UCLouvain

epl

École polytechnique de Louvain

Gamification and AI guidance for training in cyber ranges

Author: **Lucas Williot**
Supervisors: **Axel Legay, Benoît Duhoux**
Readers: **Ramin Sadre, Benoît Duhoux**
Academic year 2023–2024
Master [120] in Computer Science

Abstract

Cybersecurity training is essential for everyone who interacts with digital systems, as human error is a primary vector for cyber attacks. With the rapid emergence of new vulnerabilities due to technological advancements, continuous and role-specific training is crucial. However, maintaining motivation for ongoing education, especially among non-specialists, remains challenging. Research highlights that gamification can significantly enhance engagement and motivation, while a lack of adaptive learning difficulty can lead to frustration and disengagement.

This thesis presents a comprehensive cybersecurity training platform designed to address these challenges. The platform integrates multiple gamification elements to increase engagement and includes two key guidance techniques to support users with different levels of expertise. First, an adapted recommendation system tailored to the specific nature of cybersecurity training scenarios provides personalized guidance outside of scenarios. Second, a chatbot powered by advanced large language models (LLMs) offers real-time support within scenarios, helping users navigate tasks and answering their questions. The platform currently includes two functional scenarios focused on web vulnerabilities. While these scenarios are specific, the use of cyber ranges as a core technology enables the creation of a wide variety of realistic and diverse training environments.

By covering four primary research areas: cyber ranges, gamification, recommender systems, and LLMs, this work does not aim to push the boundaries of any single field. Instead, it innovates by combining these domains in a way that sustains motivation and provides the necessary guidance for users at different levels of expertise.

Acknowledgments

I would like to express my gratitude to all those who contributed to the completion of this thesis, particularly:

Professor Axel Legay, my supervisor, for granting me the opportunity to explore this research topic and for his valuable feedback throughout the year.

Benoît Duhoux, who also supervised me, for his support and mentorship during this year. His expertise, constant availability, and the significant time he invested have been invaluable to the completion of this thesis. His dedication and encouragement have been a tremendous source of motivation.

Finally, I extend my heartfelt thanks to my parents for their unwavering support throughout my academic journey. Their encouragement and belief in my abilities have been a constant source of strength and inspiration.

Table of contents

Chapter 1

Introduction

Rapidly evolving technologies and the growing diversity of IT tools are leading to an increasing number of vulnerabilities. Data protection and the smooth operation of IT systems have become crucial issues for businesses, governments, and individuals. In this context, cybersecurity plays a key role, requiring increasingly sophisticated skills to deal with emerging threats.

IT security policies are based on three fundamental principles: confidentiality, integrity, and availability. Confidentiality ensures that only authorized persons can access sensitive information. Integrity ensures that data is not altered in an unauthorized way. Availability ensures that systems and data are accessible when needed. These principles are essential to protect digital assets against a wide range of threats.

However, the field of IT security is vast and complex, making training in this sector particularly challenging. Upcoming cybersecurity professionals face a number of obstacles: a lack of qualified trainers, limited access to realistic training environments, an often high entry level on existing platforms, and learners' loss of motivation in the often disengaging training paths.

Another point that I want to address is that everyone who interacts with an IT system can be a potential vector of attack. Consequently, security training should not be limited to security professionals but should also extend to all users. These users must receive training adapted to their level and the level of access they have to the system. A holistic approach to cybersecurity training, covering a wide range of skills and knowledge levels, is essential to strengthen overall resilience against cyber threats.

To address these challenges, this thesis proposes the development of a cybersecurity training platform integrating realistic training scenarios, gamification

mechanisms, and guidance for the user. The aim is to provide an effective and motivating tool for cybersecurity training, capable of meeting the varied needs of learners. This online platform is distinguished by several innovative features:

- **Accessibility and adding scenarios**: The platform enables easy integration of new training scenarios, offering a variety of realistic experiences for users of all levels.
- **Gamification**: To enhance engagement and make training more enjoyable, the platform incorporates gamification elements such as profile and avatar personalization, a scoring and leaderboard system, and a tool for visualizing progress in training environments.
- **Recommendation system**: By taking into account the user's profile, strengths, weaknesses, and objectives, the platform suggests suitable scenarios, optimizing each learner's training path.
- **Personal trainer**: This digital coach plays a dual role, increasing user immersion through personalized interactions and providing guidance, allowing users to ask questions about security, scenarios, or receive additional recommendations.

The following sections will delve into various aspects of this project. Chapter 2 provides the background material, including a detailed exploration of cyber ranges, gamification, recommender systems, and the basics of Large Language Models. Chapter 3 presents the problem statement, outlining the key challenges that this thesis seeks to address. Chapter 4 details the proposed solution, covering the architecture and technologies, scenario integration, gamification mechanisms, and the recommender system. Chapter 5 validates the proposed solution through a series of evaluations and tests. Chapter 6 discusses future work, highlighting potential improvements in gamification, monitoring, and the recommender system. Finally, Chapter 7 concludes the thesis by summarizing the findings and implications of this work.

During the preparation of this thesis, I utilized various tools to enhance the quality and clarity of the writing. Specifically, I used ChatGPT[31] to rephrase and reformulate sections of the text, ensuring that the content was expressed more clearly and effectively. Additionally, I employed DeepL[7] for translation purposes, aiming to improve the linguistic quality of the document. These tools were used solely to enhance the English language quality and overall writing of the paper.

For those interested in the implementation details of this project, the source code is available on GitHub.

Chapter 2

Background Material

In this section, I introduce the key concepts that will be essential throughout the reading. I begin with cyber ranges, the foundational element upon which the platform is built, and then move on to gamification and the various mechanisms identified in the literature. Next, I provide an introduction to the fundamentals of the recommendation system, which serves as the primary guidance tool outside the scenarios. Finally, I conclude with a brief overview of the Large Language Model (LLM) used to implement the personal trainer, along with the techniques employed to enhance its capabilities.

1 Gamification

Swacha's bibliometric survey[42] shows that gamification is a rapidly evolving field of research since 2013. Hundreds of new publications come out every year all over the world. Its most common application is in the field of computer sciences, but the most frequently used keywords related to gamification show that it's an interdisciplinary subject halfway between educational theory and behavioral psychology.

To better understand gamification, it's important to define what it is and how the notions of engagement and motivation are intrinsically linked to it. Kapp [17] defines gamification as *"the use of game-based elements, techniques, and game strategies to promote learning through increasing engagement, fostering motivation, and supporting critical thinking skills"*. Numerous studies have demonstrated the veracity of this definition by measuring student engagement after the addition of game elements to their learning process. This definition also highlights the skill of critical thinking, one of the most important abilities for learning new content and adapting to new situations. Gee[13] has identified this skill as one of the 36 that video game users can learn by playing, and which are used in the context of

education. Gamification can therefore be used as a tool for learning and influencing behavior, with the aim of generating desired behavior.

Alsawaier[2] compiled the results of around twenty studies to identify which gameplay elements best increase engagement and why some studies fail to show positive results. Two particularly effective aspects of gamification are the social aspects of competition and collaboration, and the elements of reward and punishment. These aspects include game-based elements such as badges, points, leaderboards, and virtual rewards. Other gameplay elements that often yield positive results include quests or challenges, progress bars, and the acquisition of levels during progression, enabling progress to be quantified. Conversely, papers reporting negative or mixed results generally focus on a very limited number of gameplay elements, poorly integrated into the learning process, or force learners to use all the gameplay elements without giving them any choice. Nicholson[30] also notes that another element that can negatively impact engagement and provoke anxiety and frustration is when the gamified learning material is not at an appropriate level for the student.

AL-Smadi [1] has also identified the elements that are really effective in gamifying learning. He draws similar conclusions on the rewards aspect, but also on the fact that students should receive rapid feedback on their progress, and introduce the same gameplay elements. However, he pushes the social aspect a little further, showing the positive effects of a personalized profile allowing the creation of one's own avatar and encouraging social interaction such as discussion forums, comment areas, etc. Another important element of personalization is the customization of one's learning path by defining a goal to achieve the desired competencies. The design of the tasks to be carried out was also analyzed, showing that the level of difficulty must absolutely be adapted to the user, but also that it is important for them to be interactive and offer several means of resolution, enabling repetition and experimentation.

My contribution involves proposing a combination of gamification elements that have been empirically proven to be effective and adapting them to the context of cybersecurity training. The focus is on integrating elements that enhance engagement, such as rewards, personalization, progress tracking, and goal-setting, while avoiding factors that have been identified as detrimental to user engagement. In particular, this work emphasizes the importance of tailoring the difficulty level to the user.

2 Cyber ranges

While the primary focus of this thesis is not on developing Cyber ranges, it is important to provide a brief overview of this technology to contextualize the broader scope of cybersecurity training platforms. According to the National Institute of Standards and Technology (NIST) [46], *"cyber ranges are interactive, simulated representations of an organization's local network, system, tools, and applications that are connected to a simulated Internet level environment"*. The European Cyber Security Organisation (ECSO) further expands on this by defining a cyber range as *"a platform for the development, delivery and use of interactive simulation environments. A simulation environment is a representation of an organization's ICT, OT, mobile and physical systems, applications and infrastructures, including the simulation of attacks, users and their activities, and of any other Internet, public or third-party services which the simulated environment may depend upon"* [11].The growing complexity and frequency of cyber-attacks, exemplified by incidents like the Stuxnet worm, highlight the necessity for advanced training platforms such as cyber ranges. These platforms play a crucial role in enhancing organizational defenses by shifting the perception of users from being the weakest link in cybersecurity to assets that can be trained and strengthened, thereby improving overall security posture. Cyber ranges are predominantly used in academic settings, accounting for 31% of their applications, where they serve educational and research purposes. Beyond academia, they are extensively utilized in commercial sectors, military, defense, intelligence, and government, reflecting their broad applicability and importance in preparing cybersecurity professionals to tackle real-world challenges.[44]

Despite the significant benefits that cyber ranges offer, their development and management come with several challenges. One of the primary challenges is the complexity in setup and management. As Podnar et al.[35] highlighted, creating and managing a cyber range involves replicating realistic network environments with diverse hardware and software tools. This complexity is further intensified by the need to accurately mirror real-world scenarios, making the setup and ongoing management of cyber ranges a resource-intensive endeavor.

Another major challenge is scalability. As organizations expand their cyber range capabilities, ensuring that these environments can accommodate more users and larger simulations becomes increasingly difficult. Bianchi et al.[3] noted that achieving scalability requires sophisticated infrastructure and resource management strategies to support the growing demand.

Realism is another critical factor that poses a challenge in the effective deployment of cyber ranges. Grigoriadis et al.[15] emphasized that the effectiveness of a cyber range is heavily dependent on the level of realism it can achieve. However, replicating real-world network environments and attack scenarios with high fidelity

is challenging and requires substantial resources.

The cost associated with establishing and maintaining a cyber range is also a significant barrier. As Leitner et al.[21], in a previous paper, pointed out, the financial investment required is considerable, encompassing costs related to infrastructure, software licenses, and the employment of specialized personnel.

Finally, the lack of standardization in the development of cyber ranges further complicates their implementation. The absence of standardized frameworks and terminologies hinders both the development process and the widespread adoption of cyber ranges. There is a pressing need for the cybersecurity community to establish standardized practices and tools, with a particular emphasis on open-source or publicly available resources, to facilitate the creation and maintenance of cyber ranges. Addressing these challenges is essential for the effective use and expansion of cyber ranges in the future.[49]

Cyber ranges leverage a variety of technologies to create realistic and scalable environments. Virtualization technologies are central to the functioning of cyber ranges and typically involves the deployment of Virtual Machines (VMs), which provide a layer between the hardware and the host operating system, managed by a hypervisor. This method offers flexibility and security but is less scalable compared to containerization. Containerization, such as that provided by Docker, offers a lightweight alternative to virtualization. It allows applications to be containerized, stored, transported, and deployed efficiently. Although containers are more scalable, VMs remain more flexible and secure, and there is potential for merging these technologies into a form of cloud portability.[44]

Cloud computing also plays a significant role in cyber range operations. Leitner et al.[21] discussed how cloud platforms such as AWS and Microsoft Azure provide the flexibility and scalability necessary to manage large-scale cyber ranges. These platforms enable dynamic resource allocation and the execution of complex simulations, making them ideal for expanding cyber range capabilities.

Orchestration and automation are also crucial for the efficient deployment and management of cyber range environments. Leitner et al.[22] emphasized that tools like Ansible and Terraform are critical for automating the setup and teardown of simulations, thus enhancing the operational efficiency of cyber ranges.

Given the importance of virtualization and containerization technologies in the development of cyber ranges, this thesis will focus specifically on Docker as the primary tool for creating and virtualizing the scenarios used within the cybersecurity training platform. Docker is particularly suited for this purpose due to its lightweight nature and the scalability it offers, which are essential for managing multiple, concurrent simulations efficiently.

3 Recommender systems

Recommender systems, also known interchangeably as recommendation systems in the literature, are pivotal in the guidance components of this thesis. Their primary role is to predict user preferences for specific items, thus providing personalized suggestions that enhance user experience and engagement. Their ubiquitous presence across the internet highlights their importance in reducing the time and effort users need to find content or products that match their interests.

According to Thorat et al.[43], recommender systems are a type of filtering system designed to predict the preference a user may have for a particular item. These systems operate by analyzing patterns of user behavior and making inferences about future actions based on past preferences. The term "items" in the context of recommender systems is a generic term that refers to any entity being recommended by the system. This could range from products like music, movies, and books to research articles, news, and more.

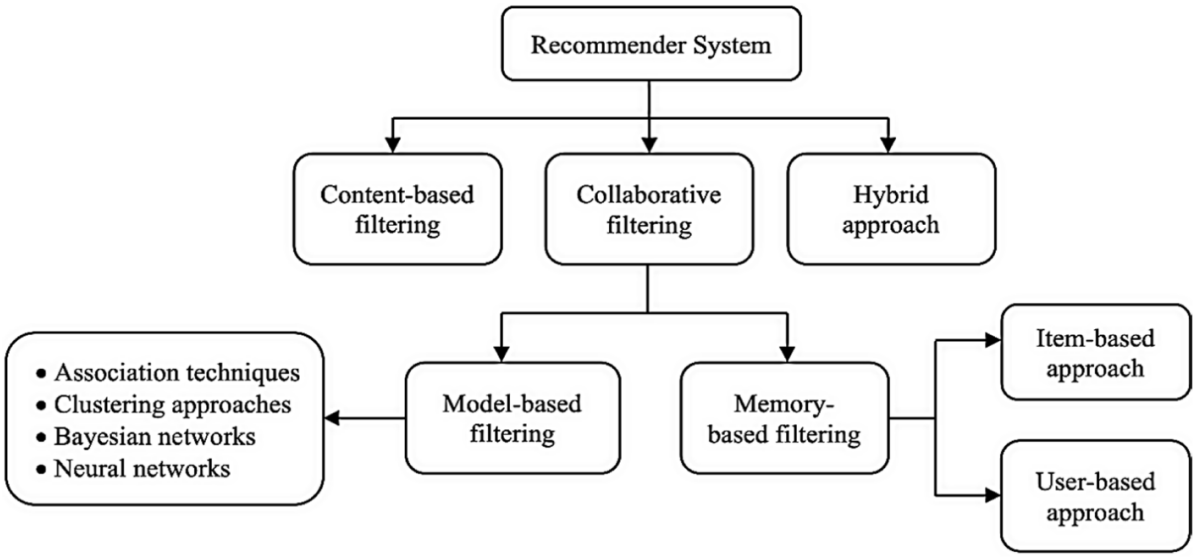


Figure 2.1: Types of recommender systems. Source:[39]

Deepjyoti et al.[39] in their literature review identify three main approaches to building recommender systems: content-based filtering, collaborative filtering, and hybrid approaches, which typically combine elements of the first two or include other less common methods. The figure 2.1 summarizes the different types and subcategories of recommender systems that are explained in this section. In Content-Based Filtering, item information is systematically collected and utilized to construct a detailed profile for each item. When a user interacts with an item, the profiles of

various items that the user has shown a preference for are aggregated to develop the user's profile. The creation of this user profile poses significant challenges due to its specificity to each type of item, necessitating comprehensive knowledge of the items to produce a profile that is both meaningful and information-rich. One of the notable advantages of this approach is its ability to rapidly adapt to changes in the user's preferences, reflecting these shifts promptly in the recommendations provided. However, it is important to note that this system predominantly recommends items similar to those previously favored by the user, which can lead to over-specialization in the recommendations. Consequently, while the system excels in responsiveness to evolving interests, it risks limiting the diversity of recommendations by consistently suggesting items of a similar nature, unless a significant change in user interest occurs. Additionally, Content-Based Filtering can recommend items that have not yet been evaluated by other users, as it relies solely on the intrinsic features of the items themselves.

