can assist developers with ensuring privacy with their products [8].

In response to these problems, we propose **Odidont**, a **GDPR learning tool** which can be used by software engineers to learn about these regulations and about handling personal data. The application contains "**learning modules**" about the rules of GDPR, explained with examples from the software engineering field, making it easier for developers to follow them. Odidont supports microlearning by splitting modules into smaller lectures, allowing users to find and learn the exact information they need. The added **gamification** elements, such as progress bar, or graying-out completed lectures encourage enjoyment and participation. Besides the lectures, Odidont also contains **quizzes** for each "learning module", so users can test their knowledge once they are done. In the end, there is a final **questionnaire** where developers can check if their system is GDPR compliant or not, which helps them put their knowledge into practice.

Following this, Section II presents the background information needed for our tool, specifically how developers learn, similar tools and information about the GDPR. Section III presents the design and implementation of Odidont. In Section IV, the system is tested using personas and user stories, while Section V discusses the results. Finally, Section VI addresses the limitations and potential future work of the application, followed by the conclusion in Section VII.

II. Background

There are three main challenges when it comes to building GDPR-compliant software. First, it is important to understand what GDPR is, it's goals and the different regulations for data protection. Second, the high-level legal requirements do not always translate into specific technical tasks, such as data minimization or automated deletion. This makes it difficult for

developers to turn a general privacy rule into working code. Finally, traditional training often fails to engage engineers because it focuses on theory rather than practical application. This section offers an explanation of the GDPR and examines how developers learn through hands-on experience and looks at why current software tools often struggle to support the technical side of privacy.

A. The General Data Protection Regulation (GDPR)

In 1995, when the internet was still in its early stages and nobody had any idea how tied it would be to our lives, the European Union (EU) adopted the **Data Protection Directive** [9]. This regulated the processing and free movement of personal data within the EU [10]. However, since this was a directive, each country could still create their own rules on data protection [9]. Because of this, and with the rise and widespread of the internet, it became a hassle to handle the transfer of data between the members of the EU. Thus, the GDPR was created in the early 2010's, and enforced in 2018 [10].

The GDPR is a set of rules about privacy and security [11]. It regulates how organizations around the world handle the personal data of EU citizens. The core motivation behind this was to harmonize data protection laws across the EU, while allowing individual to have control over their digital footprint. These rules establish **privacy** as a **fundamental human right** in the digital age.

To achieve these goals, the regulation has several main topics that dictate how software should be built [12]. First, it defined the notion of "Personal Data" (Article 4), showing that any identifiable information, from names to IP addresses, need protection. Beyond the definitions, the GDPR also introduces "Data Principles" (Article 5), mandating how the data should be processed, collected and kept in order to preserve the privacy of individuals. The concept of "Lawfulness" (Article 6) ensures that data is stored only with a valid legal reason, such as consent. Finally, the GDPR also grants individuals "Data Rights" (Articles 12-22), such as access, restrictions and erasure. These shift the power from corporations back to the users, and enforces systems to be designed with features that allows users to manage their data at any time.

B. Getting the Message Across to Developers

A persistent challenge in achieving GDPR compliance at the engineering level is that software developers frequently **lack familiarity** with data protection principles. Those who do have some awareness often cannot translate legal requirements into concrete technical implementations [1]. Studies on open-source development show that GDPR-related pull requests demand significantly more commits, more code changes, and longer review times than non-GDPR pull requests [13]. This suggests that privacy compliance is both highly **iterative** and architecturally **invasive**. The friction is made worse by the fact that traditional compliance training is typically delivered as annual lectures or text-heavy slide decks. These formats do not match the cognitive profiles and working habits of software engineers [14]. Bridging this gap requires a clear understanding of how developers actually learn.

A useful starting point is the theory of **andragogy**, popularized by Malcolm Knowles, which distinguishes adult learning from childhood education [15]. Adult learners need deep contextualization and expect new knowledge to be immediately applicable to their current work [16]. They validate new information against their existing professional experience and prefer self-directed exploration over passive instruction [17]. These traits are especially pronounced among software developers. Developers work in an environment defined by **continuous learning** and rapid technological change [18]. When confronted with a top-down mandate to memorize abstract GDPR articles, the disconnect from their daily workflow is immediate. The legal text feels entirely unrelated to the Integrated Development Environment (IDE), the sprint backlog, or the failing test suite. In contrast, training that emphasizes the purpose behind an action and offers hands-on scenarios leads to substantially higher engagement and long-term retention [15].

Research into developer learning preferences reinforces this. Developers prefer **interactive**, experiential learning over passive consumption of text or video [19]. The most effective learning moments in programming are often described as breakthroughs achieved by wrestling with a problem rather than reading documentation [19]. Modern cognitive psychology has also debunked the popular notion of fixed "learning styles." What matters is matching instruction to the nature of the content rather than to a learner's self-reported preference [18]. For a technical domain like GDPR compliance, **hands-on exercises** such as refactoring a simulated database schema to remove unnecessary personal data will outperform lectures regardless of how learners describe their preferred style.

