SOCIAL ENGINEERING SIMULATOR: AN EDUCATIONAL GUIDE FOR SOCIAL ENGINEERING DEFENCE

BY

OKAFOR DUMEBI YOUNG

PSC1908911

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCES,

UNIVERSITY OF BENIN,

BENIN CITY,

EDO STATE, NIGERIA.

JUNE 2024

SOCIAL ENGINEERING SIMULATOR: AN EDUCATIONAL GUIDE FOR SOCIAL ENGINEERING DEFENCE

BY

OKAFOR DUMEBI YOUNG

PSC1908911

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF PHYSICAL SCIENCES, UNIVERSITY OF BENIN, BENIN CITY, EDO STATE, NIGERIA

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF A BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE

JUNE 2024

T

CERTIFICATION

his is to certify that this project was carried out by **OKAFOR DUMEBI YOUNG** with Matriculation Number **PSCI9089115** under my supervision. It is adequate and satisfactory, both in scope and content, for the award of Bachelor of Science (B.Sc.) Degree in Computer Science of the University of Benin

DR. E. NWELIH. **DATE**

Project Supervisor

T

APPROVAL

his project work is hereby approved in partial fulfillment of the requirements for the award of Bachelor of Science (B.Sc.) Degree in Computer Science from the University of Benin.

PROF. GODSPOWER O. EKUOBASE, PHD. **DATE**

Head of Department

T

DEDICATION

his project is dedicated to God Almighty, my source and strength who enabled me through this programme and to my wonderful family whose love has been a source of strength.

ACKNOWLEDGEMENT

My utmost acknowledgement and undaunted gratitude go to the almighty God for his unending love and has kept me throughout my programme and for the success of his work.

I would like to voice my unreserved gratitude to my project supervisor DR. E. NWELIH for his transparency and commitment towards ensuring that this project work is completed successfully and measures up to the generally accepted standard.

I want to use this opportunity to express my appreciation to the Head of the Department of Computer Science, Prof. Godspower O. Ekuobase, PhD. for his relentless efforts and selfless dedication towards upholding excellence and integrity, and also for his tireless commitment in fighting for the interest of the student.

I would also like to specially thank my project coordinator Prof. A.A. Imianvan, and other lecturers in the Department of Computer Science who I have been opportune to cross paths with, and have impacted me immensely these past few years: Prof. (Mrs.) V.V.N. Akwukwuma, Prof. F.I.
Amadin, Dr. F.O. Chete, Dr. (Mrs.) R.O. Osaseri, Mr. K.O. Otokiti, Mr. E.C. Igodan , Prof. F.A.U. Imouokhome, and Mr. D.N. Idehen.

My Undiluted thanks goes to my lovely parents and siblings, Mr & Mrs OKAFOR, for their financial and moral support, continuous prayers throughout my programme. I will also like to acknowledge my friends, Salome Lenyie, Ezekiel, Eghosa, Wisdom, Elvis, and the entire CSC Tigers for their encouragement and guidance throughout my stay in the University of Benin.

Finally, a general thanks to all who may not be mentioned here, but has sincerely contributed to

T

the success of this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In this project, I developed a web application to Educate users on social engineering by testing them with real life scenarios to familiarize them with social engineering tricks. The application is designed using mainly the Java technology, it is designed as a web application so it can run on browsers, as a result, the user interfaces are developed using web technologies such as HTML5, CSS3 and JavaScript. The application logic is developed using Java beans and Java Servlet. The application is designed using the Model-View-Controller approach (MVC). The data persistent layer is maintained using the MySQL relational database. The simulator developed will help the users easily identify social engineering attacks, this will help in the long-lasting battle against social engineering.

CHAPTER ONE

INTRODUCTION

This chapter introduces the topic of the project work social engineering simulator: an educational guide to social engineering defense. In this chapter, we will consider the background of the study, statement of the problem, aims and objectives, methodology used to design the system, scope of the study, limitations its significance, definition of terms, and we conclude with the project layout or organization of the project work.

1.1 Background of Study

Social engineering, often hailed as the Achilles' heel of cybersecurity, represents a formidable challenge in today's digital age. While traditional cybersecurity measures focus on fortifying technical defenses against external threats, social engineering exploits the inherent vulnerabilities of human psychology and behavior, making it a uniquely insidious and difficult-to-detect threat. At its core, social engineering is a testament to the power of persuasion and manipulation. Whether through phishing emails, pretexting phone calls, or physical impersonation, social engineers adeptly exploit trust, authority, and fear to deceive unsuspecting individuals into divulging sensitive information or performing actions contrary to their best interests. From individuals to large corporations, no entity is immune to the potential consequences of falling victim to a wellexecuted social engineering attack. In this landscape fraught with deception and manipulation, the fight against social engineering demands a multifaceted approach that goes beyond technical solutions.

Phishing, pretexting, baiting, tailgating, and other techniques are among the many strategies that fall under the umbrella of social engineering. These techniques take advantage of basic human emotions including trust, fear, greed, and altruism. A skillfully written phishing email, for

example, could plausibly mimic a reliable source in order to fool the receiver into disclosing passwords or clicking on harmful links. Similar to this, an employee can be duped into divulging private information via a phone call from a fraudster pretending to be a technical support agent. These attacks are hard to protect against with traditional security measures alone since their effectiveness depends on their capacity to take advantage of human nature.

Despite the general public's understanding of social engineering, a lack of hands-on training and real-world experience in identifying and mitigating these dangers leaves many people and organizations exposed. This disparity in cybersecurity education underscores the need for creative, non-theoretical training approaches. By offering a realistic, immersive training environment where users may experience and learn to combat social engineering attacks in a safe, controlled setting, the "Social Engineering Simulator: An Educational Guide for Social Engineering Defense" seeks to fulfill this requirement.

The idea of a "Social Engineering Simulator: An Educational Guide for Social Engineering Defense" presents itself as a viable remedy for a complicated and multidimensional issue in a time when cyber threats are always changing. It is vital to critically assess the simulator's effectiveness and potential limits, even while the goal of providing people with the knowledge and abilities needed to protect themselves against social engineering attempts is unquestionably admirable. By its very nature, social engineering relies more on taking advantage of behavioral and psychological traits in people than it does on technological flaws. Therefore, it is very difficult to simulate these complex maneuvers in a controlled setting.

Additionally, the design, content, and delivery of any educational tool all play a major role in how effective it is. Two decades ago, social engineering attacks were not that serious because cybercriminals and tools had not advanced; however, as things stand, social engineering has evolved and continues to do so, forcing organizations to heavily invest in it. This project delves into the motivation of social engineering attacks, the tools and strategies used, and the approaches that can be adopted to mitigate these threats.

1.2 Problem Statement

In today's world, social engineering has become a serious issue with an increasing number of victims. The majority of internet users and those with gadgets that can connect to the internet are aware of the issue, but most are still unaware of the severity of social engineering attacks. Carpenter (2022) stated that it was anticipated that social engineering tactics will lead to a 270% increase in cyberattacks Despite advancements in cybersecurity technologies and awareness, social engineering remains a persistent and evolving threat to individuals and organizations worldwide. Traditional cybersecurity defenses often focus primarily on technical measures, overlooking the critical human element exploited by social engineering tactics. As a result, individuals are frequently targeted and manipulated into divulging sensitive information or performing actions detrimental to their security. The problem lies in the lack of effective and accessible educational resources specifically tailored to combatting social engineering attacks. While theoretical knowledge about social engineering techniques exists, the gap between understanding these concepts and applying them in real-world scenarios remains substantial. Current training methods often lack practicality and fail to adequately prepare individuals to recognize and respond to social engineering attempts effectively.

1.3 Aim and Objectives

The aim of this research project is to educate people in general on the identification of social engineering attacks.

The objectives are to:

1. Identify the framework for social engineering and its application
2. Design and implement a social engineering simulator

1.4 Research Methodology

This project made use of Rapid application development (RAD), which favours iterative development and the rapid construction of prototypes instead of large amounts of up-front planning. it also called for the use of UML diagrams.

1.5 Research Scope

The project focuses of the implementation of a social engineering simulator that helps users identify social engineering attacks using Higher institutions around Nigerian as a case study.

1.6 Significance of Study

In contrast to conventional hacking techniques that aim to exploit technical weaknesses, social engineering takes advantage of human psychology. By comprehending the workings of social engineering, we may create more effective defenses against this human aspect of cybersecurity. Research on social engineering is essential to bringing attention to these deceitful behavior Through education, we may enable people and institutions to recognize and thwart attempts at social engineering.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview of Social Engineering

Social engineering involves manipulating individuals to divulge confidential information that they would typically keep to themselves. It works well and can be done on big companies because it targets people who are the most vulnerable part of a network (Huber et al. , 2010) Recent studies show that more and more people are being tricked by social engineering. This is causing problems for both the individuals and the companies involved, as it can lead to their accounts and private information being accessed by unauthorized people. Due to a lack of understanding about the threat of social engineering, it is essential to improve awareness. This study aims to see if using a prototype can help people understand social engineering better, compared to using traditional paper materials.

2.2 Social Engineering Techniques and Terms

Social engineering tactics rely on exploiting cognitive biases, which are the ways in which people make decisions (Kirdemir, 2019). One form of social engineering involves an individual entering a building and posting a false notice on the company bulletin board, claiming that the help desk phone number has been updated. When workers ask for assistance, the person asks for their passwords and IDs, which lets them get into the company's private information. Another way of social engineering is when a hacker talks to the target on a social media site and starts a conversation with them. The hacker gradually gains the target's trust and then exploits it to obtain sensitive information such as passwords or banking details.

2.2.1 Pretexting

Pretexting involves concocting a falsehood to mislead someone into revealing information or taking actions they wouldn't typically do. A big lie that usually requires some planning and research. The individual fabricating the lie utilizes another person's personal details, such as their birthday or social security number, in order to convincingly impersonate them.

2.2.2 Water holing

Water holing is a trick where hackers take advantage of the trust people have in the websites they often go to. The person who was hurt feels like they can do stuff without being scared. A cautious individual may opt to avoid clicking on an unsolicited email link, yet they would click on a link from a familiar website. The attacker sets a trap for the unsuspecting prey at a popular drinking spot. This plan has worked to get into some very secure systems.

2.2.3 Baiting

Baiting involves being lured into a trap with something that appears harmless, but is ultimately used for stealing. In this attack, malicious individuals strategically place infected floppy disks, CD-ROMs, or USB flash drives in locations where they are likely to be discovered by people, such as bathrooms, elevators, sidewalks, parking lots, and more.) They put labels on them that make people curious and trick them into using the infected items. Then they wait for people to fall for the trick. If the computer doesn't stop viruses, putting in a USB or CD can harm the computer. Unfriendly tools can also be used. For example, a "lucky winner" gets a free digital music player that can be used with any computer. A "road apple" is a word for horse poop, which suggests that it's not something you want. It also refers to any portable storage device such as a USB drive that is deliberately left in convenient locations and contains malicious software. It could be a CD, DVD, or a small USB drive, or something else like that. Curious people put it in

a computer and it spreads to other devices connected to the computer. Hackers might use tempting names like "Employee Salaries" or "Confidential" to trick people.

In 2016, a study was done at the University of Illinois where researchers left 297 USB drives around the campus. The hard drives had files that connected to webpages owned by the researchers. The researchers saw how many drives had files opened, but they didn't see how many were put into a computer without opening a file. Out of the 297 dropped drives, 290 (98%) were picked up and 135 (45%) were "called home".

2.3 Use of Games in Cyber Security Awareness

Currently, games are becoming very popular for teaching things like programming. New research has found that people and managers often find computer security education and training to be boring and unexciting. Security games are often provided by companies to help teach and train people who work in network security. They can also be used to test the skills of people who are already working in security or who want to be hired for a security job. (Ariyapperuma and Minhas, 2015)

Using games to teach is very common, especially in network security and computer programming (Ryan et al. , 2023) One example is the control-alt-hack game that uses cards and is played on a network. The game is a card game that you play with other people. It's meant to teach about network security and hacking. This was done to help people understand the importance of computer security. It teaches them about different ways hackers can attack and which technologies are at risk of being hacked.

Just like control-alt-hack is for everyone, Microsoft Elevation of Privilege (EOP) card game is designed specifically for network administrators and has more features. This game is meant to

help people learn more about threat modeling in computer networking. The threats of attacks include things like denying access, overloading servers, and changing data. In 2014, Microsoft released a new product.

"Network nightmares" is a game that teaches people about how hackers can threaten computer security. It shows how hackers attack computer nodes. They used the angry birds game design for their research. The players are the hackers and their job is to find weak spots in the network. This could mean that many parts of the network might be at risk at the same time.

"Anti-Phishing Phil" is a game online that helps people learn about phishing so they can be more careful when they are using the internet. The game teaches people how to recognize and avoid phishing attacks by breaking it down into smaller parts. The researchers at Carnegie Mellon University made it. The online game was played by a lot of people and was popular. (CMU Laboratory, 2008).

2.4 Humans as the Weakest Link in System Security

Real-life systems can still be attacked, even with security measures in place. Better security can only happen if we understand how things can go wrong (Ross, 2024). Many times, the weakest part of the system's protections is the human element. The attack happens because the security engineers only thought about how to protect the system, not about how real users would respond to a trick from a bad person. To ensure the security of our systems, it is essential for us to understand user behavior and identify their vulnerabilities. In what manner can we acquire this knowledge? This paper suggests that scammers possess a deeper insight into human behavior compared to security experts. So, the experts can learn from the scammers..

2.5 How Companies Address Social Engineering

Right now, companies are looking at two ways to deal with the social engineering problem. First, companies can teach employees about the dangers of social engineering through security trainings. These trainings are usually required for employees and don't have a long-term impact. Security awareness campaigns are a more affordable option that tries to reach the same goal as trainings, but they also encounter the same issues. Usually, they don't handle the employees' weaknesses well. Secondly, businesses hire testing companies that attack them to find weaknesses. These types of penetration tests are not often done because they have various problems. For example, Before dealing with legal problems, a great deal of work needs to be put in (Watson et al, 2021). When penetration tests are done, the tester usually finds mistakes and companies can teach their employees how to do better. However, studies have found that these methods are hard because people lose motivation when they see the outcomes.

2.6 Computer Based Testing

Computer based testing (CBT) or Computer based Examination System (CES) is very important because it saves time, scores tests right away, is efficient, and gives quick feedback for multiple-choice question exams. Utilizing the internet is believed to be an effective method for evaluating numerous students. This method helps students improve how they work and manage their time when they answer questions. CES is a helpful tool that makes sure all students take their tests under the same conditions. CES includes providing different types of exams, storing student exam answers in databases, and the ability to make random exam questions using question banks. This method can make schools and colleges work better. It saves money on supplies and workers, and makes the test questions more varied. (Abdulkareem S) and Nathan, 2018). and Nathan, 2018). Computer based exams use different automatic systems for testing. They have a few

problems like not being able to adjust timing, not being able to handle a lot of work, not being very strong, and not being reliable.