Collaborative Filtering is a recommendation technique that identifies a group or collection of users, referred to as the neighborhood of a target user, whose preferences, likes, and dislikes are similar to those of the target user. The central idea behind this approach is to recommend items that are favored by the majority of users within this neighborhood of the target. The effectiveness of Collaborative Filtering hinges on the accuracy with which the algorithm identifies the appropriate neighborhood for the target user.

One of the key advantages of Collaborative Filtering is that it does not require any prior knowledge of item features, relying instead on user interaction with items. However, this method is not without its challenges. It suffers from the cold-start problem, which occurs when there is insufficient data on a new user or item, making it difficult to generate accurate recommendations. Additionally, privacy concerns arise due to the necessity of sharing user data to build the neighborhood profiles. Collaborative Filtering methods are broadly categorized into two types: memory-based approaches and model-based approaches.

Memory-based collaborative filtering recommends new items by directly considering the preferences of users within a defined neighborhood. This approach leverages the utility matrix, a foundational data structure that represents the preferences or ratings of users across a set of items, to make predictions. Although effective, memory-based collaborative filtering is known to be computationally intensive. This method can be further subdivided into two subcategories: user-based and item-based collaborative filtering.

In user-based collaborative filtering, if a new item receives positive ratings from users within the neighborhood, it is recommended to the target user. However, the

performance of this method can degrade with the addition of new users, as the similarity between users is often dynamic and requires frequent recalculations.

To address these performance issues, Sarwar et al.[40] introduced the item-based collaborative filtering approach. This method constructs an item-neighborhood by identifying items similar to those that the user has previously rated positively. Since the similarity between items tends to be more stable over time compared to the often-changing similarity between users, item-based filtering allows for more efficient recommendation calculations and reduces the need for frequent updates, thus improving the overall performance of the recommendation system.

On the other hand, Model-based collaborative filtering employs various data mining and machine learning algorithms to develop a model that predicts a user's rating for an unrated item. This approach involves two key steps: first, building and training the model, and second, using a prediction function to generate ratings based on the model and the user's profile. Model-based methods have the advantage of being able to make recommendations even to users who are not present in the initial model. Additionally, they address some of the traditional challenges of recommender systems, such as data sparsity and scalability, by utilizing dimensionality reduction techniques.

The hybrid approach combines multiple recommendation techniques to leverage their individual strengths and mitigate their weaknesses. A hybrid algorithm may aggregate the results from separate techniques, or it may apply one technique within the framework of another. This integration of different approaches typically leads to improved performance and greater accuracy in recommendations, as it balances the limitations of each individual method.

Lü et al.[25] summarize some of the key challenges that significantly impact their performance and usability. One of the most prominent issues is the cold start problem, which occurs when new users or items lack sufficient data, making it difficult for the system to generate accurate recommendations. This issue is closely related to data sparsity, number of users increase fast and most of them tend to rate a limited number of items, further degrading the system's performance. Scalability is another critical challenge, as recommender systems must efficiently handle the computational demands of processing millions of users and items. To manage this, algorithms need to be both computationally efficient and capable of incremental updates as the data grows. Balancing diversity and accuracy is also crucial. While recommending popular items can improve accuracy, it can reduce diversity by limiting the exposure to less obvious but potentially valuable options, which is contrary to the goal of helping users discover new products. Security concerns

arise as recommender systems are vulnerable to malicious attacks, such as shilling attacks [16], where users attempt to manipulate the system to unfairly promote or demote certain items. Developing robust defenses against these attacks is essential to maintaining the integrity of the recommendations. Finally, the evaluation of recommendations remains a complex task. Numerous metrics are available, such as mean absolute error, root mean squared error, and measures of correlation between predictions and true evaluations. However, there is no consensus on which metric should be used, as this depends on the task, making them really difficult to compare. What's more, these evaluations are mostly based on user ratings, making evaluation very difficult until the system is online with active users, underscoring the importance of empirical user studies.

The body of research on recommender systems has largely focused on domains such as movie recommendations, with significantly fewer studies addressing applications in health, tourism, and education [39]. However, Urdaneta-Ponte et al.[45] conducted a review of educational recommender systems and find that collaborative filtering is the predominant technique employed. Despite its popularity, very few studies have explored content-based filtering in this context. My contribution will focus on proposing a recommender system that primarily utilizes content-based filtering, specifically tailored to the complex structure of cybersecurity training scenarios. This approach will be similar to the work of van Meteren et al.[47], who faced challenges related to overspecialization in their recommendations. To address this, I adapted the update mechanism of the user profile to encourage diversity in recommendations.

4 Large Language Models

Large Language Models (LLMs) represent a significant evolution in natural language processing, enabling machines to understand and generate human-like text at unprecedented levels of sophistication. LLMs are typically built upon Transformer architectures, which are foundational to their ability to process and generate large amounts of text efficiently.

Transformers, introduced by Vaswani et al.[48], are the underlying architecture that powers most modern LLMs. The Transformer model is designed to handle sequences of data, such as sentences, more effectively than previous models by leveraging a mechanism called self-attention. This mechanism allows the model to focus on different parts of a sentence with varying levels of importance, thereby capturing the contextual relationships between words. Transformers form the backbone of many LLMs, providing the computational structure needed to process

vast amounts of text data efficiently.

GPT (Generative Pre-trained Transformer), which will be used for the Personal Trainer of this work, is a specific implementation of an LLM that builds on the Transformer architecture. Developed by OpenAI and presented by Radford et al.[36], the GPT model is trained in two stages: pre-training and fine-tuning. During pre-training, the model is exposed to a large corpus of text to learn general language patterns. In the fine-tuning stage, the model is further trained on a specific dataset to adapt it to particular tasks, such as text generation or translation. This dual-stage training approach allows GPT models to generate coherent and contextually relevant text across a wide range of applications. The size and capability of LLMs have grown dramatically over the years, with models like GPT-3 and GPT-4 pushing the boundaries of what these systems can achieve.

One of the tasks that has been particularly studied, as demonstrated in the work of Ouyang et al.[33], is the ability to successfully follow instructions. This new capability allows the model to detect user needs and generate reliable, structured output, which can then be used to call functions. Function calling is a recent advancement that enables models to interact more dynamically with external tools and APIs. This feature allows LLMs to perform tasks beyond mere text generation, such as retrieving data, performing calculations, or even controlling software applications[51]. OpenAI's models, for example, can use function calling to access external databases, APIs, and other services, making them highly versatile and capable of integrating with various workflows. The integration of function calling enhances the interactivity of LLMs, allowing them to execute complex sequences of actions based on user inputs or predefined triggers. This capability is particularly valuable in applications like chatbots, where the model can perform specific actions based on the context of the conversation.

Building on these advancements, Retrieval-Augmented Generation (RAG) represents a powerful approach that integrates the strengths of both retrieval-based and generation-based models. RAG enhances a model's ability to generate accurate and contextually relevant responses by first retrieving pertinent information from a knowledge base or external source and then using this data to inform the generation process. This approach is especially valuable in scenarios requiring domain-specific knowledge, as discussed by Lewis et al.[23]. In applications like chatbots, RAG enables the system to incorporate up-to-date or specialized information seamlessly, ensuring that the generated output is both relevant and grounded in factual accuracy. This integration significantly enhances the quality and reliability of the model's content.

In this work, a chatbot is developed, powered by a Large Language Model

(LLM) specifically tailored for cybersecurity training. The chatbot is equipped with tools that enable maximum personalization of user interactions on the platform by allowing the model to access user-specific information. Additionally, a Retrieval-Augmented Generation (RAG) mechanism is implemented within the chatbot. This enhancement enables the chatbot to respond to cybersecurity-related queries with precision, drawing upon existing, relevant data to provide factually accurate and contextually appropriate answers. By integrating these capabilities, the chatbot offers personalized and interactive experiences while serving as a reliable tool for delivering domain-specific knowledge in the field of cybersecurity.

Chapter 3

Problem Statement

Cybersecurity training is becoming increasingly important as the complexity and frequency of cyber threats continue to grow. Traditional training methods often struggle to engage trainees effectively, leading to suboptimal skill acquisition and retention. Gamification, a method that has been shown to enhance engagement in various educational contexts, presents a promising solution to this challenge. However, its application within cybersecurity training remains underexplored. To address this gap, it is necessary to investigate how to effectively implement gamification techniques that not only engage trainees but also enhance the overall learning experience.

Moreover, the effectiveness of cybersecurity training is heavily dependent on the adaptability of the training platform to cater to the diverse needs of users with varying levels of expertise. Current platforms often lack the flexibility to adjust to different user profiles, resulting in a one-size-fits-all approach that fails to adequately challenge more experienced users or provide sufficient support to novices. Thus, there is a critical need to design a gamified cybersecurity training platform that can be easily customized to accommodate various types of scenarios and user expertise levels.

Given these challenges, this thesis seeks to answer the following research questions:

- **Research Question 1:** How to apply gamification techniques that increase engagement of trainees in the context of cybersecurity training?
- **Research Question 2:** How to design a gamified cybersecurity training platform that is easily adaptable to various types of scenarios and user expertise levels?

By addressing these questions, this research aims to develop a cybersecurity training platform that not only utilizes effective gamification strategies to increase

trainee engagement but also ensures that the platform is adaptable and scalable, meeting the diverse needs of its users.

Chapter 4

Solution

This chapter presents the proposed solution to the two research questions that emerged from the literature review. To do this, it begins with an overview of the platform, enabling us to understand its general operation and the features developed. I then present the project architecture and the technologies used. The next section details each gamification element integrated into the platform, followed by a detailed description of the recommendation system and the personal trainer. The chapter concludes with a discussion of the integration of new scenarios.

1 Overview

The project, therefore, takes the form of a web application enabling users to practice cybersecurity by running scenarios that are as realistic as possible. It integrates various gameplay and guidance elements to support the user's learning process. The platform contains 6 main pages: a dashboard, a list of scenarios, a leaderboard, a list of badges, a list of goals, and a final page for viewing progress in a scenario.

The dashboard is the user's main page, enabling them to update their summary profile, which is visible to other users, and to track their progress. There are 2 types of progress visible on the page: progress linked to the user's goal, represented by a radar chart with each edge representing a skill required for the selected goal. There are also general progress indicators in the form of progress bars, which show how many points have been scored and how many scenarios have been completed in total on the platform. They are segmented by difficulty and scenario type. It's also via this page that you can communicate with HAL, the platform's personal trainer, who is on hand to answer any questions you may have, as well as providing quick access to recommended and recently played scenarios. Finally, the dashboard also contains a list of all the badges the user has unlocked. Figure 4.1 shows the

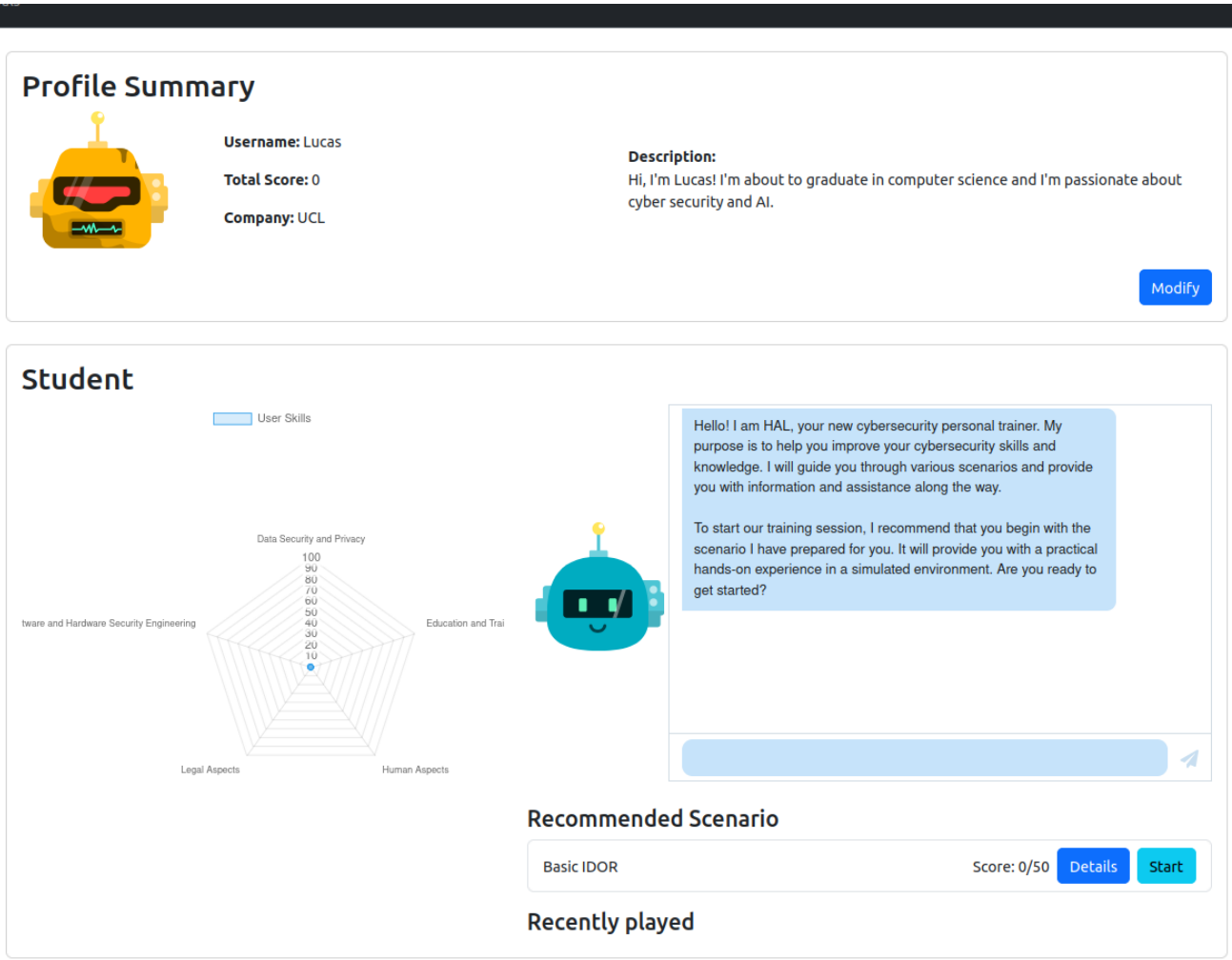


Figure 4.1: Dashboard page

top of the dashboard just after an account has been created. The profile summary has been updated, as has the avatar, and the goal selected is "Student." You can see the radar chart, which is still blank, as well as HAL presenting itself and inviting the user to start learning with the recommended scenario just below.