Given the volume and legal complexity of the GDPR, which contains 99 articles and 173 recitals [20], presenting all of this material in long sessions risks severe cognitive overload. **Microlearning** addresses this problem by delivering content in focused units of roughly three to ten minutes [3]. It is grounded in cognitive load theory and spaced repetition [21]. Breaking complex topics into small chunks prevents working memory from becoming saturated and supports transfer into long-term memory. For developers who work in Agile sprints with deep focus and frequent context switching, a short interactive scenario is far less disruptive than an hour-long training module. Microlearning also supports role-specific learning paths. A backend engineer can focus on database encryption and pseudonymization while a frontend developer concentrates on consent mechanisms and cookie management [22].

Even well-structured content will fall flat if learners lack the motivation to engage with it. **Gamification** offers a way to address this by embedding game design elements such as points, badges, leaderboards, and narrative challenges into non-game contexts [4]. Developers naturally enjoy solving complex logical problems and demonstrating technical mastery [23]. Reframing a GDPR requirement as an architectural

challenge or a vulnerability to be patched taps directly into this drive. Systematic reviews of gamified educational interventions report higher participation, enjoyment, and motivation compared to traditional approaches [24]. The "attack and defense" paradigm is particularly effective. Developers first exploit a privacy vulnerability and then switch roles to write the fix. This builds deep contextual understanding [25]. Social dynamics amplify these effects further. Team-based challenges and visible leaderboards foster a collaborative security culture. Senior developers model compliant behaviour and junior developers learn through observation and imitation, which is consistent with Bandura's Social Learning Theory [26].

C. Similar Tools on the Market

To inform our development strategy, we analyzed existing compliance and educational tools across **four** primary segments. While these platforms vary in engagement and diagnostic depth, they often lack the technical specificity required to assist software engineers in translating regulatory requirements into architectural decisions.

The first segment consists of **gamified** quiz and flashcard applications, such as Kahoot [27] and Guardey [28]. These platforms rely on microlearning and game mechanics like leaderboards to maintain interest [29]. Guardey, for instance, adopts a model similar to Duolingo [30] but is adapted for broad cybersecurity awareness. While these tools offer high engagement and mobile accessibility, they function as generalized educational platforms. They lack the domain-specific diagnostic capabilities required for an engineer to assess whether a particular software architecture is GDPR-compliant [31].

A second category focuses on targeted **GDPR self-assessments**, including tools from Microsoft [32], Orrick [33], and Sypher [34]. These surveys help organizations evaluate their baseline posture by segmenting regulations into high-level themes. Sypher, for example, generates compliance scores and actionable recommendations based on a brief assessment [34]. However, these tools are generally static and legalistic. They often miss out on providing the interactive lectures or technical details that help engineers understand the reasoning behind privacy requirements. Including this context is a beneficial way to support developers as they make design choices during active development.

For comprehensive **compliance**, organizations often turn to enterprise management software such as OneTrust [35], Osano [36], or Scrut Automation [37]. These platforms offer extensive functionality for data mapping and vendor risk management. While powerful, they are architected for dedicated legal and IT compliance departments who manage audits full-time. For a software engineer, these systems are often overly complex and time-intensive to configure, providing little assistance during the design phase where privacy-by-design must be implemented.

Finally, **corporate microlearning** platforms like OttoLearn [38] and Skillcast [39] deliver focused compliance modules to a general workforce. Skillcast offers Compliance Bites consisting of concise videos on core principles, while OttoLearn uses adaptive algorithms to reinforce daily training. Although effective for general employee awareness, such as identifying phishing attempts, these solutions do not address the specialized diagnostic and engineering needs of a technical team. They rarely touch upon technical implementation details like API security or database retention policies.

Therefore, the current market landscape offers fragmented solutions that do not serve the unique needs of the software engineer. While gamified apps drive engagement and surveys provide organizational insights, there is no integrated platform that combines continuous, technical microlearning with a readiness diagnostic tailored for those building the software. Our proposed tool, Odidont, aims to bridge this gap by grounding GDPR education in the engineering context.

III. Design and Implementation

This section presents the design and implementation of the Odidont tool. First, we provide an overview of the system's components and main design decisions that guided the development. Next, we define the scope of the educational content covered by the tool's prototype. Finally, we describe the tool's technical implementation, including the technology stack we chose and the reason behind it.

A. Design Overview

While designing Odidont, we focused on providing an interactive and developer-friendly learning experience, to make the law-heavy topic of GDPR easier to understand. Rather than presenting the regulation articles as static documentation, the application splits the content into **modular** components, that align with how developers learn and apply new concepts.

