

# A Web-based Data-Driven Security Game for Teaching Software Security

**Cristoffer Brandhaug**

Supervisor: Tosin Daniel Oyetoyan

**Master's thesis in Software Engineering at**

Department of Computer science, Electrical  
engineering and Mathematical sciences,  
Western Norway University of Applied Sciences

Department of Informatics,  
University of Bergen

June 2022



Western Norway  
University of  
Applied Sciences



# Abstract

Information security is becoming more and more important for developers. New threats and attacks regularly show up, and developers need to stay on top of these threats to protect software systems and applications. Because of the rapid changes in threats, the curriculum can easily be outdated and cumbersome to learn. Gamification has previously shown to be a successful way of engaging and motivating users. Therefore, a quiz-based game is hypothesized to motivate the adoption of teaching software security. Public information security sources that are up-to-date are incorporated into elements that are introduced to the user in the game. Others have tried to create security games to solve this problem, such as STIX and Stones, Protection Poker, and Elevation of Privilege, but they have had challenges with the adoption of their games. This thesis will look at these games and other serious games in an attempt to overcome some of the problems related to the previous security games. To evaluate whether the game affects learning and motivation for learning, a class of computer science students tested the game and answered a user survey. The results show that the game has potential, with many respondents being optimistic about the game's use in learning software security and motivation to learn using a quiz-based game. However, the game is not perfect and would require further work to succeed. Results show usability flaws and that more study is needed to conclude whether or not the use of a quiz-based game improves learning software security. Suggestions for further work are introduced to improve the game further, such as stronger metrics to improve on already implemented elements.

# Acknowledgements

I would like to express my gratitude to my supervisor, Tosin Daniel Oyetoyan, for his guidance and help throughout the period of working on my thesis. I would also like to thank his students that helped with testing the quiz-based game and answering the user survey.

Finally, I would like to thank my family, friends, and girlfriend for their continuous support.

# Contents

<b>Glossary</b>	<b>10</b>
<b>Acronyms</b>	<b>12</b>
<b>1 Introduction</b>	<b>14</b>
1.1 Research Questions . . . . .	16
1.2 Method and Evaluation . . . . .	16
1.3 Contributions . . . . .	17
1.4 Outline . . . . .	18
<b>2 Background</b>	<b>19</b>
2.1 Public Information Security Repositories . . . . .	19
2.1.1 CWE . . . . .	20
2.1.2 CAPEC . . . . .	22
2.1.3 CVE . . . . .	25
2.1.4 SEI CERT Oracle Coding Standard for Java . . . . .	25
2.1.5 Stackoverflow . . . . .	26
2.2 Gamification . . . . .	27
2.2.1 Game Types . . . . .	28
<b>3 Methodology</b>	<b>31</b>
3.1 Process . . . . .	31
3.2 Prototype . . . . .	33
3.3 Evaluation . . . . .	36
3.3.1 Learnability, Usability, and Fun . . . . .	36



3.3.2	How to Integrate Public Information Security Sources to Answer Questions Automatically . . . . .	42
3.3.3	Quiz Questions . . . . .	42
<b>4</b>	<b>Design and Implementation</b>	<b>43</b>
4.1	Architecture . . . . .	43
4.1.1	Spring Boot . . . . .	45
4.1.2	Spring JPA . . . . .	45
4.2	Conceptualization Stage . . . . .	46
4.3	Integration of Public Security Content . . . . .	47
4.4	Quiz and Question Creator for Admin Users . . . . .	51
<b>5</b>	<b>Results and Discussion</b>	<b>54</b>
5.1	Quiz Game . . . . .	55
5.2	Findings to Research Questions . . . . .	58
5.2.1	Learnability (RQ2) . . . . .	59
5.2.2	Usability (RQ1) . . . . .	65
5.2.3	User Enjoyment/Fun (RQ1) . . . . .	67
5.2.4	Ask the Audience (RQ3) . . . . .	72
5.3	Discussion . . . . .	73
<b>6</b>	<b>Literature Review</b>	<b>75</b>
6.1	Serious Games . . . . .	75
6.1.1	STIX and Stones . . . . .	75
6.1.2	OWASP Cornucopia . . . . .	77
6.1.3	Elevation of Privilege . . . . .	78
6.1.4	Kahoot! . . . . .	79
<b>7</b>	<b>Conclusion and Further Work</b>	<b>81</b>
7.1	Suggestions to Further Work . . . . .	82
<b>A</b>	<b>Source code and Link to Quiz-based Game</b>	<b>84</b>

# List of Tables

2.1	Examples of CWE records . . . . .	21
2.2	Examples of CAPEC records . . . . .	24
3.1	Goal Question Metric example . . . . .	38
3.2	Learning Software Security Metrics . . . . .	40
3.3	Fun/Enjoyment Metrics . . . . .	41
3.4	Usability Metrics . . . . .	41
5.1	Table with results from 'Ask the audience' lifeline tests. . . . .	72

# List of Figures

2.1	Public information security repositories . . . . .	20
2.2	Mechanisms of Attack . . . . .	23
2.3	Risk Assessment Summary for Rule 02. Expressions (EXP) . .	25
2.4	Non-compliant and compliant solution code examples for file deletion. . . . .	26
2.5	Tower Defense King . . . . .	29
3.1	Weekly Planning . . . . .	31
3.2	Agile development process . . . . .	33
3.3	Menu prototype . . . . .	34
3.4	Quiz question - Play view created with Figma . . . . .	35
3.5	Question 3 . . . . .	39
4.1	Simplified architectural overview . . . . .	43
4.2	Simple life-cycle of Thymeleaf template . . . . .	45
4.3	Ask the audience lifeline . . . . .	47
4.4	Call a friend lifeline - Search box with user input and selection of 'friend' . . . . .	49
4.5	Call a friend lifeline - CAPEC as selected friend and result page	50
4.6	Administrator menu . . . . .	51
4.7	Create question . . . . .	52
4.8	Create a quiz . . . . .	53
5.1	Search for a quiz by ID or name of the quiz and select what quiz to play . . . . .	55
5.2	Quiz play . . . . .	55

5.3	Explanation and correct answer for previous question . . . . .	56
5.4	Shield lifeline . . . . .	57
5.5	Submit your score page when quiz is done . . . . .	57
5.6	Leaderboard of selected quiz . . . . .	58
5.7	Results from round 1 of quiz play . . . . .	59
5.8	Results from round 2 of quiz play . . . . .	59
5.9	Results prior knowledge to topics presented in the quiz . . . . .	60
5.10	Did knowledge on topics presented improve? . . . . .	61
5.11	Was search feature helpful in answering questions? . . . . .	61
5.12	Results when asked if explanation was helpful to understand- ing the question . . . . .	62
5.13	Did explanations help improve answers? . . . . .	62
5.14	Results - Did the quiz game make users learn more about software security? . . . . .	63
5.15	Is the quiz game a good tool for learning software security? . . . . .	63
5.16	Quiz compared to other methods of learning . . . . .	64
5.17	Ask the audience - Did the lifeline help users quickly reason about the answer? . . . . .	64
5.18	Results - Easy to understand the rules and play the game. . . . .	65
5.19	Results - Did the users understand the rules and how to play from using the in-game help button? . . . . .	65
5.20	Results - Easy to use the 'shield' lifeline . . . . .	66
5.21	Results - Easy to use the '50/50' lifeline . . . . .	66
5.22	Results - Easy to use 'ask the audience' lifeline . . . . .	67
5.23	Results - Easy to use the 'call a friend' lifeline . . . . .	67
5.24	Results show that the majority of the students enjoyed playing the quiz . . . . .	67
5.25	Results when asked to try again . . . . .	68
5.26	Results for 'shield' lifeline . . . . .	68
5.27	Results for '50/50' lifeline . . . . .	69
5.28	Results for 'Ask the audience' lifeline . . . . .	69
5.29	Results for 'Call a friend' lifeline . . . . .	70
5.30	Leaderboard motivated the players to score better . . . . .	70

5.31	Score bar and levels motivated the users to try again . . . . .	71
6.1	STIX and Stones gameplay . . . . .	76
6.2	OWASP Cornucopia card game . . . . .	77
6.3	Kahoot! . . . . .	80

# Glossary

**Figma** Figma is a web-based graphics editing and user interface design app..

**GitHub** GitHub is a code hosting platform for version control and collaboration..

**Likert** A scale used to represent people's attitudes to a topic..

**product backlog** The Product Backlog is an emergent, ordered list of what is needed to improve the product..

**product owner** The Product Owner is the single point of contact for understanding customer requirements and expectations from the product..

**Spring** The Spring Framework (Spring) is an open-source application framework that provides infrastructure support for developing Java applications..

**Spring Boot** Spring Boot is a tool that makes developing web application and microservices with Spring Framework faster and easier..

**sprint review** The sprint review is one of the most important ceremonies in Scrum where the team gathers to review completed work and determine whether additional changes are needed..

**sprints** Sprints refer to short, repeating blocks of time in which key parts of the project are completed..

**Thymeleaf** Thymeleaf is a modern server-side Java template engine for both web and standalone environments..

**URL** The address of a web page..

**user stories** A user story is an informal, general explanation of a software feature written from the perspective of the end user..

# Acronyms

**API** Application programming interface.

**CAPEC** Common Attack Pattern Enumeration Classification.

**CSS** Cascading Style Sheet.

**CVE** Common Vulnerabilities and Exposures.

**CWE** Common Weakness Enumeration.

**EoP** Elevation of Privilege.

**GQM** Goal Question Metric.

**HTML** HyperText Markup Language.

**IoT** Internet of Things.

**IT** information technology.

**JPA** Java Persistence API.

**JS** JavaScript.

**JSON** JavaScript Object Notation.

**OWASP** Open Web Application Security Project.



**PP** Protection Poker.

**SO** Stack Overflow.

**SQL** Structured Query Language.

**TD** Tower Defense.

**TF** Term Frequency.

# Chapter 1

## Introduction

Cyber security is a central topic today because of its dependence on systems that are interconnected with several other systems. Unfortunately, end-users and developers are not adequately trained to mitigate threats from the cyber world [1]. A user can be exposed to many security and privacy threats in cyberspace due to insecure applications. Examples exist for different platforms such as web vulnerabilities, mobile, or Internet of Things (IoT) systems.

Open Web Application Security Project (OWASP) [2] Top-10 vulnerabilities provide a list of the top-10 vulnerabilities that can be used as an awareness document and starting point to secure coding. OWASP Top-10 can be applied to different use cases during software development, such as architecture and design, coding, testing, configuration, and using the content for developing a training curriculum for developers. Examples of categories on OWASP Top-10 are Broken Access Control, Cryptographic Failures, Injection, and Insecure Design.

Unfortunately, security in applications/software is often not the primary concern when developing a new application. Developers tend to think of everything else first; How will the user interface work? What options will the user have? and How will data be stored? A study done by Secure Code Warrior on 1,200 active software engineers/developers from around the world in

April 2022 shows that software security is not a top priority when developing an application [3]. The study shows that only 14% of the respondents listed application security as a top priority. Security is often overlooked as functionality is prioritized.

Therefore, it is essential to incorporate security content into the curriculum for software engineering. One of the challenges of teaching software security is that the curriculum can quickly be outdated because of constant changes in threats and new attacks. It is, therefore, necessary to provide software developers and students with methods and tools that can keep them up to date with all the latest threats. These tools and techniques of learning should be attractive. The user should have the option to use them for self-study and keep updated with information security content.

Gamification has become more popular and has proven to be an excellent tool for educational purposes [4]. By using games as a tool for learning, you engage the students, and they learn while getting the feeling of being rewarded (with points, ranking, achievements, etc.) [5]. There are already existing games that have tried to educate on the topic of security. Some examples include the Elevation of privilege game [6][7] and Protection Poker[8]. The challenges with these games are that some of them are card and manual-based, they have been reported to be difficult to play, and they don't seem to be well adopted [7][6]. STIX and Stones [9] is another example of an online security game incorporating data-driven security content. The game itself appears exciting and enjoyable; however, it hasn't been adopted and used by many. It is hypothesized that using a quiz-based game that is integrated with public security information databases can improve learning and provide fun at the same time.

By integrating the game with information security content from security-based online sources such as Common Attack Pattern Enumeration Classification (CAPEC) [10] through the game, the users can more easily stay up to date with new information. The educational goal of the game is to create a web-based gaming tool where it is possible to construct questions and play quiz games in order to develop cybersecurity awareness and safe practices for

both developers and end-users.

## 1.1 Research Questions

The hypothesis for this thesis is that a quiz-based security game can motivate adoption for teaching software security.

The thesis investigates three research questions:

- **Research question 1:** Does the use of a quiz-based data-driven game motivate users to learn software security?
- **Research question 2:** Does the use of a quiz-based data-driven game improve learning software security?
- **Research question 3:** How to integrate information security sources to automatically answer questions in the quiz game?

## 1.2 Method and Evaluation

This thesis has used educational gamification, a technology-based method [11] for increasing student engagement and motivation for learning, to create a game prototype. Quantitative method is used to evaluate the game prototype and answer the research questions. Specifically, the thesis has used survey and questionnaire to collect data from players, that are students who study to become software developers.

To collect quantitative data, surveys are done on a group of people, which ask the same questions to the entire group. The questions are designed to measure three key areas:

- Learnability
- Fun
- Usability

The questions are structured such that they can be measurable in a quantitative manner.

## **1.3 Contributions**

The following are the contributions of this thesis:

- A quiz-based security game for teaching software security.
- Integration of public security information content to automatically provide answers.

## 1.4 Outline

- **Chapter 1 - Introduction**

Describes the motivation for this thesis, presents the research questions, research and evaluation method, and the contributions of this thesis.

- **Chapter 2 - Background**

Presents public information security sources and gamification as background to this study.

- **Chapter 3 - Methodology**

Presents the different methods used to answer the research questions of this thesis, including research method, development methodology, design process, and how the quiz questions for the user survey were prepared.

- **Chapter 4 - Design and Implementation**

Briefly describe the tools and technology used to build the quiz-based game, how the goals of learnability, fun, and usability are conceptualized, how public information security repositories are integrated with the quiz game, and how the admin part of the game works.

- **Chapter 5 - Results and Discussion**

Presents and discusses the quiz game and results from the user survey, and answers to the research questions.

- **Chapter 6 - Literature Review**

Takes a look at other serious games, and compares similarities and differences.

- **Chapter 7 - Conclusion and Further Work**

Concludes this thesis and presents suggestions for further work to improve on the quiz game.

# Chapter 2

## Background

The thesis is built on two broad areas, namely, public information security sources and gamification. The following sections will introduce the two categories as background to this study.

### 2.1 Public Information Security Repositories

This section will introduce Common Weakness Enumeration (CWE) [12], CAPEC, Common Vulnerabilities and Exposures (CVE) [13], and SEI CERT [14] secure coding repositories as relevant security information sources for this thesis.



Figure 2.1: Public information security repositories  
 Source: [https://capec.mitre.org/about/new\\_to\\_capec.html](https://capec.mitre.org/about/new_to_capec.html)

Weaknesses are the root causes of vulnerability, as shown in CWE. How the weaknesses can be exploited is described by CAPEC. Actual exploitation in real software is demonstrated in CVE.

### 2.1.1 CWE

CWE is a community-developed list of weaknesses related to software and hardware. ”Weaknesses” are flaws, faults, bugs, or other errors in software or hardware implementation, code, design, or architecture that if left un-addressed could result in systems, networks, or hardware being vulnerable to attack” [12]. CWE’s main goal is to stop vulnerabilities by educating IT practitioners on how to tackle common mistakes before product delivery. CWE, like CAPEC, offers a few different lists categorized by how frequently they are encountered from a specific point of view. The three main lists are Software Development, Hardware Design, and Research Concepts. Some examples of software weaknesses are: code evaluation and injection, authentication errors, and user interface errors [12].

Some examples of weaknesses in hardware are: core and compute issues typically associated with CPUs, Graphics, Vision, AI, FPGA, and uControllers, and power, clock, and reset concerns related to voltage, electrical current, temperature, clock control, and state saving/restoring [12].



Table 2.1: Examples of CWE records  
Source: [12]

ID	Title	Description
CWE-478	Missing Default Case in Switch Statement	The code does not have a default case in a switch statement, which might lead to complex logical errors and resultant weaknesses.
CWE-681	Incorrect Conversion between Numeric Types	When converting from one data type to another, such as long to integer, data can be omitted or translated in a way that produces unexpected values. If the resulting values are used in a sensitive context, then dangerous behaviors may occur.
CWE-1323	Improper Management of Sensitive Trace Data	Trace data collected from several sources on the System-on-Chip (SoC) is stored in unprotected locations or transported to untrusted agents.