If the user wishes to play a scenario other than the one recommended, they can access a dedicated page that lists all the scenarios available on the platform. This list is simple and includes the name, difficulty level, type of scenario, points already obtained, points that can be earned by completing the scenario, and the average user rating for that scenario. After completing a scenario, users have the opportunity to submit a rating, contributing to this average rating. For each

scenario, additional information can be displayed, such as a description and the leaderboard, which shows the users who have scored the highest points on that specific scenario. The page also includes several filtering options, allowing users to display only certain types of scenarios, filter by difficulty, or search for a specific scenario by name.

The scenarios page can be seen in figure 4.2. As you can see, there are 4 different difficulties: easy, medium, hard, and expert. There are also 15 possible scenario types, and each scenario can combine several types. The scenario categories come from a report by the Joint Research Center of the European Commission[28]. The aim of this report is to provide a coherent and comprehensive taxonomy of cybersecurity skills. I found this taxonomy to be the most appropriate because, unlike others I've found in the literature, it doesn't focus solely on technical skills, but encompasses a much broader range. It allows for a wider variety of scenarios, which is the point of my second research question. An example of a non-technical scenario that could be classified using this taxonomy is a quiz on legal issues related to cybersecurity or on more ethical issues related to data privacy.

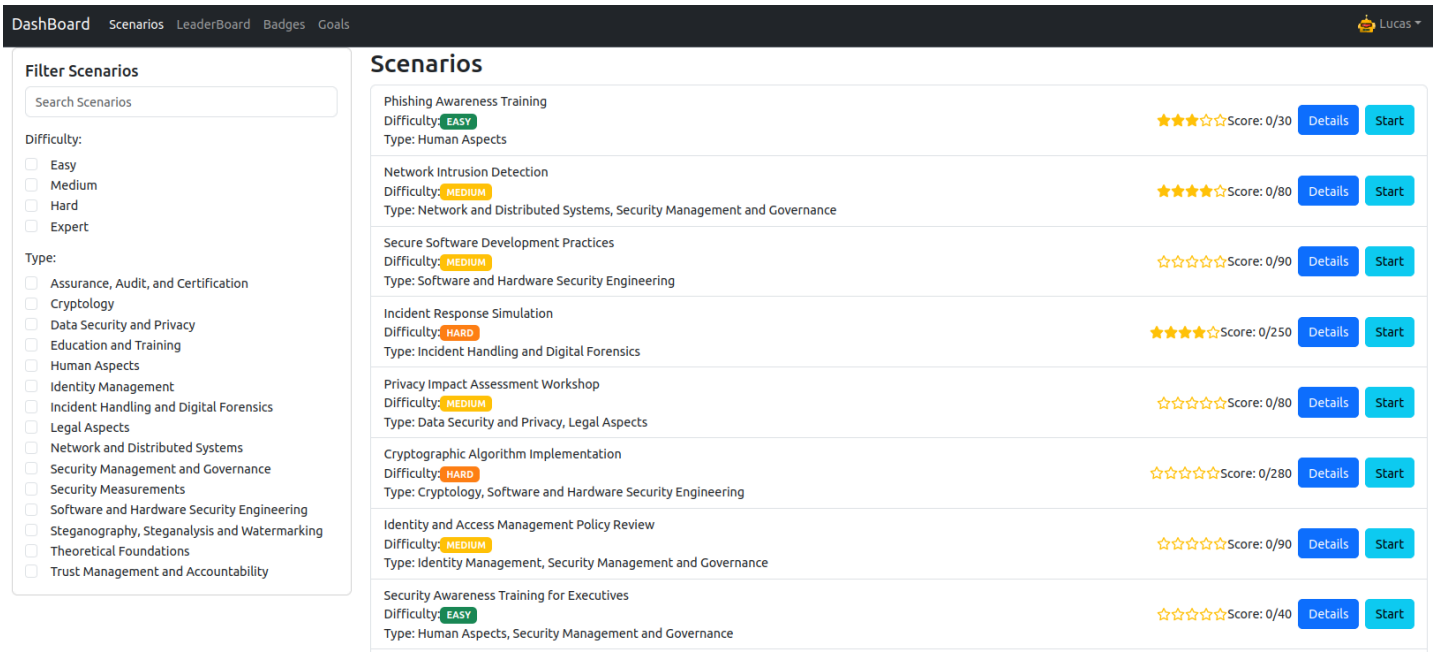


Figure 4.2: Scenarios page

The platform currently offers two playable scenarios, both centered on web security. The first scenario focuses on Insecure Direct Object References (IDORs) and is built around a simple note management application. In this scenario, users learn to manipulate URL parameters to escalate privileges, a technique

commonly associated with IDOR vulnerabilities. This scenario highlights the critical importance of securing web applications against unauthorized access. The second scenario, which is more challenging, provides less detailed hints, thereby encouraging users to engage in deeper problem-solving. It begins with the use of Nmap, a network reconnaissance tool, to identify the versions and technologies used by a web server. The scenario then advances to an SQL injection task, where users must bypass a login form, simulating a real-world attack vector.

The 3 other pages accessible via the top menu are simply the leaderboard, showing the general ranking of the platform's best players, sorted by the number of points obtained on the scenarios. The usernames of both the general leaderboard and the scenario leaderboards can be clicked to access the user's profile summary. The badges page simply displays all the badges on the platform and gives access to the conditions for obtaining badges. The last tab is the goals page, which displays the complete list of goals that the user can set and that will determine their learning path. They can view their current goal, modify it, and access a description of each goal via this page.

Finally, the last page where the user will spend a lot of time is the visualization page. This is accessed by clicking on a scenario's start button. If you haven't played the scenario yet, you'll see a large empty area on the left and a side panel on the right. The side panel allows you to do 3 things: start/stop the container responsible for the scenario, ask for a hint, or submit a flag that has just been found. Each time the user submits a flag, the empty zone is updated, gradually displaying a graph representing the path of the user inside the scenario. Each node in the graph displays a title, a description of the action, and a color: blue for starting actions, red for actions that mark the end of the scenario, and black for the others. The side panel also contains important information for the user: the points currently obtained on the scenario, a description, information to start, which is generated when the container is launched, and finally the lists of hints received and actions done. Each hint requested is added to the list of received hints, and when the corresponding action is completed, it is moved below the completed action in the actions done list, preventing the user from concentrating on the wrong hints but still allowing them to access them again if they wish to replay the scenario. Figure 4.3 shows an example of a scenario in progress.

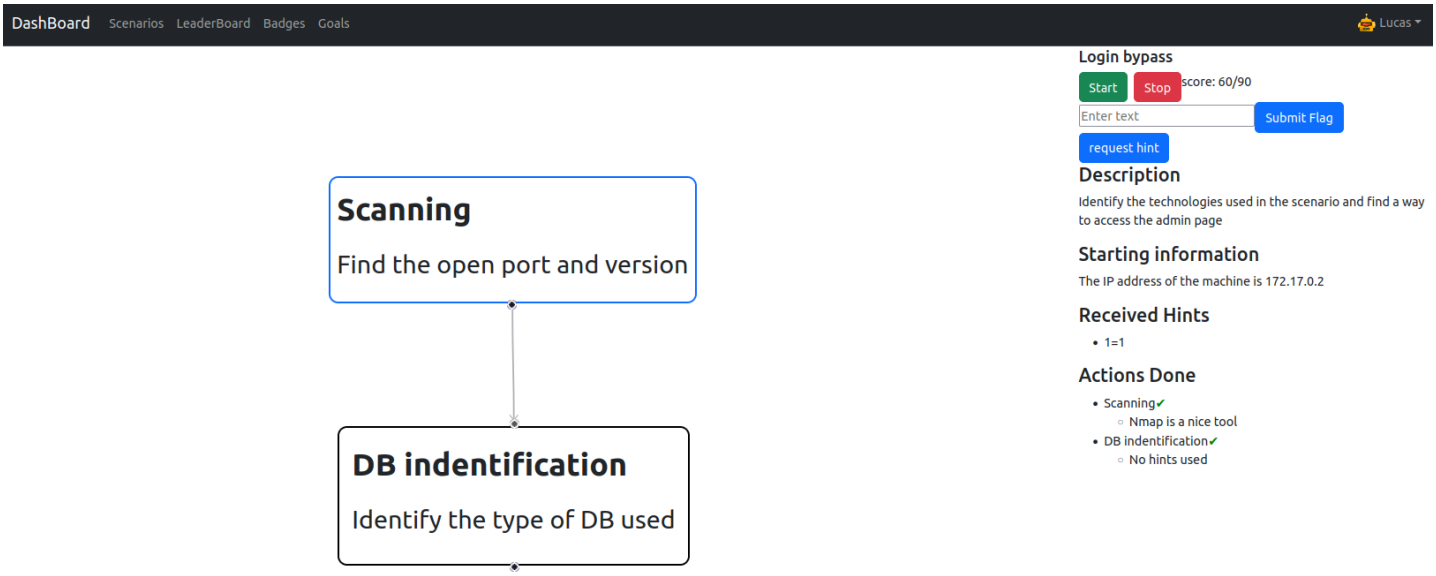


Figure 4.3: Visualization page

2 Architecture

This section takes a deep dive into the architecture, technologies, and main libraries used during the project. As mentioned in section 1 of this chapter, this is a web application. This choice was made to enable the quick production of a working prototype and easily integrate gamification mechanisms. The overall architecture, shown in figure 4.4, is therefore based on a fairly basic structure divided into 3 main components: the backend, the frontend, and the cyber ranges. The backend is made up of several sub-components, which will be described in detail later in this section. However, from an external point of view, only a RESTful application programming interface (API) is exposed to communicate with the user via the frontend. The API also enables the various cyber ranges to be deployed and stopped when a user needs them.

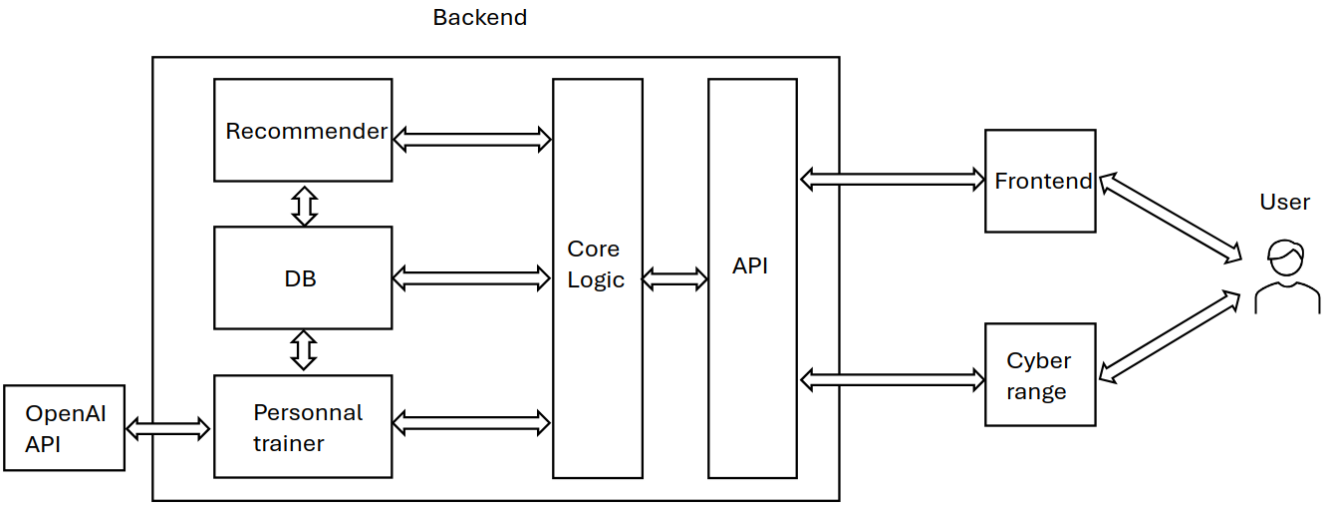


Figure 4.4: Architecture

2.1 Frontend

The frontend of the platform is designed to deliver a fluid, interactive user experience. To achieve this goal, I have selected several cutting-edge technologies, each of which brings specific benefits.

React[37] is the main library for the development of the user interface. React is renowned for its flexibility and performance, thanks to its virtual DOM system, which enables the interface to be updated quickly and efficiently. What's more, the composability of React components facilitates code reuse and improves application maintainability. Although React isn't the most comprehensive library in terms of functionality, it allows easy integration of other libraries and tools, making it a highly modular choice. What's more, its large community offers support and a lot of resources to facilitate development.

To style the application, I rely on Bootstrap[4]. This CSS framework provides a comprehensive range of prefabricated components and responsive styles, enabling rapid and consistent development. Bootstrap facilitates the creation of adaptive layouts that work well on a variety of devices, ensuring a smooth and enjoyable user experience.

Avatars and badges, essential elements in the personalization and gamification of the platform, are generated using Dicebear[9]. Dicebear is a library for creating unique, customizable avatars using a simple API. This solution boosts user engagement by offering them a playful way of representing their profile and successes on the platform.

For scenario visualization, I use ReactFlow[38]. This open-source tool helps to create and customize interactive and dynamic graphs. React Flow is particularly effective for managing complex structures, while offering excellent performance and ease of use.

2.2 API

The API is built with Python Flask[12], a lightweight framework that enables web applications to be developed quickly and with great flexibility. Flask is particularly well suited to this type of application due to its simplicity and ability to extend with extensions for features such as security, session management when needed, and more. The RESTful API plays a crucial role in the platform's architecture. Designed to be stateless, it doesn't record session information, which lightens the load on the server and improves application scalability. This approach also favors architecture modularity, enabling, for example, the frontend to be replaced or updated without impacting other system components.

2.2.1 Resources

In terms of API structure, there are 4 main types of resources to interact with: users, results, scenarios, and messages. The first one is simply for creating a profile, logging in, and accessing/modifying information. The second is mainly used when playing a scenario, to interact with the results resource and thus request hints, submit flags, or simply retrieve results from a scenario. You can also submit a rating when you finish a scenario. The scenario resource simply lets you access scenario information or launch scenario deployment for a regular user, but admins can also add or modify scenarios. Finally, the last main resource is messages, which are exchanged with the personal trainer. The user can add a message to the discussion or retrieve its conversation history. You can find the entire API in table 4.1. The last 4 sections of the table are read-only, allowing the user to access all badges, goals, and the leaderboard with just one call to the API. The Dashboard resource, also read-only, was introduced to reduce the load on the server. This is the central page of the platform and therefore the most frequently used; however, it contains a lot of different information, which would require several calls to the API to retrieve all the necessary information without this addition.