We split the application into **two main sections**: lecture topics with interactive quizzes and a compliance questionnaire.

The first component consists of **lecture topics**, that organize the educational content into structured modules. Each topic represents a specific subject area, as further described in III-B. By using a micro-learning approach and making these sections concise and focused, developers can research relevant material fast, while still providing them enough context to understand the reasoning behind each regulation.

To complement these lectures, the tool also leverages **interactive quizzes**, closely connected to the introduced topics. Their purpose is to reinforce learning, by making the learner apply recently-gained knowledge. At the same time, they are useful as a feedback tool, ensuring that there are no gaps in the user's understanding. By interleaving lessons and quizzes, the tool encourages active learning.

For example, after a lesson introducing the purpose of GDPR and the role of developers in ensuring compliance, the user is given a short quiz, to make sure they understood the content. One question asks about the main goal of GDPR, while the other focuses on why developers should care about it. Each answer is followed by immediate feedback, explaining why the selected answer is correct or not. An example of such a quiz is shown in Figure 1.

Fig. 1: Example of an interactive quiz testing users on GDPR knowledge

The second component is a **compliance questionnaire**, which can be used as a 'diagnostic' tool [1]. Unlike the topic-specific quizzes, this questionnaire evaluates the developer's system against a larger set of GDPR regulation. Users answer questions about their system's data handling strategy, after which the tool provides feedback (i.e. if the system is or is not GDPR compliant) and references lectures that could help fill any compliance gaps. This feature bridges the gap between theoretical knowledge and its practical application, encouraging developers to reflect upon their practices [2].

Together, these components create a useful learning loop: developers gain knowledge through the lectures, validate their understanding through quizzes and apply both through the compliance questionnaire. This ensures that Odidont supports not only the gain of theoretical knowledge, but also practical evaluation and application, which is essential for reaching GDPR compliance in the software development context.

B. Content Scope and GDPR Coverage

The main challenge we had when designing Odidont was choosing a **relevant subset** of GDPR topics to include in the educational section of the application. Given that the regulation consists of 99 articles, it was not feasible for a functional prototype to cover the entire scope. Instead, we focused on **four main topics**: "GDPR Basics", "Data Protection", "Lawful Principles" and "Privacy & Data Rights". Together they provide a good foundation for software developers about the GDPR and its regulations, and about how data should be stored and collected. The main reason we had for choosing these topics was that they are foundational and easily transferable into design decisions in software:

- **"GDPR Basics"** and **"Lawful Principles"**: By defining what personal data is and the reasons to store it, these topics can influence the design of a database schema or an API, ensuring that data collection is tied to a specific purpose. This prevents "data hoarding" and helps developers implement data minimization by only creating fields that are legally justified.
- **"Data Protection"**: This topic is directly tied to security requirements. It encourages developers to use, for example, encryption and hashes to protect the personal data of users from unauthorized access or data breaches.
- **"Privacy & Data Rights"**: These rights represent functional requirements that need to be added to software. For example, developers need to add a method for users to be able to delete their data or limit how much of it can be collected and processed.

Of course, there are many other regulations in the GDPR of as much importance as these ones. Since this is only a prototype, we decided to limit the topics to only four. A finished and deployed application would include the rest of the articles as well, in order for Odidont to serve as a comprehensive GDPR learning tool.

C. Implementation

For the development of Odidont, we chose to create a unified, client-side cross-platform architecture. The framework uses .NET 10.0 as the main framework together with MAUI. For the frontend, the app applies Blazor WebAssembly for a responsive, web-standard User Interface (UI). Together, this allows the app to be shipped as a single C# codebase. That way, the app can act as a native application across Windows, Android, macOS, and iOS.

We made these stack choices to keep in mind the following three primary requirements for the tool:

- **Ubiquity**: By using .NET MAUI, the app maintains 100% logic parity between the features across all devices. The app needs the validation logic only once. After that it runs identically on an iPhone as it does on a Windows desktop.
- **Stability**: The GDPR framework consists of established legal principles that do not undergo frequent, volatile changes. Implementing the application fully client-side eliminates the need for a constant internet connection and reduces server overhead. Updates only occur when significant legislative changes (like new guidelines) are released.
- **Performance**: Blazor WebAssembly, paired with .NET 10, offers near-native execution speeds. This ensures that the interactive elements of the app remain fluid and that complex branching logic does not imply latency for the user.

From a design perspective, Blazor also allows standard CSS/Sass. This gave us more **creative freedom** to make the interface look modern and engaging, allowing easier implementation of findings from II-B and II-C without being limited by the usual mobile UI constraints.