2.7 Advantages of CBT

According to Jacob and Chase, computers can present test content in ways that paper tests are unable to, including 3-D imagery, dynamic visuals, rotating shapes, and diverse plant perspectives. More ideas recommend using computer simulations to test how well someone does something. In the project called "Electric Mysteries", students had to copy circuits by using image icons of batteries, light bulbs, and wires on a Macintosh computer.

When giving tests, computers can be used to create personalized testing conditions, so students can take tests whenever they are ready. Also, teachers can change the test questions to be easier or harder and focus on specific topics for students (Alessi, 2021). Computer tests can give people their test results right away. However, Wise and Plake discovered that getting feedback right away might make students feel more anxious about tests (Wise, 2019). In 2020, it was also found that people who are generally anxious about using computers may be affected when taking tests. They thought that even though anxiety can be different for everyone, it should be recognized and taken care of. (Bernt, 2020). Jacob and Chase also recommended stopping giving feedback on each question until more research has been done on the problem of anxiety during computer tests (Jacob, 2022).

Improvements in computer networking have given individual computers the ability to communicate well. This allows for a different way to measure how well students are learning and their attitudes. Students who are far away from each other may be able to choose where and when they want to take the test. As well as the regular multiple choice, fill in the blank, and short

essay questions. There are numerous proven benefits of using a computer for administering tests that have been well-documented. Some of these benefits are:

1. Reduced testing time.
2. Increased test security.
3. Provision of instant scoring (the test can be discussed while the whole thing is fresh in the subject's mind; in selection where the number of candidates again immediate results are valuable; where a huge number of subjects is tested this facility is not so important).
4. Better use of professional time.
5. Reduced time lag.
6. Greater availability: individuals can be tested in a computer setting individually or in groups, usually in more user-friendly environments than the large classroom auditoriums where paper-and-pencil tests have been administered traditionally. The computer format is also much more flexible than the printed page: for example, split screens could show stimuli such as a picture, as well as the possible responses. In addition, the computer format allows each examinee to work at his or her own pace, much more so than the paper-andpencil version.
7. Greater accuracy: computers can combine a variety of data according to specific rules; human are less accurate and less consistent when they attempt to do this. Computers can handle extensive amounts of normative data, but humans are limited. Computers can use very complex ways of combining and scoring data, whereas most humans are quite limited in these capabilities. Computers can be programmed so that they continuously update the norms, predictive regression equation, etc., as each new case is entered.

8. More consistency is needed with the computer. It requires following specific rules for tests and how test results are understood. It doesn't usually accept any differences from these rules.
9. More control: this is about having more power over the situation. It means that the mistakes made by the person examining something are much less likely or even completely gone.
10. Using computers to test special students and groups has big advantages. For example, it can help severely disabled students who may struggle with regular paper tests.
11. Saving money in the long run: Even though buying computer equipment and making program software can be expensive at first, after automating a test, it can be done again and again without spending a lot more money.
12. Simpler testing: using a computer to make the test shorter and save time. The test can also be customized for the person taking it.

2.8 Review of Related Research

In the article "Control-alt-hackTM: A card game for computer security outreach and education" by Denning et al. (2013), the game Control-Alt-Hack is introduced as a way to help people learn about computer security. Players can pretend to be good hackers and learn how to protect against bad ones. Control-Alt-Hack doesn't focus on studying threats or finding out security rules, instead, it emphasizes on being aware of potential security issues. So, it is in a made-up story. Also, the players use predetermined attacks on the cards and do not have to come up with their own attacks. The goal is to make people aware of hacking and the harm it can cause, so they can learn how to protect themselves.

Kristian and Sebastian (2016) suggest using a card game to figure out what security measures a company needs. All employees can play the game to learn about potential threats and write down what security measures are needed. The game looks at how each company is different and shows how people can be manipulated by social engineers. It also gives examples of specific ways that attacks are carried out. They checked their method with many different groups of researchers, IT administrators, and industry professionals.

Abdus-Samad and Nur (2015) aim to illustrate how playing a game can enhance understanding of social engineering, a method of manipulating individuals to obtain their personal information.This game was made for people who like technology. It is supposed to be both informative and enjoyable. In this paper, they talked about what they wanted to find out and how they did the research. A science test where people are put into different groups to see what happens. The game helped the players learn a lot about social engineering. On average, their knowledge and awareness went up by 71%. They found the game useful and informative. The game is not easy to use and needs to be more focused on the player. They finish by saying that more research is needed on using games for teaching, especially in the field of network security, which is facing more and more threats.

In 2005, Irvine and others created a video game called CyberCIEGE, where players make decisions about information security in a company. Players have to make sure the company stays safe while still letting users get things done. Just like Persuaded, the game shows how important it is to balance productivity and security in a realistic way. As game makers, players get to decide things that affect users. How thorough will the background checks be. Also, will they include checking computer records. How will computers be connected to each other and kept safe. Who can go into a zone and keep an eye on what happens when they make decisions. We found that

CyberCIEGE had a lot of benefits similar to Persuaded. For example, when players are trying to protect something and have to choose what to do, they see what happens because of their choices. CyberCIEGE includes things in the game that Persuaded doesn't have, like assets and resources. However, the game takes a while to learn and play.

Newbould and Furnell (2009) made a game called PlayingSafe to teach people about social engineering. It has questions where you can choose from a few options, and it's presented like a regular board game. Since the questions are all about social engineering, the game is very much like ours. The biggest difference is in what we concentrate on when it comes to social engineering. PlayingSafe asks questions about Phishing, scams, and spam. It covers less common types of attacks. Our game covers a lot of different things, but doesn't go into detail on any one subject. Furthermore, our game includes strategy to make it more fun and improve the overall experience for players.

In their article "PERSUADED: Fighting Social Engineering Attacks with a Serious Game," Dina and her team created a game to help people protect themselves from social engineering scams using social psychology techniques. Our study shows that the game can help people learn about social engineering in a fun way.

Gaurav et al (2017) created a game called "Phish Phinder" to help users feel more confident in recognizing phishing scams.

"Mitigating Phishing Attacks" introduces Phish Phinder, a game that helps users become more confident in stopping phishing attacks by teaching them about the concept and how to prevent them. The user will learn about phishing in a fun way. They will complete challenges that are

like games. These challenges will be on a computer or phone screen and will help the user understand important phishing concepts.

We found important parts of the game design by studying it, so that we can make it easier for people to play the game. They used persuasive design principles when making Phish Phinder to help people avoid phishing.

Agus Tedyyana and others (2017) studied how to make tests on computers using Unified Modeling language.

Language is used to make models for Computer-based Tests. It uses Unified Modeling Language (UML) and includes Use Case diagrams, Activity diagrams, and sequence diagrams. When designing the app, it's important to focus on how to make sure the test questions are secure when being shown by using encryption and descriptions for the password. The RSA code system was used in this process. Next, the questions in the question banks were mixed up using the Fisher-Yates Shuffle method. The computer test application used a client-server network and a local area network. They had a big challenge to make new ways to mix up the answers so that new questions could lead to new answers.

In 2018, Abdulkareem and his team set up a Computer Based Testing (CBT) System for GST Exams at Adamawa State University, Mubi. This system helps to speed up the process of getting students' exam results. The CBTS was created using a fast and flexible approach to building software. We used open source technologies like XAMPP server, MySQL, PhP, JavaScript, CSS, and HTML to implement the project. The CBT tells students their test results fast, lets them take the test again if needed, and gives different kinds of questions.

In a study from 2017, Omorogiuwa and his team made a computer center using biometric fingerprint and linked all the system's MAC addresses to a program server. They also connected the computers to the server using quantum mechanics distribution for security. This was done to make sure that only authorized people could access the system and to make the system very secure. We used Visual Studio to create the front part of the Computer Based Test (CBT) authentication software and we used Visual Basic. NET to make the software. The backend was made with the MySQL server application. This means all the data for the CBT authentication software is kept in the MySQL database. In processing, data is taken from the database and shown on the front end. The test showed that Biometric verification/authentication (average 1sec) is much faster than Manual verification/authentication (average 6. 6sec) The server stops people who don't have permission from logging in and sends them a warning message. This stops people who aren't supposed to be there from getting into the exams. All systems can see the network ID, but not everyone can access the exam. The writer stated that we may need to look into ways to identify students using a mixture of facial and iris features. They found this to be a big challenge in their research.

Prince and his colleagues devised an internet-based test platform for colleges. They used Microsoft Visual Studio 2008 for the design, C# 3. 5 for coding, and Microsoft SQL Server for the database. ASPNET is the most commonly used technology for making websites. The system checks and grades multiple choice questions automatically when they are entered into the system.

CHAPTER THREE

System Analysis and Design Methodology

3.1 Introduction

This chapter provides an overview of system analysis, including all the modules and components used to design the system, and how they work together. It also shows us how the users of the system interact with the system.

3.2 Analysis of the Existing System

The computer game PERSUADED was created by (Dina et. al, 2023) to educate individuals on safeguarding themselves from social engineering deceptions. They used ideas from social psychology to design the game. The

Rephrase

findings from our study indicate that the game can be an effective tool for educating people about social engineering in an entertaining way.

In order to aid players in learning, the game incorporated attack scenarios and methods for defending against them. They picked eight ways that people might trick others to get information or access to a place. These include things like pretending to be someone else, sending fake emails, and using pop-up messages.

The attacks were caused by a game that helps find security needs. The defense cards gave us problems because it's not easy to go against our natural behavior, which is what social engineers use to control us. They found specific ways that security departments in various companies recommend for protecting against threats. At first, defenses were supposed to work for many

types of attacks. But when they looked at the different options, they saw that they all had the same basic defenses. to check where the information or person is from. So, they decided to have matches with just one person and made eight special Defense cards.

Following Don Norman's design principles, they chose a user interface that is easy to use for both new and experienced players. The new design was tested more and changed based on the feedback they got during the testing phase. They used different colors for each kind of cards. For the attack and defense situations, they made the text really short and split the information into up to three bullet points. Action cards have pictures that show what they do, while attack and defense cards have names that tell you what they are about. However, the titles of matching pairs are different. We purposely made this design choice to make sure players have to read the cards. The Game Setting was made to be easy to understand and give a lot of information. As shown in figure 3.0

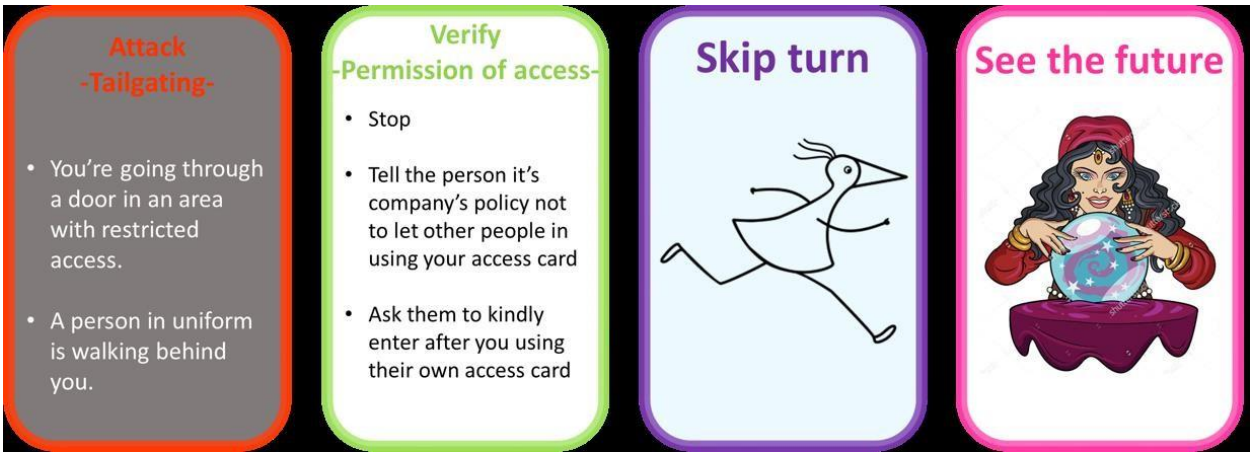


Figure 3.0: User Interface Design of Existing System

3.1.2 Problems of the Existing System

1. **Accessibility:** the existing system is a table game which require physical presence in a specific location. This can be inconvenient for those who don't have easy access to such places or who prefer the convenience of playing from home.

2. **Time and Travel:** Playing table games often involves travel time and expenses to reach the venue. Additionally, arranging schedules with friends can be time-consuming.
3. **Social Distractions:** Playing at a physical table may expose players to various social distractions, such as noise from other players, or unwanted interactions with strangers. These distractions can sometimes detract from the gaming experience.

4. **Higher Costs**: Table games typically come with higher costs associated with travel, accommodation (if necessary).

5. **Limited Operating Hours:** Table games at physical locations are subject to operating hours, meaning players may not always be able to find a game when they want to play, especially during off-peak times or late at night.

3.2 Analysis of the Proposed System

First the administrator uploads social engineering attack emails or messages as questions and their corresponding options “a” to “e”, with one as the correct answer. The first level starts with quiz like asking questions on the basics of social engineering. At this stage, players were introduced to the concepts of social engineering. In the game, people get confronted with a possible social engineering threat and have to select a defense mechanism. The users login and answers the social engineering questions and in the end the student score is displayed to the

student. A score above 50 indicates that the user is well knowledgeable and defend him/herself against social engineering attack.

3.2.1 Justification of the Proposed System

1. People learn about different facets of social engineering acts e.g. how social engineering attacks are composed. The learning and application of social engineering while having fun playing creates lasting knowledge on the subject.
2. It warns people about threats they may face in their daily lives and develop a sense of suspicion when being attacked.
3. Ease to learn: A low level of complexity allows learning the game more easily, and thus is more attractive to novices in game play.
4. Ease to play: To be easily integrated into the players' daily routine, the game should have a minimum of necessary preparations and a short play time. Given online games require less preparation than tabletop games, it should be online.

3.3 Architecture of the Proposed System

The system is made with a 3-tier application architecture. This means it has three parts: the part that you see and interact with, the part that runs the program, and the part that stores the information. The data tier holds information, the application tier handles how things work, and the presentation tier is the part you see and interact with on the screen. The three levels are not physical, they are logical. They might be on the same server or different ones. The presentation level of the new system is a web client made with HTML, CSS, JavaScript and JSP. You can see the presentation tier with browsers like Mozilla Firefox, Google Chrome, and Microsoft Edge. The middle part of the new system is made using Java Standard.