Note that some API responses can return a notifications field in the JSON response, allowing notifications to be managed without a session stored on the server. This is useful, for example, for notifying users that they have obtained a new badge or avatar after submitting a new flag.

2.2.2 Authentication

User authentication is provided by JSON web tokens (JWT). These tokens are returned in response to login or register requests, and the frontend must then include the token in each request that requires higher levels of access. These tokens are signed and contain the user's ID, which ensures that they are not modified and that I can easily recover a user's identity without having to maintain a session.

There are 3 different levels of access in the application, the lowest allowing access to public data such as the leaderboard, the list of scenarios, goals, or badges. The higher access level is for users who are registered on the platform, and gives access to personal information such as their profile, scenario progress, recommendations, etc. And finally, the last level is for admins, who can modify or add scenarios.

Endpoint	Method	Description
Users		
`/api/register`	POST	Register a new user.
`/api/login`	POST	User login.
`/api/users/{ID}`	GET	Retrieve information of a specific user by ID. Requires JWT.
`/api/users`	GET	Retrieve own user information. Requires JWT.
`/api/users`	PUT	Update user information. Requires JWT.
Scenarios		
`/api/scenarios`	GET	Retrieve a list of all scenarios.
`/api/scenarios`	PUT	Update a scenario. Only accessible by admins. Requires JWT.
`/api/scenarios`	POST	Add a new scenario using the GUI. Only accessible by admins. Requires JWT.
`/api/scenarios/dev`	POST	Add a new scenario with simpler JSON format. Only accessible by admins. Requires JWT.
`/api/scenarios/start/{ID}`	GET	Start a specific scenario. Requires JWT.
`/api/scenarios/stop/{ID}`	GET	Stop a specific scenario.
`/api/scenarios/list`	GET	Retrieve a list of scenarios by their IDs.
Results		
`/api/results`	POST	Submit a flag. Requires JWT.
`/api/results/hint`	POST	Request a hint. Requires JWT.
`/api/results`	GET	Retrieve all results of a user. Requires JWT.
`/api/results/rating`	POST	Add a rating to a scenario. Requires JWT.
Messages		
`/api/messages`	GET	Get the conversation history. Requires JWT
`/api/messages`	POST	Send a message to the personal trainer. Requires JWT
Leaderboard		
`/api/leaderboard`	GET	Retrieve the global leaderboard.
`/api/leaderboard/{ID}`	GET	Retrieve the leaderboard of a specific scenario.
Dashboard		
`/api/dashboard`	GET	Retrieve the dashboard of a user. Requires JWT.
Goals		
`/api/goals`	GET	Retrieve all goals.
Badges		
`/api/badges`	GET	Retrieve all badges.

Table 4.1: API Endpoints

2.3 Database

For the database, I chose MongoDB[27], a NoSQL database. Its ability to store JSON documents is particularly useful for managing scenarios that contain a lot of nested structures. Its flexibility is also very useful for objects such as the conditions for obtaining badges, which can have different structures depending on the type of badge. Storing JSON also facilitates the use of the various libraries used that manipulate objects that I can directly store in the database.

Database collections showed in figure 4.5 follow a structure similar to that of the API. The checkpoints_chat collection corresponds to the API's message resource. The latter stores more information than just messages, to facilitate communication with the LangGraph library, which manages user interactions with the chatbot. Despite the correspondence with the API, not all collection fields can be modified via the API. Take, for example, the user collection, which contains fields such as the list of unlocked badges, the score, or the recommendations, which cannot be modified directly by the user, but are updated automatically in the backend when the user interacts with the platform.

In addition to these 6 main collections, I've also added 2 views to provide quicker access to certain information. The first view is the Leaderboard, which maintains an ordered list of users by score, and the second is ScenarioStats, which maintains statistics used by the progress indicators. For each level of difficulty, and each type of scenario, it contains the total number of scenarios and the total amount of points.

2.4 Core Logic

This component, also developed in Python, contains most of the application's logic. Once requests have been authenticated and authorized by the API, they will be passed to it, and it will be responsible for building the response. Its role is to query the recommendation or personal trainer components when necessary. It will also be able to retrieve and modify information in the database, mainly data related to the user and the results. It will, for example, check the validity of flags, add badges or new avatars to the user if necessary, and include a notification in the response. It also manages hints and their attribution to finished actions in result objects, as well as the list of scenarios recently played by the user. A final important function is the creation and initialization of new users, as well as the verification of password hashes when the user needs to log in again after their JWT has expired.

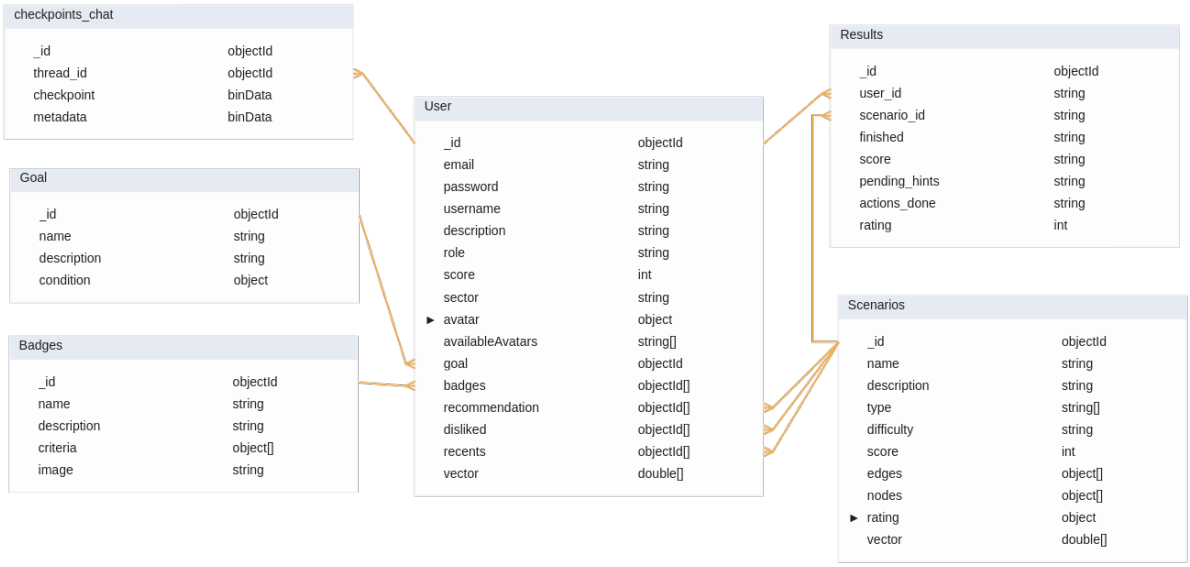


Figure 4.5: Database Schema

2.5 Cyber Ranges

The primary feature of the platform is its cyber ranges, which enable scenarios to be played out in realistic environments. These cyber ranges have been developed using Docker[10], a containerization technology. Although this choice of architecture is simpler compared to more complex solutions found in the literature, it is intentionally designed this way to align with the project's focus on gamification and user guidance.

The simplicity of Docker offers several key advantages that make it well-suited for this project. Firstly, the prototype needs to run locally on my machine, and Docker's lightweight nature allows the entire project to operate efficiently without overwhelming system resources. This lightweight approach also facilitates potential scaling of the application, enabling multiple containers to run simultaneously at a very low cost, as they only include the necessary components. Additionally, Docker's simplicity makes it easier to develop, store, and deploy scenarios through images, which can even be hosted on platforms like DockerHub.

Despite its straightforward nature, Docker is versatile enough to simulate complex network architectures by combining multiple containers using tools like Docker Compose. This flexibility supports the project's educational goals by providing realistic scenarios without the overhead of more complicated infrastructures.

One limitation encountered with Docker is the challenge of incorporating automated progress tracking within the scenarios. Since the project aims to offer a wide variety of scenarios, any monitoring solution would need to be customized for each type of scenario. To address this, I implemented a flag system where users manually submit their progress, eliminating the need for direct interaction between the container and the backend. The only interactions are the launching and stopping of containers, managed by scripts that handle connection information and communicate it to the user.

By opting for this simpler architecture, the project remains accessible and manageable while focusing on enhancing the learning experience through gamification and guidance, rather than on the complexity of the underlying infrastructure.

2.6 Guidance Components

The architecture also contains 2 guidance-related components that I'll briefly introduce here. These are the recommendation system and the personal trainer, which are covered in greater detail in sections 4 and 5 of this chapter.

The recommendation system is responsible for recommendation generation and managing user and scenario embeddings. It relies mainly on the Scikit-learn[34] library for pipeline implementation and data preprocessing. But also on the Gensim

library, which provides an algorithm called Doc2vec[52] for generating my scenario embeddings.

The personal trainer is simply a chatbot designed to help the user learn. The aim of the component is to process the user's messages and generate a response. To do this, I mainly use 2 libraries. LangChain[18], which facilitates the development of applications based on LLMs. It makes it easy to define templates, create query and response strings, structure outputs, etc. The second is LangGraph[19], a LangChain extension that lets you define agents and model their behavior via a state diagram. Inference is performed using the OpenAI API[32]. The first version used the Llama model in its uncensored version. Llama had several advantages, the first being the uncensored version, which is not available via the OpenAI API, since by default the model will refuse to answer many cybersecurity-related questions due to its usage policy. The second advantage is simply data privacy: the model is fed with information from the user to enable personalized interaction. Unfortunately, my configuration doesn't allow the model to run smoothly alongside the rest of the project. However, the libraries abstract from the model used, so it's relatively simple to switch from one to the other with a few adjustments.

3 Gamification

This section describes in detail each gamification element integrated into the platform and selected on the basis of my literature review. I've used a large number of elements, as one of the reasons for negative results is that there are too few of them, but I've tried to maintain consistency in their integration. Another point that appeared essential to me during my analysis was not to make any gamification element mandatory; a user who isn't interested can simply go to the scenarios page and start training.

3.1 Points

The scoring system is the central point of my gamification, as most of the other gameplay elements will depend on the points you earn when you perform actions in a scenario. It is strongly inspired by the scoring system proposed by Diakoumakos et al.[8]. On the platform, points are used as an indicator of difficulty: the more points an action earns, the more difficult it is. The 4 levels of scenario difficulty (easy, medium, hard, expert) are associated with a point range, and the scenario creator can choose a value from the range when creating the scenario. This makes it possible to create a more precise difficulty scale. For example, the point range for an easy scenario is 20 to 50. Once the number of points has been determined,

it's time to define all the actions that can be carried out in the scenario, and to distribute all the points among the different actions, always keeping in mind that the more difficult an action is, the more points it must earn. Each action is associated with its own flag, and points are earned when the flag is submitted, thus laying the foundations of the reward system.

Each action, in addition to the number of points and the flag, can have a certain number of hints. The use of a hint can introduce a penalty, also defined by the scenario creator. They'll choose a percentage penalty to be applied, which takes effect when all the action's hints are used. If the user does not use all the hints, the penalty is applied in proportion to the amount used. For example, if an action earns 100 points and has two hints with a penalty of 50%, a user who submits the flag using only one hint will suffer a penalty of 25% and will receive 75 points.

3.2 Badges

Another gamification element that takes the form of a reward is the badges. There are currently 21 on the platform, designed with the Dicebear library and generated on the basis of a seed. Each badge has a name, a description, a seed, and a set of conditions that must all be met for the badge to be awarded. These conditions are stored in the criteria field of the Badges collection in figure 4.5. For the moment, there are 3 types of conditions for badges, whose structure depends on the type of condition, but they all depend on the number of points obtained. The first is based on scenario type and contains a type and score required on that type of scenario; the second is based on difficulty and contains a difficulty and score to be achieved with scenarios of that difficulty. Finally, the last type of condition is based on the total score the user has achieved on the platform and requires only a total score value. The first field of each condition is the type, which lets the backend know which type of conditions it needs to check.

3.3 Progress Tracking

Progress tracking is a vital part of the gamification strategy implemented on this platform. This offers learners a clear and motivating view of their achievements and areas to be improved. The system employs three forms of tracking progress, each tailored to offer comprehensive insights into three different dimensions of user performance and progress.

The first type of progress tracking focuses on the global progress of users across the entire platform. This metric aggregates user performance data, showcasing the total points accumulated by type and difficulty of scenarios. For a more

engaging visual representation, this progress is illustrated through both linear and circular progress bars. Through these visuals, the user has an immediate appreciation of their general progress that fosters a sense of accomplishment and greater engagement with the platform.

The second type of tracked progress is related to the users' individual goals within the platform. The tracking that aligns with these goals is in the form of a radar chart, which provides a multi-faceted view of how the user is performing across different types of scenarios tied to their personalized goals. By scoring against certain types of scenarios, the radar chart lets the user know their strength and weakness areas for their defined objectives. This targeted feedback is really important to let users know what to do in order to improve.

The third kind of progress tracking is the visualization of scenario flow as scenarios are being played. This involves a dynamic graph that updates in real-time as users progress through a scenario and submit flags. This graph will not only help the user to understand the current position within a scenario but also provide immediate feedback on their actions and decisions.

3.4 Social Interactions

Social interactions are a significant aspect of gamification that makes the user engaged and motivated through the experience of community and competition. There are two main kinds of social interaction on the platform: leaderboards and personalized profiles.

Leaderboards are a very important feature because they incite competition among users and provide motivation for users to strive for performance improvement. The leaderboards are further divided into two categories:

- **Global Leaderboard**: All users are ranked across the platform based on the total points that they've earned, showing all top performers and encouraging a platform-wide competition.
- **Scenario Leaderboard**: Each scenario will have its own leaderboard based on the points earned in a given scenario. This type of leaderboard fosters targeted competition and allows users to see how they perform in particular scenarios.

By clicking on a username within the leaderboards, users can view other participants' profile summaries. This feature enables users to benchmark their progress against their peers.

Personalized profiles give users an individual identity on the platform, increasing their involvement by allowing them to express themselves. They include the user's username, total points earned, company name, brief description, and a unique avatar.

Users can fully customize avatars, a key aspect of personalization. As users continue to earn points, they unlock new kinds of avatars, incentivizing them to remain active. Currently, there are three different types of avatars, each with various customization options. These avatars not only allow users to showcase their uniqueness but also add a fun and interactive element to the platform.

3.5 Goals and Guidance

Goals are at the core of a guidance system for the platform. Each goal was designed to help users achieve specific cybersecurity skills that are relevant to their job roles. The goals are labeled by job names, making them not only clear but also relatable for any trainee who will use the system to attain proficiency in their area of operation. Each goal is composed of five types of scenarios and a required score. The progress towards these goals is visually represented using the radar chart on the dashboard, allowing users to track their advancement comprehensively. Up to now, the system has six goals, mainly technical ones within cybersecurity. However, the framework is flexible enough to incorporate non-technical job roles, recognizing that cybersecurity awareness is essential across all sectors. For instance, the "Student" goal addresses the vulnerabilities that students might face and the good practices they should adopt regarding the access level they have in university infrastructure.