Table 2.1 shows an example of a CWE entry, with ID, title, and description of the weakness. In addition to this, a CWE record can contain: **Relationships**, that are weaknesses related to the specific weakness. **Related Attack Patterns**, which are CAPEC attack pattern(s) related to the weakness. **Modes of Introduction**; How and When the specific weakness may be introduced. **Applicable Platforms**, possible areas where the weakness could appear, i.e. specific programming languages. **Common Consequences** associated with the weakness **Likelihood of Exploit**, Low, Medium, or High. **Demonstrative Examples**, such as code examples for Software Development. **Potential Mitigations** to avoid the specific weakness, and **Observed Examples**.

CAPEC and CWE are related in that a CAPEC entry lists related weaknesses associated with the attack pattern. A CAPEC entry can contain multiple

weaknesses. Related weaknesses are identified with a CWE ID.

### **2.1.2 CAPEC**

CAPEC [10] is a public website working as a repository with familiar cyber attack patterns that users can use to better understand how weaknesses are exploited and how to mitigate the risk of these exploits happening. Attack patterns are descriptions of how to exploit known weaknesses in software. Each attack pattern contains information on how the attacks work and how to mitigate the attack's effectiveness. This is helpful for a developer to understand better how to avoid allowing such weaknesses in their application or software system. CAPEC entries are mainly listed in two lists: Mechanisms of Attack (as seen in figure 2.2) and Domains of Attack. Mechanisms of Attack lists attack patterns that are often used when exploiting a vulnerability, whereas Domains of Attack lists items by the target domains.

## 1000 - Mechanisms of Attack

- ⊕ **C** Engage in Deceptive Interactions - (156)
- ⊖ **C** Abuse Existing Functionality - (210)
  - ⊕ **M** Interface Manipulation - (113)
  - ⊕ **M** Flooding - (125)
  - ⊕ **M** Excessive Allocation - (130)
    - ▪ **M** Resource Leak Exposure - (131)
  - ⊕ **M** Functionality Misuse - (212)
  - ⊕ **M** Communication Channel Manipulation - (216)
  - ⊕ **M** Sustained Client Engagement - (227)
  - ⊕ **M** Protocol Manipulation - (272)
  - ⊕ **M** Functionality Bypass - (554)
- ⊕ **C** Manipulate Data Structures - (255)
- ⊖ **C** Manipulate System Resources - (262)
  - ⊖ **M** Software Integrity Attack - (184)
    - ▪ **S** Malicious Software Download - (185)
    - ⊕ **S** Malicious Software Update - (186)
    - ▪ **S** Exploitation of Transient Instruction Execution - (663)
    - ▪ **S** Alteration of a Software Update - (669)
  - ⊕ **M** Hardware Integrity Attack - (440)
  - ⊕ **M** Infrastructure Manipulation - (161)
  - ⊕ **M** File Manipulation - (165)
  - ⊕ **M** Configuration/Environment Manipulation - (176)
  - ⊕ **M** Obstruction - (607)
  - ⊕ **M** Modification During Manufacture - (438)
  - ⊕ **M** Manipulation During Distribution - (439)
  - ⊕ **M** Malicious Logic Insertion - (441)
    - ▪ **M** Contaminate Resource - (548)
- ⊕ **C** Inject Unexpected Items - (152)
- ⊕ **C** Employ Probabilistic Techniques - (223)
- ⊕ **C** Manipulate Timing and State - (172)
- ⊕ **C** Collect and Analyze Information - (118)
- ⊕ **C** Subvert Access Control - (225)

Figure 2.2: Mechanisms of Attack

Source: <https://capec.mitre.org/data/definitions/1000.html>

Table 2.2: Examples of CAPEC records  
Source: [10]

ID	Title	Description
CAPEC-25	Forced Deadlock	The adversary triggers and exploits a deadlock condition in the target software to cause a denial of service. A deadlock can occur when two or more competing actions are waiting for each other to finish, and thus neither ever does. Deadlock conditions can be difficult to detect.
CAPEC-115	Authentication Bypass	An attacker gains access to application, service, or device with the privileges of an authorized or privileged user by evading or circumventing an authentication mechanism. The attacker is therefore able to access protected data without authentication ever having taken place.
CAPEC-240	Resource Injection	An adversary exploits weaknesses in input validation by manipulating resource identifiers enabling the unintended modification or specification of a resource.

Table 2.2 shows examples of CAPEC entries with the related ID, Title, and Description. A CAPEC record also includes information on **Related Weakness(es)** exploited by the attack pattern. **Execution Flow**, instructions on how the attack is performed. **Prerequisites** for the attack to succeed, and **Mitigations**, how to reduce the risk of a particular attack to happen/succeed.

### 2.1.3 CVE

CVE is a public repository for known cybersecurity vulnerabilities and exposures [13]. A CVE record identifies one vulnerability in the catalog. Vulnerabilities are found and assigned before being published by organizations partnered with the CVE program. A vulnerability is a weakness that the attacker can take advantage of directly to perform unauthorized activities as an authorized user. In contrast, exposure is a mistake that allows attackers to break into a system. To ensure information technology (IT) and cybersecurity experts are talking about the same vulnerability, they use CVE records as they are identified with ids. CVE aims to make sharing information on known vulnerabilities easy to stay updated on the latest cybersecurity threats.

### 2.1.4 SEI CERT Oracle Coding Standard for Java

SEI CERT Oracle Coding Standard for Java consists of rules and recommendations for programming with Java as a programming language [14]. The rules give requirements to code, whereas recommendations work as a guide that should improve the security, safety, and reliability of software systems when followed. Rules come with a risk assessment summary with severity, the likelihood, remediation cost, priority, and level.

Risk Assessment Summary

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
EXP00-J	Medium	Probable	Medium	P8	L2
EXP01-J	Low	Likely	High	P3	L3
EXP02-J	Low	Likely	Low	P9	L2
EXP03-J	Low	Likely	Medium	P6	L2
EXP04-J	Low	Probable	Low	P6	L2
EXP05-J	Low	Unlikely	Medium	P2	L3
EXP06-J	Low	Unlikely	Low	P3	L3

Figure 2.3: Risk Assessment Summary for Rule 02. Expressions (EXP)

Source: <https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487704>

The rules and recommendations come with useful example codes on weak code and how to improve it to secure better code. An example of this is seen in figure 2.4 where the non-compliant code doesn't check if a file is successfully deleted. The compliant solution shows how it can be fixed by checking if the file has been successfully deleted. If not, a function to handle the failure is called.

#### Noncompliant Code Example (File Deletion)

This noncompliant code example attempts to delete a file but fails to check whether the operation has succeeded:

```
public void deleteFile(){  
  
    File someFile = new File("someFileName.txt");  
    // Do something with someFile  
    someFile.delete();  
  
}
```

#### Compliant Solution

This compliant solution checks the Boolean value returned by the `delete()` method and handles any resulting errors:

```
public void deleteFile(){  
  
    File someFile = new File("someFileName.txt");  
    // Do something with someFile  
    if (!someFile.delete()) {  
        // Handle failure to delete the file  
    }  
  
}
```

Figure 2.4: Non-compliant and compliant solution code examples for file deletion.

Source: <https://wiki.sei.cmu.edu/confluence/display/java/EXP00-J.+Do+not+ignore+values+returned+by+methods>

## 2.1.5 Stackoverflow

Stack Overflow (SO) [15] is a Question-and-Answer (Q&A) website used by programmers to share knowledge with each other. According to StackExchange [16], SO has close to 18 million registered users, 23 million questions, and 33 million answers, with 70% of the questions being answered. SO enables users to ask questions or post answers on specific topics related to coding. It is easy to post code snippets, which is helpful to get assistance on specific coding errors, as users can make reproducible examples to help find

a solution to the user posting's question. SO has a voting system that lets users upvote or downvote questions and answers. This voting system allows good answers and questions to rise in popularity, making them more noticeable, whereas bad ones are filtered out. Users can earn reputation points and 'badges,' which are gamification elements, for being a helpful contributor. Users with a high reputation can unlock privileges that let them flag, vote, comment, and edit other people's posts.

## 2.2 Gamification

A literature review done by Hamari et al. [17] on the effects of gamification shows that gamification comes with positive results and benefits in most cases. Gamification is when elements usually seen in games are added to other environments outside games. Gamification used in lectures has proven to be useful and can also lead to more students showing up to class and reading the lecture notes [18]. There are mixed opinions on whether gamification affects the student's grades, where for one study, it did not seem to affect student grades [19]. Another study was done on two groups of students [20]. One of the groups participated in a gamified learning environment and the other in a traditional learning environment, which shows that the students in the gamified group performed better and achieved higher grades than the other group following the conventional learning environment. However, both studies agree that more research must be done to create a more robust and thorough foundation for these findings.

Literature on gamification and serious games was investigated to find which features/elements bring the most user engagement, fun, and motivation for learning [4][17][18][21][22]. This is essential to see what elements can be used for the software security quiz game. It is found that leaderboards, a scoring system (points/coins), and achievements are the most popular elements used in gamification [18][17][4][23]. A scoring system works so that the players are rewarded with points when playing, for example, when answering the correct answer in a quiz. Receiving points motivates the user to score better than

classmates to climb the leaderboard [22] and receive positive feedback [18]. Leaderboards create a competitive environment between classmates, where they compete against each other to get the highest score or get their name on the top-10 leaderboard. According to O'Donovan et al. [23], leaderboard was the most motivating gamification element by far. Achievements are badges the players achieve by completing different tasks in the game. This is an effective motivator to keep the students engaged and motivated to keep playing by having them work towards a goal/prize that gives positive feedback when achieved [23][24].

### **2.2.1 Game Types**

Laurie Williams et al. [8] introduced Protection Poker (PP), which is a software security card game. PP is described as a collaborative and lightweight game that can estimate software security risks and is particularly suited for agile teams [7]. PP shows potential in that it leads to discussions about software security within the team; however, the game has shown little to no usage. Reasons for this may be that the game requires someone with knowledge about software security to be a success and that it requires a lot of effort and time to play [7]. Another challenge with PP is that it is not a single-player game. The problem with this is that it can be cumbersome to gather enough people to play.

STIX and Stones [9] is another security game. This Tower Defense (TD) type of game integrates CAPEC to create entities for enemies and defensive towers. TD is a strategy game where the goal is to defend against waves of attacks that gradually become more and more difficult to defend against. The enemies try to reach the player's territory, and they have to be defeated before they reach it to stay alive and eventually beat the game by defeating all waves of attacks. To defend against the attacks, the player is given options on defensive structures to build against each wave. These defensive structures have strengths and weaknesses against different types of enemies.





Figure 2.5: Tower Defense King

Source: <https://www.researchgate.net/figure/Screenshot-of-a-classic-Tower-Defense-game-Game-shown-Tower-Defense-Kingfig2339428392>

Figure 2.5 shows how a typical TD game is built, with a path enemies have to walk through and defensive structures built around this path defending against the enemies. The challenges with this type of game can be that it is not as familiar to most, and it can be too much information to focus on the important part, learning software security.

The primary motivation for this study is thus to improve the adoption of software security games by creating a quiz-based software security game that can motivate learning and teaching this subject. There are multiple examples of successfully using gamification in education, such as the popular game-based learning platform Kahoot! [25].

This study incorporates some of the successful features investigated in previously mentioned games and articles in a quiz-based software security game and fill in the missing gaps in previous software security games. By selecting quiz as the game type, the obstacles of users being thrown off by the complexity and initial information are removed. A quiz-based game is easy to understand and can easily incorporate features such as leaderboards, scoring systems, and other fun elements. Integrating public repositories revolving

around software security and secure coding into in-game features adds an innovative and different approach to serious games about software security that has not been done previously.

# Chapter 3

## Methodology

### 3.1 Process

The software engineering methodology used for this project is adapted from agile development methodology [26]. The different project phases have been divided into sprints with tasks/activities, similar to user stories per sprint, to help keep track of what is done and what is next on the agenda. The supervisor has been acting as a light version of a product owner, primarily helping with prioritizing needs, defining a vision, and reviewing and giving feedback on the work regularly.

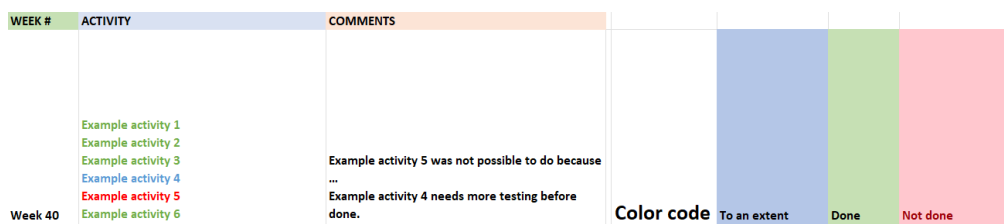


Figure 3.1: Weekly Planning

Meetings have taken place every other week or every week. A plan made in excel was used throughout the project, where weekly activities got updated after meetings with the supervisor. The plan works similar to a product backlog where activities are listed for all prior weeks and filled with activities

for the following weeks. The activities can vary, some examples of activities are:

- Read and look into question-and-answer style for quiz-based games.
- Fix database so that it works with Heroku.
- Allow adding image as question.
- Change design of quiz creation page.
- Finish draft 4.

Comments have been used, mainly for help in finishing activities, remembering valuable sources, and general comments for the activities. In the weekly meetings, activities are discussed, what is done, what is not done, and what is still in progress. A part of the meeting works as a sprint review, where implemented features are reviewed and presented to the supervisor. This helps look into what is working or not working and what needs to change and further discuss approaches to the activities. Each activity is color-coded as it is done, not done, or if done to an extent and still a work in process. If an activity is done, it is left in the plan for the week it was done, with the color code green to distinguish it from the other activities easily. If an activity is not done, it is discussed why, options to replace the activity, or whether it is better not to continue the activity altogether. The plan is divided into four different phases:

- Literature review/research phase.
- Implementation, design and creating prototype.
- Evaluation of prototype.
- User survey.

For phase 1, the main object was to gather resources, create research questions, do literature reviews, look into related work, and start working on design documents for the prototype. Phase 2 focused on creating a workable prototype of the quiz-based software security game, integrating the differ-

ent public repositories revolving around software security and secure coding into the quiz game. Phase 3 focused on evaluating, testing, and continuously improving the quiz game to meet the expectations. Phase 4 focused on conducting user surveys on the correct target group of the quiz game and evaluating the results gathered.

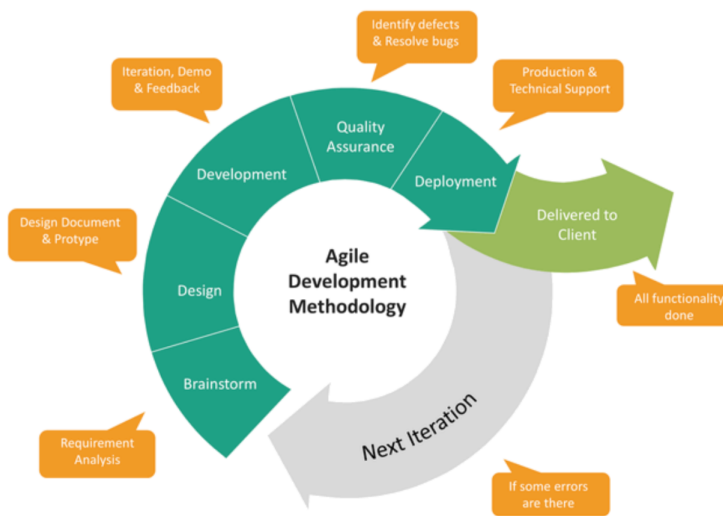


Figure 3.2: Agile development process  
Source: Copied from [27]

Each timeline between meetings can be seen as a sprint, where each session starts with a demo and feedback, followed by a requirement analysis for the upcoming weeks of development work. During the demo, the supervisor is presented with work that has been done since the last meeting. How new features are working and what ideas are behind them. After a demo, the supervisor gives overall feedback and further discussion on what needs to be improved, implemented, and changed until the next meeting.

## 3.2 Prototype

To answer the research questions, a prototype is first developed. This thesis uses educational gamification, which is a technology-based method [11], to

build a prototype where the quiz-based software security game is created as a prototype from requirements and design documents.

Figma was used to visualize and design how the quiz game should look before starting the development phase and how it should react to user inputs. Making a prototype makes it easier to communicate ideas and picture how the quiz should look and how each feature should look and work before having a product or results to show.