[1]The questionnaire can be found here: https://github.com/BestTUDRepos/Odidont/tree/main/Appendix

[2]The content of the questionnaire, along with our prototype can be found here: https://github.com/BestTUDRepos/Odidont/.

IV. User Scenarios

To evaluate whether Odidont meets its intended goals, we define a set of **user scenarios**. Each scenario describes a realistic situation in which a software engineer would interact with the tool. These scenarios guided both the design of the application and the evaluation of its usability. Importantly, each scenario also demonstrates how the learning principles discussed in Section II are applied in practice.

A. Scenario 1: Onboarding a New Backend Developer

Tom is a junior backend developer who recently joined a Dutch fintech startup. During his first sprint, he is assigned to build an API endpoint that stores customer payment information. He has **no prior experience** with GDPR and does not know what rules apply to storing financial data.

Tom opens Odidont and browses the available lectures. He selects the module on data minimization (see figure 2a) because it is directly relevant to his current task. This reflects the principle of andragogy, where adult learners are most engaged when new knowledge is immediately applicable to their work [15]. The lecture explains the principle in plain language and provides a short code-level example showing how to avoid collecting unnecessary fields. By presenting a concrete technical scenario rather than abstract legal text, the tool supports experiential learning, which research identifies as the most effective approach for developers [19].

After reading the lecture, Tom takes the accompanying quiz. He answers incorrectly on a question about data retention periods (see figure 2b), so he revisits that section before moving on. This quiz-based feedback loop aligns with spaced repetition and active recall, two mechanisms that support transfer into long-term memory [21]. The entire lecture takes roughly five to ten minutes, following the microlearning model that prevents cognitive overload during focused work sessions [3].

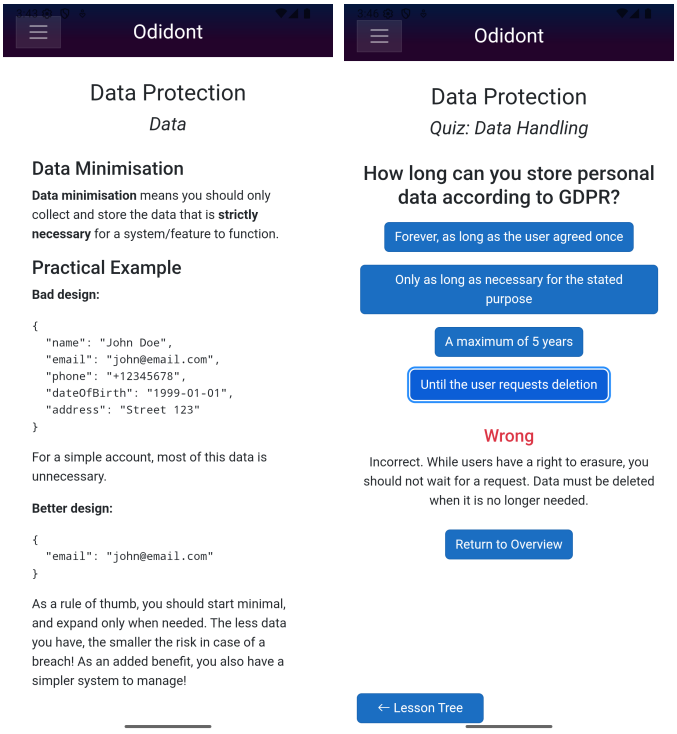
At the end of the session, Tom has a **basic understanding** of which personal data fields he is allowed to store and for how long. He applies this knowledge when designing the database schema for his endpoint.

B. Scenario 2: Preparing for a Compliance Audit

Eva is a senior full-stack developer at a mid-sized Software-as-a-Service (SaaS) company. Her team lead informs her that an external **GDPR audit** is scheduled for next month. Eva has some general awareness of GDPR but is unsure whether the features she built over the past year are fully compliant.

Eva opens Odidont and navigates to the compliance questionnaire. She answers a series of questions about her application's data handling practices (see figure 3a). These include questions about consent mechanisms, data access requests and breach notification procedures. This self-assessment approach supports self-directed learning, a core principle of andragogy where adult learners prefer to evaluate their own knowledge gaps rather than follow a fixed curriculum [16].