Guidance on the platform is conducted through two main components: the recommender system and the chatbot personal trainer. These elements are crucial in ensuring a smooth and adaptive learning experience. As identified in the literature review, proper guidance helps mitigate frustration and anxiety by aligning scenario difficulty with the user's skill level. It also prevents them from being stuck on challenging scenarios for too long.

The recommender system plays a pivotal role by using the goals as starting points. It dynamically adjusts recommendations based on the user's interactions with the scenarios and their ratings, ensuring that the suggestions are tailored to the user's evolving profile. This system provides guidance outside the scenarios, helping users choose the next steps in their learning journey.

The chatbot personal trainer, on the other hand, assists users within the scenarios. It explains general ideas in cybersecurity, answers user questions, and at the same time, motivates them. Additionally, it introduces recommendations that can be considered as a form of challenges or quests, adding a gamified element to

the training.

The implementation details of these guidance components are explained in the following sections (4, 5). Together, they push the boundaries of interactive and personalized cybersecurity training, ensuring that users receive the necessary support to develop their skills effectively and efficiently.

4 Recommender System

As discussed in the previous sections, it is crucial that trainees engage with scenarios whose difficulty is appropriately matched to their skill level. To achieve this, the integration of an effective recommendation system became a logical and necessary component of the platform.

The pipeline of the recommender system 4.6 can be conceptualized as a function that accepts a user profile and a list of all available scenarios as input, and subsequently returns an ordered list of the most suitable scenarios for that user. The first stage in this process is rule-based filtering, which, as its name suggests, eliminates scenarios that do not meet certain predefined criteria. Following this, the pipeline diverges into two distinct components: content-based filtering and collaborative filtering. Each component processes the user profile and the filtered scenario list to generate a ranked list of recommendations.The final stage of the pipeline, the Aggregator, is designed to combine the rankings from the previous components into a cohesive and optimal recommendation list.

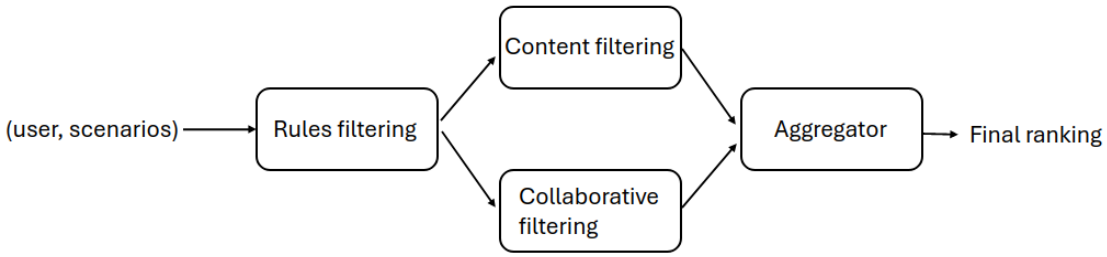


Figure 4.6: Recommender pipeline

However, it is important to note that, due to time constraints and the current lack of user data on the platform, the collaborative filtering component has not been implemented. Consequently, the Aggregator component has also been omitted, as its function relies on the output of both filtering methods. Despite this, the pipeline is presented in its entirety because the decision to pursue a hybrid recommendation approach can significantly influence the design of other components. Moreover,

the platform already integrates a rating system, which will facilitate the rapid incorporation of the collaborative filtering component when sufficient data becomes available.

4.1 Rule-based filtering

The rule-based filtering component of the recommender system functions by applying a set of predefined rules that are aligned with the user's profile. By applying these rules at an early stage, the rule-based filtering significantly reduces the number of scenarios that need to be processed in the subsequent steps, thereby enhancing the overall performance and efficiency of the system.

The first rule implemented within this component is the removal of any scenarios that the user has already completed. This prevents redundancy and ensures that the user is continuously challenged by new content. As discussed in the gamification section3, each goal that a user can select is associated with several categories of scenarios. Therefore, the second rule filters out any scenarios that do not fall within these relevant categories, aligning the training experience with the user's chosen objectives.

Finally, if a user expresses dissatisfaction with a specific recommendation, they can communicate this to their personal trainer, who can then exclude the scenario from future recommendations, effectively filtering it out at this stage.

4.2 Content-based filtering

Content-based filtering is a recommendation method that utilizes the characteristics of scenarios to generate suggestions. Each scenario is represented by a vector of features, and similarly, a feature vector is created for the user to represent their ideal scenario based on their profile. By calculating the similarity between these vectors, the system can identify the scenario most closely aligned with the user's preferences. The initial step in this process involves analyzing and processing the data within each scenario to construct a feature vector that encapsulates the most relevant information. The subsequent steps involve creating the user's initial feature vector, updating it as they interact with the platform, and performing similarity measurements to identify the most suitable scenario. These steps will be detailed in the following subsections.

4.2.1 Scenario vector

The first type of data incorporated into the scenario vector relates to difficulty. Each scenario contains both a difficulty field and a score field. However, only the

score is retained in the vector, as each difficulty level corresponds to a specific score range, making the inclusion of the difficulty field redundant.
The second type of data added to the vector is the scenario's category information. Since a scenario can belong to multiple categories, a variation of one-hot encoding, called multi-label binarization, is used. This technique creates a vector where each component represents a category, assigning a value of 1 to the categories present in the scenario, and 0 to those absent.

The third type of data pertains to the graph structure and scenario visualization. To retain essential information about the scenario's structure while keeping the representation compact, a set of graph features is included in the vector. The key features identified for capturing the scenario's complexity are the number of nodes, the number of edges, the maximum degree of a node, the maximum node score, the minimum node score, and the average node score.

The final and most complex type of data is the textual information within the scenario. Given that the platform's categories are broad and general, they do not precisely define a scenario's classification. Therefore, leveraging the textual content is crucial for defining the scenario's topic, which can be considered a sub-category. This text includes the scenario's name, description, and the details and hints for each action. To effectively process this textual data, the Doc2Vec model is employed. Doc2Vec[20] is an extension of the Word2Vec[26] model, designed to generate vector representations of entire documents, rather than individual words. It assigns a unique vector to each document, allowing the model to capture the semantic and contextual meaning of the text as a whole. This capability makes Doc2Vec particularly well-suited for classification tasks and similarity measurements, which are essential in recommender systems that handle textual information([29][14]. Given the technical nature of the scenarios, the training data for this model consists of guides on solving CTFs (Capture the Flag challenges)[1]. This approach enables the model to build a highly specialized vocabulary and familiarize itself with the relevant technologies and contexts.

The final step is to combine all these sub-vectors into a comprehensive vector representing the scenario. This vector is generated when the scenario is added or modified, and for performance reasons, it is stored in the database to avoid recalculating it each time a recommendation is needed.

[1] `https://www.hackingarticles.in/ctf-challenges-walkthrough/`

4.2.2 Initial User Vector

Having established the scenario data and the method for creating scenario vectors, we now turn to the construction of the user vector. For the similarity measure to function effectively, the user vector must mirror the scenario vector in terms of features and processing methods. In this section, I will revisit the four different vector data types and explain how each is populated in the user vector. The user vector is generated each time a user selects or changes their goal, as this choice is the primary factor influencing recommendations.

Regarding the difficulty feature, represented by the score, it is initialized as the average score of scenarios previously completed by the user that belong to at least one category related to their current goal. If no such scenarios exist, the score is set to 20, which is the minimum possible score for a scenario. A similar approach is applied to the graph features, as these are considered sub-features of difficulty. These features are also initialized to the average value, or to 0 if no relevant scenarios are found, as a value of 0 will not affect the cosine similarity calculation.

For the category features, I have deliberately chosen to enforce a uniform progression through the various categories associated with the user's goal. This approach is intended to mitigate the issue of over-specialization, which is discussed in the background material. By encouraging broader category engagement, the system ensures that users do not become overly focused on a narrow set of skills, which could limit their overall development.

In practical terms, if a user has a score of 0 in a category required for their goal, a value of 1 is assigned to the corresponding feature in the vector. Conversely, as the user's score in a category approaches the required value, the corresponding feature value is adjusted closer to 0.

Thus, if the completion percentage of a category i relative to the user's goal is denoted by P_i, the corresponding feature F_i in the vector is calculated as in Equation 4.1

$$F_i = 1 - P_i \tag{4.1}$$

where P_i ranges from 0 to 1. This method ensures that categories in which the user has the lowest scores are prioritized.

4.2.3 User Vector Update

Once the user vector is established, it must be updated as necessary. Two actions require updating the vector: when the user completes a scenario and when the user rates a scenario.

Completing a Scenario When a user completes a scenario, the vector is updated using two key concepts: the *learning rate* (LR) and the *range* (R). The learning rate controls the speed of score progression, while the range determines whether the score of the completed scenario (SS) is close enough to the user's current score (CS) to warrant an update. The update process is governed by the following rules:

If the scenario score falls within a specific range around the user's current score, the user's score is increased by a factor that includes the learning rate. However, if the scenario score is significantly higher than this range, the score is adjusted more aggressively to accelerate progress. On the other hand, if the scenario score is below this range, no update is made to the user's score.

The user vector update is mathematically expressed as in Equation4.2

$$\text{New\_Score} = \begin{cases} CS \times (1 + LR) & \text{if } SS \in [CS \times (1 - R), CS \times (1 + R)] \\ CS \times (1 + LR \times \frac{SS}{CS}) & \text{if } SS > CS \times (1 + R) \\ CS & \text{if } SS < CS \times (1 - R) \end{cases} \tag{4.2}$$

Additionally, the category features are recalculated in the same manner as during the initial vector creation to keep providing uniform progression through the categories.

Rating a Scenario The user vector is also updated when the user rates a scenario. This process focuses on adjusting the textual (TF_i) and graph features (GF_i) based on the user's preferences. The rating, which is a score between 1 and 5, determines whether these features are adjusted positively or negatively relative to the scenario vector, thereby influencing the similarity measure used in subsequent recommendation steps.

For ratings of 1 or 2, which indicate a "dislike," the similarity between the user vector and the scenario vector is reduced by applying a negative adjustment. A rating of 3 is considered "neutral," resulting in no changes to the user vector. For ratings of 4 or 5, which indicate a "like," the similarity is enhanced by applying a positive adjustment.

The direction of this adjustment is determined by the *Sign Modifier* (SM), where negative ratings apply a negative impact and positive ratings apply a positive impact. The magnitude of this adjustment is controlled by a fixed adjustment value (δ), which starts with small values to reflect that changes will be more pronounced with extreme ratings. However, these δ values should be considered as initial hyperparameters that may be adjusted as the system collects more user ratings and is able to be evaluated and fine-tuned.

The *Sign Modifier* is defined as follows:

$$SM = \begin{cases} -1 & \text{if rating} \in \{1, 2\} \\ 0 & \text{if rating} = 3 \\ 1 & \text{if rating} \in \{4, 5\} \end{cases} \tag{4.3}$$

The initial values for δ are set as follows:

$$\delta = \begin{cases} 0.1 & \text{if rating} \in \{2, 4\} \\ 0.2 & \text{if rating} \in \{1, 5\} \end{cases} \tag{4.4}$$

The updates to the graph and text features of the user vector are then calculated using equations 4.6 and 4.6

$$\text{New\_GF}_i = \text{Current\_GF}_i + SM \cdot \delta \cdot (\text{Scenario\_GF}_i - \text{Current\_GF}_i) \tag{4.5}$$

$$\text{New\_TF}_i = \text{Current\_TF}_i + SM \cdot \delta \cdot (\text{Scenario\_TF}_i - \text{Current\_TF}_i) \tag{4.6}$$

4.2.4 Similarity Measure

The final step in the recommendation system is to measure the similarity between the user vector and the scenario vectors. This similarity measure is crucial for identifying which scenarios are most relevant to the user based on their profile. Cosine similarity was selected as the metric for comparing vectors due to its robustness in high-dimensional spaces and its ability to handle sparse vectors effectively. Unlike other similarity measures, cosine similarity calculates the cosine of the angle between vectors, focusing on their direction rather than magnitude. This characteristic is ideal for our recommendation system, as it emphasizes the patterns shared between user preferences and scenario features. By disregarding magnitude, cosine similarity ensures that comparisons are based solely on the presence and relative importance of features, making it a fair and effective metric.

To ensure all features contribute proportionately to the similarity measure, min-max scaling is applied to the vectors, transforming features to a fixed range, typically [0, 1]. This scaling is particularly important as it allows for more controlled application of feature weighting in subsequent steps, ensuring that no single feature dominates the similarity calculation. Additionally, by restricting all feature values to a non-negative range, min-max scaling prevents the introduction of negative values that could otherwise distort the effect of feature weighting.

Following scaling, feature weighting[6] is applied to prioritize certain features, specifically emphasizing difficulty and category features, which are critical for

scenario recommendation. This weighting is achieved by multiplying each feature by a weight factor, thereby altering the vector's direction in the feature space. The weighted vector $\mathbf{v}_{\text{weighted}}$ is computed as:

$$\mathbf{v}_{\text{weighted}} = \mathbf{v} \circ \mathbf{w}$$

where $\circ$ denotes element-wise multiplication, and $\mathbf{w}$ is the weighting vector. For instance, if difficulty and category features are considered twice as important as others, their weights are set to 2. As for the value of equation 4.4 this is considered as hyperparameter of the system and will need to be fine-tuned to find the trade of between favoring difficulty learning curve and user's preference.

After preprocessing, the similarity between the user vector $\mathbf{u}$ and a scenario vector $\mathbf{s}$ is calculated using the cosine similarity formula:

$$\text{cosine\_similarity}(\mathbf{u}, \mathbf{s}) = \frac{\mathbf{u} \cdot \mathbf{s}}{\|\mathbf{u}\| \|\mathbf{s}\|}$$

where $\cdot$ denotes the dot product of the vectors, and $\|\mathbf{u}\|$ and $\|\mathbf{s}\|$ represent the Euclidean norms of $\mathbf{u}$ and $\mathbf{s}$, respectively.

In summary, the combination of cosine similarity with the preprocessing steps of min-max scaling and feature weighting enables the recommendation system to effectively match users with scenarios that align with their current level and interests, while maintaining control over critical features for educational purposes.

4.3 Collaborative Filtering

Collaborative filtering is a recommendation technique that leverages the preferences and behaviors of multiple users to suggest relevant items to individuals. Unlike content-based filtering, which relies on the attributes of items, collaborative filtering identifies patterns among user interactions to make predictions. Based on my literature review, a model-based approach is the most effective when sufficient data is available. This approach involves using historical data to train a predictive model that can generalize user preferences and make recommendations.

In the system, the vectors representing a user's profile at specific points in time can be effectively used to train the prediction model. As more user data becomes available, the model will learn to identify similarities between users and predict scenarios that a particular user is likely to find engaging based on the preferences of others with similar profiles.