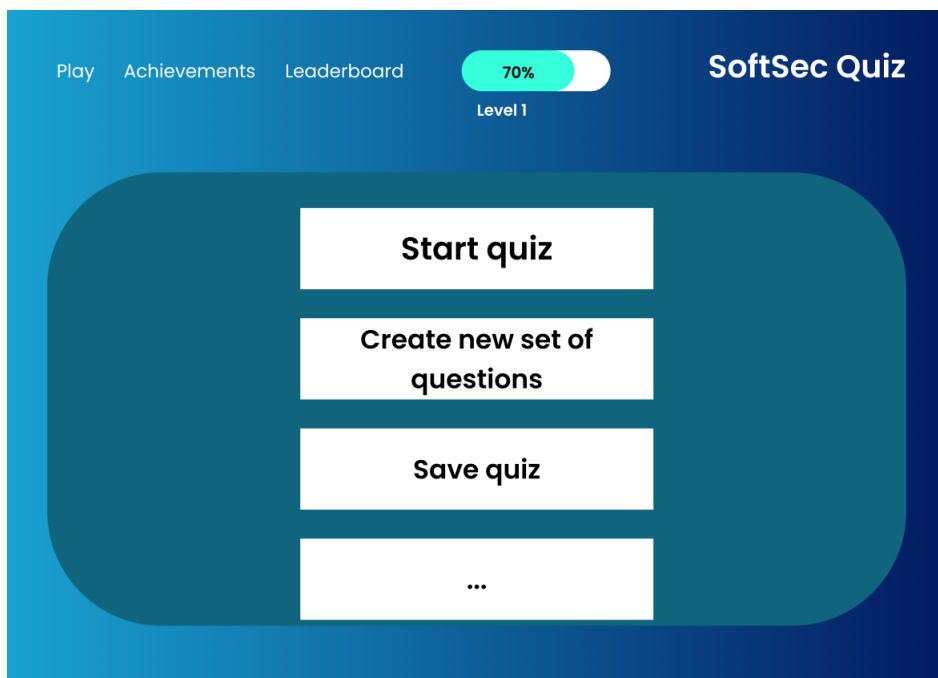


Figure 3.3: Menu prototype

Figure 3.3 shows what the main menu was envisioned to look like before beginning the project's development phase.

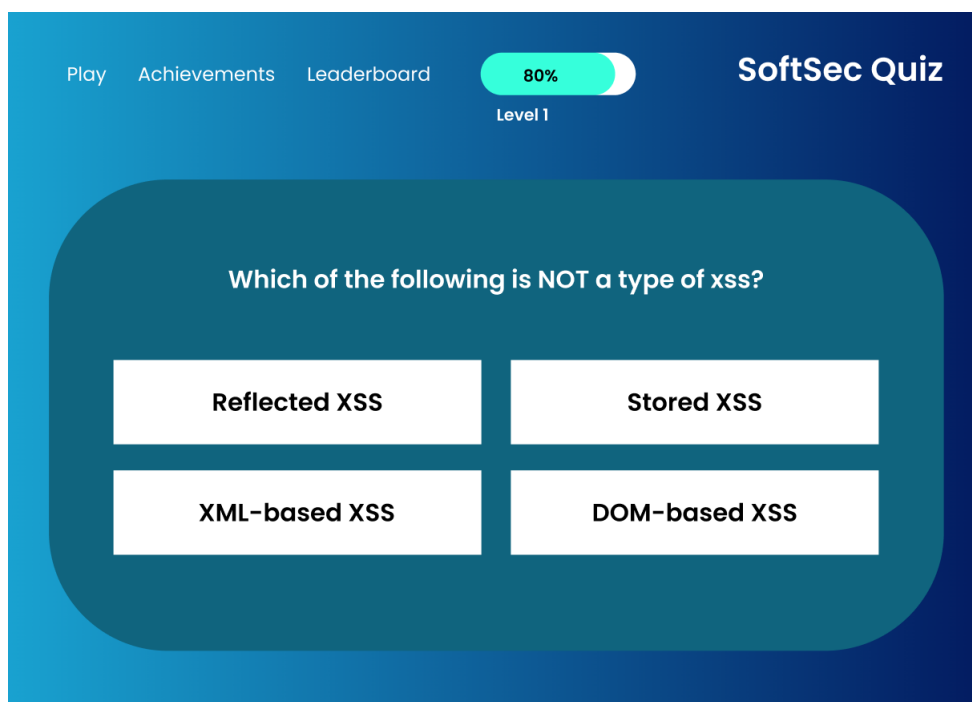


Figure 3.4: Quiz question - Play view created with Figma

Figure 3.4 shows how the 'play' part of the quiz was envisioned, with a question and four answer options to answer the question, only one of them being the correct answer. The idea behind the bar with a percentage score of '80%' is a scoring system where the user would level up after answering several correct answers. Leveling up would gradually increase the question's difficulty per level, continuously challenging the user once they got better at the selected topic. As seen in the final prototype of the quiz game, the design is very much like the sketches created early in Figma, as it is easier to build something with helpful illustrations than to make it only on the go, only from ideas and thoughts. Figma also has a prototype feature, which allows for clickable and interactive demos directly inside the website where the prototype is designed and conceptualized. This was helpful when sharing early ideas with the supervisor that needed feedback or confirmation.

## 3.3 Evaluation

### 3.3.1 Learnability, Usability, and Fun

To answer RQ1 and RQ2, a user survey was done to collect quantitative data by using the finished prototype of the quiz-based software security game. The user surveys are essential to measure the three main categories critical for the quiz game's success in self-study of software security. The questionnaire's three categories focused on capturing fun, learnability, and usability. The usability and fun goals address research question 1 (RQ1). The learnability goal addresses RQ2. The user survey respondents are all students studying to become software developers and are, therefore, a suitable user group and fit the target group. Most of the respondents are students of the supervisor, who teach a course on software security. Only five out of twenty-two respondents to the user survey are classmates. They were asked to respond in order to get some more respondents. Since most users are not classmates, the answers will be less biased or affected by kindness and anxiety about hurting feelings.

The supervisor's class is presented with the quiz game beforehand to collect feedback and ensure enough respondents answered the user survey. The presentation mainly introduced them to the game and its idea and let them know what was required of them to answer the user survey. To not influence the answers in the user survey, it was important not to give information about how they should play the game but rather only give them an overview of what to expect. After presenting the quiz, the class was asked to play through a quiz of ten questions on a few different topics revolving around software security. They were asked to play the quiz two times and record their scores, as this data is of interest in the evaluation process and one of the questions for the survey. After finishing playing the quiz, the class answered the user survey.

Since capturing the three main categories: fun, learnability, and usability, is of the highest importance, Goal Question Metric (GQM) [28] was used as a method to design the questionnaire. GQM is a systematic approach used for measuring software quality [29]. Following the GQM approach made



finding quantitatively measurable metrics easier. GQM has three levels: the conceptual level, where specific goals are identified. The operational level, where the questions for each goal are created to investigate how the goals should be accomplished, and thirdly; the quantitative level, where metrics are constructed to answer the questions related to their specific goal. The three goals set are: How to capture fun, learnability, and usability for the quiz game. The goals are set up individually in each column of an excel sheet, with their related questions, followed by the metrics for each question. Questions are constructed to capture the goals individually. An example is seen in table 3.1, where the goal of learnability is investigated, the questions to capture learnability are made, and then the metrics to measure it quantitatively are found. The same applies to the two other goals, usability and fun.

Table 3.1: Goal Question Metric example

Goal: Learning software security	
Question:	Metrics:
Did you learn more about software security from playing through a quiz?	Metric 1: I had prior knowledge on the topic presented in the quiz. Metric 2: My knowledge of the security topics presented in the quiz improved by playing the quiz.
Did the lifeline 'call a friend' help you learn more about software security and where to find information about the topic?	Metric 1: The search feature provided me information that helped me answer the question. Metric 2: The search feature introduced me to public software security content that I previously did not know about.
Did the 'Explanation for previous question' help you learn more about software security?	Metric 1: The explanation gave me more information that helped me better understand the answer to the question. Metric 2: The explanation helped me to improve my answer in the next round.
Do you consider the quiz game to be a good tool to learn about software security?	Metric 1: The quiz game made me learn more about software security. Metric 2: The quiz game is a good tool for learning about software security. Metric 3: I learn more about software security from playing the quizzes than other methods of learning.

The Likert scale is used for most of the questions in the user survey to measure the metrics quantitatively. The scale goes from one to five:

- 1: Strongly disagree
- 2: Disagree
- 3: Neither agree nor disagree
- 4: Agree
- 5: Strongly agree

I had prior knowledge on the topic presented in the quiz.

	1	2	3	4	5	
Strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly agree

Figure 3.5: Question 3

The metrics created using the GQM method are the same that the users had to answer using the Likert scale in the user survey. Figure 3.5 shows a question using the Likert scale from the user survey. All questions used this scale as their answer, except for three questions where one was optional feedback to get some qualitative feedback from users that were willing to leave a comment with their opinions. The other two asked what they scored in rounds one and two of the quiz.

The questions the users were asked to answer using the Likert scale in the user survey, categorized by the goals they were to capture, are presented in tables 3.2, 3.3, and 3.4 found below:

Table 3.2: Learning Software Security Metrics

Learning Software Security
I had prior knowledge on the topic presented in the quiz.
My knowledge of the security topics presented in the quiz improved by playing the quiz.
The search feature provided me information that helped me answer the question.
The explanation gave me more information that helped me better understand the answer to the question.
The explanation helped me to improve my answer in the next round.
The quiz game made me learn more about software security.
The quiz game is a good tool for learning about software security.
I learn more about software security from playing the quizzes than other methods of learning.
The lifeline 'Ask the audience' helped me quickly reason about the answer.

Table 3.3: Fun/Enjoyment Metrics

Fun/Enjoyment
I enjoyed playing the quiz.
I would like to try it again, several times.
I liked using the 'shield' lifeline.
I liked using the '50/50' lifeline.
I liked using the 'Ask the audience' lifeline.
I liked using the 'Call a friend' lifeline.
I got more motivated to score better because I wanted to place better on the leaderboard.
The scorebar and levels motivated me to try again to score better than last time.

Table 3.4: Usability Metrics

Usability
It was easy to understand the rules and play the game.
I understood the rules and how to play from using the in-game help button.
It was easy to use the 'shield' lifeline.
It was easy to use the '50/50' lifeline.
It was easy to use the 'Ask the audience' lifeline.
It was easy to use the 'Call a friend' lifeline.

### **3.3.2 How to Integrate Public Information Security Sources to Answer Questions Automatically**

To answer RQ3, the prototype incorporates the 'ask the audience' lifeline integrated with public information security sources and answer explanations. Tests were done to evaluate how accurate the lifeline 'Ask the audience' was and how well it did in predicting the correct answer out of the four answer options given per question in the quiz game. The quiz used for evaluating the lifeline is the same as in the previously mentioned user survey done on students. The lifeline was used for each question in a quiz of ten questions. The lifeline was used for each question, and the answer with the highest percentage score from the audience was picked as the submitted answer. This was done to see how many times it predicted the correct answer out of ten questions. When the audience scored more than one answer as the highest, meaning two or more answers had the same percentage score, the answer submitted was selected at random between these options. The results for each question were recorded and can be seen in section 5.2.4.

### **3.3.3 Quiz Questions**

The supervisor prepared the quiz questions and spread them across different security topics such as secure coding and cryptography. Ten questions allow the players to experience all the in-game features and learn how the game works. Ten questions are considered a good enough number that the participants can answer without overburdening them, and according to Jot-form [30], a quiz typically has ten or fewer questions. Considering there are four lifelines, ten questions seem appropriate for the learning outcome, as the player has to answer 60% of the questions independently.

# Chapter 4

## Design and Implementation

### 4.1 Architecture

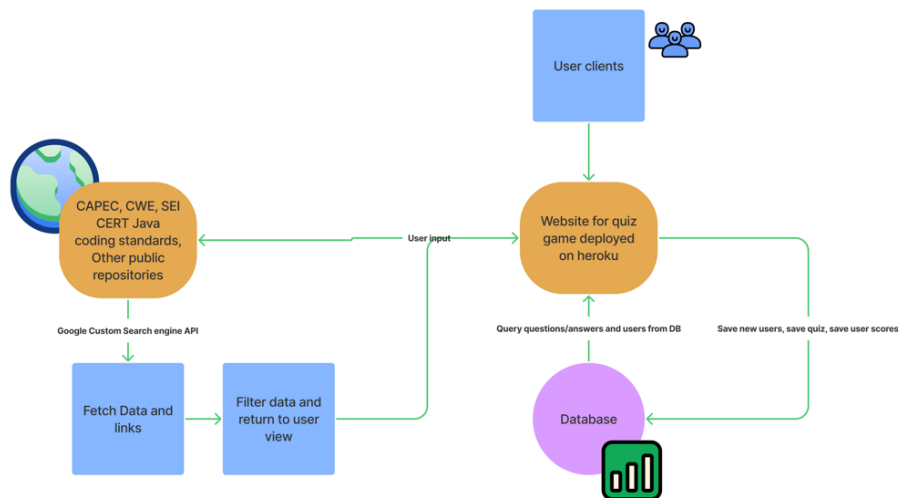


Figure 4.1: Simplified architectural overview

The application is built with Spring Boot as the backend framework and Java as the programming language. Spring Boot framework allows fast development and gets an application running quickly with few challenges. The web application is deployed to Heroku [31] to make it more easily accessible for the end-user. Heroku is a Platform as a Service (PaaS) that runs the appli-

cation on a cloud platform. Using Heroku allows the end-user to access the application through a URL link, which simplifies the use compared to having the end-user run the application locally. For automatic deployment of the most recent web application version, GitHub automatically pushes code to Heroku when new code gets pushed to the GitHub repository. Deploying it means Heroku will build and deploy every web application version that gets pushed to the specific GitHub branch. Structured Query Language (SQL) databases are being used to store data, an H2 database locally, and a Heroku extension called Heroku Postgres to store data when the web application is accessed through the Heroku deployment. To manage, view, fetch, and edit data inside the databases, Java Persistence API (JPA) repositories are used. These have been very useful as it makes it possible to create queries in Java language that is more compact and easily readable than SQL queries. For the front-end it is mostly used HyperText Markup Language (HTML) and vanilla Cascading Style Sheet (CSS), with some JavaScript (JS). Thymeleaf, a Java library template engine, is used to parse and render data/objects produced by the application to template files. Using Thymeleaf allows for easy rendering of data, collections of data, and binding data to objects.



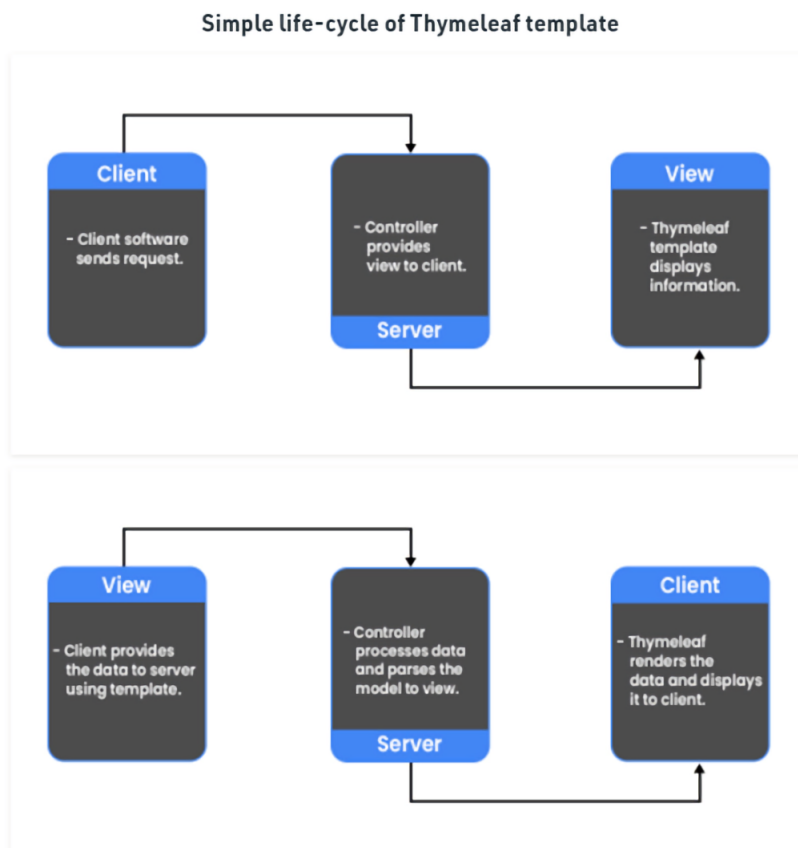


Figure 4.2: Simple life-cycle of Thymeleaf template

### 4.1.1 Spring Boot

Spring Boot [32] allows the creation of Spring-based applications in an 'easy' way that runs with minimal effort. The Spring initializer makes for an effortless start to any project by simply picking out dependencies according to the type of project. Spring Boot does a lot of the configuration for the user by being a framework built on top of the Spring framework, which enables the user to quickly "bootstrap" a Spring application from scratch.