After completing the questionnaire, Odidont presents her with a summary that highlights areas where her application



(a) Lecture on Data Minimization (b) Quiz on Data Retention Period

Fig. 2: Example content in the Data Protection Module

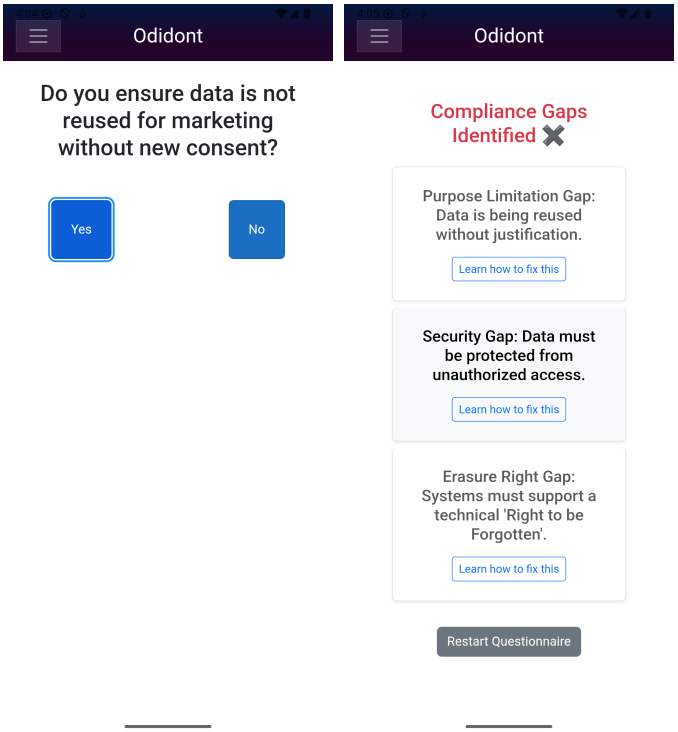
may fall short (see figure 3b). Crucially, each identified gap links back to a specific lecture module. This connection between diagnosis and education ensures that Eva does not just receive a compliance score but also understands the reasoning behind each requirement. This reflects the andragogical principle that adults learn best when they understand the purpose behind the material [15].

Based on this feedback, Eva identifies that her application lacks an automated process for handling data deletion requests. She raises this as a technical debt item in the next sprint planning meeting.

C. Scenario 3: A Frontend Developer Learning About Consent

Lisa is a frontend developer working on a consumer-facing web application. She is tasked with implementing a new cookie consent banner. She knows that cookies require user consent under GDPR but is not sure what counts as valid consent or how granular the options need to be.

Lisa opens Odidont and searches for lectures related to consent. She finds a module that covers the legal requirements for obtaining consent, including the need for an opt-in mechanism and the ability for users to withdraw consent at any time. The lecture includes a **practical example** of a compliant consent flow. This role-specific content demonstrates how microlearning can be tailored to different developer profiles. A frontend developer like Lisa focuses on consent mechanisms and cookie management, while a backend developer would focus on database encryption and pseudonymization [22].



(a) Example questionnaire question (b) Questionnaire Summary and Compliance Gaps

Fig. 3: GDPR Compliance Questionnaire Assessment

After completing the lecture, Lisa takes the accompanying quiz. The current prototype uses multiple-choice questions for this purpose. However, the planned design for Odidont includes additional question formats such as drag-and-drop exercises or code-based scenarios, which would better support hands-on learning [19]. For instance, a question could ask the user to identify which elements of a given consent banner are non-compliant and select the correct fixes. This type of question reframes the GDPR requirement as a technical problem to solve, tapping into the developer's natural drive for problem-solving [25].

After the quiz, Lisa understands the difference between essential and non-essential cookies. She uses this knowledge to implement a banner that allows users to accept or reject each cookie category individually.

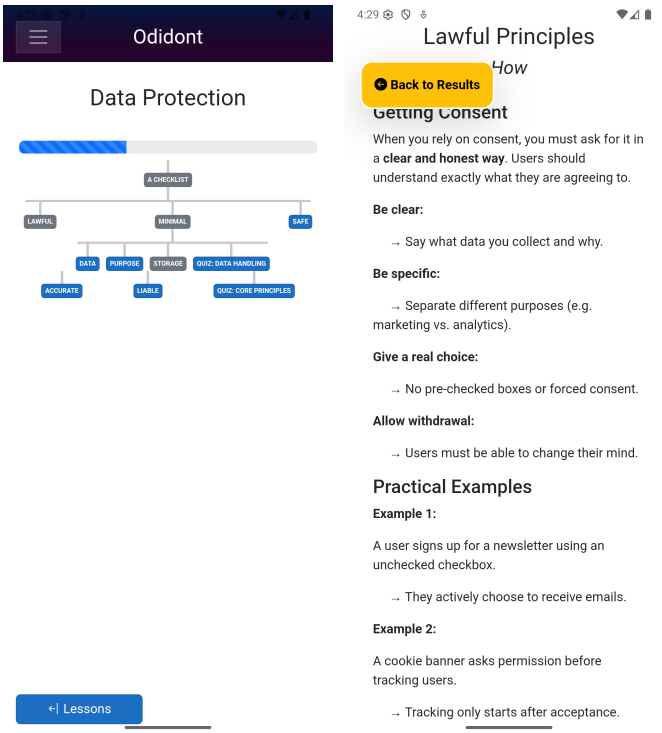
D. Scenario 4: Self-Study During Downtime

Mark is a medior developer who wants to **improve** his understanding of GDPR during quieter periods in his sprint. He does not have a specific task that requires GDPR knowledge right now, but he wants to be better prepared for future projects.