4.4 Aggregator

The final step in the recommendation pipeline is to aggregate the outputs of the content-based and collaborative filtering methods into a single ranked list. The

aggregator could be designed to assign weights to each method's recommendations to construct an overall ranking of scenarios. These weights can be dynamically adjusted based on the amount of data available in the system, with a greater emphasis on collaborative filtering as the dataset grows larger. This flexible approach not only optimizes the accuracy of recommendations but also allows for the seamless integration of additional methods as the system evolves. However, similarly to the collaborative filtering component, the aggregator has not yet been implemented due to the current lack of data and has been left for future work.

5 Personal Trainer

This section provides a detailed overview of the functionality of the personal trainer. The personal trainer is implemented as a chatbot designed to address both general cybersecurity inquiries and questions specific to a scenario. It is particularly valuable in guiding users who have exhausted all available hints within a scenario yet find themselves unable to progress. The chatbot enhances the user experience by offering relevant security information without requiring extensive documentation review, thereby reducing friction in accessing critical knowledge. It delivers precise answers tailored to the user's questions and the context of the scenario being played.

The chatbot leverages OpenAI's API for inference using a generative AI model, enabling it to respond effectively to user queries. Beyond traditional inference, recent AI models are equipped with function-calling capabilities, allowing the model to autonomously invoke specific tools as needed. When the model detects a requirement for an external tool, it can independently call a function, utilize its output, and incorporate it into the subsequent response generation.

On the platform, the chatbot is equipped with various tools that allow it to access scenario information, retrieve data from a database to enhance the reliability of its outputs, and modify the recommendations provided to the user. To ensure more precise control over the chatbot's behavior, LangGraph is employed to model its actions within a graph structure. This enables the system to, for instance, request user confirmation before modifying a recommendation.

5.1 Tools

Before delving into the modeling of the chatbot's behavior, it is essential to introduce the tools it utilizes. These tools are categorized into two main types: writing tools and reading tools.

The personal trainer is equipped with a single writing tool, which allows it to modify the current recommendation provided to the user by making a request to

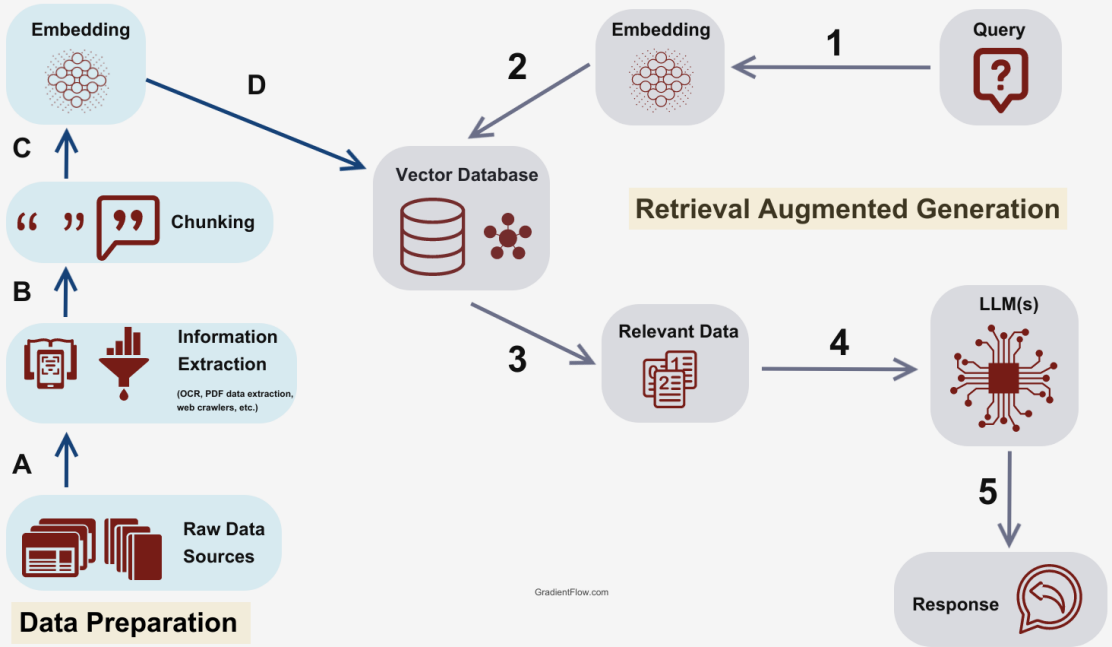


Figure 4.7: RAG Process. Source: [24]

the recommendation module. This capability is crucial for dynamically adapting the guidance offered based on evolving user interactions.

For reading operations, the trainer has two distinct tools designed to access the JSON representation of a scenario. These tools enable the chatbot to better understand the context of the user's questions by retrieving either the recommended scenario for the user or the most recently launched scenario from the user's scenario history.

Additionally, the chatbot is equipped with a retriever tool that enhances the reliability of its generated responses through the Retrieval-Augmented Generation (RAG) method. This approach involves augmenting the context of the AI's responses with information drawn from a trusted source. To implement this, I utilized OpenAI's model for generating text embeddings.

The content was sourced from PortSwigger Academy[2], a high-quality resource for explaining web vulnerabilities, which aligns with the focus of my demo scenarios, both of which involve web-based scenarios. The text from PortSwigger was scraped, split into smaller overlapping sections to ensure no information was lost during the segmentation, and then the embeddings for these text segments were computed and stored in a database alongside their associated text.

When a user asks a question that could potentially be answered by the PortSwig-

[2]`https://portswigger.net/web-security/all-topics`

ger content, an embedding of the question is generated. A similarity measure is then calculated between the question's embedding and the vectors in the database to quickly identify the most relevant segments that could assist in providing a response. The text corresponding to the most similar embedding is then incorporated into the context for generating the answer. The entire process is shown in figure 4.7.

The database is stored locally on the server and, while it currently uses PortSwigger content for the demonstration, it could be adapted to include a wiki specifically created for the platform. This wiki could contain documentation for all scenarios, making it a versatile resource. The segmentation of content into smaller parts allows for precise identification of relevant sections, thereby enabling the chatbot to redirect the user to the exact part of the wiki that contains the answer, thus minimizing the user's need to manually search for information.

5.2 Workflow

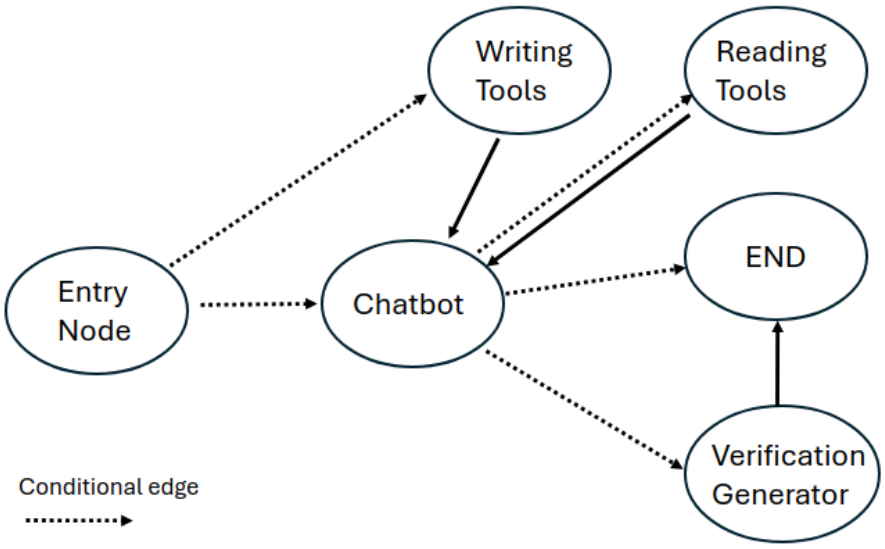


Figure 4.8: Chatbot Workflow

The workflow of the personal trainer chatbot is structured using LangGraph, a specialized library for creating stateful applications that operate based on predefined workflows. LangGraph supports persistence, reasoning cycles, and fine-grained control over operations, which are essential for maintaining consistent and reliable chatbot interactions. This library is built on LangChain, facilitating the integration of various transformer-based models and managing both message flow and prompt formatting.

The workflow graph consists of the following elements:

- **State**: A comprehensive record that includes all past messages, critical user details (such as username, user ID, current goal), and a field for pending tool calls.
- **Nodes**: These are functions that receive the current state, modify it, and pass it on to the next node.
- **Edges**: These are the pathways between nodes that can be either conditional or unconditional, dictating the transition between different states.

The state plays a crucial role in managing the chatbot's workflow. It ensures that all user interactions are preserved, and it supports user integration by requiring confirmation before the chatbot utilizes any writing tools. This distinction between writing and reading tools is foundational to the workflow's design, ensuring that actions with potential data implications are carefully controlled.

Each user message initiates the workflow by being added to the state's message list and processed by the entry node. The workflow progresses through various nodes until it reaches the END node, where the final response is returned to the user. Like nodes, which are implemented by functions, edges are also functions that take the current state as an argument, but instead of returning a new state, they return the name of the next node.

The core of the workflow is the chatbot node, linked to a GPT-4 mini model instance. Based on the context that is dynamically built with last messages, instructions, and tool responses, the chatbot may generate a direct response or initiate a tool call. If the call involves a writing tool, it is placed in the pending field of the state, instead of the message history, for user confirmation. In contrast, reading tool outputs are immediately added to the message history. When a tool call is pending, the workflow moves to the verification generator node, which produces a formatted message requesting the user's permission to proceed with data-modifying actions. This verification step is crucial for ensuring user control over writing tool usage.

The entry node checks for pending tool calls and checks the user's response, either confirming the action or discarding it and redirecting the workflow accordingly. If no tool calls are pending, the state remains unchanged, and control returns to the chatbot node. Tool nodes have unconditional edges leading back to the chatbot after their execution. This design prevents non-human-readable tool responses from being directly returned to the user. Instead, the chatbot interprets the tool output, constructs a human-readable response, or triggers additional tool calls if necessary. The workflow concludes upon generating a human-readable response, reaching the END node.

Each new step in the graph is recorded in the database in the chat_checkpoints collection of figure 4.5. The collection follows the structure defined by LangGraph

to include persistence in the application. The thread_id field refers to the user ID and is used to retrieve all the documents specific to the user during interactions. This is very useful for retrieving the history of messages and also for debugging the chatbot's behavior.

5.3 Messages

In the chatbot's workflow, there are three distinct types of messages managed by the graph: user messages, AI messages, and system messages.

User Messages are straightforward and consist of the direct inputs provided by the user during their interaction with the chatbot. These messages serve as the primary means through which users engage with the system.

AI Messages are the outputs produced by the chatbot in response to user inputs. These can either be human-readable messages intended to communicate with the user or calls to various tools as part of the chatbot's internal processes.

System Messages play a crucial role in guiding the behavior of the AI model. These messages are not visible to the user and are used internally to instruct the model on how to act in different scenarios. They often provide the model with foundational information and context, which do not always require a direct response from the model. For instance, when a user selects a goal and the chatbot is activated, the initial system message outlines the overall behavior expected of the chatbot. This message assigns the chatbot its role as a personal trainer, identifies the trainee, and clarifies that the questions posed will be within the context of a structured cybersecurity training exercise in a simulated environment. This is important to minimize the model's default censorship, which might otherwise prevent it from responding to questions related to cyber-attacks. Additionally, this initial message introduces the tools available to the chatbot.

System messages are also employed to transmit notifications to the model, prompting it to generate a response. For example, upon the creation of the initial recommendation, the chatbot will proactively engage with the user, encouraging them to try the suggested action. Similarly, when the user completes a recommended task, the system notifies the chatbot, prompting it to congratulate the user.

The implementation of these interactions relies on two core functions: one that sends messages to the graph and retrieves the AI-generated response, and another that retrieves the conversation history to load into the user's interface. System messages are sent from the backend via the first function. However, they are filtered out by the second function that retrieves the chat history to ensure that these internal instructions are not displayed in the front-end chat interface.

6 Scenario Integration

In order to accommodate the diverse needs of developers, the platform offers two distinct methods for adding scenarios. The first method is through a user-friendly graphical user interface (GUI), which is designed for the straightforward integration of individual scenarios. For those who require a more automated and scalable solution, the platform also provides the capability to add scenarios programmatically via the API. By offering both a GUI and API, the platform ensures flexibility and efficiency, catering to a wide range of user preferences and technical requirements.

6.1 GUI

The GUI is designed to facilitate the easy integration of individual scenarios, making it an ideal tool for users who prefer a visual and intuitive approach. This interface allows for a seamless setup process, where users start by setting the basic characteristics of the scenario, such as its name, description, difficulty, score, and types. To avoid the upload of external files to the server, the platform allows users to specify a Docker image available on Docker Hub during the initial setup phase. This makes it easy to store, deploy, and update images.

Once the basic scenario characteristics are defined, the GUI provides a drag-and-drop interface that simplifies the creation process. Users can easily and visually add and arrange various scenario actions by dragging them into the workspace, creating a clear visual representation of the scenario's structure. Customization is straightforward within the GUI, allowing users to click on each node to modify its properties, such as defining the flag value, allocating points, adding hints, and setting penalties. Additionally, actions can be linked together using the handles provided on each node, with the interface automatically managing the creation and configuration of edges, the assignment of IDs, and the formatting of objects. This ensures that the position and size of nodes are tracked in a format compatible with ReactFlow, streamlining the entire process and reducing the potential for errors.

The platform supports three types of nodes: `startingAction`, which marks the initial step of a scenario flow; `endingAction`, which means that the scenario is complete even if all other actions have not been executed; and `action` nodes, which serve as intermediate steps. Additionally, there are two types of logical gate nodes: `AND` and `OR`, which allow for the creation of complex scenarios with multiple resolution paths or prerequisite steps needed for subsequent actions.

While these nodes offer powerful tools for scenario design, it is the responsibility of the scenario creator to use them effectively, as the platform does not impose strict rules on their usage during scenario resolution. For instance, a user might submit as a first flag a node different from the `startingAction` or succeed in submitting a flag at a node following an `AND` gate without completing the preliminary steps.

The intent is not to restrict a user's ability to find alternative solutions or skip steps but to accommodate diverse problem-solving approaches. In such cases, when a user discovers a flag further down the scenario flow, the preceding nodes will become visible, allowing the user to potentially recover all points. However, logical gates play a crucial role in the overall visualization and hint management within a scenario. Hints are provided according to the resolution path chosen by the user, and when a logical gate is encountered, the system will trace back through the scenario flow to give hints for all necessary actions. This mechanism ensures that while flexibility is maintained, the logical structure of the scenario is preserved.

6.2 API

For developers who need to integrate multiple scenarios more efficiently, the platform offers an API-based approach. This method is particularly useful for users with administrative rights, allowing them to submit scenarios programmatically by sending a simple JSON file that represents the scenario structure.

When using the API, the same scenario characteristics that are defined in the initial setup phase through the GUI must also be specified. In addition to these basic characteristics, developers must include arrays of `nodes` and `edges` that define the scenario's structure. The API endpoint is designed to minimize the amount of information that needs to be provided manually; much of the necessary data is handled by the backend. However, the user must still manually manage `IDs` to define the graph's structure, as well as specify the position and size of the nodes.

Figure 4.9 shows each field that must be included in the JSON object sent to the API. In addition to `nodes` and `edges`, there are also `hints` objects that are embedded within the nodes. The `source` and `target` fields of the `edges`, as well as the `node_id` field of the `hints`, all reference the corresponding node `IDs`.