### 4.1.2 Spring JPA

Spring Data JPA [33] is provided by Spring to make it easier for developers to implement the data access layer. While the developer focuses on program-

ming the project's vision, Spring JPA will implement all necessary methods for accessing all the data from repositories for the developer. The developer only has to write the entities, repository interfaces, and custom finder methods, and Spring will automatically take care of the implementations. This reduces a lot of boilerplate code required by JPA.

## 4.2 Conceptualization Stage

To conceptualize fun, a few different educational games [25][34] and other quiz games [35] were investigated to gather inspiration for features that could be used for the prototype quiz game. Some features that were considered implementing consist of:

- Reward systems
- Lifelines
- Score
- Leaderboard
- Achievements

The end goal is to improve user engagement and motivation to continue playing by implementing such features [36]. Leaving these elements out of the game would leave the game bland and not any different from a multiple choice test. The idea behind adding a leaderboard is to have the users come back to play more, to either beat their previous score or their classmates. This increases motivation to continue playing more than once. A scoring system while answering the questions is a fun way for the users to keep track of their correct and wrong answers. Lifelines work in a way that helps users to continue playing; instead of being stuck and not knowing the answer, the user can use a lifeline that will help them reason about the correct answer and move on to the next question. Achievements could be used as a motivation to come back and play more, for example, if the users got an achievement for playing multiple days in a row, scoring better than 50% on a quiz, playing

ten quizzes, etc. All these elements help encourage the users to play, learn and have fun at the same time.

The idea to conceptualize learning is to integrate public cyber security content in some of these elements, like 'call a friend' and 'ask the audience' lifelines. By incorporating these websites that cover software security and secure coding with lifelines, the user is introduced to content that will help them answer the quiz questions in a fun way. Because the lifelines are only allowed to be used once per quiz, they will only work as help in need when stuck. This forces the users to play the game and not only use the lifelines forever but only whenever they are stuck.

### 4.3 Integration of Public Security Content

Integrating public repositories containing data on threats, mitigation strategies, weaknesses, secure coding, and in general cyber security content has been a big part of this thesis. It has previously been tried in previous software security game STIX and Stones [9], but in a different way. The way the public repositories are integrated into the quiz game created for this thesis is innovative and has not been previously done. Using them in lifelines such as 'Call a friend' and 'Ask the audience,' the users are introduced to them through the game and can use them inside the game to search for answers or reason about the correct answer.

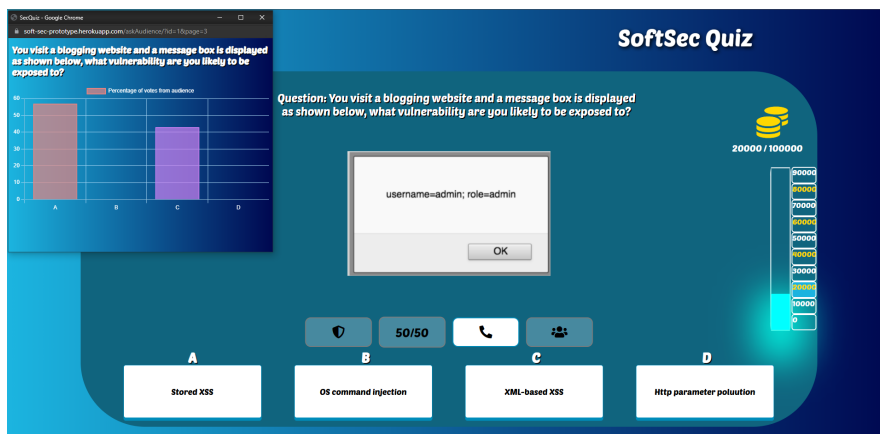


Figure 4.3: Ask the audience lifeline

'Ask the audience' is a lifeline where inspiration is taken from the game 'Who Wants to Be a Millionaire.' From the TV show, it works in a way where the audience takes their votes on what answer option they believe to be the correct one, and the player of the game show is presented with a bar graph with percentage scores given for each answer option. The player then has to choose based on the results presented, usually picking the highest scored option. The quiz game differs in that it doesn't use actual humans to cast a vote on the answer options they believe to be the correct; it uses Term Frequency (TF) [37] and similarity measure. First a custom search JavaScript Object Notation (JSON) Application programming interface (API) is used to fetch data from a selection of public repositories that specialize in cyber security and secure coding. The custom search is based on the question given in the quiz game, and fed into the custom search JSON API which returns data in JSON format. This data, with the explanation (as seen in figure 5.3) for the belonging question is fed into a TF function. TF is an information retrieval metric for determining features in document classification problems [37]. Term frequency is computed by tokenizing each document in a document corpus and collecting the number of occurrences (frequency) of the unique terms in the entire corpus. By removing stop words that doesn't help gather information from the document, processing time is saved and there is more focus on the important information in the document. Cosine similarity, which is a similarity measure, is used to measure the similarity between two documents. It calculates the cosine of the angle between two vectors projected in a multi-dimensional space. Given two documents with vectors A and B, the cosine of the angle between them can be computed as:

$$\cos(A, B) = \frac{A \cdot B}{|A| \times |B|}$$

The bigger the cosine value, the higher the similarity between the two documents. Two documents that have been vectorized using TF are compared by computing their cosine similarity. This is done in the 'Ask the audience' lifeline to see which of the four answer options has the highest similarity with the search result using the custom search JSON API and the explanation to

the question given. The results are then presented to the user in a bar graph that shows each answers percentage as to how high similarity they have. The similarity score is converted to show percentage score compared to each other answer option, meaning if only one of the answers has any similarity with the search result and explanation, it gets 100% of the 'votes' from the audience. In figure 4.3, option A has received over 55% of the votes, C has over 40%, and B and D have received 0%. It should be easier to answer the question and reason around the correct answer with this information.

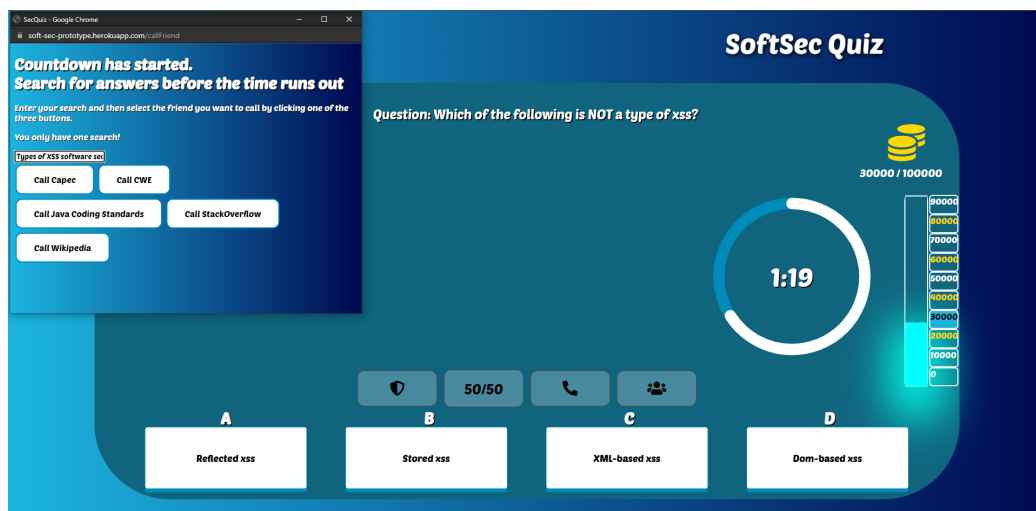


Figure 4.4: Call a friend lifeline - Search box with user input and selection of 'friend'

'Call a friend' is a lifeline that also has taken inspiration from the game show 'Who Wants to Be a Millionaire' and tweaked it to integrate public cyber security content into the game. Instead of calling an actual human, public repositories specializing in software security, secure coding, and coding, in general, are used as 'friends.' This is done using custom google search engines for each website. When using this lifeline, the user has to type in their search word and select their 'friend' before being redirected to a search result page where they can look through and open links. This way, the player can search for up-to-date content from carriers specializing in cyber security to find the answer to their in-game question. When the lifeline has been clicked, a timer of two minutes will start counting down, and the counter will show up, as

seen in figure 4.4. Suppose no answer has been submitted before the time runs out. The user will lose the opportunity to answer and be redirected to the next question. If the life option is active for the particular quiz, the user will lose a life when the time runs out. By adding a countdown timer of two minutes and not allowing for more than one search, the user is prohibited from continuing using the lifeline for other questions.

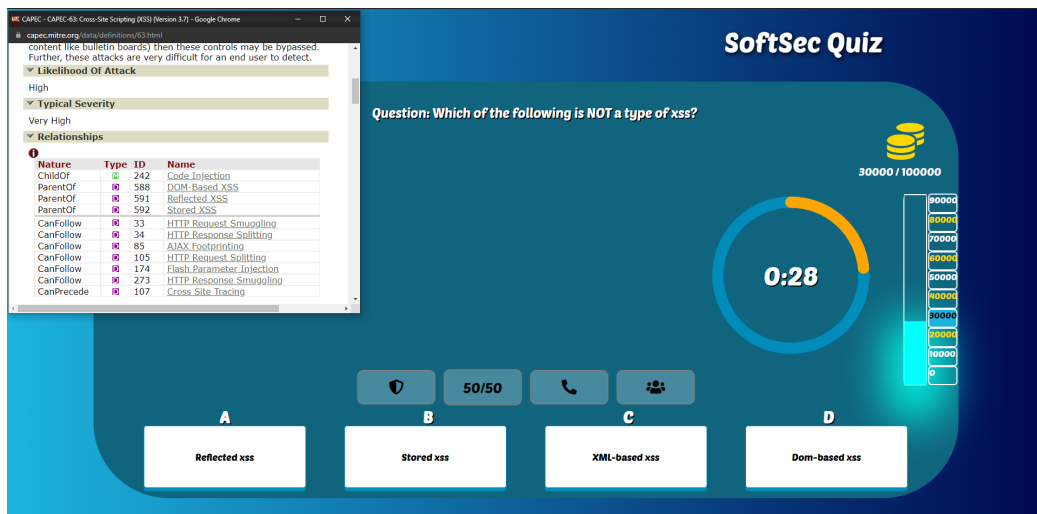


Figure 4.5: Call a friend lifeline - CAPEC as selected friend and result page

Figure 4.5 shows how the lifeline can be used to find the correct answer. For this example, a search with 'Types of XSS Software Security' as the input was done, and CAPEC was used as a 'friend.' As seen in figure 4.5, DOM-based XSS, Reflected XSS, and Stored XSS can be found in the relationships table from CAPEC on Cross-Site Scripting (XSS). Therefore, the correct answer to this quiz question should be XML-based XSS, which is true.

The 50/50 lifeline uses a randomization function to remove two of the wrong answers, leaving only the correct answer and one wrong answer. This can be useful if the player is unsure of the correct answer, leaving them with a 50% chance of getting the correct answer if they were to guess between the two answers.

## 4.4 Quiz and Question Creator for Admin Users

This section will introduce the administrator part of the quiz-based game. This is where the admin users can create and edit or delete quizzes and questions.

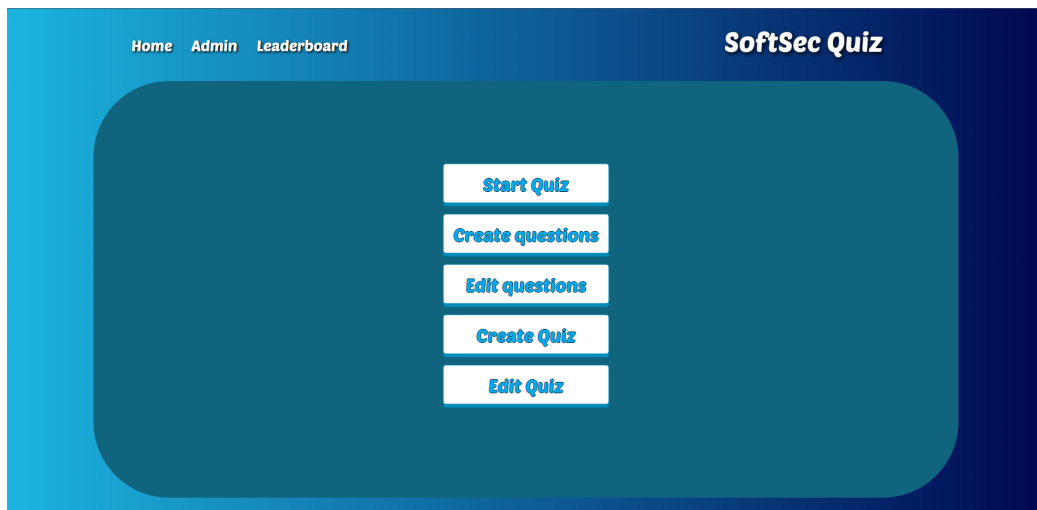


Figure 4.6: Administrator menu

Figure 4.6 shows the administrator's main menu. It presents the admin user with a set of buttons that directs them to different views where it is possible to play quizzes, create questions, create quizzes, or edit them. There is also a header menu available from all the views. From the header menu, it is possible to go to the regular users' 'home,' which is where they select what quiz to play, the admin 'home,' which is the menu from figure 4.6, and lastly the leaderboard. The leaderboard view will direct the user to a table presented with all the quizzes; Here, the user can search for quizzes by name or ID and select which quiz leaderboard to see.

Figure 4.7: Create question

Figure 4.7 is where the admin user, preferably someone with knowledge of the topic, can make questions by filling in the input fields. This view is presented when clicking the 'Create questions' button in the admin menu from figure 4.6. When creating a question, adding an image to complement the question is possible. All fields except the tag need to be filled for the question to be valid and not return an error message. Since the quiz game offers four answer options per question, three wrong, and one correct answer has to be filled for a complete question. The explanation field allows the question creator to give the quiz players an explanation of the correct answer and the question given. After answering a question, the explanation will be presented to the user in the 'game' part of the web application, as seen in figure 5.3. When creating questions, three helpful links will direct them to different websites; CAPEC, SEI CERT Oracle Coding Standard for Java, and CWE. If help is needed when constructing questions, the 'Need help?' button will offer a pop-up window similar to the one in figure 5.3 but helpful for constructing a question.



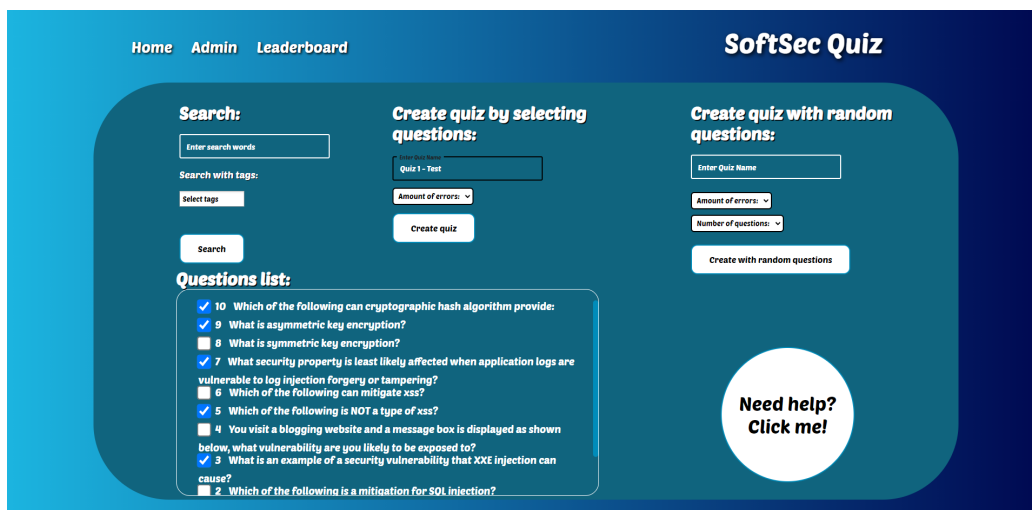


Figure 4.8: Create a quiz

When creating a quiz, it is possible to search for questions either by tag, word(s) that the question contains, or both, and the questions matching the search will show up in the question list. There are two ways to create a quiz, either by selecting desired questions or randomly selecting a chosen number of questions. For method one: check the check-boxes of the desired questions to be added to the quiz, type chosen name for the quiz, and select the number of errors that the quiz should contain, either infinite or from one to five. The number of errors represents the amount of in-game 'lives' a user is given when playing the constructed quiz. Answering a question wrong will be penalized by losing a 'life' represented as hearts, as seen in figure 5.4. The quiz will end when the user runs out of 'lives.' For method two of creating a quiz by random selection of questions: Type quiz name, select amount of errors as in the previous method, and choose the number of questions wanted, from one to ten. The questions will be randomly added from the question list, so if it is desired only to have questions from one specific topic/tag, it can be done by selecting the desired tag. 'Need help?' works the same way as previously mentioned in the question creation section, where a pop-up window with helpful text will show up as shown in figure 5.3.