Mark opens Odidont and starts with the first lecture in the recommended order. Each lecture takes roughly five to ten minutes to complete, which fits between his other tasks without breaking his focus. This structure is a direct application of microlearning, which is grounded in cognitive load theory [21]. By keeping sessions short and focused, the tool avoids the cognitive overload that comes with hour-long training modules and fits naturally into the Agile sprint workflow [14].

He completes two lectures and their quizzes during a single afternoon. The quizzes give him **immediate feedback** on what he understood and what he needs to review. Over the course of two weeks, Mark works through all the available lectures. The structured progression through topics supports the principle of self-directed exploration, where adult learners build on their existing professional experience at their own (see figure 4a) [16].

He then takes the compliance questionnaire using a past project as a reference. The results confirm that his previous project handled data subject access requests correctly, but lacked proper documentation for its data processing activities. By validating his knowledge against a real project, Mark engages in experiential learning, which research shows leads to deeper understanding than passive content consumption (see figure 4b) [19].



(a) Example module structured progression (b) Learning about specific compliance gap

Fig. 4: Using the tool to learn about GDPR generally

V. Discussion

The design of Odidont transforms the GDPR education into a **developer-oriented** learning tool [8]. By translating the abstract regulatory principles into concrete, technical scenarios, the tool bridges the gap between the legal text and the software engineers [1]. This contextualization makes it

easier for users to understand why these regulations matter and how they should be integrated in the design process [15]. The combination of microlearning modules and interactive quizzes encourages software developers to engage with the GDPR contents in a way that is aligned with their learning style and preferences [3]. The addition of gamification elements keeps the learning process engaging and provides a feeling of accomplishment [4].

The addition of the compliance questionnaire moves the goal of the system from passive learning to practical application [19]. While its main goal is to check whether a system is GDPR compliant, it also encourages software engineers to reflect on their own design choices and to evaluate how well they apply them in practice [16]. This process promotes a **deeper understanding** of privacy and data protection [25]. Thus, Odidont supports both short-term understanding and long-term behavioral changes [21].

Odidont is designed to shift GDPR education away from abstract legal theory and into the technical environment where software engineers actually work [8]. By translating high-level regulations into practical technical tasks, the tool helps developers see how privacy rules directly impact software design [1]. For example, the scenario involving Tom shows how breaking content into small, focused modules allows a junior developer to learn about data minimization without derailing their current sprint [3]. This reflects the finding that engineers often learn best through hands-on "breakthrough" moments rather than just reading documentation [19]. Instead of being overwhelmed by the entire regulation, this microlearning approach keeps the focus on what is immediately applicable to the code [3].

The inclusion of a diagnostic questionnaire moves the tool from passive learning toward a more active evaluation of a developer's own system [19]. As seen in Eva's scenario, using the questionnaire as a self-assessment helps developers identify specific technical debt, like missing automated deletion processes, that might otherwise be overlooked [16]. This creates a "learning loop" where a diagnostic result links directly to a specific technical lecture, ensuring the education stays relevant to the user's professional context [15]. By tailoring these resources to specific roles, such as focusing on consent for frontend tasks or encryption for the backend, the tool reduces the friction usually found in one-size-fits-all compliance training [8]. This practical approach encourages developers to reflect on their own design choices and helps turn theoretical knowledge into long-term technical habits [25].

Overall, the proposed system shows a way to teach GDPR compliance to software engineers in an engaging and domain-specific way, promoting the integration of privacy regulations since the early stages of the system's design.

VI. Limitations and Future Work

Odidont uses a microlearning approach to instruct developers about GDPR topics, which fits with the cognitive profile of developers. However, this design choice presents a **trade-off** between being concise and being thorough enough to capture all necessary details. Some GDPR topics are inherently complex, so condensing them into short lessons risks oversimplifying important details. This could leave learners with an incomplete education, or even affecting their ability to apply these principles in practice. Future improvements could explore **adjustable lessons** or optional extra content, to keep material detailed without overwhelming users.

Another design challenge lies in keeping the quality of the lessons **consistent**. As the tool expands, it becomes difficult to ensure that each lecture and quiz is technically accurate, and as easy to understand as the other ones. Inconsistency can confuse learners or reduce trust in the tool. To address this limitation, we suggest a strong content **review process**, together with collaboration between legal, educational and technical experts.

In addition to design challenges, Odidont is limited by the **restricted** number of GDPR topics covered, as the system focuses on a small subset of chapters relevant for software engineers. While this provides a targeted learning experience, it doesn't cover all the regulations, which might become relevant during the design of a system. Future work could **expand** the topic coverage to include all the regulations from the GDPR, ensuring that the tool can provide assistance for any privacy concern.