For the special case of logical gate nodes, only the `id` and `type` fields are required, along with the necessary information about the node's size and position. This reduces the complexity for these specific nodes, as they do not require the additional fields needed for action nodes.

It is important to note that the API will reject a submission if it does not comply with the same constraints enforced by the visual interface during scenario creation. Specifically, the `difficulty` and `type` fields must be selected from the authorized lists, and the total `score` must fall within the range allowed for the specified difficulty. Additionally, the sum of the scores assigned to individual nodes must equal the total score of the scenario. This ensures consistency and validity across scenarios, regardless of the integration method used.

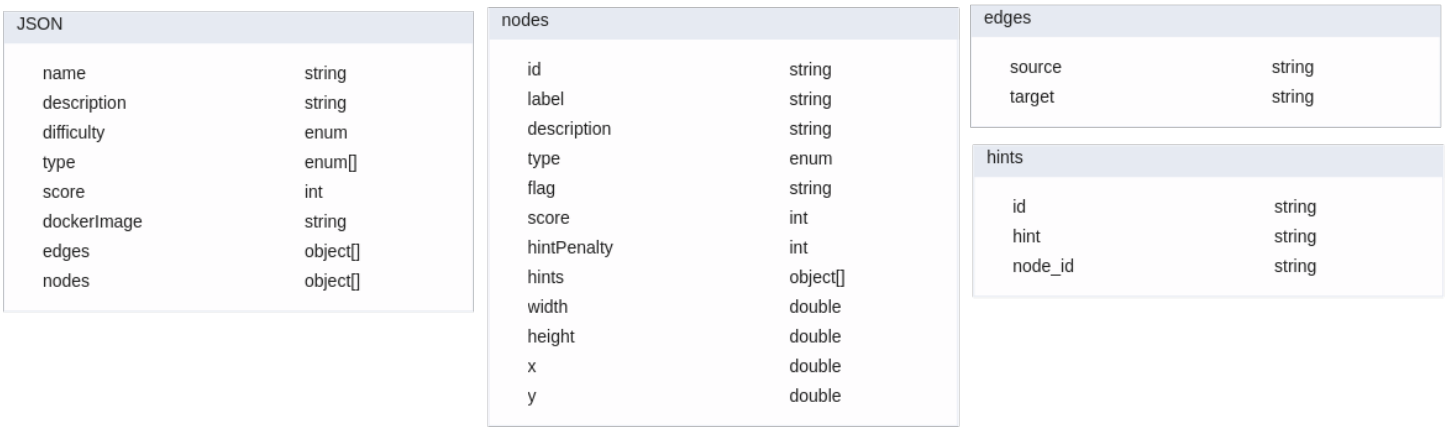


Figure 4.9: JSON structure of a scenario

Chapter 5

Validation

1 Methodology

The validation of the cybersecurity training platform was conducted using surveys. This method was designed to provide comprehensive answers to the two research questions developed in Chapter 3.

In the survey phase, two participants were invited to engage with the two available scenarios and explore the platform independently. During this step, their interactions with the platform were closely observed, with intervention limited to addressing any critical issues that might prevent the experience from continuing, which fortunately did not occur. This observation was crucial for understanding how users naturally navigated and utilized the platform. After completing the scenarios, participants were presented with a survey divided into three parts, each addressing different parameters related to the research questions.

The first part consisted of a series of Likert scale questions. These questions were inspired by the work of Alsawaier [2], who has studied the effects of gamification on engagement and developed an evaluation method. However, these questions were adapted to the specific context of the platform, with additional questions included to address the second research question 3. Engagement and gamification questions provide information on how engaged the participants thought they were, whether there was an impact from specific gamification elements, and whether they would be willing to take part in further cybersecurity training that used gamification. The other questions focused more on the difficulty of finding scenarios at their level and their variety.

The second part of the survey featured four open-ended questions designed to elicit in-depth qualitative feedback from participants. These questions were crafted to uncover specific elements of the platform that users found either compelling

or problematic. Participants were encouraged to share their thoughts on which gamification elements (such as points, badges, leaderboards, avatars, or personalized trainers) they found most motivating and to explain why. Additionally, they were asked to provide suggestions for enhancing the platform's engagement and motivational aspects. The questions also sought to identify any challenges or obstacles the participants encountered while using the platform, along with their recommendations for overcoming these issues. Lastly, participants were invited to propose any additional features they believed would enrich their learning experience, thereby providing valuable insights for further platform development.

The third part of the survey implemented the System Usability Scale (SUS), a tool developed by John Brooke [5], to gauge the usability of the cybersecurity training platform. Usability, defined as the appropriateness of a system to its intended purpose, is a fundamental component that underpins both engagement and adaptability, central themes of this research. For Research Question 1, which investigates how gamification can enhance user engagement, the platform's usability is crucial. A well-designed, user-friendly system allows participants to engage with the content seamlessly, thereby amplifying the impact of gamification elements. If the platform is difficult to navigate, it could detract from the user experience, reducing overall engagement despite the presence of motivating game mechanics. In relation to Research Question 2, which focuses on the adaptability of the platform to various scenarios and user expertise levels, usability ensures that the platform is accessible to a broad range of users. A highly usable system can accommodate both novice and experienced users, facilitating the platform's ability to adjust to different training scenarios. The SUS provides a standardized measure of usability that helps determine whether the platform can effectively support users with varying backgrounds and needs.

2 Results and Discussion

The survey results provided valuable insights into the platform's strengths and areas for improvement. Results can be found in Appendix A and B. The first part of the survey, which measured user engagement and the perceived effectiveness of gamification elements on a scale of 1 to 5, revealed generally positive results. The average scores of 4.2 and 3.8 across the two participants suggest a favorable perception of engagement, a desire to learn more, and satisfaction with the guidance provided by the platform. However, certain aspects received lower ratings. The leaderboard, for instance, averaged a score of 2, indicating it was not well-received by the participants. Additionally, the questions regarding the variety of scenarios and the customization of the learning path also scored lower, highlighting areas for improvement. These three areas of concern—leaderboard effectiveness, scenario

variety, and learning path customization—can be attributed to two main factors. Firstly, the limited number of participants during the testing phase reduced the effectiveness of the leaderboard, as there was not enough competition to make it engaging. Secondly, the small number of available scenarios likely hindered the participants' ability to fully experience and appreciate the platform's adaptability and potential for personalized learning paths.

The second part of the survey, which focused on open-ended qualitative feedback, identified several critical points. Participants particularly appreciated the avatars and badges, aligning with existing literature that identifies rewards as one of the most effective gamification techniques. This is further supported by suggestions from participants to increase the number of collectible elements available on the platform. However, participants also pointed out several usability issues. The primary concerns revolved around interface intuitiveness, such as the lack of clear indicators showing whether a scenario was active and the subtlety of the indicators for badge possession. These issues suggest that while the platform's gamification elements were well-received, its usability could be improved. Participants suggested adding a tutorial to help users learn the basic functions available on the site, which could mitigate some of these usability challenges.

Despite these concerns, the System Usability Scale (SUS) scores of 80 and 62.5 out of 100 indicate that the platform is still considered usable, though there is room for improvement. Despite these usability challenges, participants expressed confidence in their ability to use the platform autonomously. They also believed that other users would be able to navigate and utilize the platform effectively.

The observation of user interactions with the platform, particularly with the chatbot feature, revealed significant differences in usage patterns between the two participants. The participant with a weaker background in cybersecurity relied heavily on the chatbot, using it to successfully navigate scenarios where they lacked prior knowledge of the vulnerabilities. In contrast, the other participant, who had more cybersecurity experience, used the chatbot less frequently and tended to ask less clear questions, which negatively impacted their interaction with the personal trainer feature. This disparity in chatbot usage highlights the importance of clear communication with AI-driven tools and suggests that the platform could benefit from improvements in how users are guided to interact with the chatbot.

Overall, the results indicate that while the platform is effective in engaging users and providing a gamified learning experience, there are notable areas for improvement, particularly in usability and the variety of content available. The feedback gathered suggests that with enhancements to the interface, the addition of more scenarios, and improvements to the chatbot functionality, the platform

could significantly increase its effectiveness in training users across varying levels of expertise.

3 Threats to Validity

Several factors potentially limit the validity of the findings. One of the main limitations is the small participant pool, with only two participants involved in the validation process. This small sample size is partly due to the platform's complex installation process, which requires a paid OpenAI key and necessitates significant time availability for assistance during setup. Additionally, the short duration of the testing period presents another limitation, as a longer testing period would be needed to obtain a meaningful measure of engagement. The current short-term evaluation restricts the ability to draw conclusions about sustained engagement over time.

The quantitative analysis of engagement is also biased, as it relies on participants' self-reported feelings. A more objective measure would involve server-side metrics, such as login duration, login frequency, points earned, scenario difficulty progression, and badge accumulation. However, implementing such measures would require a larger user base and enhanced user tracking and logging capabilities, although some aspects of the current gamification system could directly support these measurements.

Lastly, the expertise level of the participants may have influenced the results. The participants were computer science students with basic knowledge of cybersecurity, but not specialized in the field. This background might have shaped their interaction with the platform and the feedback they provided. To achieve a more comprehensive and objective evaluation of engagement, it would be necessary to involve a larger and more diverse user base and employ more sophisticated tracking and logging mechanisms to capture detailed user interactions over time.

Chapter 6

Future Work

The current work lays a solid foundation for a comprehensive cybersecurity training platform that integrates gamification, cyber ranges, recommender systems, and an AI-powered personalized trainer. However, there are several areas across all components that warrant further exploration and development.

Although the implemented gamification techniques have effectively increased user engagement, there is significant potential for advancing these methods. Future research should investigate more sophisticated gamification strategies to further enhance user motivation and retention. A promising avenue is the incorporation of collaborative and competitive elements, such as scenarios where trainees participate in defense-against-attack simulations, thereby mirroring real-world cybersecurity challenges and deepening the learning experience.

Improving the platform's usability and user experience is another critical area for future work. This includes optimizing notification management and integrating a chatbot directly within the visualization page to facilitate real-time interactions. The current RESTful API presents limitations, particularly in supporting real-time notifications and chatbot streaming responses. Transitioning to a more flexible API framework could address these challenges and significantly enhance the overall user experience.

The absence of a monitoring solution currently limits effective management and analysis of the cyber ranges. Implementing a monitoring solution exclusively based on Docker may present challenges. Therefore, transitioning to a hybrid environment that combines containerization with virtual machines is essential. This approach would enable the development of a more comprehensive and flexible monitoring system, allowing for sophisticated and robust simulations. Future work should also focus on refining deployment and upgrade processes, particularly for cloud-based systems, to ensure scalability and ease of maintenance.

The current recommender system primarily employs content-based filtering. Future efforts should explore the integration of collaborative filtering techniques, as discussed in Section 4of Chapter 4, to enhance the personalization of training scenarios. Additionally, exploring demographic or knowledge-based filtering approaches could broaden the system's ability to cater to a diverse range of user profiles. Fine-tuning the system's hyperparameters and conducting a thorough evaluation based on real-time ratings and feedback while the system is online is also crucial for improving recommendation accuracy and relevance. Furthermore, exploring the use of other embeddings, potentially leveraging the same model as the personal trainer, could enhance the system's capacity to understand and recommend scenarios that align closely with individual users' needs.

A particularly promising direction for future development involves enhancing the recommender system by integrating it more deeply with the chatbot, thereby creating a conversational recommender system[41]. This would allow users to interact with the platform more naturally, using dialogue to refine their training recommendations. The personal trainer already leverages a large language model (LLM) with a Retrieval-Augmented Generation (RAG) mechanism, and there are numerous opportunities to further develop this component. For example, designing more intricate graph flows could improve the evaluation of RAG outputs[50]. Additionally, integrating more tools into the chatbot could significantly enhance its ability to provide personalized and contextually relevant advice, further enriching the user experience.

Chapter 7

Conclusion

This thesis has investigated the application of gamification techniques to enhance engagement in cybersecurity training, with the goal of adapting the training experience to a wide range of scenarios and user proficiency levels. The resulting platform integrates a comprehensive gamified ecosystem, emphasizing point collection, rewards, personalization, competition, and guided learning mechanisms to create an engaging and effective training environment.

A standout feature of the platform is the specialized recommendation system, which aligns training scenarios with user profiles by analyzing complex scenario characteristics. Additionally, the platform incorporates a powerful chatbot, leveraging recent advancements in Large Language Models (LLMs). This chatbot is equipped with function-calling capabilities and can retrieve information from trusted sources, making it an invaluable tool for learning. It provides users with easy access to relevant information and delivers personalized solutions and interactions based on their specific context.

The platform also supports the seamless integration of new scenarios, demonstrated through two playable demo scenarios that highlight its versatility. However, despite these innovations, the platform is not without its limitations. One significant challenge is the integration of monitoring solutions, which is constrained by the simplistic architecture of containerized cyber ranges.

Looking ahead, there are numerous opportunities for further development. One particularly promising direction is the enhancement of the chatbot's capabilities. By increasing the complexity of its workflow, improving its ability to evaluate the reliability of its sources, and integrating it more closely with the recommendation system, the chatbot could become an even more central tool in education, potentially transforming the way learning is approached.

The implications of this work extend beyond academic contributions, offering

practical solutions for organizations aiming to improve their cybersecurity training programs. The platform's ability to simulate realistic scenarios within a gamified environment presents a valuable resource for training both professionals and non-experts, significantly contributing to ongoing efforts to secure the digital landscape.

Bibliography

[1] AL-Smadi, M. Gameducation: using gamification techniques to engage learners in online learning. *Immersive Education: 4th European Summit, EiED 2014, Vienna, Austria, November 24-26, 2014, Revised Selected Papers 4* (2015), 85–97.

[2] Alsawaier, R. S. The effect of gamification on motivation and engagement. *The International Journal of Information and Learning Technology 35*, 1 (2018), 56–79.

[3] Bianchi, F., Bassetti, E., and Spognardi, A. Scalable and automated evaluation of blue team cyber posture in cyber ranges. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing* (2024), pp. 1539–1541.

[4] Bootstrap. Bootstrap – the most popular html, css, and js library in the world. `https://getbootstrap.com/`. Accessed: 2024-08-05.

[5] Brooke, J., et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry 189*, 194 (1996), 4–7.

[6] Debnath, S., Ganguly, N., and Mitra, P. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of the 17th international conference on World Wide Web* (2008), pp. 1041–1042.

[7] DeepL GmbH. Deepl translator. `https://www.deepl.com/en/translator`, 2024. Accessed: 2024-08-15.

[8] Diakoumakos, J., Chaskos, E., Kolokotronis, N., and Lepouras, G. Cyber-range federation and cyber-security games: a gamification scoring model. In *2021 IEEE international conference on Cyber Security and Resilience (CSR)* (2021), IEEE, pp. 186–191.

[9] DiceBear. Dicebear – avatar library for designers and developers. `https://www.dicebear.com/`. Accessed: 2024-08-05.

[10] Docker. Docker – empowering app development for developers. https://www.docker.com/. Accessed: 2024-08-05.

[11] European Cyber Security Organisation. Understanding cyber ranges: From hype to reality, 2020. WG5 Paper.