# Chapter 5

## Results and Discussion

The result of this research is a web-based quiz game integrated with public software security content. By using gamification elements such as leaderboards, score bars, and lifelines, the game has the potential to be an attractive tool for learning about software security. This chapter includes figures from the quiz-based software security game's finished prototype and explains different features. The results from the evaluations are presented and discussed, and the research questions are answered.

## 5.1 Quiz Game

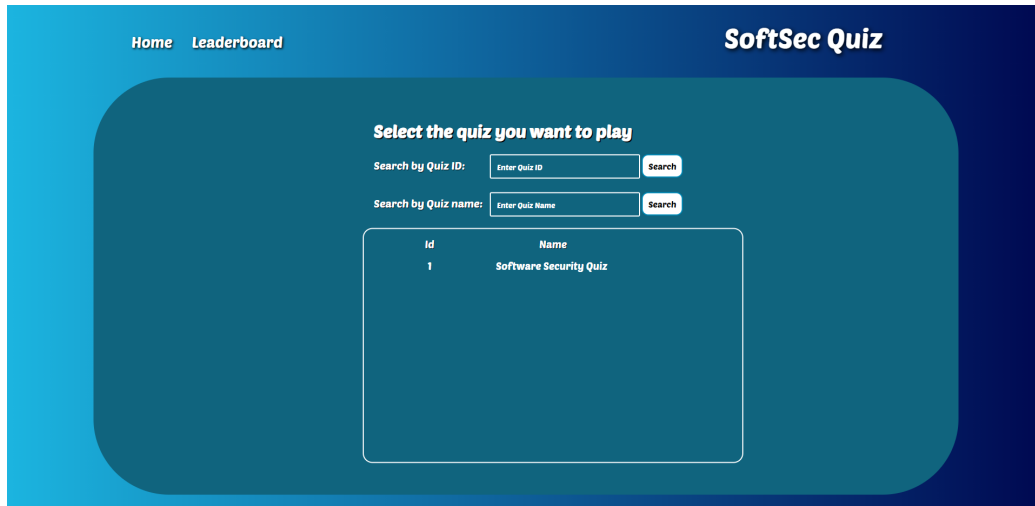


Figure 5.1: Search for a quiz by ID or name of the quiz and select what quiz to play

The 'Start quiz' button brings the user to a table listing all the quizzes created. Search by ID or name to find the desired quiz. When hitting search, the user is presented with a table containing only quizzes that match the search word, or if the search is done by ID, only the quiz with the searched ID will show up in the list. To play a quiz, simply click the name of it.

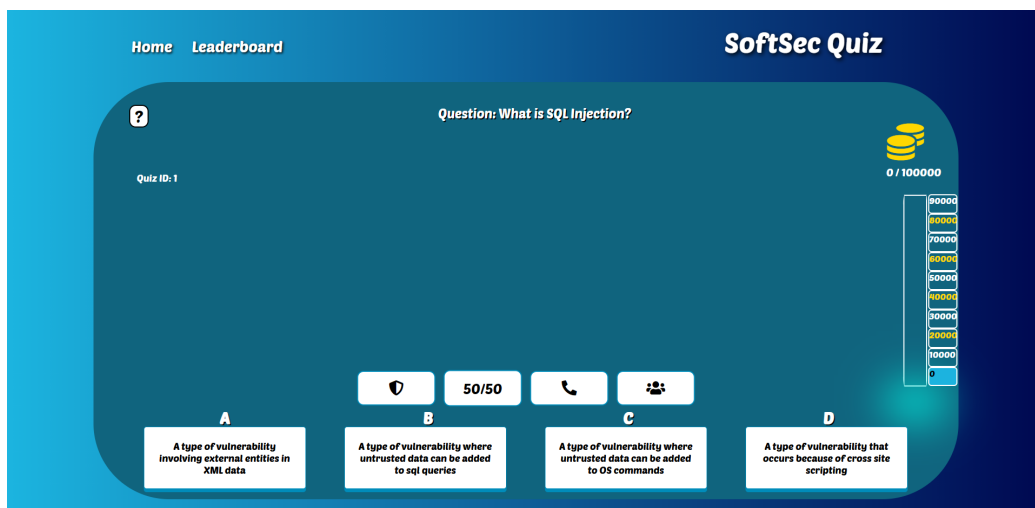


Figure 5.2: Quiz play

After selecting a quiz, a new window shows up, as shown in figure 5.2. This is the game view where the user is presented with a question, four answer options, A, B, C, and D, four lifelines, and a score bar. For each correct answer the score bar will raise by 10 000 points. As seen in figure 5.2, some of the scores are gold in color; they are plateaus that save the player from dropping score. If the user were to lose on a non-golden score, they would drop down to their previous achieved golden score. The player also starts with four different lifelines that can help them throughout the quiz differently. Each lifeline is only available for use once per quiz.

- Shield - Protects the player from losing a life if they were to answer wrong while the shield lifeline is active.
- 50/50 - Removes two wrong answers
- Call a friend - Allow the player to ask a 'friend' out of a selection of public repositories specialized in software security and secure coding, among other sites.
- Ask the audience - Asks the audience what they think is the correct answer. A bar graph with percentage scores for what the audience has 'voted' is the right answer will show up.

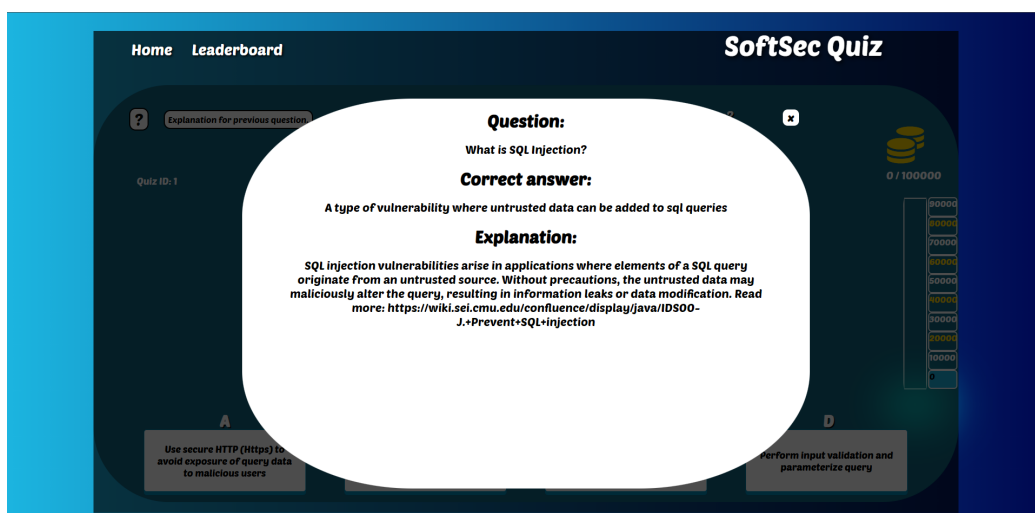


Figure 5.3: Explanation and correct answer for previous question

When an answer has been submitted, the next question will show up and there will be an option to read the explanation and correct answer for the previous question. The explanation will be shown in a pop-up window as seen in figure 5.3.

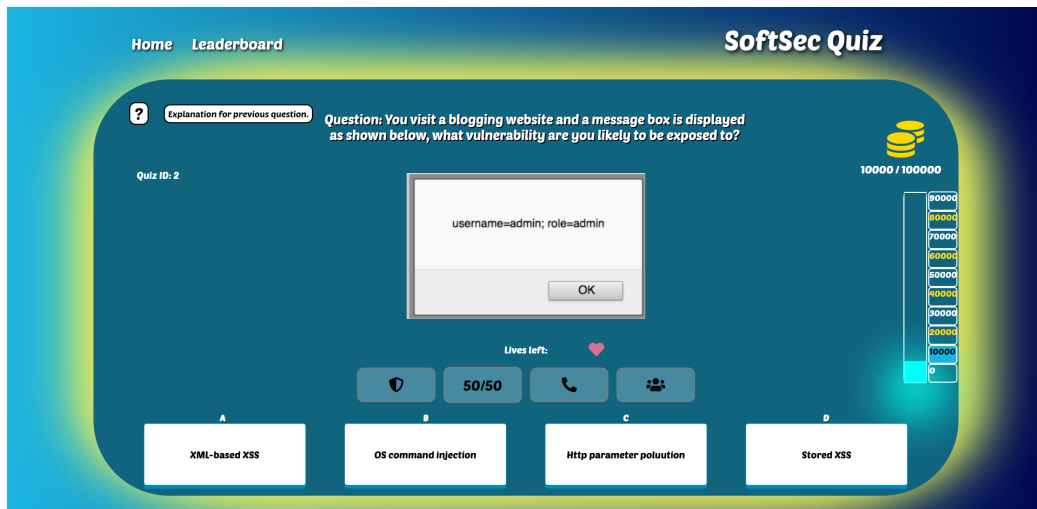


Figure 5.4: Shield lifeline

The 'shield' lifeline will activate a glowing effect to indicate the lifeline is active. While the 'shield' lifeline is active, it is impossible to lose a life if the submitted answer is wrong.

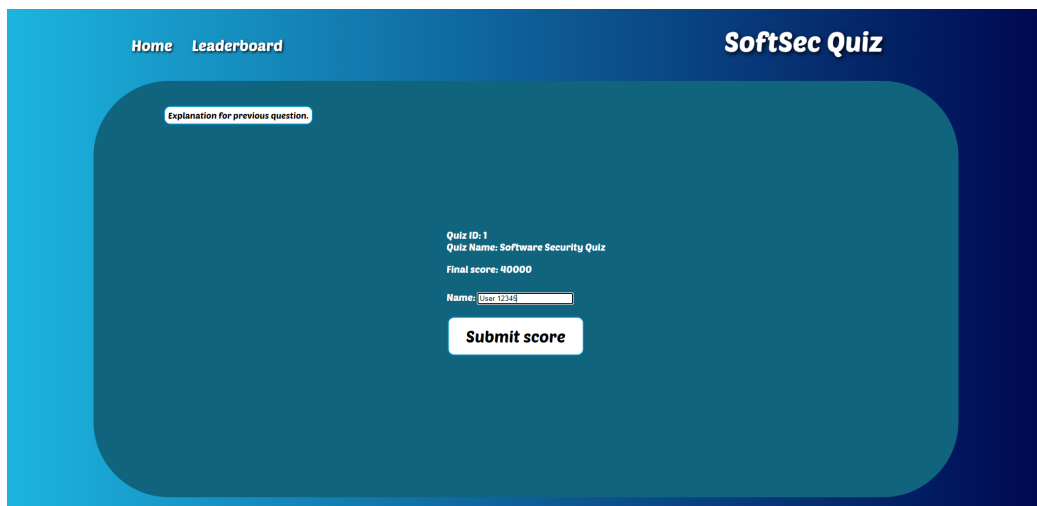


Figure 5.5: Submit your score page when quiz is done

The quiz finishes either by losing all lives or answering all the questions. When the quiz finishes, the user is redirected to the view in figure 5.5. The user can submit their score to the leaderboard by entering a nickname and clicking the 'Submit score' button.



User ID	Name	Score
52	x	100000
50	x	80000
15	Halldor	60000
65	Asd	60000
31	Moses	60000
53	Bollen	40000
39	bad-fred	40000
77	no	40000
51	Bollen	20000
16	Evan	20000

Figure 5.6: Leaderboard of selected quiz

There are leaderboards for all created quizzes, with the top ten scores listed with their submitted nickname.

## 5.2 Findings to Research Questions

This section introduces the user survey results answering RQ1 and RQ2 and the results from the evaluation done on the lifeline 'Ask the audience' that answers RQ3. The user survey is divided into three different categories: learnability, usability, and user enjoyment/fun. The results should tell a lot about the user experience based on these three categories, if they learned anything, if the quiz game is fun, and if it is easy to use.

## 5.2.1 Learnability (RQ2)

What was your score the first time you played the game?  
22 svar

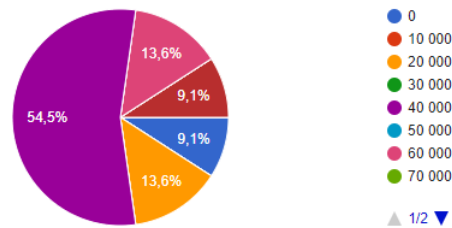


Figure 5.7: Results from round 1 of quiz play

Figure 5.7 shows the scores that the survey users achieved the first time they played the quiz game. The majority of the users (54,5%) scored 40 000 out of 100 000 possible points, which is also the average score:

$$\frac{880000}{22} = 40000$$

22,7% scored above average, and 22,7% scored below the average score. This is a good beginning, as it is possible to see if any improvement is made for round two.

What was your score the second time you played the game?  
21 svar

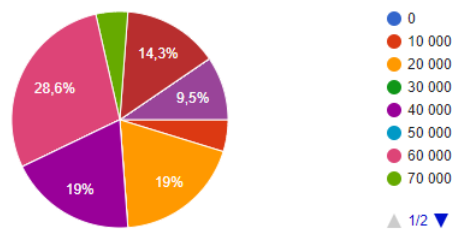


Figure 5.8: Results from round 2 of quiz play

Results from round two (see figure 5.8) of the users quiz scores show that there has been some improvement overall, where the majority of the users scored 60 000, whereas in round one the majority scored 40 000. However,

the average score hasn't changed much. The average score for round two is:

$$\frac{1020000}{21} = 48600$$

Round two scores show that the scores have increased some overall, but it is hard to tell if this slight increase in scores are because they learned something or just memorized the correct answer and used the lifelines for different questions than in round one. A proper experiment would be required to evaluate this goal. This is work for future study.

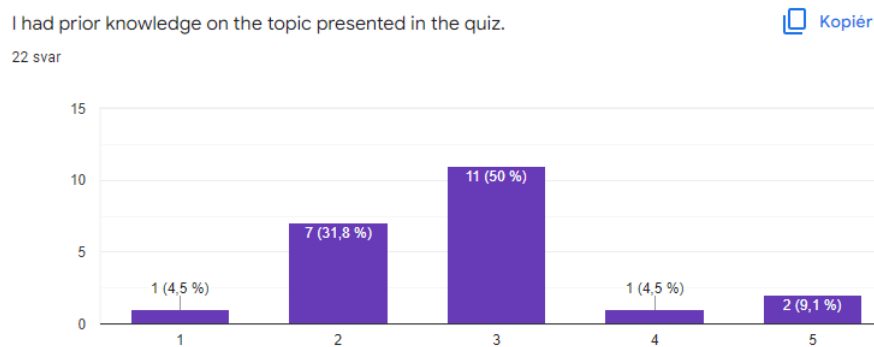


Figure 5.9: Results prior knowledge to topics presented in the quiz

To check the users background knowledge on the topics presented in the quiz, they were asked whether they had any prior knowledge on the topics (see figure 5.9). The majority of the users answered 'neither agree nor disagree', which can be interpreted as they had some basic knowledge but is not confident in their knowledge on the topics. The average is leaning towards 'disagree', which can be interpreted that the average user did not have prior knowledge to the topics presented.