Version (JSE) and the more advanced Java Enterprise Edition (JEE) technologies like Servlet and JDBC. The part of the application that runs on the Apache Tomcat Server is called the application tier. The data tier uses the MySQL database. In figure 3. 1, as seen in the figure below

.

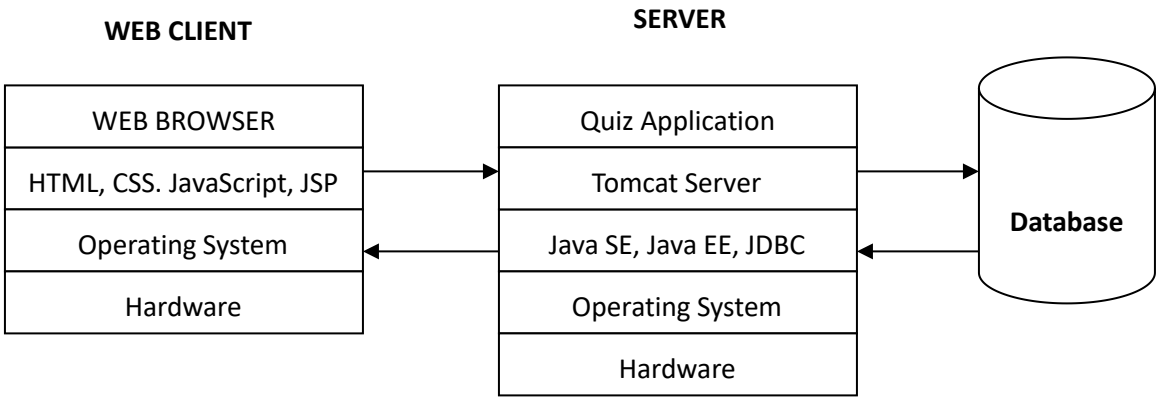


Figure 3.1 System Architecture

3.4 Functional Requirements of the Proposed System

The functional requirements detail the desired behavior and capabilities of the system. This is about what the system needs to do, how it will do it, what information it will keep, and how users will interact with it. The system needs to meet these requirements to work the right way.

1. Users should be able to register on the system
2. Administrator should be able to set social engineering questions.
3. The system should be able to assess user performance and display results.
4. Users should be able to take and submit quiz.
5. Users should be able to view performance and results.

3.5 Methodology of the Study

The software development method we are using for this project is known as Rapid Application Development (RAD). The RAD prefers to make things in stages and quickly create examples instead of planning everything in detail at the beginning. The process of planning and writing software using RAD happens at the same time.

- The planning and writing of software using RAD occur concurrently.

- Software planning and writing with RAD are simultaneous processes.

- The process of RAD involves planning and writing software simultaneously.

Not planning too much in advance makes it quicker to write software and easier to make changes to what is needed. Due to our limited time, using this method will be ideal for rapidly producing prototype versions of the desired system.

3.6 Use Case Diagrams

The use case diagram is used to show the interaction between the system use cases and its clients without much detail. A use case diagram displays an actor and its use cases, the actors are also the users of the system. The users or actors of our system include:

1. Administrator
2. Users

As shown in figure 3.2.

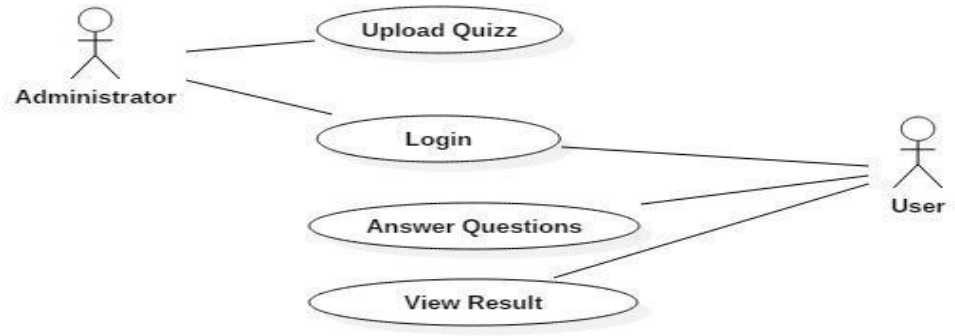


Figure 3.2 Use Case Diagram

3.7 Sequence Diagram

Sequence diagrams are basic parts of interaction diagrams. They plan out the order of events in a project to make things run more smoothly. Sequence diagrams are drawings that show how things work together in a certain situation. A sequence diagram illustrates the sequence of events over time, starting from the top and ending at the bottom. Lower means something that is below or under something else. Later denotes an event taking place subsequent to the current moment. As shown in figure 3.3

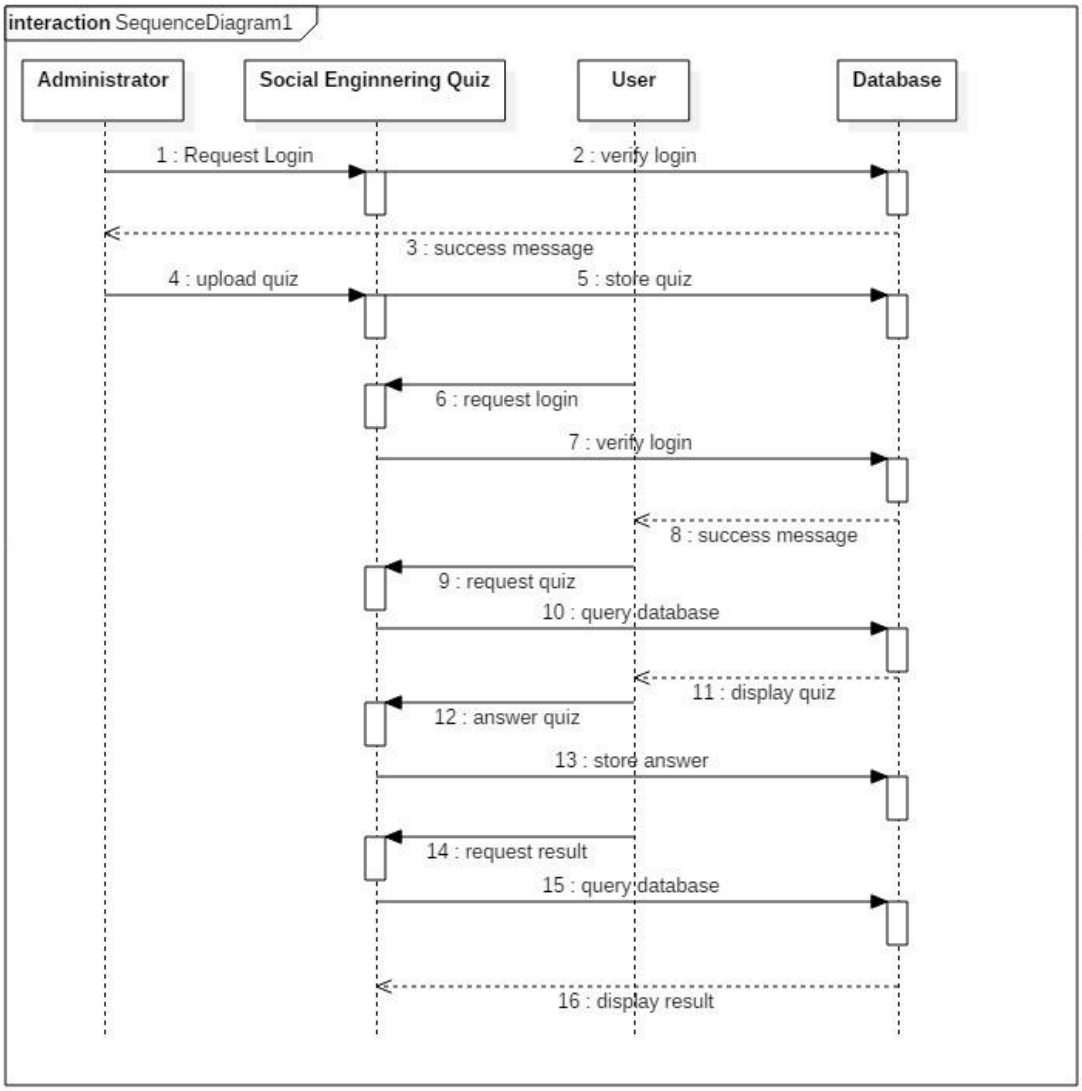
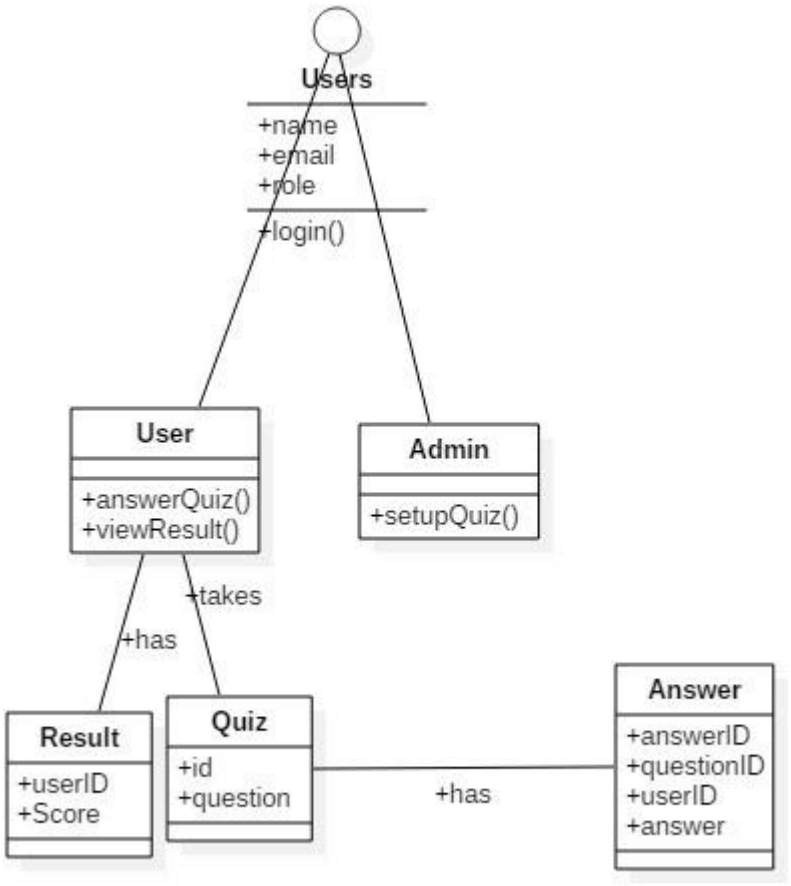


Figure 3.3 Student Sequence Diagram

3.8 Class Diagrams

To begin making the diagram, we identify the classes required for the conference management system. We draw pictures of the classes and write code for them in Java. The class diagram helps us make models using rectangles. Each rectangle represents a class and has three compartments. The first one has the class name centered in bold letters. The middle part has the things that

describe the class, and the bottom part has the things the class can do. Below is the class diagram for the system as shown in figure 3.4



v

Figure 3.4 Class Diagram

3.9 Entity-Relationship Diagram

An Entity-relationship model (ERM) is an abstract and conceptual representation of data. E-R modelling is a database modelling method, used to produce a type of conceptual schema of a system. Diagrams created by this process are called ER diagrams. In this section we present a set of E-R diagrams the conceptual database schema of the system. As shown in figure 3.5

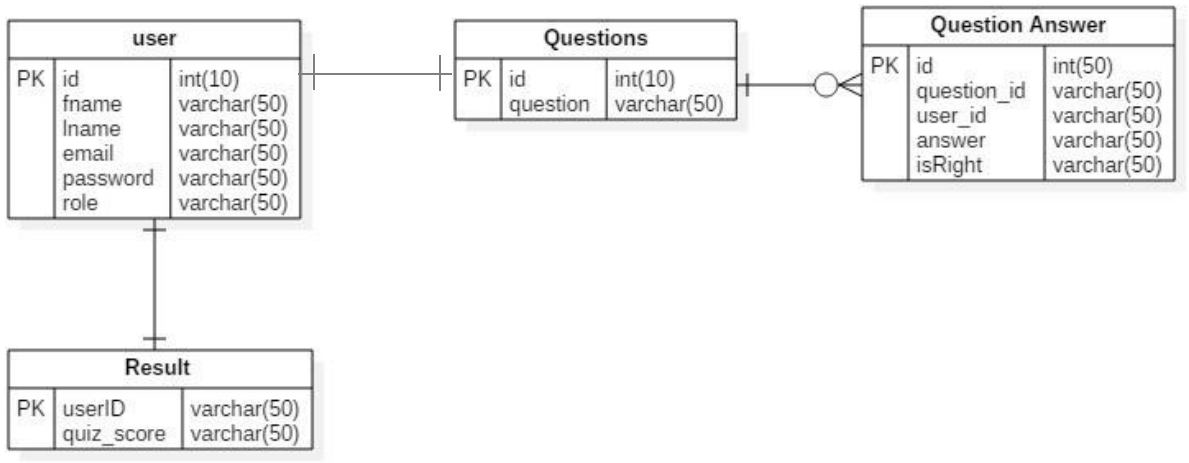


Figure 3.5 E-R Diagram

3.10 Database Design

In this section, the basic structure of the tables composing the database for the project are shown along with information about primary and foreign keys.

Table 1: user

Field	Type	Null	Key	Default	Extra
id	Autoincrement	NO	PRI	NULL	
fname	Varchar(50)	YES		NULL	
lname	Varchar(50)	YES		NULL	
email	Varchar(50)	YES		NULL	
password	Varchar(50)	YES		NULL	
role	Varchar(50)	YES		NULL	

Table 2: Questions

Field	Type	Null	Key	Default	Extra
id	Autoincrement	NO	PRI	NULL	
question	Varchar(500)	YES		NULL	

Table 3: Question Answer

Field	Type	Null	Key	Default	Extra
id	Auto_increment	NO	PRI	NULL	
question_id	Varchar(50)	YES		NULL	
user_id	Varchar(50)	YES		NULL	
answer	Varchar(50)	YES		NULL	
isRight	Varchar(50)	YES		NULL	

Table 4: Result

Field	Type	Null	Key	Default	Extra
Userid	Auto_increment	NO	PRI	NULL	
quiz_score	Varchar(50)	YES		NULL	

3.11 System Flow Chart

This is a graphical representation of the sequence of operations in an information system or program. Information system flowcharts show how data flows from source documents through the computer to final distribution to users. The following figure 3.6 and figure 3.7 are the system flow chart for our system.