[12] Flask. Flask – a microframework for python. https://flask.palletsprojects.com/en/3.0.x/. Accessed: 2024-08-05.

[13] Gee, J. P. What video games have to teach us about learning and literacy. *Computers in entertainment (CIE) 1*, 1 (2003), 20–20.

[14] Geng, J., Cao, B., Ye, H., Chen, J., Peng, M., and Liu, J. Web service recommendation based on knowledge graph convolutional network and doc2vec. *2020 IEEE World Congress on Services (SERVICES)* (2020), 95–100.

[15] Grigoriadis, A., Darra, E., Kavallieros, D., Chaskos, E., Kolokotronis, N., and Bellekens, X. Cyber ranges: The new training era in the cybersecurity and digital forensics world. In *Technology Development for Security Practitioners*. Springer, 2021, pp. 97–117.

[16] Gunes, I., Kaleli, C., Bilge, A., and Polat, H. Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review 42* (2014), 767–799.

[17] Kapp, K. M. *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons, 2012.

[18] LangChain, Inc. Langchain. https://www.langchain.com/, 2024. Accessed: 2024-08-15.

[19] LangChain, Inc. Langgraph documentation. https://langchain-ai.github.io/langgraph/, 2024. Accessed: 2024-08-15.

[20] Le, Q., and Mikolov, T. Distributed representations of sentences and documents. In *International conference on machine learning* (2014), PMLR, pp. 1188–1196.

[21] Leitner, M., Frank, M., Hotwagner, W., Langner, G., Maurhart, O., Pahi, T., Reuter, L., Skopik, F., Smith, P., and Warum, M. Ait cyber range: flexible cyber security environment for exercises, training and research. In *Proceedings of the 2020 European Interdisciplinary Cybersecurity Conference* (2020), pp. 1–6.

[22] Leitner, M., Frank, M., Langner, G., Landauer, M., Skopik, F., Smith, P., Akhras, B., Hotwagner, W., Kucek, S., Pahi, T., Reuter, L., and Warum, M. Enabling exercises, education and research with a comprehensive cyber range. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl. 12* (2021), 37–61.

[23] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[24] Lorica, B. Gradient flow - best practices in retrieval-augmented generation. `https://gradientflow.substack.com/p/best-practices-in-retrieval-augmented`, 2024. Accessed: August 18, 2024.

[25] Lü, L., Medo, M., Yeung, C. H., Zhang, Y.-C., Zhang, Z.-K., and Zhou, T. Recommender systems. *Physics reports 519*, 1 (2012), 1–49.

[26] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems 26* (2013).

[27] MongoDB. Mongodb – the most popular database for modern apps. `https://www.mongodb.com/`. Accessed: 2024-08-05.

[28] Nai-Fovino, I., Neisse, R., Hernandez-Ramos, J. L., Polemi, N., Ruzzante, G., Figwer, M., and Lazari, A. A proposal for a european cybersecurity taxonomy. Tech. Rep. EUR 29868, Joint Research Centre (JRC), European Commission, 2019.

[29] Nandi, R. N., Zaman, M. A., Al Muntasir, T., Sumit, S. H., Sourov, T., and Rahman, M. J.-U. Bangla news recommendation using doc2vec. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)* (2018), IEEE, pp. 1–5.

[30] Nicholson, S. A recipe for meaningful gamification. *Gamification in education and business* (2015), 1–20.

[31] OpenAI. Chatgpt. `https://chatgpt.com/`, 2024. Accessed: 2024-08-15.

[32] OpenAI. Openai api. `https://openai.com/index/openai-api/`, 2024. Accessed: 2024-08-15.

[33] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems 35* (2022), 27730–27744.

[34] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[35] Podnar, T. G., Dobson, G. B., Updyke, D. D., and Reed, W. E. Foundation of cyber ranges. *Software Engineering Institute, Pittsburgh, PA* (2021).

[36] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training.

[37] React. React – a javascript library for building user interfaces. `https://react.dev/`. Accessed: 2024-08-05.

[38] React Flow. React flow – an open-source library for building node-based uis. `https://reactflow.dev/`. Accessed: 2024-08-05.

[39] Roy, D., and Dutta, M. A systematic review and research perspective on recommender systems. *Journal of Big Data 9*, 1 (2022), 59.

[40] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (2001), pp. 285–295.

[41] Sun, Y., and Zhang, Y. Conversational recommender system. In *The 41st international acm sigir conference on research & development in information retrieval* (2018), pp. 235–244.

[42] Swacha, J. State of research on gamification in education: A bibliometric survey. *Education Sciences 11*, 2 (2021), 69.

[43] Thorat, P. B., Goudar, R. M., and Barve, S. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications 110*, 4 (2015), 31–36.

[44] Ukwandu, E., Farah, M. A. B., Hindy, H., Brosset, D., Kavallieros, D., Atkinson, R., Tachtatzis, C., Bures, M., Andonovic, I., and

Bellekens, X. A review of cyber-ranges and test-beds: Current and future trends. *Sensors 20*, 24 (2020), 7148.

[45] Urdaneta-Ponte, M. C., Mendez-Zorrilla, A., and Oleagordia-Ruiz, I. Recommendation systems for education: Systematic review. *Electronics 10*, 14 (2021), 1611.

[46] U.S. Department of Commerce National Institute of Standards and Technology. Cyber ranges. Tech. rep., NIST, 2018.

[47] Van Meteren, R., and Van Someren, M. Using content-based filtering for recommendation. In *Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop* (2000), vol. 30, Barcelona, pp. 47–56.

[48] Vaswani, A. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[49] Yamin, M. M., Katt, B., and Gkioulos, V. Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. *Computers & Security 88* (2020), 101636.

[50] Yan, S.-Q., Gu, J.-C., Zhu, Y., and Ling, Z.-H. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884* (2024).

[51] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[52] Řehůřek, R. Gensim doc2vec tutorial. `https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html`, 2024. Accessed: 2024-08-15.

Appendix A

Survey 1

	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
The use of game elements (points, badges, leaderboards) increased my interest in the training scenarios.	○	○	○	○	⊗
I felt absorbed and focused while participating in the gamified training scenarios.	○	○	○	⊗	○
The gamified activities made challenging cybersecurity topics more interesting and engaging.	○	○	○	○	⊗
I did not feel interested in the gamified activities.	⊗	○	○	○	○
I enjoyed customizing my avatar and felt it gave me a sense of ownership in the training.	○	○	○	⊗	○
Being able to set personal goals within the platform increased my motivation to complete the scenarios.	○	○	○	○	⊗
The platform allowed me to choose scenarios that matched my expertise level.	○	○	○	⊗	○
Seeing my progress on the leaderboard motivated me to improve my performance.	○	○	○	⊗	○
Competing with others on the leaderboard made the training more exciting.	○	○	⊗	○	○
Achieving high scores and earning badges motivated me to engage more deeply with the training content.	○	○	○	⊗	○
Overall, I am satisfied with my experience in the gamified training scenarios.	○	○	○	⊗	○
I would prefer gamified training activities over traditional ones for future cybersecurity training.	○	○	○	⊗	○
I will participate again in gamified training scenarios if offered in the future.	○	○	○	○	⊗
I feel that more game elements could be incorporated into the training to further increase engagement.	○	○	○	⊗	○
The training platform was flexible enough to accommodate various types of scenarios.	○	○	⊗	○	○
The platform allowed me to easily select scenarios appropriate for my skill level.	○	○	○	⊗	○
The scenario recommendations based on my profile were helpful in guiding my training.	○	○	○	○	⊗
I found the visual progress graph helpful in understanding my advancement through the scenarios.	○	○	○	○	⊗
The ability to customize my training path (choose scenarios, set goals) increased my engagement.	○	○	○	⊗	○

Open-Ended Questions

1. Which specific game element (points, badges, leaderboards, avatars, personnal trainer,...) did you find most motivating and why?

   Avatar modification because it represents visually my improvements and it is always visible

   I also like the badges because it can open the horizons and set global goals.

2. How could the gamified training platform be improved to better engage and motivate you?

   Show in transparence the avatar / badges that are not already unlocked or completed.

   This would improve the global view of what we have already done and what there is still to do.

3. Describe any challenges you faced while using the platform and suggest possible improvements.

   How to launch the scenario because even if we have opened it, we still need to click on the "start" button.

   Similar remark for "request hint" button. We do not know when we do not have hints anymore.

   Maybe show it in transparence if so?

4. What additional features would you like to see to enhance your learning experience?

   - Have access to Hal on the scenario page.
     It will be more intuitive to ask him for help
   - A tutorial page explaining what we can collect when improving our skills through scenario.

System Usability Scale

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently				X	
	1	2	3	4	5
2. I found the system unnecessarily complex	X				
	1	2	3	4	5
3. I thought the system was easy to use					X
	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system		X			
	1	2	3	4	5
5. I found the various functions in this system were well integrated				X	
	1	2	3	4	5
6. I thought there was too much inconsistency in this system		X			
	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly					X
	1	2	3	4	5
8. I found the system very cumbersome to use		X			
	1	2	3	4	5
9. I felt very confident using the system				X	
	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system			X		
	1	2	3	4	5

Appendix B

Survey 2

	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
The use of game elements (points, badges, leaderboards) increased my interest in the training scenarios.	○	○	○	⊗	○
I felt absorbed and focused while participating in the gamified training scenarios.	○	○	○	⊗	○
The gamified activities made challenging cybersecurity topics more interesting and engaging.	○	○	○	○	⊗
I did not feel interested in the gamified activities.	○	⊗	○	○	○
I enjoyed customizing my avatar and felt it gave me a sense of ownership in the training.	○	○	○	○	⊗
Being able to set personal goals within the platform increased my motivation to complete the scenarios.	○	○	○	⊗	○
The platform allowed me to choose scenarios that matched my expertise level.	○	○	⊗	○	○
Seeing my progress on the leaderboard motivated me to improve my performance.	○	○	○	○	⊗
Competing with others on the leaderboard made the training more exciting.	⊗	○	○	○	○
Achieving high scores and earning badges motivated me to engage more deeply with the training content.	○	○	○	⊗	○
Overall, I am satisfied with my experience in the gamified training scenarios.	○	○	○	⊗	○
I would prefer gamified training activities over traditional ones for future cybersecurity training.	○	○	○	○	⊗
I will participate again in gamified training scenarios if offered in the future.	○	○	○	○	⊗
I feel that more game elements could be incorporated into the training to further increase engagement.	○	○	○	○	⊗
The training platform was flexible enough to accommodate various types of scenarios.	○	○	⊗	○	○
The platform allowed me to easily select scenarios appropriate for my skill level.	○	○	○	⊗	○
The scenario recommendations based on my profile were helpful in guiding my training.	○	○	⊗	○	○
I found the visual progress graph helpful in understanding my advancement through the scenarios.	○	○	⊗	○	○
The ability to customize my training path (choose scenarios, set goals) increased my engagement.	○	⊗	○	○	○

Open-Ended Questions

1. Which specific game element (points, badges, leaderboards, avatars, personnal trainer,...) did you find most motivating and why?

avatars + badges

2. How could the gamified training platform be improved to better engage and motivate you?

add more elements to earn after completing a training.

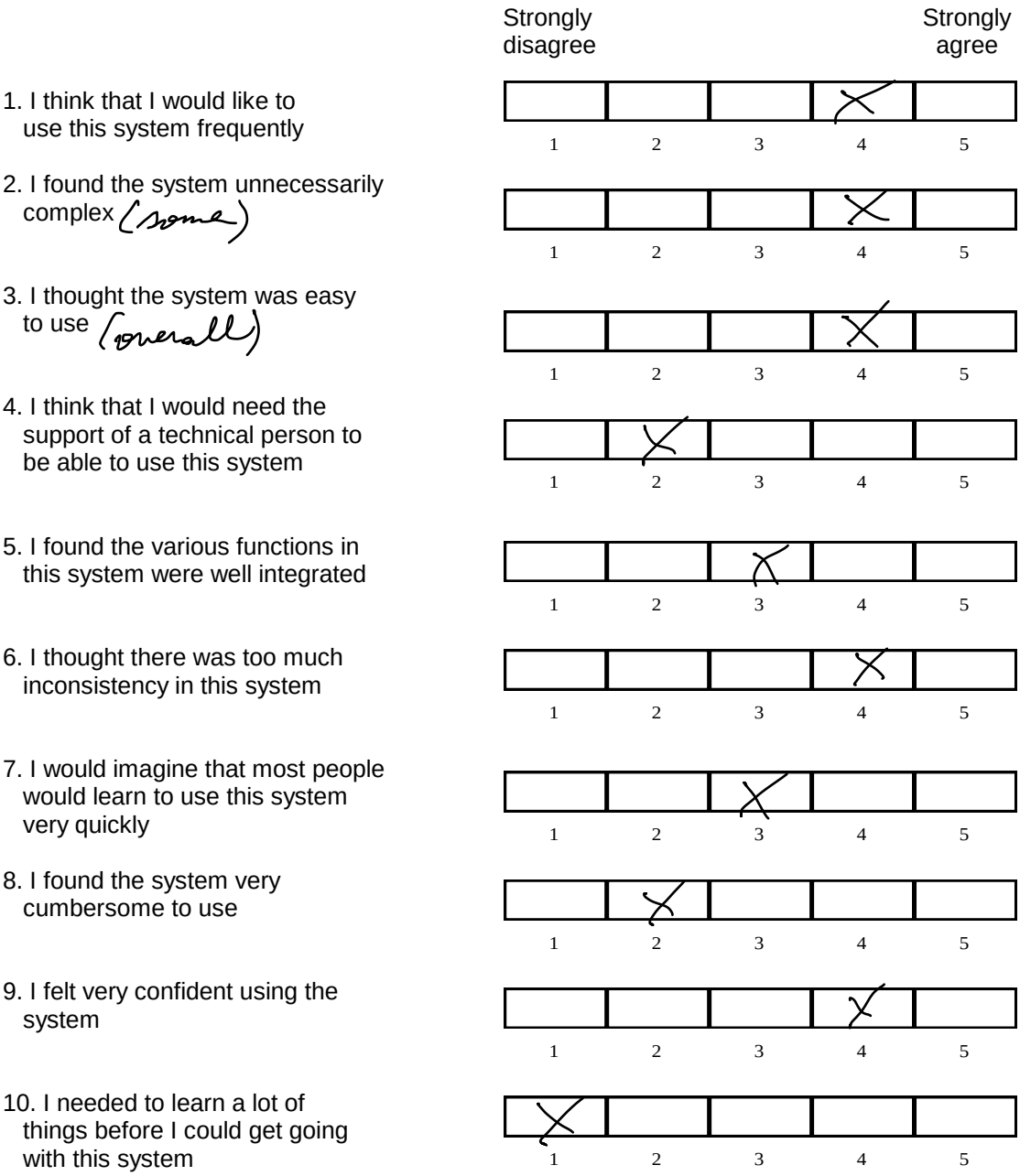
3. Describe any challenges you faced while using the platform and suggest possible improvements.

the use of the AI chat + button interactions in scenario
Need more information on what to do (a guide or a demo).

4. What additional features would you like to see to enhance your learning experience?

the possibility to hide not founded flags above the founded flags in the graph (during a scenario)

System Usability Scale



UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**