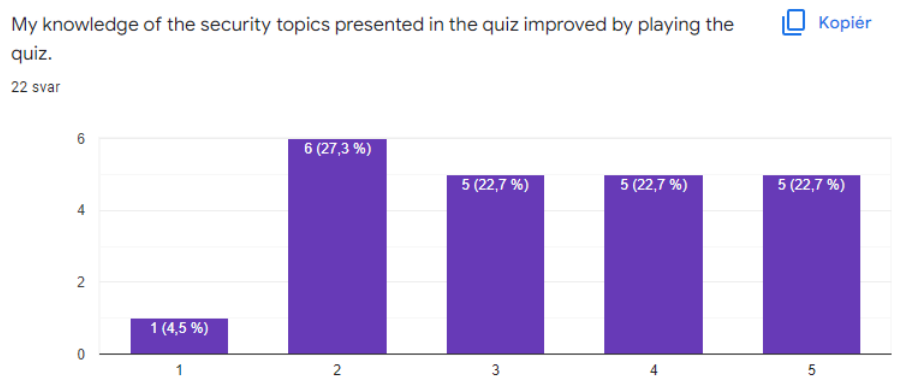


Figure 5.10: Did knowledge on topics presented improve?

Looking at figure 5.10 shows that the users knowledge in the presented topics improved after playing the quiz, at least for some of the users (45,4%). 31,8% of the users disagreed in that their knowledge in the presented topics increased from playing the quiz, and 22,7% of the users neither agree nor disagree. Seeing that almost 50% of the users did in fact learn from playing the quiz is a positive sign and support the hypothesis in that the quiz game can be used to motivate adoption for teaching software security.

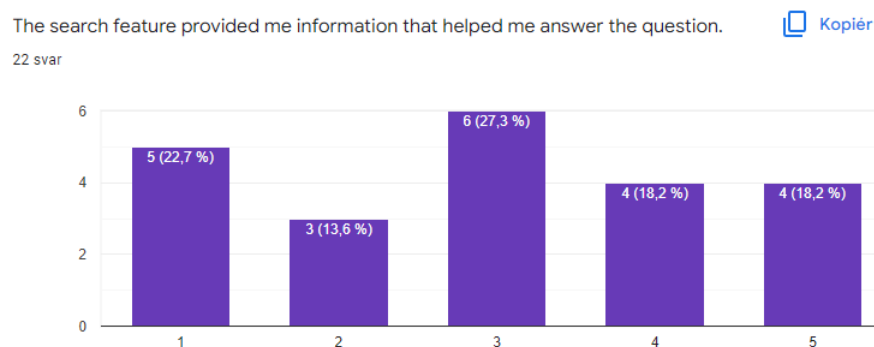


Figure 5.11: Was search feature helpful in answering questions?

The 'call a friend' lifeline had some mixed results, for some it helped them answer the question, whereas for others it didn't. During the session where the users took the quiz, there was some feedback on this lifeline that could be taken into account for further work. Some misunderstood the usage, meaning it is not as user-friendly as it was intended to be.

The explanation gave me more information that helped me better understand the answer to the question.

 Kopier

22 svar

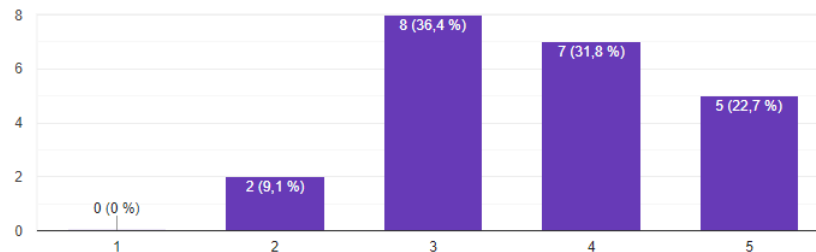


Figure 5.12: Results when asked if explanation was helpful to understanding the question

The explanation to the questions got positive feedback in that most (54,5%) of the users understood the answer to the questions better after reading it. Many (36%) of the users were indifferent to the explanation, whereas only two of the users disagreed in that it helped them understand the answer to the question better.

The explanation helped me to improve my answer in the next round.

 Kopier

22 svar

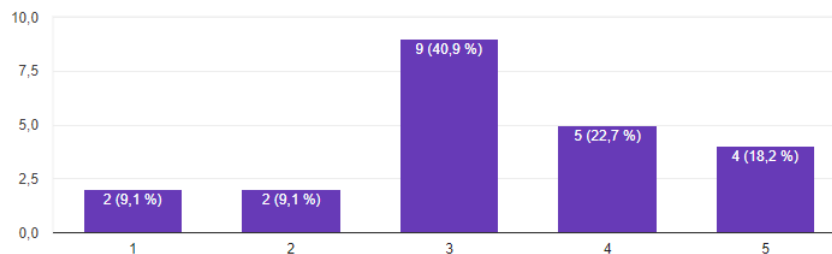


Figure 5.13: Did explanations help improve answers?

For some users (40,9%), the explanation helped them answer better in round two of playing the quiz game (see figure 5.13). The same amount of the users were indifferent, while two of the players disagreed, and two strongly disagreed in that the explanation helped them improve their answer for the next round.

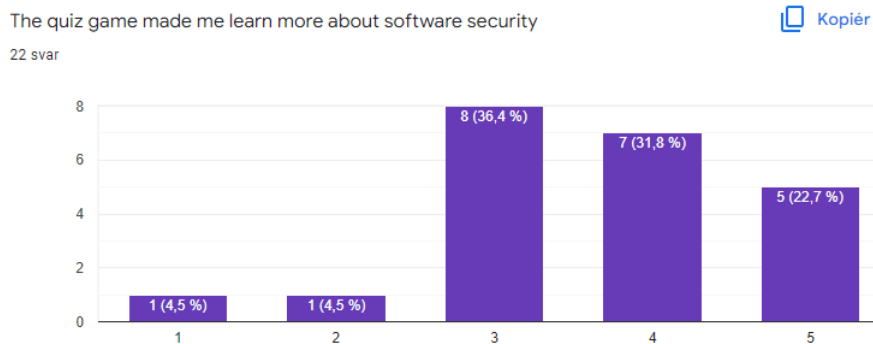


Figure 5.14: Results - Did the quiz game make users learn more about software security?

When asked if the quiz game made them learn more about software security (see figure 5.14), the majority answered that they agreed (31,8%) or strongly agreed (22,7%), whereas only two of the users (9%) disagreed or strongly disagreed. This again supports the hypothesis.

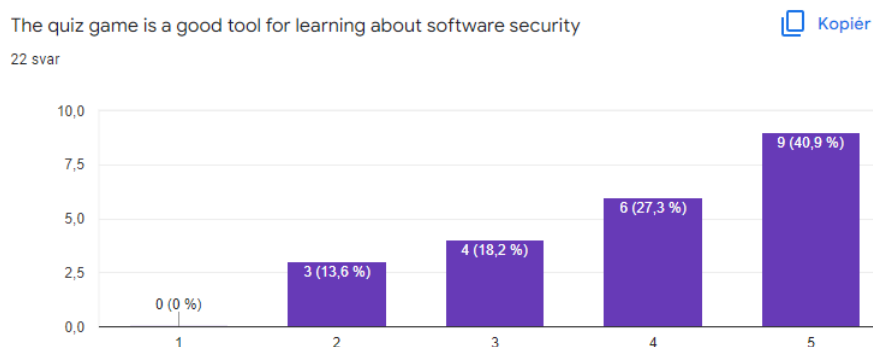


Figure 5.15: Is the quiz game a good tool for learning software security?

When asked whether or not they think the quiz game is a good tool for learning about software security (see figure 5.15), even more of the users agreed or strongly agreed (68%) that yes, they think it is a good tool for learning about software security. This result amplifies the support of the hypothesis. Only three out of twenty-two users disagreed, whereas four were indifferent. In fact, 'strongly agreed' was the most selected answer, with 40,9% of the users strongly agreeing.

I learn more about software security from playing the quizzes than other methods of learning

Kopier

22 svar

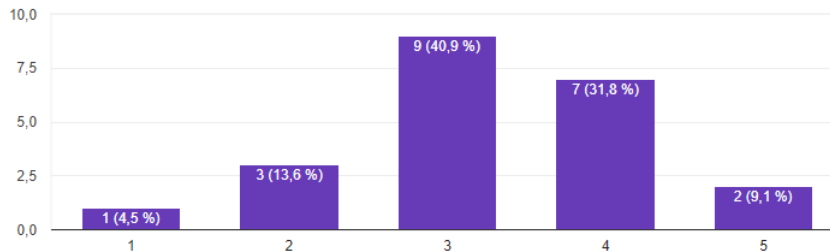


Figure 5.16: Quiz compared to other methods of learning

The results from figure 5.16 show that most users are indifferent when asked if they learn more about software security from playing the quiz game compared to other methods of learning. While some agree and some disagree, the results favor that the users agree that they learn more from playing the quiz game than other learning methods, with 40,9% of the users agreeing against 18,1% disagreeing.

The lifeline 'Ask the audience' helped me quickly reason about the answer.

Kopier

22 svar

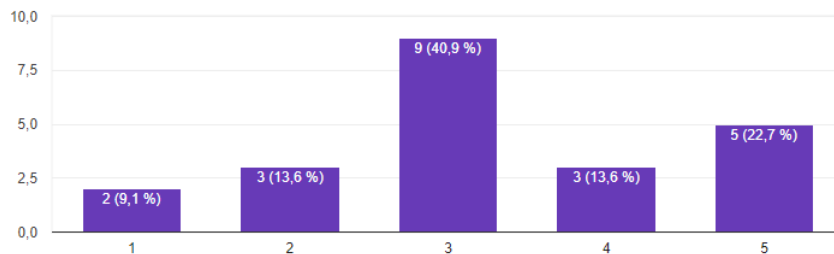


Figure 5.17: Ask the audience - Did the lifeline help users quickly reason about the answer?

The lifeline 'ask the audience' did not have much impact on how the users reason about the answer. Still, the results slightly favor that the lifeline did help them quickly reason about the answer, where about 36% of the users agreed or strongly agreed. 22,7% of the users did not agree, whereas 40,9% were indifferent.

## 5.2.2 Usability (RQ1)



Figure 5.18: Results - Easy to understand the rules and play the game.

Looking at the results for usability, it is clear that almost all users (81,8%) agreed or strongly agreed in that it was easy to play the game and understand the rules (see figure 5.18). Only one out of twenty-two users disagreed, while three were indifferent.

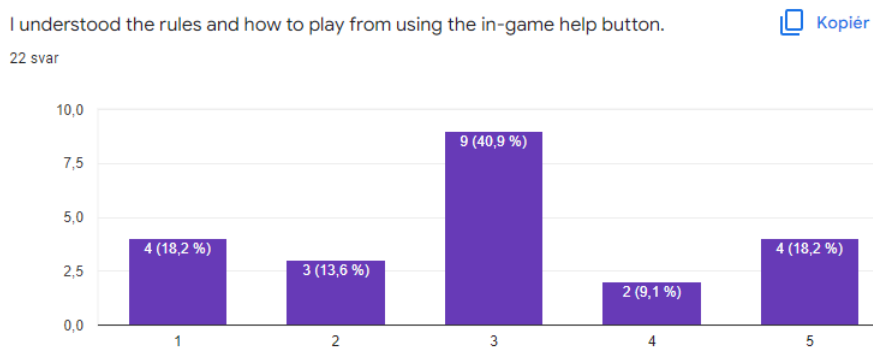


Figure 5.19: Results - Did the users understand the rules and how to play from using the in-game help button?

The results are indifferent whether or not the users understood the rules on how to play the game from the in-game help (5.19).

It was easy to use the 'shield' lifeline.

Kopier

21 svar

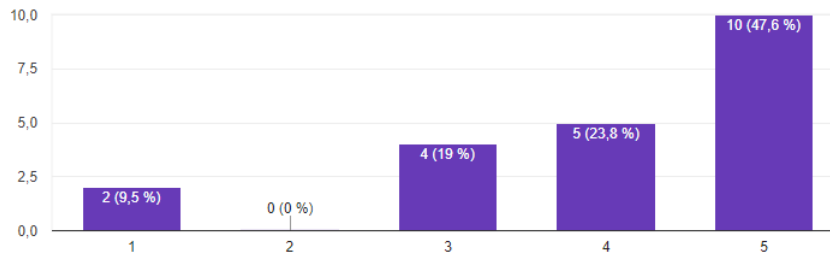


Figure 5.20: Results - Easy to use the 'shield' lifeline

When asked if the lifelines were easy to use, most users agreed or strongly agreed that the lifelines were easy to use (see figures 5.20, 5.21, and 5.22). However, the results were different for the 'call a friend' lifeline (see figure 5.23). Five (22,7%) of the users were indifferent, while eight (36%) strongly disagreed or disagreed, and nine (40,9%) agreed or strongly agreed that 'Call a friend' was easy to use.

It was easy to use the '50/50' lifeline.

Kopier

22 svar

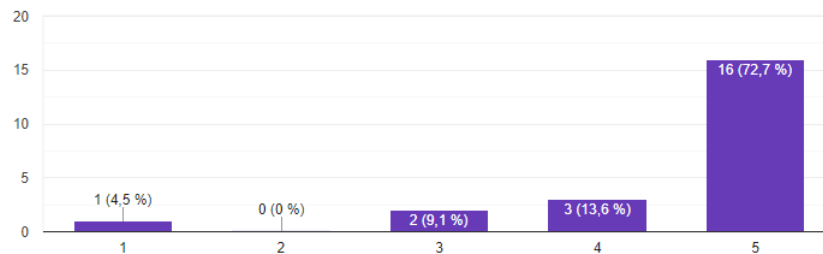


Figure 5.21: Results - Easy to use the '50/50' lifeline

It was easy to use the 'Ask the audience' lifeline.

Kopier

22 svar

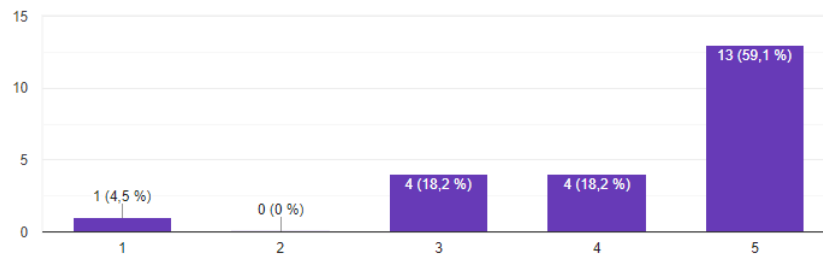


Figure 5.22: Results - Easy to use 'ask the audience' lifeline

It was easy to use the 'Call a friend' lifeline.

Kopier

22 svar

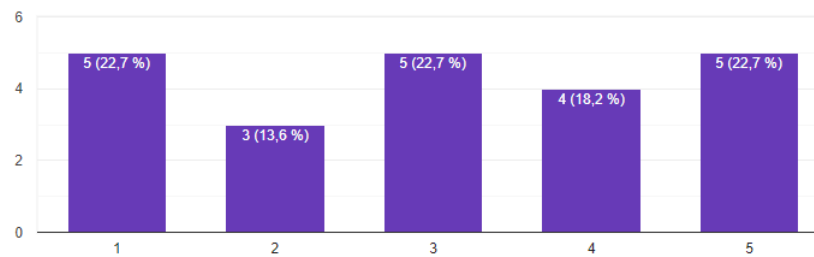


Figure 5.23: Results - Easy to use the 'call a friend' lifeline

### 5.2.3 User Enjoyment/Fun (RQ1)

I enjoyed playing the quiz.

Kopier

22 svar

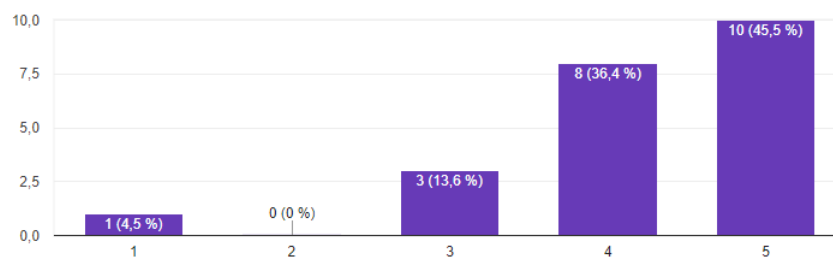


Figure 5.24: Results show that the majority of the students enjoyed playing the quiz

When looking at the results for user enjoyment/fun, only one out of twenty-two users disagreed (strongly) in that they enjoyed playing the quiz game (see

figure 5.24). Three of the users were indifferent, whereas eighteen (81,8%) of the users agreed or strongly agreed in that they enjoyed playing the quiz game.