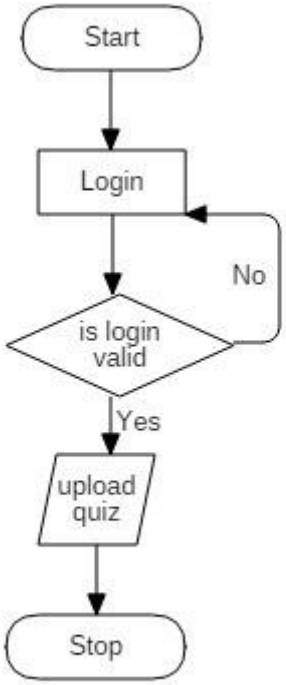


Figure 3.6 Admin Flowchart

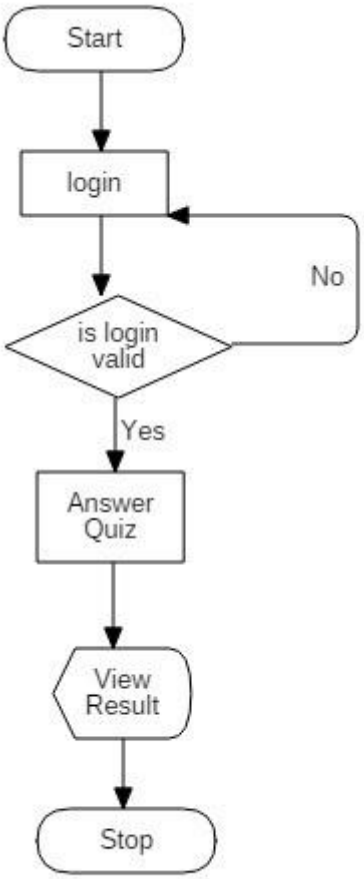


Figure 3.7 User Flowchart

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.1 Choice of Implementation Platform

Java Server Pages (JSP) is a Java-based technology that is run on a server to facilitate the processing of Web-based requests. It helps software developers create dynamically generated web pages based on HTML, Architecturally, JSP may be viewed as a high-level abstraction of Java servlets. JSPs are translated into servlets at runtime; each JSP servlet is cached and re-used until the original JSP is modified. JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, with the resulting page being compiled and executed on the server to deliver a document.

4.2 Justification for Choice of Implementation Platform

JSP has a number of advantages over many of its alternatives. The reason for choosing JSP as a development platform is as follows:

1. Programming keeps the design and functionality of a website separate.
2. JSP programming lets you work on web apps at the same time.
3. JSP programming makes pages compile on their own.
4. JSP environment automatically handles exceptions.
5. Website creators can modify the set design of pages without messing up the website's functionality. In the same way, developers can change how something works without having to change every single page that uses it.
6. The developer can put the java code into the HTML code in JSP.

7. JSP uses the HTTP protocol to communicate and is a very important technology for enabling websites..

4.3 Requirements to Run the System

To run the system, we will need certain hardware and software specification, for both the client and the server computer. This section will outline these specifications

4.3.1 Server Hardware Specification

1. 1GB of RAM
2. 320 GB Hard Disk
3. 2.0 GHZ processor speed.
4. 15 inches Monitor Screen

4.3.2 Server Software Specification

1. 32-bit Operating System, Windows or Linux
2. Java Runtime Environment, version 7
3. Apache Tomcat Server version 7
4. MySQL version 5

4.3.3 Client Hardware Specification

1. 1GB of RAM
2. 320 GB Hard Disk
3. GHZ processor speed
4. 15 inches Monitor Screen

4.3.4 Client Software Specification

1. 32-bit Operating System, Windows or Linux

2. Web browser

4.4 System Sample Output

This section displays the sample interface, and describes the functions of each web page in the system.

4.4.1 The Home Screen

This is the first page the user sees; it contains a form for users to register on the system and it also contains a 'Login' button on the top right corner of the page which redirects to the login page.

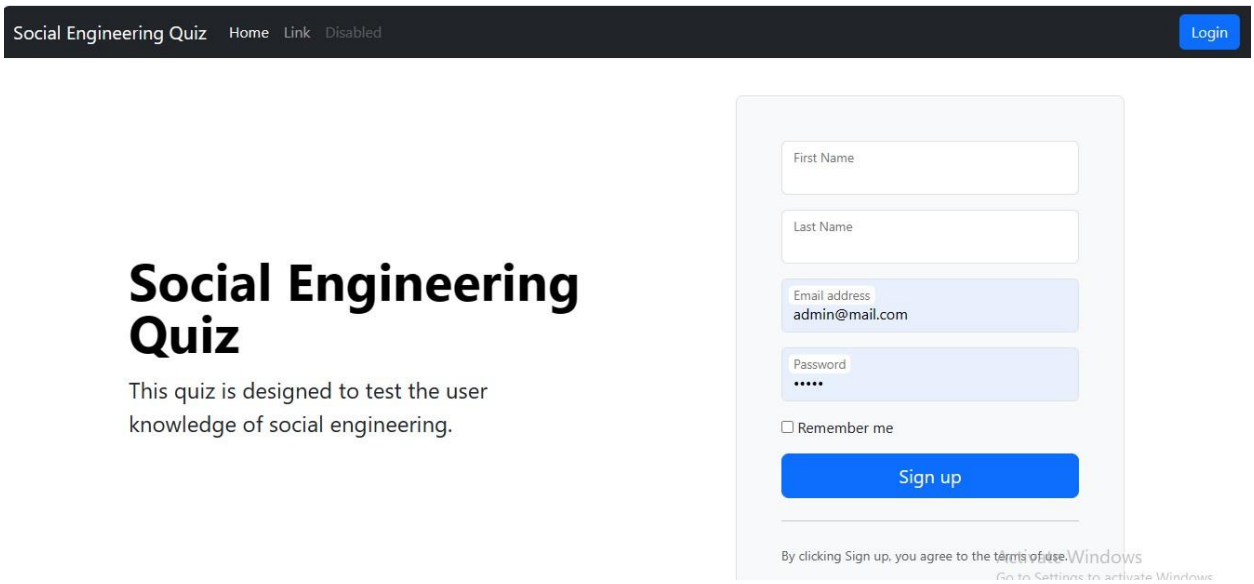


Figure 4.1 Screenshot of home screen

4.4.2 The Login Screen

The login screen contains a form for the user to enter their login credentials (username and password) and a login button. As shown in figure 4.4.2

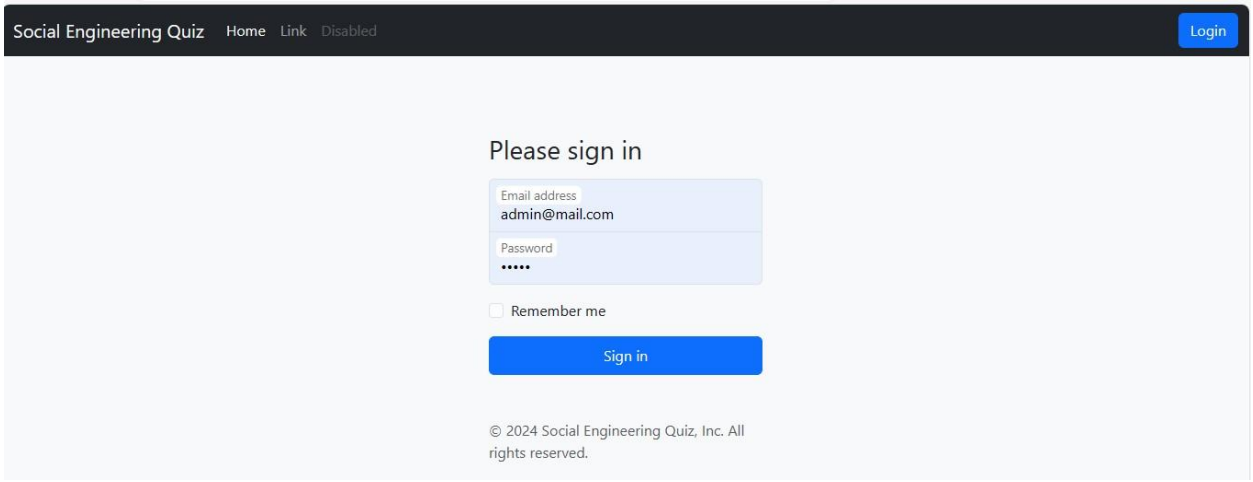


Figure 4.2: screenshots of login page

4.4.3 The Create New Test Page

This page is used by the administrator to create a new social engineering test, it contains a form to specify the name, instruction, description and duration of the test. As shown in figure 4.4.3

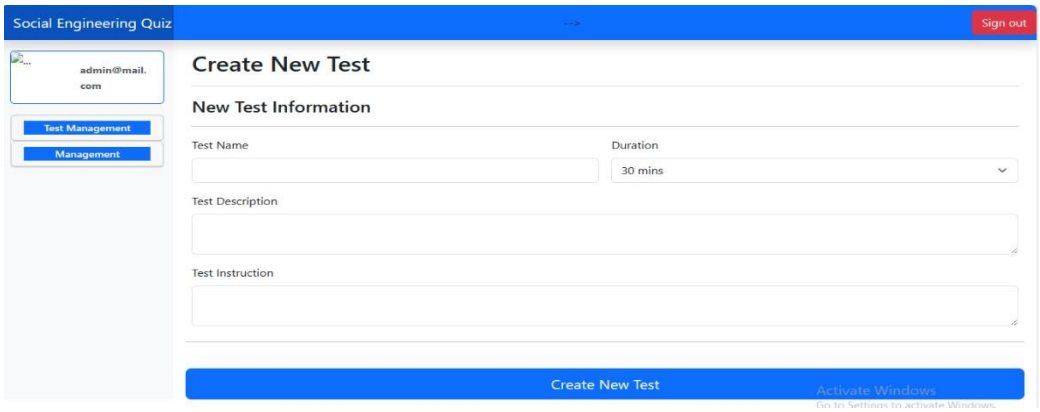


Figure 4.4.3: Screenshot of Create New Test Page

4.4.4 View All Test Page

This page is used by the administrator to view a list of all the tests created, it also contains a button in each item of the list from which the administrator can add questions to the test. As shown in figure 4.4.4

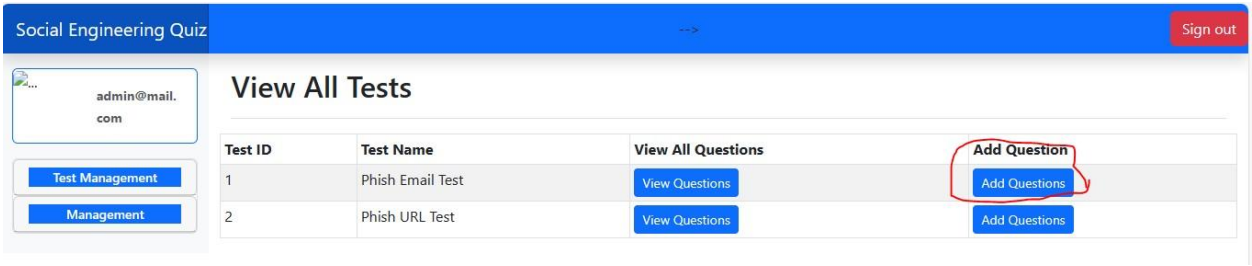


Figure 4.4.4: screenshot of view all Test Page

4.4.5 Add Question Page

This page is used by the administrator to add questions to the test, it contains a text box to enter in the questions and an add options button that redirects to the add option page. As shown in figure 4.4.5

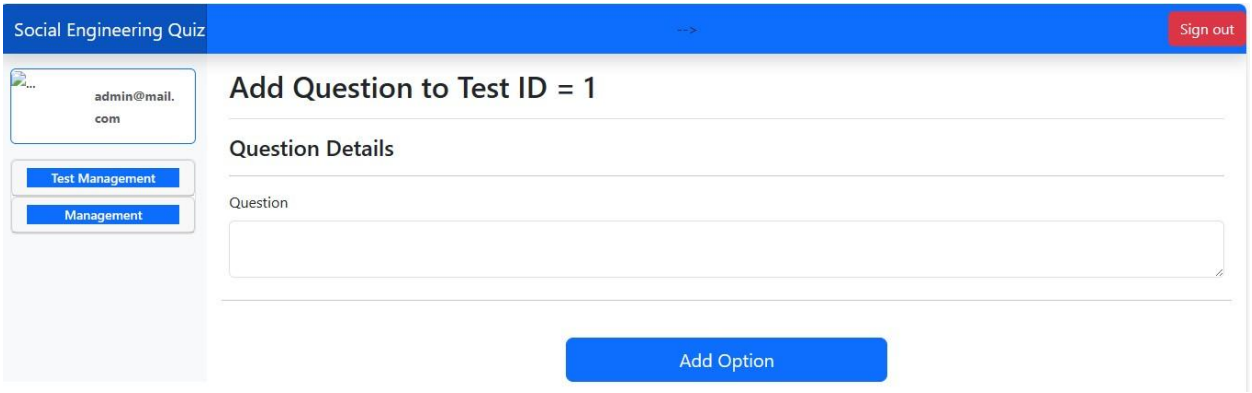


Figure 4.4.5: screenshot of add question page

4.4.6 Add Option Page

This page is used by the administrator to add options to the recently added question, it contains a text box for adding the option and a drop down to specify if the option is the correct one or not. As shown in figure 4.4.6

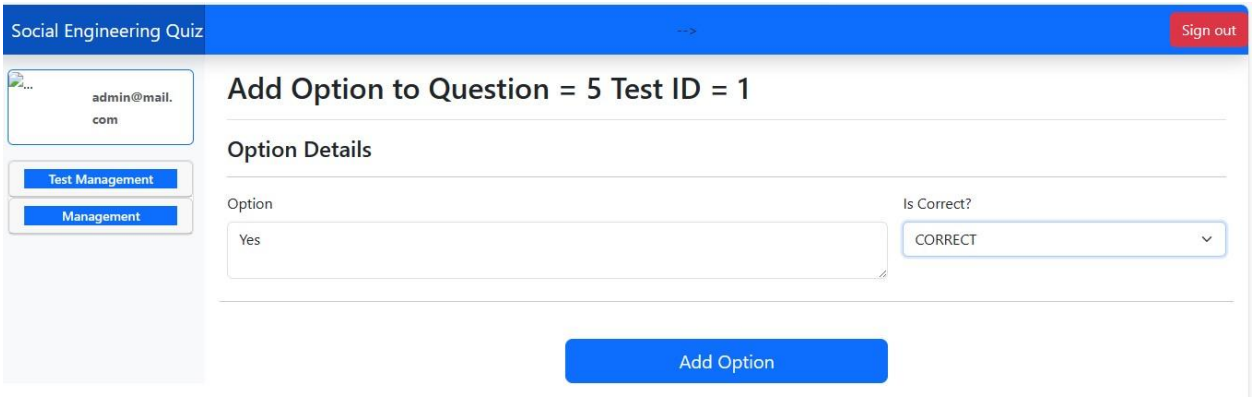


Figure 4.4.6: Screenshot of Add Option Page

4.4.7 User Dashboard

This is the first page the users see after login; it displays a list of the test currently in the system with a description of the test and a button to start each test. As shown in figure 4.4.7