Another limitation of the system is the lack of empirical evaluation using real users. Due to time constraints and lack of approval for user studies, the tool has not been tested with software developers. Instead, personas and **scenarios** were created in IV in order to evaluate how the tool could be used by different people with different needs. While these provide insights into the applicability and usability of the tool, they do not replace direct feedback from actual users. Thus, future work should include a **user study** where software developers with different levels test the tool and provide feedback, in order to asses how Odidont support GDPR and privacy regulations learning.

Together, these improvements highlight how Odidont can evolve as a learning tool, by covering all the GDPR regulations and by collaborating with different experts.

VII. Conclusion

In conclusion, this paper talked about the **challenges** of learning and understanding GDPR requirements. And while these regulations are important to be implemented in order to improve the privacy of a system, their abstract and non-technical nature makes it **difficult for developers** to follow and apply them. Therefore, these rules are often overlooked during the design phase of a system, which can have catastrophic consequences, such as data breaches.

To address these problems, we introduced Odidont, a learning tool that provides lectures, quizzes and a compliance questionnaire specifically tailored towards software engineers. The introduction of microlearning and gamification elements promotes **active learning** and engagement, while the transformation of abstract laws into technical scenarios provides an easier way of understanding. These together prove that "Privacy by Design" is an achievable standard.

While the current version of Odidont is just a functional prototype, focusing on only four topics, it creates a scalable framework for GDPR education. By providing developers with the knowledge of **data protection**, the application plays a crucial role in preventing future data breaches and ensuring privacy becomes a **standard** in our society.

References

[1] A. Alhazmi and N. A. G. Arachchilage, "I'm all ears! Listening to software developers on putting GDPR principles into software development practice," *Personal and Ubiquitous Computing*, vol. 25, no. 5, pp. 879–892, May 2021. DOI: 10.1007/s00779-021-01544-1.

[2] Y. Zhang, F. Ebbers, and N. Martin, "Implementing gdpr: A perspective from developers," *FIfF-Kommunikation*, 2024. DOI: 10.24406/h-486832.

[3] EQS Editorial Team, *How to Use Microlearning in Compliance Trainings*, https://www.eqs.com/compliance-blog/how-to-use-microlearning-in-compliance-trainings/, Accessed: March 31, 2026, 2024.

[4] V. Di Nardo, R. Fino, M. Fiore, G. Mignogna, M. Mongiello, and G. Simeone, "Usage of Gamification Techniques in Software Engineering Education and Training: A Systematic Review," *Computers*, vol. 13, no. 8, p. 196, 2024. DOI: 10.3390/computers13080196.

[5] V. Ayala-Rivera, A. O. Portillo-Dominguez, and L. Pasquale, "Gdpr compliance via software evolution: Weaving security controls in software design," *Journal of Systems and Software*, vol. 216, p. 112 144, 2024, Available: https://www.sciencedirect.com/science/article/pii/S0164121224001894, ISSN: 0164-1212. DOI: 10.1016/j.jss.2024.112144.

[6] Odido, *Information page odido cyber incident.* https://www.odido.nl/veiligheid-eng, Accessed: March 31, 2026, 2026.

[7] Z. Delev, *The odido case: A gdpr transparency lesson*, https://gdprlocal.com/ga/gdpr-transparency-lesson/, Accessed: March 31, 2026, 2026.

[8] I. Theophilou and G. Kapitsaki, "Examining software developers' needs for privacy enforcing techniques: A survey," *arXiv preprint arXiv:2512.14756*, 2025. DOI: 10.48550/arXiv.2512.14756.

[9] iHasco, *A very brief history of the GDPR*, https://www.ihasco.co.uk/blog/brief-history-of-the-gdpr, Accessed: April 1st, 2026, 2018.

[10] European Data Protection Supervisor, *The history of the General Data Protection Regulation*, https://www.edps.europa.eu/data-protection/data-protection/legislation/history-general-data-protection-regulation_en, Accessed: April 1st, 2026.

[11] B. Wolford, *What is GDPR, the EU's new data protection law?* GDPR.eu, Accessed: April 1st, 2026.

[12] European Parliament and Council of the European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*, Official Journal of the European Union, L 119/1, 2016.

[13] L. Franke, H. Liang, S. Farzanehpour, A. Brantly, J. C. Davis, and C. Brown, "An Exploratory Mixed-methods Study on General Data Protection Regulation (GDPR) Compliance in Open-Source Software," in *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '24, Barcelona, Spain: Association for Computing Machinery, 2024, pp. 325–336, ISBN: 9798400710476. DOI: 10.1145/3674805.3686692.