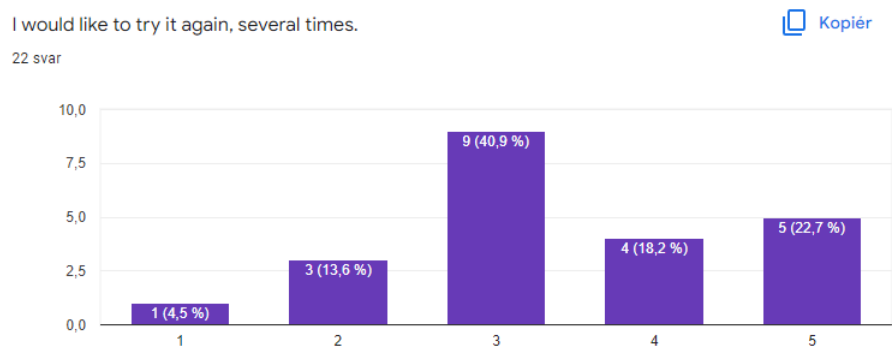


Figure 5.25: Results when asked to try again

When asked if they would like to try it again several times, most users neither agreed nor disagreed (see figure 5.25). However, more people (40,9%) agreed or strongly agreed than people disagreed (18%) in that they would like to try it again.

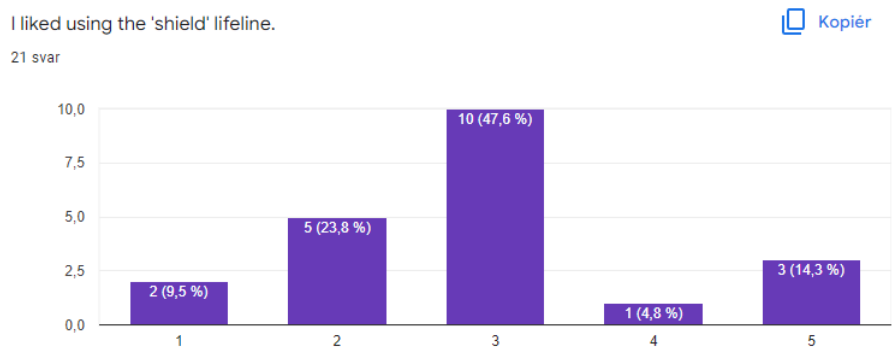


Figure 5.26: Results for 'shield' lifeline

Only four respondents agreed or strongly agreed that they liked using the 'shield' lifeline. This lifeline only has any use if the quiz played has 'lives' active, as it protects the player from losing a life when answering a question incorrectly. For the particular quiz played by the users answering the user survey, this was not relevant as they had an unlimited number of 'lives.' This can have impacted the results for this metric.



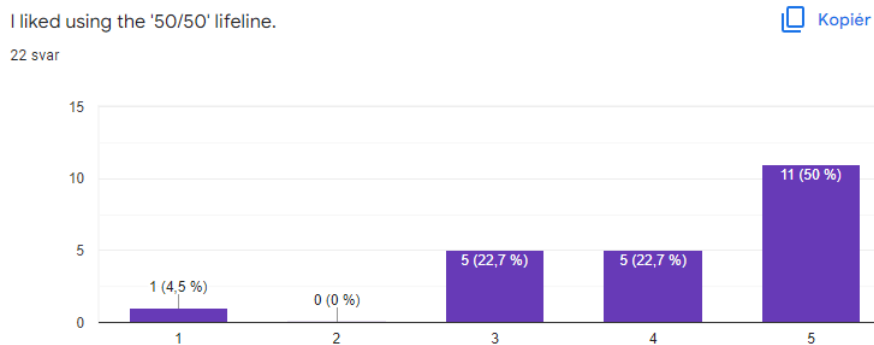


Figure 5.27: Results for '50/50' lifeline

The majority of the respondents liked using the '50/50' lifeline, as the results show in figure 5.27. Only one out of twenty-two strongly disagreed, and five out of twenty-two were indifferent. 50% of the users strongly agreed in that they liked using this lifeline, and 22,7% agreed.

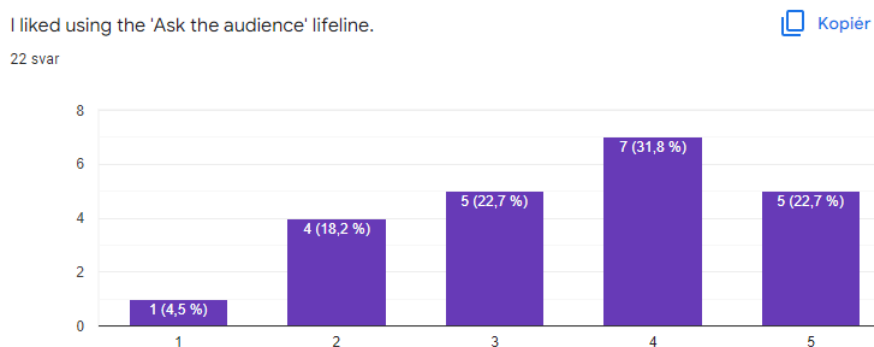


Figure 5.28: Results for 'Ask the audience' lifeline

'Ask the audience' lifeline got some mixed results from the respondents when asked if they liked using this lifeline (see figure 5.28). 54,5% of the users agreed or strongly agreed in that they liked using it, 22,7% were indifferent, and 22,7% disagreed or strongly disagreed.

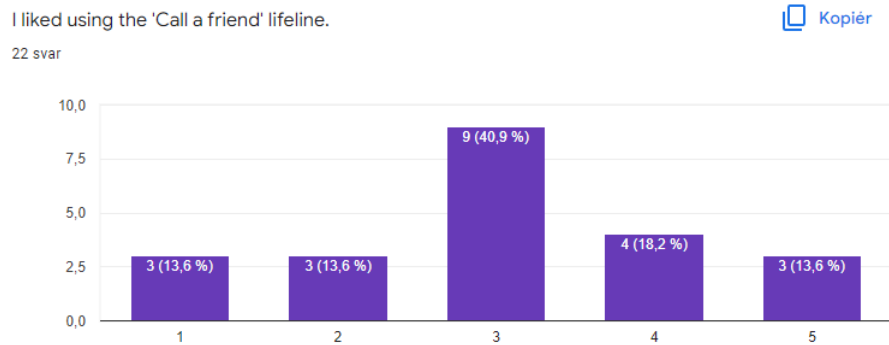


Figure 5.29: Results for 'Call a friend' lifeline

'Call a friend' received results indicating the users were indifferent about the lifeline. About 27% of the users disagreed or strongly disagreed in that they liked using the lifeline, 40,9% were indifferent, and 31,8% agreed or strongly agreed.

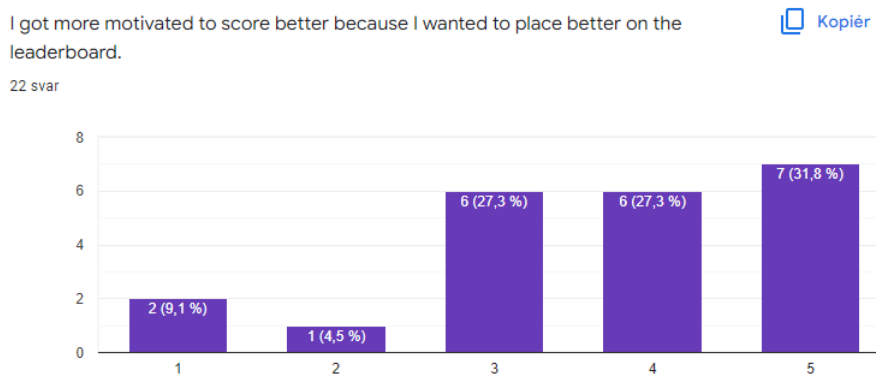



Figure 5.30: Leaderboard motivated the players to score better

Looking at figure 5.30, the results show that most of the users participating in the user survey got more motivated to score better because they wanted their name on the leaderboard. Only three participants disagreed or strongly disagreed with this statement, 59,1% agreed or strongly agreed, and 27,3% neither agreed nor disagreed.

The scorebar and levels motivated me to try again to score better than last time.  Kopier  
21 svar

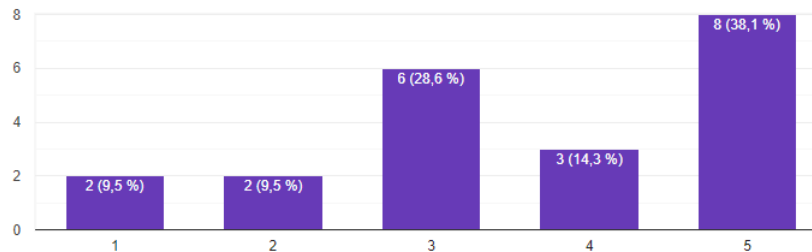


Figure 5.31: Score bar and levels motivated the users to try again

38,1% of the students strongly agreed in that the score bar and levels motivated them to try again to score better than last time, and 14,3% agreed. Four of the students (19%) did not agree or strongly disagreed, whereas 28,6% of the students were indifferent.

Answers from the optional feedback question presented in the user survey:

- "Timer on call lifeline should not start before you post the search so you have a little time to read/look for the answer."
- "It was a bit uneventful when i got answers right, i had to look at the score to find out, but thats a first world problem, nice webquiz!"
- "I must have missed explanation when answering wrong, maybe more pop-out/dominant before next round if wrong?"
- "It would be nice to be able to review all the explanations, at least after finishing the quiz."
- "It has the potential to be good, though this iteration feels very incomplete."
- "I think it's a great game!"
- "Very cool"
- "Fun, good game!"

### 5.2.4 Ask the Audience (RQ3)

Previously mentioned in 3.3.2 it was talked about how the lifeline 'Ask the audience' was evaluated. This section will display the results collected from doing these tests and show how well the lifeline did in predicting the correct answers. Table 5.1 shows every result from the testing. Green indicates the correct answer, while the percentages colored red are instances where the lifeline was wrong. The rows where it only shows one green cell and no red cell indicate that the correct answer is the one the lifeline predicted it to be.

After using the lifeline on ten questions, the recorded results were that the lifeline got six out of ten (6/10) answers correct and four out of ten (4/10) wrong. When looking at the instances where the lifeline was incorrect, three out of four (3/4) times the percentage score differed by only two (2%), three (3%) and six (6%) percent to the correct answer. In contrast, in one case, it was off by thirty-three percent (33%) and also the lowest scored answer, with zero percent (0%) of the votes. Out of the four incorrect guesses made by the audience, the correct answer had the lowest percent score in two of the cases.

Table 5.1: Table with results from 'Ask the audience' lifeline tests.

Question n	A	B	C	D
1	30%	20%	32%	18%
2	30%	23%	20%	27%
3	0%	0%	100%	0%
4	0%	57%	0%	43%
5	23%	27%	22%	28%
6	0%	100%	0%	0%
7	33.33%	0%	33.33%	33.33%
8	27%	22%	25%	26%
9	22%	28%	25%	25%
10	23%	23%	31%	23%

## 5.3 Discussion

This section will discuss the results of the research questions.

**RQ1:** *Does the use of a quiz-based data-driven game motivate users to learn software security?*

The results indicate that the game was easy to play, and the students participating in the user survey understood the rules. The in-game help button did not seem to affect this. It is hard to interpret the results as if they understood the rules from prior knowledge of the game type. Whether it is intuitive and they just understood how to play, or that the in-game help button was not informative enough to make them understand the rules and how to play the game. With a few exceptions, most of the players found it easy to use the lifelines, except for the 'call a friend' lifeline. This lifeline needs further work to improve usability to be a successful feature in the game. Except for four out of twenty-two respondents, most users strongly agreed that they enjoyed playing the quiz. However, when asked if they would like to try it again several times, the results were not as positive, but still somewhat positive as about 40% agreed or strongly agreed that they would try it again several times. Most users found that the leaderboard and score bar motivated them to score better. All in all, the results indicate that the use of the game can motivate users to learn software security. By polishing the game, adding new fun features such as achievements, and improving on already implemented elements, the game can further motivate users to learn software security.

**RQ2:** *Does the use of a quiz-based data-driven game improve learning software security?*

When looking at the results from the user survey on learnability, most are optimistic that the game is a good tool for learning software security, and most users agree or strongly agree that the quiz game made them learn more about software security. However, the results also show that seven out of twenty-two of the users strongly disagreed or disagreed in that their knowledge of the topics presented in the quiz improved by playing, whereas ten out of twenty-two of the users agreed or strongly agreed. This indicates that the answer to RQ2 is that the game can potentially improve learning software

security. Still, more testing is needed to conclude whether or not the game can improve learning software security.

**RQ3:** *How to integrate information security sources to automatically answer questions in the quiz game?*

When evaluating the 'Ask the audience' lifeline, the results showed that the lifeline predicted the correct answer six out of ten times. The lifeline automatically answers questions with the help of information security sources, but not as accurate as was hoped. The lifeline does answer RQ3 by presenting a way of integrating websites such as CAPEC and CWE into the game. A stronger metric should be used to improve the accuracy of the predictions the lifeline makes.

The optional feedback shows there is still room for improvement to make the quiz more user-friendly and eventful and make learning more attractive. Overall, the results are positive and show that a quiz-based game can be an attractive way of learning software security, but more testing must be done before concluding this.

# Chapter 6

## Literature Review

This literature review investigates gamification and serious games used to teach developers about software security. Previous research has shown that gamification effectively increases user engagement and motivation to learn through serious games [17]. The aim is to find why previous games have not succeeded in being used for learning software security. What gamification elements are used in successful serious games, and what is lacking/missing to increase the use of teaching software security through a serious game. Serious games are games with a goal to educate, and not only for entertainment. There have been earlier studies on the topic of gamification in software security. Some examples of these types of games are STIX and Stones [9] and Protection Poker [8].

### 6.1 Serious Games

#### 6.1.1 STIX and Stones

STIX and Stones is a single-player, data-driven TD educational game to teach software engineering students about information security knowledge. The game's goal is to pick the most suitable mitigation strategy to defend against various attack patterns. Having a data-driven game eliminates the obstacles

of manually adding and editing rules, threats, and mitigations to stay up to date with the rapid evolution of software security. The game automatically fetches security information from CAPEC, which stay updated on common attack patterns. The game does seem like it could offer a lot of fun, especially with the exciting pick of tower defense as the game type. However, it has not been successful in being used for teaching software security or being played.



Figure 6.1: STIX and Stones gameplay  
Source: Copied from [9]

When looking at the results section from the thesis written on STIX and Stones [9], some reasons why the game has not been successful in being used for teaching information security can be:

- Too much initial information in the game for the user to easily "plug and play." The users are missing a gradual introduction (tutorial) to how to play the game.
- The game is too slow-paced, with too much waiting in between rounds.
- Too much randomness, little learning value.

STIX and Stones is similar to this project's quiz-based game in that they



both integrate public information security repositories with the intention of learning software security. They both use gamification elements such as leaderboards and score. However, the game types differ a lot where STIX and Stones is a TD game, whereas this project's game is quiz-based.

### 6.1.2 OWASP Cornucopia

OWASP Cornucopia is a card game whose idea is to assist, mainly Agile, development teams in identifying application security requirements and developing user stories that focus on security [38]. OWASP recommends playing it with a mix of roles but suggests including someone with knowledge of information security. The game recommends 1.5 to 2.0 hours play time for 4-6 people.



Figure 6.2: OWASP Cornucopia card game

Thompson and Takabi investigated OWASP Cornucopia's effect on teaching threat modeling for secure web application developments [39]. They performed an experiment where students undergoing a software security course

were split into two groups, where one played the Cornucopia card game, whereas the other group did not. There were run quizzes before and after the activity to evaluate the effectiveness. They also issued a survey to measure the students' experiment viewpoint. The results showed that the card game was somewhat effective; however, it was not easy to understand, and they needed more time to know how to play the game.

OWASP Cornucopia is different from this project's quiz-based software security game in that it is a card game, which requires more effort to keep up-to-date. Cornucopia is more cumbersome to start playing, as the users have to download and print out the playing cards. It is recommended to play with a mix of roles, and more than one player, which requires people to be available simultaneously, whereas the quiz-based game only requires one player and can be played anywhere. Cornucopia generally requires more effort than the quiz-based software security game. They are similar in that they both aim to teach information security.