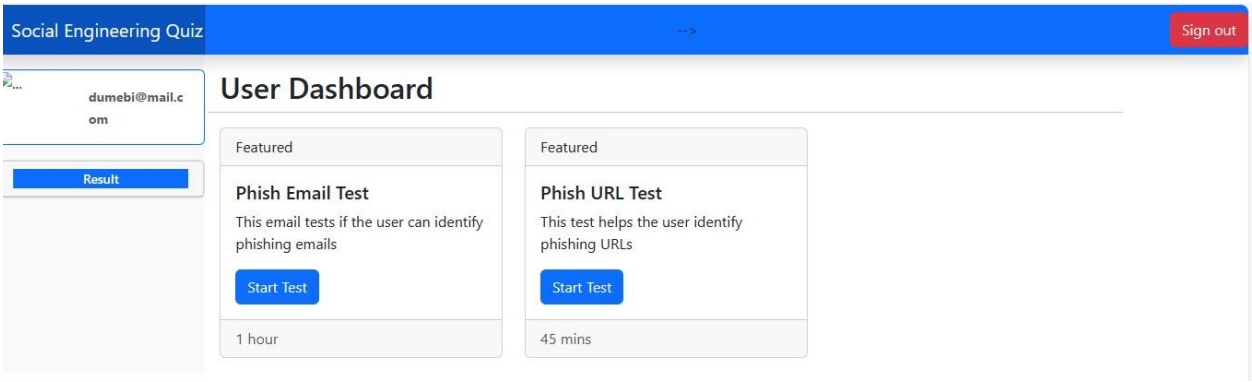


Figure 4.4.7: Screenshot of User Dashboard

4.4.8 The Test Page

After clicking on start test, the user is taken to the test page that display the questions and their options, after takin the test, the user will click on the submit button below to submit the test. As shown in figure 4.4.8

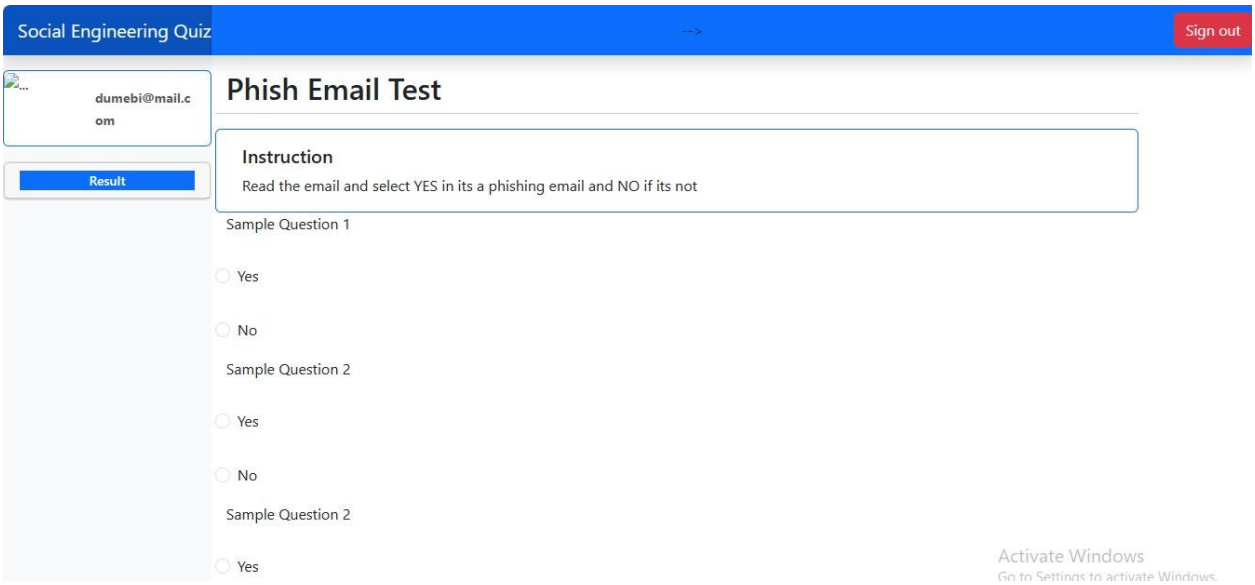


Figure 4.4.8: Screenshot of Test Page

4.5 How to Install the Program

1. Ensure that the server meets the above software and hardware requirements.
2. The software will be built in to a .war file, copy the .war file into the webapps folder of the apache tomcat folder.
3. The complementing database will be distributed as .sql which contains all the tables. Create a database called “secbt” and import the .sql file.

4.6 How to Run the Program

1. Startup Tomcat by running the file C:\jakarta-tomcat-5.0.28\bin\startup.bat
2. Open web browser and enter the address: http://localhost:80/SocialEngineertingQuiz to start.

4.7 Software Testing and Evaluation

Testing, as the final stage of the development of the website, plays a vital role in the process of creating high-quality software. After the website testing, the customer is provided with a ready

project without errors, with good readability, perceived ease, convenience, and reliability. The basic rules for testing of a website are steps that show the user how easy and logical the project is, how easy and possible to find the required information. The following tests were carried out

4.7.1 Website Functionality Test

This test includes the following

Links Testing: all links in the website redirects to their respective URL successfully and there are no broken links.

Form Validation: All forms in the system are validated to accept only the right input in the right format and form data submitted to database successfully.

Database Connection: There is a strong connection to the database and this was tested through the login and form data submission.

4.7.2 Usability Test

This test includes the following

Navigation Testing: The navigation was tested and it was found that there is a way to navigate successfully on the site. The website does not have dead end pages (pages in which you cannot navigate out of). Also, the navigation is intuitive, logical and supported by breadcrumbs.

Content Testing: All spellings and grammar on the website were checked

4.7.3 User Interface Test

This test includes the following

Mobile Friendliness: We tested the website from different screen resolution and all the User Interface elements displayed properly, we also visited the website from a mobile phone and it displayed properly in mobile view.

Browser Compatibility: The website was tested in different browsers such as Mozilla Firefox, Opera, and Google Chrome and the behavior across all browsers was stable.

4.7.4 Security Test

This test includes the following

SQL Injection: SQL injection was tried in the user login form but failed, the SQL injuction could not get past the login page which proves that the website is secure.

Accessing Internal Links: Internal links were tried to be accessed without first logging in, the website refused to display internal links that are visible to only privileged users. This show that the website is secure as someone who is not a user of the website cannot access internal links like the user dashboard.

4.8 Training Requirement

Users were trained for a period of 3 to 6 months to enable them acclimatizes themselves fully with system. The pre-requisite for training includes basic computer appreciation, and the ability to surf the internet.

4.9 File Conversion

When you receive a new computer program, you need to switch from the old one to the new one. This process is called system changeover. It refers to the act of switching from conventional business practices to modern ones. There are different ways to make a change in a strategy. Some

ways are doing the change all at once, doing a small test first, or making changes in stages. This study recommends using the parallel changeover method. This means using both the new and old systems at the same time until you are sure the new system is working well and is not risky. The plan makes sure we can go back to the old system if something goes wrong with the new one. The plan also gives the user time to get used to the new system and feel more confident.

4.10 Commissioning the Project

The system has been thoroughly researched and design is detailed. I am confident that the proposed system designed will serve it purpose with little or no technical glitches. The system can be commissioned as is without further analysis.

CHAPTER FIVE

SUMMARY AND CONCLUSION

5.1 Summary

This study which was carried out as a step in improving the awareness on social engineering was done using a game prototype, this was done in an effort in raising awareness in a fun and relatively easy way so as to make the users have a live experience of how the attacks are being carried out. It gave the participants an opportunity to be acquainted with the terms, terminologies and techniques of social engineering

This research developed a Social Engineering Simulator an Educational guide to mitigate social engineering. The software development methodology employed in this project work is the Rapid application development (RAD). The application is designed using mainly the Java technology, it is designed as a web application so it can run on browsers, as a result, the user interfaces are developed using web technologies such as HTML5, CSS3 and JavaScript. The application logic is developed using Java beans and Java Servlet. The application is designed using the Model-ViewController approach (MVC). The data persistent layer is maintained using the MySQL relational database.

5.2 Conclusion

Social engineering threat assessment usually involves testers who try to break into a company's security system and then tell the company how to fix any problems they find. This indicates deceiving others and potentially infringing on their privacy rights. It only demonstrates a limited number of methods by which individuals can be targeted. We recommend finding an alternative approach that doesn't rely on dishonesty, bypasses the need for external security specialists, and respects individuals' privacy.

We believe that knowing how con artists use human psychology will help people avoid scams and help security experts create stronger systems.

5.3 Recommendation

After the full implementation of the system, it is therefore recommended that;

1. The system can be put on a Tomcat server and accessed online by all users from wherever they are (more details about this are in chapter four).
2. Add Secure Socket Layer to the website hosting the app to make sure data is exchanged safely.
3. The equipment and programs needed to set up the system must be installed.

5.4 Future Work

Due to the limited time involved in developing this project work, some key features could not be integrated, it is my recommendation that in future work, the following features be added.

1. Effort should be made to develop a mobile version of the system for the IOS and Android platforms.
2. More UI/UX (User Interface/User Experience) improvement should be done on the web application.
3. Restful Web services should be developed and published for the system to enable other developers integrate the system into their own portal.

Reference

Abdulkareem S. and Nathan N. (2018), Computer Based Testing (CBT) System for GST Exams in Adamawa State University, Mubi, Asian Journal of Research in Computer Science 2(1): 1-11, 2018; Article no.AJRCOS.44060.

Abdus-Samad T. O., Nur H. Z., (2015) "Social Engineering Awareness Game (Seag): An Empirical Evaluation of Using Game Towards Improving Information Security Awareness", Proceedings of the 5th International Conference on Computing and Informatics, ICOCI.

Agus Tedyyana et al (2017), Design of Computer Based Test Using The Unified Modeling Language, IOP Conf. Ser.: Earth Environ. Sci. 97 012006.

Alessi S. M., Trollip S. R. (2021) Computer-Based Instruction: Methods and Development. – Englewood Cliffs, NJ: Prentice-Hall,. – P. 205-243.

Anderson, Ross. (2024) "Why Cryptosystems fail". Communications of the ACM, 37(11):32–40, Nov. http://www.cl.cam.ac.uk/~rja14/Papers/wcf.pdf.

Ariyapperuma, S., Minhas, A. (2015). Internet Security Games as a Pedagogic Tool for Teaching Network Security, 1–5.

Bernt F.M., Bugbee A.C., Arceo R.D. (2020) Factors influencing student resistance to computer administered testing // J. Res. Comput. Educ. – Vol. 22. – № 3. – P. 265-275.

Coelho, A., Kato, E., Xavier, J., & Gonçalves, R. (2021). Serious game for introductory programming. In Lecture Notes in Computer Science (Vol. 6944, pp. 61–71). doi:10.1007/978-3642-23834-5_6.

Conklin, Wm. Arthur; White, Greg; Cothren, Chuck; Davis, Roger; Williams, Dwayne (2015). Principles of Computer Security, Fourth Edition (Official Comptia Guide). New York: McGrawHill Education. pp. 193–194. ISBN 978-0071835978.

Deschatres, S. (2024). Social Engineering: Attacking the Weakest Link in the Security Chain. http://www.symantec.com/connect/blogs/social-engineeringattacking-weakest-link-securitychain.

Denning T., Kohno T., Shostack A., (2013) "Control-alt-hackTM: A card game for computer security outreach and education" in Proceeding of the 44th ACM Technical Symposium on Computer Science Education, ser. SIGCSE '13. ACM, , pp. 729–729.

Denning, T., Lerner, A., Shostack, A., & Kohno, T. (2023). Control-Alt-Hack: the design and evaluation of a card game for computer security awareness and education. CCS13: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, 915–928. doi:10.1145/2508859.2516753

Dina A., Kristian B. Sebastian P.(2023), "PERSUADED: Fighting Social Engineering Attacks with a Serious Game".

Dimkov T., van Cleeff A., Pieters W., Hartel P., (2020) "Two methodologies for physical penetration testing using social engineering," in Proceedings of the 26th Annual Computer Security Applications Conference, ser. ACSAC '10. ACM, , pp. 399–408.

Evans, Jonathan St.B. T.; Newstead, Stephen E.; Byrne, Ruth M. J. (2019). "Introduction". Human Reasoning: The Psychology of Deduction. Taylor & Francis. ISBN 978-1317716266.

Gaurav M., Arachchilage N.A.G. Berkovsky S. (2017), "Phish Phinder: A Game Design

Approach to Enhance User Confidence in Mitigating Phishing Attacks" https://www.researchgate.net/publication/320464583.

Hatfield, Joseph M (2019). "Virtuous human hacking: The ethics of social engineering in penetration-testing". Computers & Security. 83: 354–366. doi:10.1016/j.cose.2019.02.012.

S2CID 86565713.

Huber, M., Kowalski, S., Nohlberg, M., & Tjoa, S. (2010). Towards Automating Social Engineering Using Social Networking Sites. CSE 09: Proceedings of the 12th IEEE International

Conference on Computational Science and Engineering, 3, 117–124. doi:10.1109/CSE.2009.205.

Irvine C. E., Thompson M. F., Allen K.. (2005) "Cyberciege: gaming for information assurance".

IEEE Security & Privacy, 3(3):61–64.

Jacobs L. C., Chase C. I. (2022) Developing and Using Tests Effectively: A Guide for Faculty. – San Francisco, CA: Jossey-Bass,. – P. 168-177.

Karakasiliotis, A., Furnell, S., Papadaki, M. (2016). Assessing end-user awareness of social engineering and phishing. Information Warfare and Security Conference, 60.

Kirdemir, Baris (2019). "Hostile Influence And Emerging Cognitive Threats In Cyberspace".

Centre for Economics and Foreign Policy Studies.