[14] Thomson Reuters, *Avoiding the Knowledge Gap with Microlearning: The Importance of Relevant and Repetitive Compliance Training*, White paper, Accessed: March 31, 2026, 2024.

[15] N. Graf, *10 Simple Principles of Adult Learning*, https://www.wgu.edu/blog/adult-learning-theories-principles2004.html, Accessed: March 31, 2026, 2025.

[16] M. Feder, *Adult Learning Theory and the Principles of Andragogy*, https://www.phoenix.edu/articles/education/adult-learning-theory-and-the-principles-of-andragogy.html, Accessed: March 31, 2026, 2024.

[17] University of San Diego, *15 Top Strategies for Teaching Adult Learners*, https://pce.sandiego.edu/15-top-strategies-for-teaching-adult-learners-faqs/, Accessed: March 31, 2026, 2024.

[18] N. C. C. Brown, F. F. J. Hermans, and L. E. Margulieux, "10 things software developers should learn about learning," *Commun. ACM*, vol. 67, no. 1, pp. 78–87, Dec. 2023, ISSN: 0001-0782. DOI: 10.1145/3584859.

[19] M. Cneude, *The art of learning for software developers*, https://thevaluable.dev/learning-developer-efficiently-effectively/, Accessed: March 31, 2026, 2021.

[20] M. Miri, F. H. Foomany, and N. Mohammed, "Complying With GDPR: An Agile Case Study," *ISACA Journal*, vol. 2, 2018, Accessed: March 31, 2026.

[21] I. Oyeyipo et al., "Investigating the effectiveness of microlearning approaches in corporate training programs for skill enhancement," *Gulf Journal of Advance Business Research*, vol. 2, Mar. 2025. DOI: 10.51594/gjabr.v2i6.122.

[22] D2L, *10 Best Compliance Training Best Practices*, https://www.d2l.com/blog/compliance-training-best-practices/, Accessed: March 31, 2026, 2026.

[23] SecurityJourney, *How To Create a Successful Secure Coding Training Plan*, https://www.securityjourney.com/post/how-to-create-a-successful-secure-coding-training-plan, Accessed: March 31, 2026, 2024.

[24] S. Korniienko, P. Zahorodko, A. Striuk, A. Kupin, and S. Semerikov, "A Systematic Review of Gamification in

Software Engineering Education," in *CEUR Workshop Proceedings*, Accessed: March 31, 2026, vol. 3844, 2024.

[25] Application Security Master, *Secure Coding Challenges: Master Developer Security Skills*, Medium, Accessed: March 31, 2026, 2025.

[26] J. Hallett et al., "The Social Psychology of Software Security (Psycurity)," *IEEE Transactions on Software Engineering*, vol. 50, no. 11, 2024. DOI: 10.1109/TSE.2024.3431602.

[27] Kahoot! *Learning games and quizzes for everyone*, https://kahoot.com/, Accessed: March 31, 2026, 2026.

[28] Guardey, *Gamified cybersecurity awareness training*, https://www.guardey.com/, Accessed: March 31, 2026, 2026.

[29] S. Upadhyay, *Drive compliance training completion with scalable microlearning*, https://www.learningguild.com/articles/drive-compliance-training-completion-with-scalable-microlearning, Accessed: March 31, 2026, 2025.

[30] Duolingo, *The world's best way to learn a language*, https://www.duolingo.com/, Accessed: March 31, 2026, 2026.

[31] TalentCards, *Create laser focused training*, https://www.talentcards.com/tour, Accessed: March 31, 2026, 2026.

[32] EXIN, *The 5 best gdpr compliance tools*, https://www.exin.com/article/the-5-best-gdpr-compliance-tools/, Accessed: March 31, 2026.

[33] Orrick, *Gdpr readiness assessment tool*, https://www.orrick.com/Solutions/GDPR-Readiness, Accessed: March 31, 2026.

[34] Sypher, *Free gdpr compliance tools*, https://www.sypher.eu/free-tools, Accessed: March 31, 2026, 2026.

[35] OneTrust, *Gdpr compliance software*, https://www.onetrust.com/solutions/gdpr-compliance/, Accessed: March 31, 2026, 2026.

[36] Osano, *Gdpr compliance software*, https://www.osano.com/solutions/gdpr-compliance-software, Accessed: March 31, 2026, 2026.

[37] Scrut Automation, *Top 5 gdpr compliance software*, https://www.scrut.io/hub/gdpr/top-5-gdpr-compliance-software, Accessed: March 31, 2026, 2025.

[38] OttoLearn, *Microlearning pricing plans and gamification*, https://www.ottolearn.com/microlearning-pricing-plans, Accessed: March 31, 2026, 2026.

[39] Skillcast, *Compliance bites microlearning*, https://www.skillcast.com/course-library/compliance-bites-microlearning, Accessed: March 31, 2026, 2026.