### **6.1.3 Elevation of Privilege**

Elevation of Privilege (EoP) [40] is another card game focused on teaching developers about threat modeling. Threat modeling is the process of identifying threats and mitigations to protect an application or software system. EoP is designed for 3-6 players. Before starting the game, a diagram of the system to be threat modeled should be available for all players. A card deck is shuffled and dealt to the players. A round of EoP is played by having each player read their card, announce the threat and write it down. The highest card played in the suit that was led wins, unless it is trumped by an EoP card. Then it is the highest value EoP card played that wins. A case study done by Tøndel et al. [6] investigated the challenges with adoption of EoP. Some viewpoints from this study are that the game seems to be too advanced for beginners to play. It requires previous knowledge of the topic or someone who's an expert on the matter. The game lacks the 'fun' part. Some students feel like they are "playing a random card game while discussing security issues," [6] and others dislike the gamification aspect and

suggest dropping it altogether. EoP is probably best suited for experienced software developers and not students, as too much preparation is needed to play and understand the game entirely. The game does have some positive effects related to learning about software security. However, it seems to be too much work to play. This projects quiz-based game is different in that it doesn't require more than one player. It is more straightforward in that it doesn't require much preparation other than having a computer to open the website. The rules are simple and intuitive. Both EoP and the quiz-based game provide learning on information security.

#### **6.1.4 Kahoot!**

Kahoot! [25] is a gamified learning platform that engages the students to actively learn by introducing a scoring system where you compete against other co-students in a fun and educational way. Kahoot! is commonly used in classroom sessions to measure the students' knowledge of a particular subject. This is done by creating quizzes where each answer has four answer options. The players are scored by how fast they submit the correct answer before the time runs out. In between questions, a leaderboard is shown to the players where it is possible to see who is leading. This motivates players to score better to have their names displayed on the leaderboard [22]. Competing against classmates engages the players to focus on scoring better. When each question has been answered by the end of the quiz, a top-3 leaderboard is shown. Research shows that Kahoot! mostly has positive effects on learning in various fields/topics, including technical and engineering fields [41].

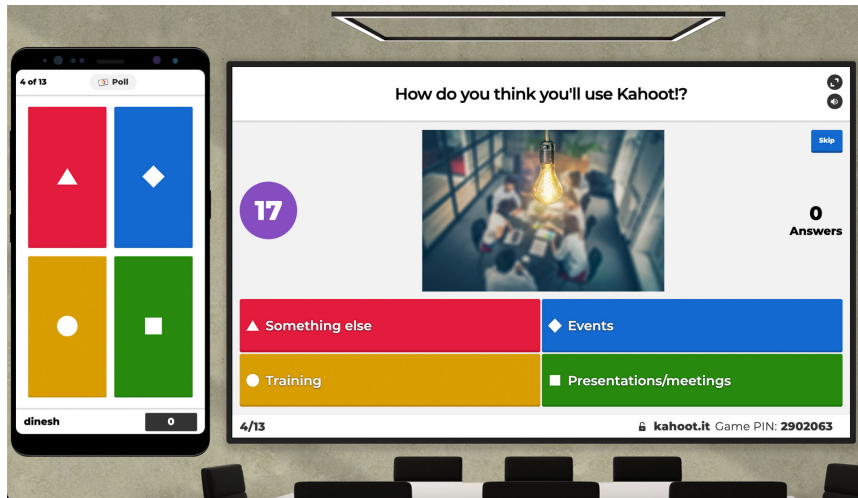


Figure 6.3: Kahoot!

Source: <https://uk.pcmag.com/education/132563/kahoot>

Kahoot! and this projects quiz-based game is similar in that they both are quiz games with four answer options and incorporate gamification elements such as leaderboards and score. They are different in that the quiz-based game is focused on teaching software security. It doesn't require players to play simultaneously and has implemented other gamification elements to engage further and motivate the players.

## Chapter 7

# Conclusion and Further Work

The main goal of this thesis has been to create a quiz-based game for teaching information security in a fun and motivating way. Gamification has been studied to find elements that increase motivation and engagement. Previous successful games have been looked into to discover why they are successful and find inspiration on features to add to the quiz-based software security game. Existing security games have attempted to teach software security but have struggled with being adopted to teach the subject. Information security is a topic with rapid changes as new threats and attacks against software appear regularly. Therefore, it is essential to incorporate up-to-date public repositories on the latest weaknesses and attack patterns that threaten software systems and applications. This is done by introducing them through in-game features such as the lifelines 'call a friend' and 'ask the audience.' RQ3 is answered with the lifeline 'ask the audience,' which shows one way of integrating information security sources to answer questions inside the quiz game automatically. A group of students studying to become software developers tested the quiz-based game and answered a user survey with its primary goal of capturing three goals; learnability, usability, and fun, to answer RQ1 and RQ2. The response from the user survey showed that the participants gained some knowledge on the topic of software security, and they believe it can be a good tool for learning this topic. The game showed

some weak points when it came to usability, as some users overlooked features that are meant to increase the learning outcome or didn't understand how to use them. Results capturing fun showed that the users enjoyed playing the quiz game, but not as many were optimistic about playing it again several times. More elements will need to be implemented to encourage users to come back to play the game more than a few times. Therefore, the answer to RQ1 is: Yes, a quiz-based data-driven game can motivate users to learn software security. However, some work needs to be done for this quiz-based game to achieve that more successfully. The answer to RQ2 is that more testing needs to be done. The results are not enough to conclude whether or not the quiz-based game can be used to improve knowledge on the topic of information security. However, it has shown the potential that it can become an attractive tool for learning this subject with a bit of polishing and further work.

## 7.1 Suggestions to Further Work

This section will suggest different ideas for further development of the quiz game that improves the usability of lifelines, usability in general, stronger functions to enhance existing lifelines, and improvements to increase motivation for users to play the game more.

First of all, general usability improvements are suggested. This includes highlighting essential features for better learning outcomes, such as the 'explanation' feature. Feedback shows that some of the users did not even see this feature and that it should be more visible. An idea could be to highlight the explanation when a user submits the wrong answer to a question. Another concept that was given through optional feedback was to provide the user with an option to review all explanations after finishing the quiz. All these suggestions are doable with minimal effort and can increase the learning outcome of the game.

The general design of the quiz could be worked on, as it only works for desktop computers at the moment. By making the game mobile-friendly,

more users can play it whenever, whether they are on the bus or somewhere they do not have access to a computer.

Gamification elements that can motivate users to continue playing, such as future rewards in the form of achievements, will help boost motivation for the user to keep coming back to play more quizzes [4]. Achievements/badges are a way to reward the users when they have achieved an in-game goal, for example, when completing five quizzes with a score higher than 60%. The achievements can motivate learning in a fun way by rewarding good scores. Achievements/badges work as positive feedback that keeps users continuing to play to work towards future goals.

The lifeline 'call a friend' can be worked on to improve usability as feedback from the user survey implied that the lifeline had challenges in this area. One idea could be to automatically fetch the quiz question and insert it into the search field, making it easier for the user. Some users also struggled to understand that they had to insert a search before selecting their 'friend' for the lifeline. This could be improved upon to increase ease of use and remove any doubts about how the lifeline work. Feedback from users also suggested making the timer start counting after the user has selected their friend. This is a good idea and should be an easy fix to improve the lifeline.

Ask the audience is an important feature that incorporates websites revolving around software security to answer questions given in the quiz automatically. Therefore it should be worked on to improve the accuracy of how well it predicts the correct answers. This can be done using a stronger metric to measure how relevant a word is to a document. Term Frequency-Inverse Document Frequency [42] (TF-IDF) is an example of such a statistical measure. TF-IDF is a combination of TF and IDF, where they are multiplied. TF is used to find how many times a word appears in a document, whereas IDF measures how common a word is in the document set. By multiplying these, it is possible to find the relevance of a word in a document.

# Appendix A

## Source code and Link to Quiz-based Game

The source code for the Quiz-based game is available at: <https://github.com/crissb3/soft-sec-prototype>.

The quiz-based game can be played using this link: <https://soft-sec-prototype.herokuapp.com/>



# Literature and References

- [1] Tosin Daniel Oyetoyan, Daniela Soares Cruzes, and Martin Gilje Gilje Jaatun. “An Empirical Study on the Relationship between Software Security Skills, Usage and Training Needs in Agile Settings.” In: *2016 11th International Conference on Availability, Reliability and Security (ARES)* (2016), pp. 548–555.
- [2] Open Web Application Security Project. *OWASP*. URL: [https://owasp.org/..](https://owasp.org/)
- [3] Secure Code Warrior. *Where does secure code sit on the list of development team priorities?* URL: <https://www.securecodewarrior.com/blog/where-is-secure-code-in-development-team-priorities>.
- [4] Fiona Fui-Hoon Nah et al. “Gamification of Education: A Review of Literature.” In: *HCI in Business - First International Conference, HCIB 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014. Proceedings*. Ed. by Fiona Fui-Hoon Nah. Vol. 8527. Lecture Notes in Computer Science. Springer, 2014, pp. 401–409. DOI: 10.1007/978-3-319-07293-7\\_39. URL: [https://doi.org/10.1007/978-3-319-07293-7%5C\\_39](https://doi.org/10.1007/978-3-319-07293-7%5C_39).
- [5] Fiona Fui-Hoon Nah et al. “Flow in gaming: literature synthesis and framework development.” In: *Int. J. Information Systems and Management*. Vol. Vol. 1. 2014, pp. 83–124.
- [6] Inger Anne Tøndel et al. “Understanding Challenges to Adoption of the Microsoft Elevation of Privilege Game.” In: *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security*. HoTSoS '18. Raleigh, North Carolina: Association for Comput-

- ing Machinery, 2018. ISBN: 9781450364553. DOI: 10.1145/3190619.3190633. URL: <https://doi.org/10.1145/3190619.3190633>.
- [7] Inger Anne Tøndel et al. “Understanding Challenges to Adoption of the Protection Poker Software Security Game.” In: *Computer Security*. Ed. by Sokratis K. Katsikas et al. Cham: Springer International Publishing, 2019, pp. 153–172. ISBN: 978-3-030-12786-2.
- [8] Laurie Williams, Andrew Meneely, and Grant Shipley. “Protection Poker: The New Software Security ”Game”,” in: *IEEE Security Privacy* 8.3 (2010), pp. 14–20. DOI: 10.1109/MSP.2010.58.
- [9] Dag Erik Homdrum Løvgren. “Data-driven Security Game-STIX and Stones.” MA thesis. NTNU, 2018.
- [10] Common Attack Pattern Enumeration and Classification. *CAPEC*. URL: <https://capec.mitre.org>.
- [11] Kristie Cameron and Lewis A. Bizo. “Use of the game-based learning platform KAHOOT! to facilitate learner engagement in Animal Science students.” In: (2019).
- [12] Common Weakness Enumeration. *CWE*. URL: <https://cwe.mitre.org/>.
- [13] Common Vulnerabilities and Exposures. *CVE*. URL: <https://cve.mitre.org/>.
- [14] Software Engineering Institute Carnegie Mellon University. *SEI CERT Oracle Coding Standard for Java*. URL: <https://wiki.sei.cmu.edu/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java>.
- [15] *StackOverflow*. URL: <https://stackoverflow.com/>.
- [16] *StackExchange*. URL: <https://stackexchange.com/>.
- [17] Juho Hamari, Jonna Koivisto, and Harri Sarsa. “Does Gamification Work? – A Literature Review of Empirical Studies on Gamification.” In: *2014 47th Hawaii International Conference on System Sciences*. 2014, pp. 3025–3034. DOI: 10.1109/HICSS.2014.377.
- [18] Gabriel Barata et al. “Engaging Engineering Students with Gamification.” In: *2013 5th International Conference on Games and Virtual*

- Worlds for Serious Applications (VS-GAMES)*. 2013, pp. 1–8. DOI: 10.1109/VS-GAMES.2013.6624228.
- [19] Gabriel Barata et al. “Improving Participation and Learning with Gamification.” In: *Proceedings of the First International Conference on Gameful Design, Research, and Applications*. Gamification ’13. Toronto, Ontario, Canada: Association for Computing Machinery, 2013, pp. 10–17. ISBN: 9781450328159. DOI: 10.1145/2583008.2583010. URL: <https://doi.org/10.1145/2583008.2583010>.
- [20] Víctor Arufe Giráldez et al. “Can Gamification Influence the Academic Performance of Students?” In: *Sustainability* 14.9 (2022). ISSN: 2071-1050. DOI: 10.3390/su14095115. URL: <https://www.mdpi.com/2071-1050/14/9/5115>.
- [21] G. Ivanova, V. Kozov, and P. Zlatarov. “Gamification in Software Engineering Education.” In: *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2019, pp. 1445–1450. DOI: 10.23919/MIPRO.2019.8757200.
- [22] S.A. Licorish, H.E. Owen, and Daniel B. et al. “Students’ perception of Kahoot!’s influence on teaching and learning.” In: (2018). URL: <https://doi.org/10.1186/s41039-018-0078-8>.
- [23] Siobhan O’Donovan, James Gain, and Patrick Marais. “A Case Study in the Gamification of a University-Level Games Development Course.” In: *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*. SAICSIT ’13. East London, South Africa: Association for Computing Machinery, 2013, pp. 242–251. ISBN: 9781450321129. DOI: 10.1145/2513456.2513469. URL: <https://doi.org/10.1145/2513456.2513469>.
- [24] Carlos Santos et al. “Students’ Perspectives on Badges in Educational Social Media Platforms: The Case of SAPO Campus Tutorial Badges.” In: *2013 IEEE 13th International Conference on Advanced Learning Technologies*. 2013, pp. 351–353. DOI: 10.1109/ICALT.2013.108.
- [25] J. Brand A. I. Wang M. Versvik and J. Brooker. *Kahoot!* URL: <https://kahoot.it/>.

- [26] Henrik Kniberg. *Scrum and XP from the Trenches: Enterprise Software Development*. Lulu.com, 2007. ISBN: 1430322640.
- [27] chercher tech. *agile methodology*. URL: <https://chercher.tech/jira/agile-methodology>.
- [28] Rini Solingen et al. “Goal Question Metric (GQM) Approach.” In: Jan. 2002. ISBN: 9780471028956. DOI: 10.1002/0471028959.sof142.
- [29] Nooralisa Mohd Tuah and Gary B. Wills. “Measuring the Application of Anthropomorphic Gamification for Transitional Care; A Goal-Question-Metric Approach.” In: *Computational Science and Technology*. Ed. by Rayner Alfred et al. Singapore: Springer Singapore, 2020, pp. 553–564.
- [30] *Jotform*. URL: <https://www.jotform.com/blog/quiz-vs-test/>.
- [31] *Heroku*. URL: <https://www.heroku.com>.
- [32] *Spring Boot*. URL: <https://spring.io/projects/spring-boot#overview>.
- [33] *Spring Data JPA*. URL: <https://spring.io/projects/spring-data-jpa>.
- [34] Severin Hacker Luis von Ahn. *Duolingo*.
- [35] Sony Pictures Television. *Who Wants to Be a Millionaire?*
- [36] Adam Atkins, Vanissa Wanick, and Gary Wills. “Metrics Feedback Cycle: measuring and improving user engagement in gamified eLearning systems.” In: *International Journal of Serious Games* 4.4 (Dec. 2017). DOI: 10.17083/ijsg.v4i4.192. URL: <https://journal.seriousgamesociety.org/index.php/IJSG/article/view/192>.
- [37] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. “Introduction to Informational Retrieval.” In: (2008), pp. 100–123.
- [38] OWASP. *OWASP Cornucopia*. URL: <https://owasp.org/www-project-cornucopia/>.
- [39] Mark Thompson and Hassan Takabi. “EFFECTIVENESS OF USING CARD GAMES TO TEACH THREAT MODELING FOR SECURE WEB APPLICATION DEVELOPMENTS.” In: *Issues in Information Systems* 17.3 (2016).

- [40] Adam Shostack. “Elevation of privilege: Drawing developers into threat modeling.” In: *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*. 2014.
- [41] Alf Inge Wang and Rabail Tahir. “The effect of using Kahoot! for learning – A literature review.” In: *Computers and Education* 149 (2020), p. 103818. ISSN: 0360-1315. DOI: <https://doi.org/10.1016/j.compedu.2020.103818>. URL: <https://www.sciencedirect.com/science/article/pii/S0360131520300208>.
- [42] Fayola Peters et al. “Text Filtering and Ranking for Security Bug Report Prediction.” In: *IEEE Transactions on Software Engineering* 45.6 (2019), pp. 615–631. DOI: 10.1109/TSE.2017.2787653.
- [43] Common Configuration Enumeration. *CCE*. URL: <https://cce.mitre.org/about/index.html>.
- [44] Cyber Observable eXpression Archive Website. *CybOX<sup>TM</sup>*. URL: <https://cyboxproject.github.io/>.
- [45] Malware Attribute Enumeration and Characterization. *MAEC<sup>TM</sup>*. URL: <https://maecproject.github.io/about-maec/>.
- [46] Open Source Vulnerability Database. *OSVDB*. URL: <https://cve.mitre.org/data/refs/refmap/source-OSVDB.html>.