Kristian B., Sebastian P. (2016), "A Serious Game for Eliciting Social Engineering Security

Requirements"

Microsoft. (2014). Elevation of Privilege (EoP) Card Game. Retrieved April 2, 2015, from http://www.microsoft.com/en-us/sdl/adopt/eop.aspx.

Newbould M. Furnell S. (2009) “Playing safe: A prototype game for raising awareness of social engineering”. In Australian Information Security Management Conference, page 4.

Omorogiuwa O., and F. N. Nwukor (2017), Design and Implementation of a Computer Based Test Centre Using Biometric for Authentication, American Journal of Advanced Research, 2017, 1–1 Sept. 2017, pages 13-18.

Prince Ana and Paul Tawo Bukie, (2013), Design and Implementation Of Online Examination Administration System for Universities, Global Journal Of Mathematical Sciences Vol. 12, 2013: 39-51.

Ross Anderson. (2024) “Why Cryptosystems fail”. Communications of the ACM, 37(11):32–40, Nov. http://www.cl.cam.ac.uk/~rja14/Papers/wcf.pdf.

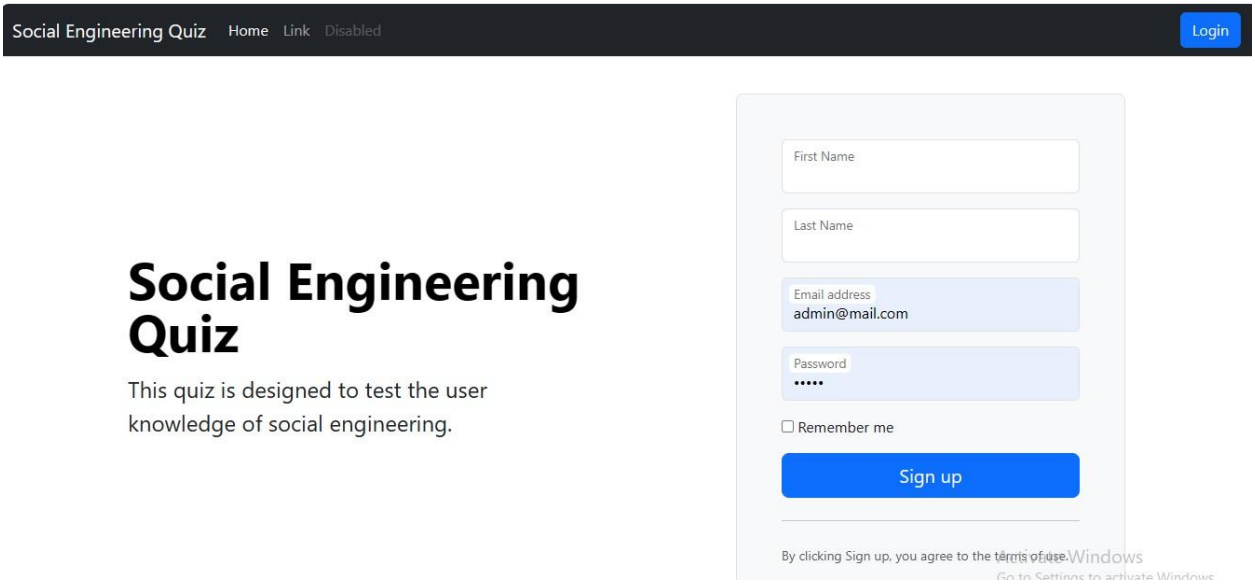
Ryan, W., Stewart, J., Verleger, D. (2023). Network Nightmares : Using Games to Teach Networks and Security.

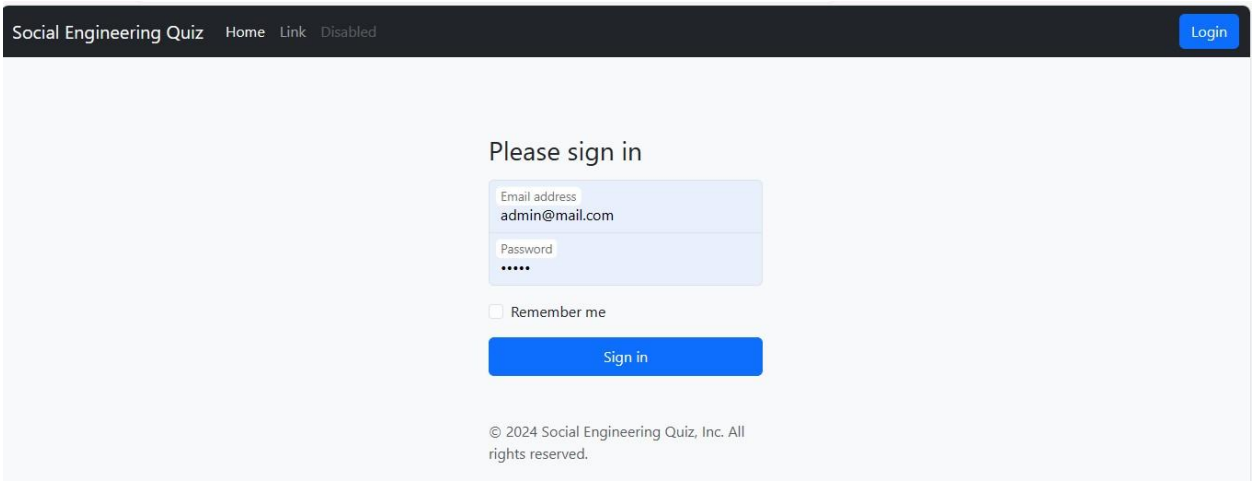
Shavelson R. J., Baxter G. P., Pine J.(2022) Performance assessments: Political rhetoric and measurement reality // Educ. Res. –. – Vol. 21. – №. 4. – P. 22-27.

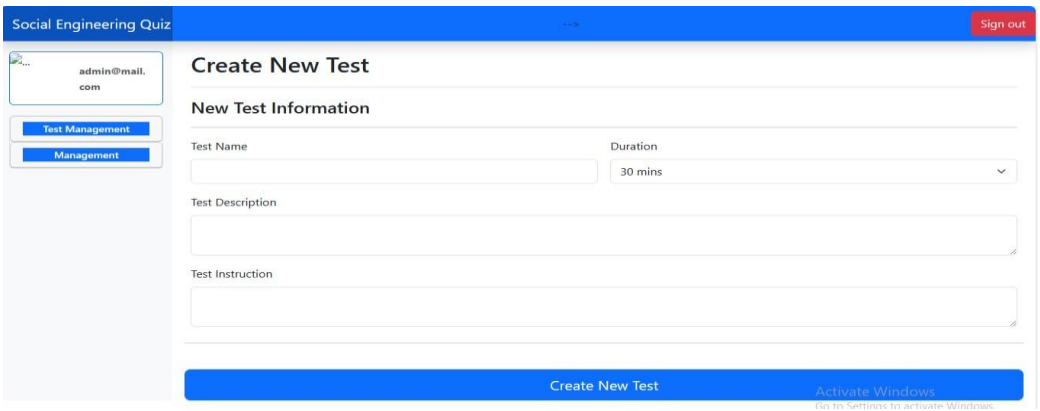
Watson G., Mason A., Ackroyd R., (2021) “Social Engineering Penetration Testing: Executing Social Engineering Pen Tests, Assessments and Defense.” Syngress.

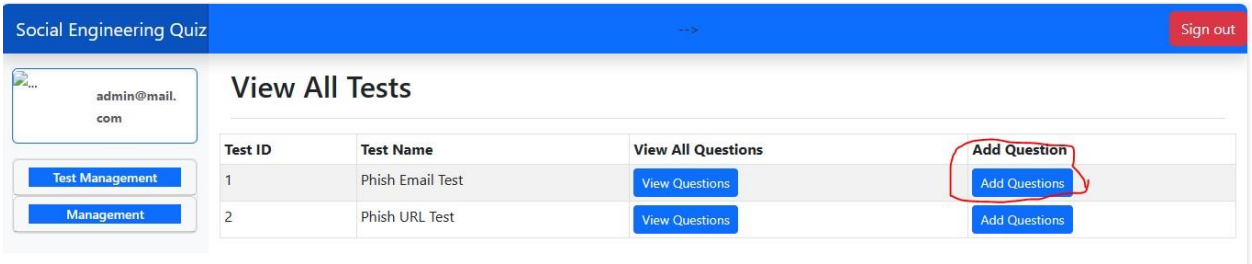
Wise S. L., Plake B. S. (2019) Research on the effects of administering tests via computers // Educ. Meas.: Issues Practice. –. Vol. 8. – № 3. – P. 5-10.

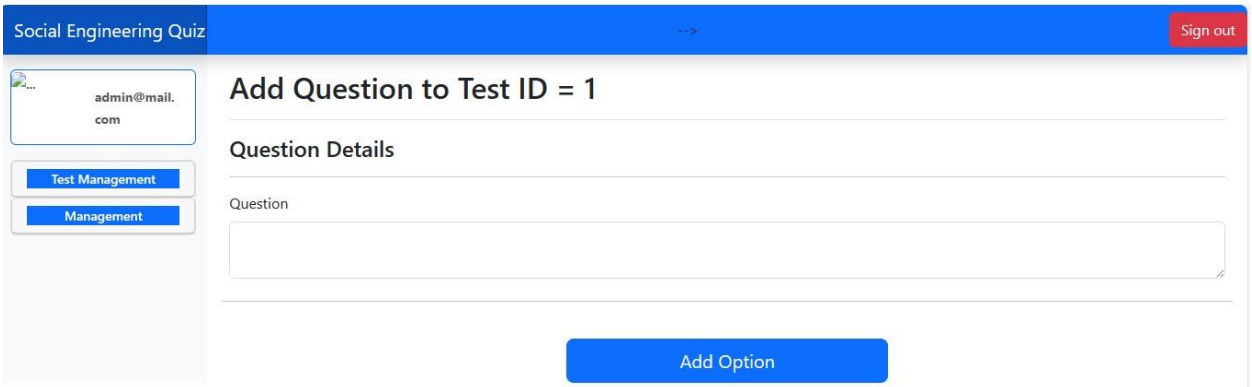
APPENDIX A SAMPLE OUTPUT

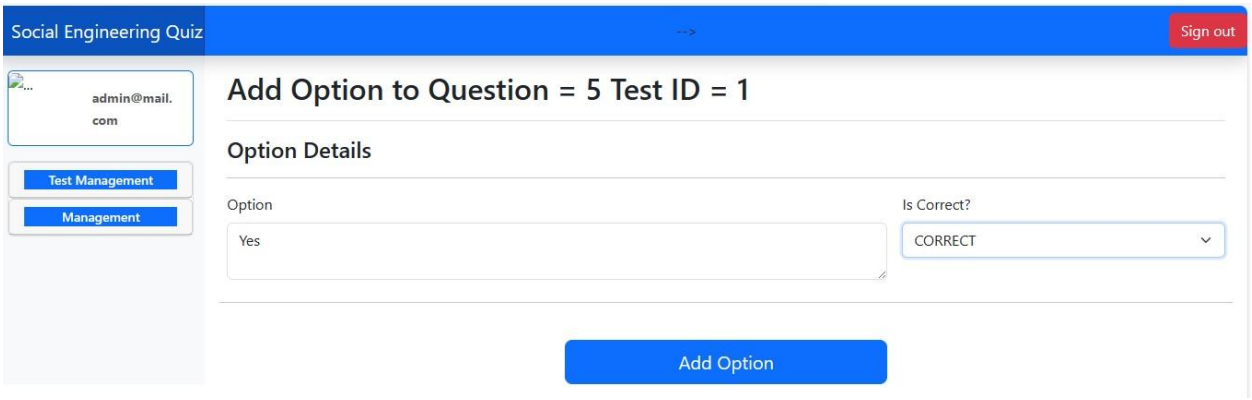


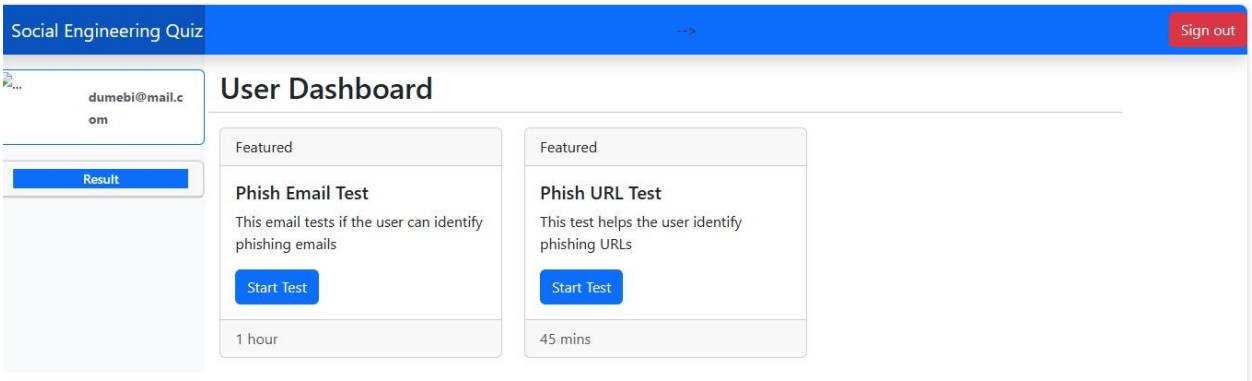





Social Engineering Quiz
Sign out
admin@mail.com
Test Management
Management
View All Tests
Test ID
Test Name
View All Questions
Add Question
1
Phish Email Test
View Questions
Add Questions
2
Phish URL Test
View Questions
Add Questions


Social Engineering Quiz
Sign out
admin@mail.com
Test Management
Management
Add Question to Test ID = 1
Question Details
Question
Add Option


Social Engineering Quiz
Sign out
admin@mail.com
Test Management
Management
Add Option to Question = 5 Test ID = 1
Option Details
Option
Yes
Is Correct?
CORRECT
Add Option


Social Engineering Quiz
Sign out
dumebi@mail.com
Result
User Dashboard
Featured
Phish Email Test
This email tests if the user can identify phishing emails
Start Test
1 hour
Featured
Phish URL Test
This test helps the user identify phishing URLs
Start Test
45 mins


Social Engineering Quiz
Sign out
dumebi@mail.com
Result
Phish Email Test
Instruction
Read the email and select YES in its a phishing email and NO if its not
Sample Question 1
Yes
No
Sample Question 2
Yes
No
Sample Question 2
Yes
Activate Windows

APPENDIX B

User.java

```
package model;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class User {

   private String userID;
private String last_name;
private String email;
private String password;
private String role;     private
String profile_pic;     private
String phone;
}
```

testPage.jsp

```
@page import="model.Option"%>
<%@page import="model.Question"%>
<%@page import="java.util.Iterator"%>
<%@page import="java.util.List"%>
<%@page import="dao.DbConnection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <jsp:include page="/pageParts/meta.jsp"/>
      <title>Social Engineering Quiz - Test Page</title>
      <jsp:include page="/pageParts/css.jsp"/>

       <!-- Custom styles for this template -->
      <link href="/css/dashboard.css" rel="stylesheet">
      <link href="/css/sidebars.css" rel="stylesheet">
   </head>
   <body>
      <%
         String sessionId = (String) request.getAttribute("sessionId");
         String instruction = (String) request.getAttribute("instruction");
         String testId = (String) request.getAttribute("testId");
```

```
        String testName = (String) request.getAttribute("testName");

        DbConnection dbobject = new DbConnection();

        getServletContext().setAttribute("questionList", dbobject.viewAllTestQuestions());
List<Question> questionList = (List<Question>) application.getAttribute("questionList");
Iterator<Question> iterator = questionList.iterator();
    %>
    <jsp:include page="pageParts/user_header.jsp"/>

    <main class="d-flex flex-nowrap">
       <jsp:include page="pageParts/user_left_menu.jsp"/>
       <div class="flex-shrink-0 p-3" style="width: 1000px;">
          <div class="row">
             <h1 class="h2"><%=testName%></h1>
             <hr>
          </div>

             <div class="row">
                <div class="card border-primary">
                   <div class="card-body">
                      <h5 class="card-title">Instruction</h5>
                      <p class="card-text"><%=instruction%></p>
                   </div>
                </div>
</div>
          <form method="post" action="/UserController?user_action=goToTestPage">
<div class="row">

             <%
                while (iterator.hasNext()) {
                   Question q = (Question) iterator.next();
             %>

             <div class="row">
                <p> <%=q.getQuestionText()%></p>
                <%
                   List<Option> optionList = dbobject.viewQuestionsOptions(q.getQuestionID());
                   Iterator<Option> iterator1 = optionList.iterator();
while (iterator1.hasNext()) {
                   Option o = (Option) iterator1.next();
                %>
                <p>
                <div class="form-check">
                   <input type="radio" name="answer" class="form-check-input" id="exampleRadio1"
value="<%=o.getOptionText()%>">
```

```
                    <label class="form-check-label" for="exampleRadio1"> <%=o.getOptionText()%>
</label>
                </div>

                </p>
                <%
                    }
                %>
            </div>

            <%
                }
            %>



<!--                <div class="col"></div>-->
        </div>
            <div class="row">
                <div class="col-md-4"></div>
                <div class="col-md-4"><button  class="btn btn-primary"
type="submit">Submit</button></div>
                <div class="col-md-4"></div>
            </div>
        </form>

      </div>
    </main>


    <script src="/js/bootstrap.bundle.min.js"></script>
    <script src="/js/sidebars.js"></script>
    <script src="/js/dashboard.js"></script>
  </body>
</html>
```

DbConnection.java

```
package dao;

import java.io.FileNotFoundException;
import java.sql.Connection; import
java.sql.DriverManager; import
java.sql.PreparedStatement; import
java.sql.ResultSet; import
java.sql.SQLException; import
java.sql.Statement; import
```

```
java.util.ArrayList; import
java.util.List; import model.Option;
import model.Question;
import model.Test;

public class DbConnection {

   private static final String DATABASE_URL = "jdbc:mysql://localhost/secbt?serverTimezone=UTC";
private static final String USERNAME = "root";     private static final String PASSWORD = "root";
private static Connection connection = null;

   private Statement selectUser = null;     private
Statement checkUsernameEntry = null;     private
Statement viewAllTests = null;
   private Statement viewAllTestQuestions = null;
private Statement getLastQuestionID = null;
   private Statement viewQuestionsOptions = null;



   private PreparedStatement createUserAccount = null;
private PreparedStatement createNewTest = null;
private PreparedStatement addQuestions = null;     private
PreparedStatement addOption = null;




   private ResultSet checkUsernameResultEntry = null;
   private ResultSet selectUserResult = null;     private
ResultSet viewAllTestsResult = null;     private
ResultSet getLastQuestionIDResult = null;     private
ResultSet viewAllTestQuestionsResult = null;     private
ResultSet viewQuestionsOptionsResult = null;



   public static void connectToDataBase() throws ClassNotFoundException {

      try {

         // Load (and therefore register) the  Driver
         Class.forName("com.mysql.cj.jdbc.Driver");

         // Get a Connection to the database
         connection = DriverManager.getConnection(DATABASE_URL, USERNAME, PASSWORD);

      } catch (SQLException error) {
```

```
            System.out.print(error.getMessage());
error.printStackTrace();
            System.out.print("What is trying to say here is that, it could not connect to the database");

        }
    }

    private void closeConnection(Connection connection)
{       if (connection == null) {
            return;
        }
try {
            connection.close();
        } catch (SQLException ex) {
        }
    }

    public String[] userLogin(String username, String password) throws ClassNotFoundException
{       String firstName = "";
        String lastName = "";
        String role = "";
        String username1 = "";
        String password1 = "";
        String profile_pic = "";
        String user_id = "";

        try {

            connectToDataBase();

            String sql = "SELECT * FROM user WHERE email = '" + username + "'";

            selectUser = connection.createStatement();

            selectUserResult = selectUser.executeQuery(sql);

            while (selectUserResult.next()) {               username1 =
selectUserResult.getString("email");              password1 =
selectUserResult.getString("password");               firstName =
selectUserResult.getString("first_name");              lastName =
selectUserResult.getString("last_name");              role =
selectUserResult.getString("role");              profile_pic =
selectUserResult.getString("profile_pic");              user_id =
selectUserResult.getString("user_id");
System.out.println("profile_pic is " +profile_pic);
            }
```

```
        } catch (SQLException error) {

            error.printStackTrace();
            System.out.print(error);
        } finally {
            closeConnection(connection);
        }
        return new String[]{password1, firstName, lastName, username1, role,profile_pic,user_id};
    }

    public int checkUsernameEntry(String email) throws ClassNotFoundException{

        int result = 0;
        try{
            connectToDataBase();

            String sql = "SELECT * FROM user WHERE email = '"+email+"'";

            checkUsernameEntry = connection.createStatement();

            checkUsernameResultEntry = checkUsernameEntry.executeQuery(sql);
if(checkUsernameResultEntry.next()){

              result = 1;
            }

        }catch (SQLException ex)
{           ex.printStackTrace();
        } finally {
            closeConnection(connection);
        }
        return result;
    }

    public void createUserAccount(String email,String first_name, String last_name, String password)
throws ClassNotFoundException, FileNotFoundException, SQLException {        try {

            connectToDataBase();

            String role = "user";

            createUserAccount = connection.prepareStatement("INSERT INTO user(email,first_name,"
+ "last_name,password,role)VALUES(?,?,?,?,?)");

            createUserAccount.setString(1, email);
createUserAccount.setString(2, first_name);
```

```
createUserAccount.setString(3, last_name);
createUserAccount.setString(4, password);
            createUserAccount.setString(5, role);

            createUserAccount.executeUpdate();

        } catch (SQLException error) {

            error.printStackTrace();
            System.out.print(error);
        } finally {
            closeConnection(connection);
        }
    }

    public void createNewTest(String testName,String duration, String description, String instruction)
throws ClassNotFoundException, FileNotFoundException, SQLException {        try {

            connectToDataBase();

            createNewTest = connection.prepareStatement("INSERT INTO tests(testName,duration,"
+ "description,instruction)VALUES(?,?,?,?)");

            createNewTest.setString(1, testName);
createNewTest.setString(2, duration);            createNewTest.setString(3,
description);            createNewTest.setString(4, instruction);


            createNewTest.executeUpdate();

        } catch (SQLException error) {

            error.printStackTrace();
System.out.print(error);
        } finally {
            closeConnection(connection);
        }
    }

    public List<Test> viewAllTests() throws ClassNotFoundException {

        List<Test> result = new ArrayList();

try {

            String sql = "SELECT * FROM tests";
```

```
            connectToDataBase();

            viewAllTests = connection.createStatement();

            viewAllTestsResult = viewAllTests.executeQuery(sql);

            while (viewAllTestsResult.next()) {

               Test t = new Test();

               t.setTestID(viewAllTestsResult.getString("testID"));
               t.setTestName(viewAllTestsResult.getString("testName"));
               t.setDescription(viewAllTestsResult.getString("description"));
               t.setInstruction(viewAllTestsResult.getString("instruction"));
               t.setDuration(viewAllTestsResult.getString("duration"));

               result.add(t);
            }

         } catch (SQLException ex)
{            ex.printStackTrace();
         } finally {
            closeConnection(connection);
         }

         return result;
      }

      public void addQuestions(String testID,String questionText) throws ClassNotFoundException,
FileNotFoundException, SQLException
{        try {

            connectToDataBase();

            addQuestions = connection.prepareStatement("INSERT INTO
questions(testID,questionText)VALUES(?,?)");

            addQuestions.setString(1, testID);
            addQuestions.setString(2, questionText);

            addQuestions.executeUpdate();

         } catch (SQLException error) {

            error.printStackTrace();
```

```
            System.out.print(error);
        } finally {
            closeConnection(connection);
        }
    }

    public String getLastQuestionID() throws ClassNotFoundException {

        String questionID = "";

        try {

            connectToDataBase();

            String sql = "SELECT questionID FROM questions ORDER BY questionID DESC LIMIT 1";

            getLastQuestionID = connection.createStatement();

            getLastQuestionIDResult = getLastQuestionID.executeQuery(sql);

            while (getLastQuestionIDResult.next()) {

                questionID = getLastQuestionIDResult.getString("questionID");
            }

        } catch (SQLException error) {

            error.printStackTrace();
            System.out.print(error);
        } finally {
            closeConnection(connection);
        }

        return questionID;
    }

    public void addOption(String questionID,String option_text,String IsCorrect) throws
ClassNotFoundException, FileNotFoundException, SQLException
{       try {

            connectToDataBase();

            addOption = connection.prepareStatement("INSERT INTO
options(questionID,option_text,IsCorrect)VALUES(?,?,?)");
```

```
            addOption.setString(1, questionID);
addOption.setString(2, option_text);            addOption.setString(3,
IsCorrect);

            addOption.executeUpdate();

        } catch (SQLException error) {

            error.printStackTrace();
            System.out.print(error);
        } finally {
            closeConnection(connection);
        }
    }

    public List<Question> viewAllTestQuestions() throws ClassNotFoundException {

        List<Question> result = new ArrayList();

try {

            String sql = "SELECT * FROM questions";

            connectToDataBase();

            viewAllTestQuestions = connection.createStatement();

            viewAllTestQuestionsResult = viewAllTestQuestions.executeQuery(sql);

            while (viewAllTestQuestionsResult.next()) {

                Question q = new Question();

                q.setQuestionID(viewAllTestQuestionsResult.getString("questionID"));
                q.setQuestionText(viewAllTestQuestionsResult.getString("questionText"));

                result.add(q);
            }

        } catch (SQLException ex)
{           ex.printStackTrace();
        } finally {
            closeConnection(connection);
        }

        return result;
```

```
    }

    public List<Option> viewQuestionsOptions(String questionID) throws ClassNotFoundException {

        List<Option> result = new ArrayList();

try {
            String sql = "SELECT * FROM options WHERE questionID = '"+questionID+"'";

            connectToDataBase();

            viewQuestionsOptions = connection.createStatement();

            viewQuestionsOptionsResult = viewQuestionsOptions.executeQuery(sql);

            while (viewQuestionsOptionsResult.next()) {



                Option o = new Option();

                o.setOptionID(viewQuestionsOptionsResult.getString("optionID"));
                o.setQuestionID(viewQuestionsOptionsResult.getString("questionID"));
                o.setOptionText(viewQuestionsOptionsResult.getString("option_text"));
                o.setIsCorrect(viewQuestionsOptionsResult.getString("IsCorrect"));

                result.add(o);
            }

        } catch (SQLException ex)
{           ex.printStackTrace();
        } finally {
            closeConnection(connection);
        }

        return result;
    }
}
```

Test.java

```
package model;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class Test {
```

```
   private String testID;
private String testName;
private String description;
private String instruction;
private String duration;
}
```

Question.java package
model;

```
import lombok.Getter;
import lombok.Setter;

@Setter @Getter
public class Question {

   private String questionID;
private String testID;     private
String questionText;
}
```

Option.java package
model;

```
import lombok.Getter; import
lombok.Setter;

@Setter @Getter
public class Option {

   private String optionID;
private String questionID;
private String optionText;
private String IsCorrect;
}
```

Answer.java package
model;

```
public class Answer {

   private String answerID;
private String userID;     private
String questionID;     private
String AnswerText;
}
```

UserController.java import
java.sql.SQLException; import
java.util.HashMap; import
javax.servlet.RequestDispatcher; import
javax.servlet.ServletException; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse; import
javax.servlet.http.HttpSession;

```
public class UserController extends HttpServlet {

   private java.util.Map<String, String[]> sessionMap = new HashMap<String, String[]>();// create
HashMap

   public void createAccount (HttpServletRequest request, HttpServletResponse response) throws
ClassNotFoundException, FileNotFoundException, IOException, ServletException, SQLException,
SQLException{

      String firstName = request.getParameter("firstName");
      String lastName = request.getParameter("lastName");
String email = request.getParameter("email");
      String password = request.getParameter("password");

      DbConnection dbConnection = new DbConnection();

      dbConnection.createUserAccount(email,firstName, lastName,password);

      RequestDispatcher rd = request.getRequestDispatcher("/signupSuccessful.jsp");

      rd.forward(request, response);
   }

   public void userLogin(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{

      String[] sessionData = (String[])request.getAttribute("sessionData");

      HttpSession session = request.getSession(true);

      String sessionId = session.getId();

      sessionMap.put(sessionId, sessionData);

      String[] sessionMapValues = sessionMap.get(sessionId);
```

```
        String sessionUserName = sessionMapValues[0];
        String sessionFirstName = sessionMapValues[1];
        String sessionLastrName = sessionMapValues[2];
        String sessionUserImage = sessionMapValues[3];


        request.setAttribute("sessionId", sessionId);
request.setAttribute("sessionUserName", sessionUserName);
request.setAttribute("sessionFirstName", sessionFirstName);
request.setAttribute("sessionLastrName", sessionLastrName);
request.setAttribute("sessionUserImage", sessionUserImage);

        RequestDispatcher rd = request.getRequestDispatcher("/user/userDashboard.jsp");

        rd.forward(request, response);
    }


    public void goToTestPage(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{

        String sessionId = request.getParameter("sessionId");
        String testId = request.getParameter("testId");
        String instruction = request.getParameter("instruction");
        String testName = request.getParameter("testName");


        String[] sessionMapValues = sessionMap.get(sessionId);

        String sessionUserName = sessionMapValues[0];
        String sessionFirstName = sessionMapValues[1];
        String sessionLastrName = sessionMapValues[2];
        String sessionUserImage = sessionMapValues[3];


        request.setAttribute("sessionId", sessionId);
request.setAttribute("sessionUserName", sessionUserName);
request.setAttribute("sessionFirstName", sessionFirstName);
request.setAttribute("sessionLastrName", sessionLastrName);
request.setAttribute("sessionUserImage", sessionUserImage);
        request.setAttribute("testId", testId);
request.setAttribute("testName", testName);
        request.setAttribute("instruction", instruction);

        RequestDispatcher rd = request.getRequestDispatcher("/user/testPage.jsp");
```

```
        rd.forward(request, response);
    }

    public void logout(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        String sessionId = request.getParameter("sessionId");

        sessionMap.remove(sessionId);

        RequestDispatcher rd = getServletContext().getRequestDispatcher("/login.jsp");
rd.forward(request, response);
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {         doPost(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

try {
            String user_action = request.getParameter("user_action");

            switch (user_action) {

                case "createAccount":
createAccount(request, response);
break;                  case "userLogin":
userLogin(request, response);
                    break;                  case
"goToTestPage":
goToTestPage(request, response);
break;
                /**case "goToViewStudentNotificationListPage":
goToViewStudentNotificationListPage(request, response);
break;                  case "viewNotification":
                    goToViewStudentNotificationPage(request, response);
break;**/                   case "logout":                  logout(request,
response);
                    break;
            }

        } catch (Exception error) {
```

```
            error.printStackTrace();
        }
    }

    /**
* Returns a short description of the servlet.
     *
* @return a String containing servlet description
     */
    @Override
    public String getServletInfo()
{        return "Short description";
    }// </editor-fold>

}
```

AdministratorController.java

```
package controller;

import dao.DbConnection; import
java.io.FileNotFoundException; import
java.io.IOException; import
java.io.PrintWriter; import
java.sql.SQLException; import
java.util.HashMap; import
javax.servlet.RequestDispatcher; import
javax.servlet.ServletException; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse; import
javax.servlet.http.HttpSession;

public class AdministratorController extends HttpServlet {

    private java.util.Map<String, String[]> sessionMap = new HashMap<String, String[]>();// create
HashMap

    public void administratorLogin(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{

        String[] sessionData = (String[])request.getAttribute("sessionData");

        HttpSession session = request.getSession(true);

        String sessionId = session.getId();

        sessionMap.put(sessionId, sessionData);
```

```
        String[] sessionMapValues = sessionMap.get(sessionId);

        String sessionUserName = sessionMapValues[0];
        String sessionFirstName = sessionMapValues[1]; String
        sessionLastrName = sessionMapValues[2]; String
        sessionUserImage = sessionMapValues[3];


        request.setAttribute("sessionId", sessionId);
        request.setAttribute("sessionUserName", sessionUserName);
        request.setAttribute("sessionFirstName", sessionFirstName);
        request.setAttribute("sessionLastrName", sessionLastrName);
        request.setAttribute("sessionUserImage", sessionUserImage);

        RequestDispatcher rd = request.getRequestDispatcher("/admin/adminDashboard.jsp");

        rd.forward(request, response);
    }

    public void goToDashboard(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{

        String sessionId = request.getParameter("sessionId");

        String[] sessionMapValues = sessionMap.get(sessionId);

        String sessionUserName = sessionMapValues[0];
        String sessionFirstName = sessionMapValues[1];
        String sessionLastrName = sessionMapValues[2];
        String sessionUserImage = sessionMapValues[3];


        request.setAttribute("sessionId", sessionId);
request.setAttribute("sessionUserName", sessionUserName);
request.setAttribute("sessionFirstName", sessionFirstName);
request.setAttribute("sessionLastrName", sessionLastrName);
request.setAttribute("sessionUserImage", sessionUserImage);

        RequestDispatcher rd = request.getRequestDispatcher("/admin/adminDashboard.jsp");

        rd.forward(request, response);
    }

    public void goToCreateNewTestPage(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException{
```

```
        String sessionId = request.getParameter("sessionId");

        String[] sessionMapValues = sessionMap.get(sessionId);

        String sessionUserName = sessionMapValues[0];
        String sessionFirstName = sessionMapValues[1];
        String sessionLastrName = sessionMapValues[2];
        String sessionUserImage = sessionMapValues[3];

        request.setAttribute("sessionId", sessionId); request.setAttribute("sessionUserName",
        sessionUserName); request.setAttribute("sessionFirstName", sessionFirstName);
        request.setAttribute("sessionLastrName", sessionLastrName);
        request.setAttribute("sessionUserImage", sessionUserImage);

        RequestDispatcher rd = request.getRequestDispatcher("/admin/createNewTestPage.jsp");

        rd.forward(request, response);
    }

    public void createNewTest (HttpServletRequest request, HttpServletResponse response) throws
ClassNotFoundException, FileNotFoundException, IOException, ServletException, SQLException,
SQLException{

        String testName = request.getParameter("testName");
        String duration = request.getParameter("duration");
        String testDescription = request.getParameter("testDescription");
        String testInstruction = request.getParameter("testInstruction");
         String sessionId = request.getParameter("sessionId");

        DbConnection dbConnection = new DbConnection();

        dbConnection.createNewTest(testName,duration, testDescription,testInstruction);

        String[] sessionMapValues = sessionMap.get(sessionId);

        String sessionUserName = sessionMapValues[0];
        String sessionFirstName = sessionMapValues[1];
        String sessionLastrName = sessionMapValues[2];
        String sessionUserImage = sessionMapValues[3];

        request.setAttribute("sessionId", sessionId);
request.setAttribute("sessionUserName", sessionUserName);
request.setAttribute("sessionFirstName", sessionFirstName);
request.setAttribute("sessionLastrName", sessionLastrName);
request.setAttribute("sessionUserImage", sessionUserImage);
```

```
        RequestDispatcher rd = request.getRequestDispatcher("/admin/createNewTestSuccessful.jsp");

        rd.forward(request, response);
    }

    public void goToViewAllTestPage(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{

        String sessionId = request.getParameter("sessionId");

        String[] sessionMapValues = sessionMap.get(sessionId);

        String sessionUserName = sessionMapValues[0];
        String sessionFirstName = sessionMapValues[1];
        String sessionLastrName = sessionMapValues[2]; String
        sessionUserImage = sessionMapValues[3];
        request.setAttribute("sessionId", sessionId);
        request.setAttribute("sessionUserName",
        sessionUserName);
        request.setAttribute("sessionFirstName",
        sessionFirstName);
        request.setAttribute("sessionLastrName",
        sessionLastrName);
       request.setAttribute("sessionUserImage", sessionUserImage);

        RequestDispatcher rd = request.getRequestDispatcher("/admin/viewAllTestPage.jsp");

        rd.forward(request, response);
    }

    public void goToAddQuestionPage(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{

        String sessionId = request.getParameter("sessionId");
        String testId = request.getParameter("testId");

        String[] sessionMapValues = sessionMap.get(sessionId);

        String sessionUserName = sessionMapValues[0];
        String sessionFirstName = sessionMapValues[1];
        String sessionLastrName = sessionMapValues[2];
        String sessionUserImage = sessionMapValues[3];


        request.setAttribute("sessionId", sessionId);
request.setAttribute("sessionUserName", sessionUserName);
```

```
request.setAttribute("sessionFirstName", sessionFirstName);
request.setAttribute("sessionLastrName", sessionLastrName);
request.setAttribute("sessionUserImage", sessionUserImage);
request.setAttribute("testId", testId);

        RequestDispatcher rd = request.getRequestDispatcher("/admin/addQuestionPage.jsp");

        rd.forward(request, response);
    }

    public void goToAddOptionPage(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ClassNotFoundException, FileNotFoundException, SQLException{

        String questionID = "";

        String sessionId = request.getParameter("sessionId");
        String testId = request.getParameter("testId");
        String question = request.getParameter("question");

        DbConnection dbConnection = new DbConnection();

        dbConnection.addQuestions(testId, question);

        questionID = dbConnection.getLastQuestionID();

        String[] sessionMapValues = sessionMap.get(sessionId);

        String sessionUserName = sessionMapValues[0];
String sessionFirstName = sessionMapValues[1];
        String sessionLastrName = sessionMapValues[2];
        String sessionUserImage = sessionMapValues[3];



        request.setAttribute("sessionId", sessionId);
request.setAttribute("sessionUserName", sessionUserName);
request.setAttribute("sessionFirstName", sessionFirstName);
request.setAttribute("sessionLastrName", sessionLastrName);
request.setAttribute("sessionUserImage", sessionUserImage);
        request.setAttribute("testId", testId);
        request.setAttribute("questionID", questionID);

        RequestDispatcher rd = request.getRequestDispatcher("/admin/addOptionsPage.jsp");

        rd.forward(request, response);
    }
```

```
   public void addOption(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ClassNotFoundException, FileNotFoundException, SQLException{

      String questionID = "";

      String sessionId = request.getParameter("sessionId");
      String testId = request.getParameter("testId");
      String question = request.getParameter("question");
      String option = request.getParameter("option");
      String is_correct = request.getParameter("is_correct");

      DbConnection dbConnection = new DbConnection();

      questionID = dbConnection.getLastQuestionID();

      dbConnection.addOption(questionID, option, is_correct);

      String[] sessionMapValues = sessionMap.get(sessionId);

      String sessionUserName = sessionMapValues[0];
      String sessionFirstName = sessionMapValues[1];
      String sessionLastrName = sessionMapValues[2];
      String sessionUserImage = sessionMapValues[3];

      request.setAttribute("sessionId", sessionId); request.setAttribute("sessionUserName",
      sessionUserName); request.setAttribute("sessionFirstName", sessionFirstName);
      request.setAttribute("sessionLastrName", sessionLastrName);
      request.setAttribute("sessionUserImage", sessionUserImage);
      request.setAttribute("testId", testId); request.setAttribute("questionID",
      questionID);

      RequestDispatcher rd = request.getRequestDispatcher("/admin/addOptionsPage.jsp");

      rd.forward(request, response);
   }


   public void logout(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

      String sessionId = request.getParameter("sessionId");

      sessionMap.remove(sessionId);

      RequestDispatcher rd = getServletContext().getRequestDispatcher("/login.jsp");
rd.forward(request, response);
```

```
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {         doPost(request, response);
    }


    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

try{
            String admin_action = request.getParameter("admin_action");

            switch (admin_action) {

                case "administratorLogin":
                    administratorLogin(request, response);
break;                  case "goToDashboard":
                    goToDashboard(request, response);
break;                  case
"goToCreateNewTestPage":
                    goToCreateNewTestPage(request, response);
break;                  case "createNewTest":
createNewTest(request, response);                       break;
case "goToViewAllTestPage":
goToViewAllTestPage(request, response);
break;                  case "goToAddQuestionPage":
                    goToAddQuestionPage(request, response);
break;                  case "goToAddOptionPage":
                    goToAddOptionPage(request, response);
break;                  case "addOption":
                    addOption(request, response);
                    break;
case "logout":
                    logout(request, response);
                    break;
            }

        }catch (Exception error) {

            error.printStackTrace();
        }
    }

    /**
```

```
* Returns a short description of the servlet.
    *
* @return a String containing servlet description
    */
   @Override     public String
getServletInfo() {         return
"Short description";
   }// </editor-fold>

}
```

LoginRouterController.java package controller;

```
import dao.DbConnection; import
java.io.IOException; import
java.io.PrintWriter; import
javax.servlet.RequestDispatcher; import
javax.servlet.ServletException; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class LoginRouterController extends HttpServlet {



   protected void loginRouter(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ClassNotFoundException {

       try {

           String username = request.getParameter("username");
           String password = request.getParameter("password");

           DbConnection user_login = new DbConnection();

           String[] user_details = user_login.userLogin(username, password);

           String user_password = user_details[0];
           String firstName = user_details[1];
           String lastName = user_details[2];
           String username1 = user_details[3];
           String user_role = user_details[4];
           String profile_pic = user_details[5];
           String user_id = user_details[6];

           System.out.println("user name is " + username1);
```

```
        System.out.println("Password is: " + user_password);

        String[] sessionData = {username1, firstName, lastName,profile_pic,user_id};

        //check if fetched username is empty
if (username1.isEmpty()) {

            RequestDispatcher rd1 = request.getRequestDispatcher("/wrongLoginCredentials.jsp");

            rd1.forward(request, response);
        }else {
            int result;

            result = user_login.checkUsernameEntry(username);

            //check if username is in database
            if (result == 1) {
                System.out.println("username is in database");//username is in database

                //check password
                if (password.equals(user_password)) {

                    System.out.println("password matches");//password matches

                    //check if role is admin
if (user_role.equals("admin")) {

                        request.setAttribute("sessionData", sessionData);

                        RequestDispatcher rd2 =
request.getRequestDispatcher("/AdministratorController?admin_action=administratorLogin");
rd2.forward(request, response);

                    }

                    //check if role is user
                    if (user_role.equals("user")) {

                        request.setAttribute("sessionData", sessionData);

                        RequestDispatcher rd2 =
request.getRequestDispatcher("/UserController?user_action=userLogin");

                        rd2.forward(request, response);
                    }
```

```
            }else {

                System.out.println("password does not match");//password does not match

                RequestDispatcher rd3 = request.getRequestDispatcher("/wrongLoginCredentials.jsp");

                rd3.forward(request, response);
            }
          }else{

                System.out.println("username is not in database");//username is not in database

                RequestDispatcher rd4 = request.getRequestDispatcher("/wrongLoginCredentials.jsp");

                rd4.forward(request, response);
          }
        }

      }catch (Exception error) {

        error.printStackTrace();
      }
   }

   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {         doPost(request, response);
   }

   /**
* Handles the HTTP <code>POST</code> method.
    *
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
    */
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

try{
        String login_action = request.getParameter("login_action");
```

```
            switch (login_action) {

                case "loginRouter":
                    loginRouter(request, response);
                    break;
            }

        } catch (Exception error) {

            error.printStackTrace();
        }
    }

    /**
* Returns a short description of the servlet.
     *
* @return a String containing servlet description
     */
    @Override     public String
getServletInfo() {         return
"Short description";
    }// </editor-fold>

}
```