

Design and Implementation of a Quiz Application to Combat Social Engineering Risks

Deborah Knowles

Florida Polytechnic University
Lakeland, Florida 33805, USA
Dknowles1204@floridapoly.edu

Devin Williams

Florida Polytechnic University
Lakeland, Florida 33805, USA
Dwilliams2290@floridapoly.edu

Nicholas Murphy

Florida Polytechnic University
Lakeland, Florida 33805, USA
Nmurphy1658@floridapoly.edu

Knoelle Grassi

Florida Polytechnic University
Lakeland, Florida 33805 USA
Kgrassi2693@floridapoly.edu

Abstract—The project aimed to develop an interactive quiz application designed to raise awareness about social engineering risks. The primary objectives were to implement a system with user authentication, score tracking, a leaderboard, and a quiz using a MySQL database. The solution features a user interface built with Tkinter, providing navigation for users to log in, create accounts, or play as guests. A database-backed system, dynamically fetched quiz questions, tracked user performance, and answer feedback were successfully implemented. The project includes a user-friendly design, secure authentication, and an engaging quiz experience.

Keywords—Quiz Application, User Authentication, Question Fetching, Tkinter, MySQL

I. PROBLEM STATEMENT

Our project addresses the growing risk of social engineering attacks by developing an interactive game that educates users on identifying and mitigating these threats, making cybersecurity awareness engaging and accessible.

II. RELATED WORK

In a world that heavily relies on technology for everything from day-to-day tasks to business activities, there is a growing focus on the prevention of social engineering attack prevention. While social engineering does not necessarily rely on modern technologies, there are more avenues to execute these social engineering attacks now. Due to this, there has been a large push in recent years to provide greater awareness to end users about the risks of these social engineering attacks and how they are executed. This has led to a large variety of resources in this field. There are multitudes of online quizzes and study guides to teach and evaluate users. Many companies require mandatory annual training to ensure their employees are aware of social engineering risks. There are even certifications, like CompTIA Security+, that test knowledge of social engineering attacks. Most of these resources use a similar approach utilizing quizzes to teach and test user knowledge.

Additionally, there are many quiz game platforms like our product that inform users about social engineering risks. These platforms have become popular tools for educating users about social engineering and other cybersecurity risks. For example, Kahoot! offers an interactive, game-based approach that makes learning engaging and competitive, which is effective for corporate training sessions or classrooms. Similarly, Quizlet provides flashcards, quizzes, and other learning tools that allow users to study and reinforce their understanding of key concepts

at their own pace. The project itself uses a quizlet from Quizlet. Reference [5] is a Quizlet about social engineering that the project has questions from. Both Quizlet and Kahoot! Emphasize accessibility and user-friendly interfaces, making them suitable for diverse audiences.

III. CORE DESIGN

The solution was designed to implement a quiz game application that allows users to create accounts, log in, and play quizzes while tracking their progress. The system uses MySQL 8.0 as the Database management System (DBMS) to store and manage user accounts and quiz questions. The application is developed in Python 3.9, leveraging the Tkinter library for the graphical user interface (GUI) to ensure an intuitive and interactive experience for users and the MySQL-connector-python library to access and interact with the database. Key tools included MySQL Workbench for database management and Anaconda for the Python development environment.

The system architecture consists of three layers: the User Interface (UI) layer, the Application Logic Layer, and the Database layer. The UI layer, built with Tkinter, handles user interactions like account creation, login, and gameplay, using widgets such as Entry, Button, and Label. The application Logic layer processes their interactions, validates user inputs, and ensures smooth navigation between screens while maintaining game state and scores. The Database layer executes SQL queries for data storage and retrieval, maintaining the integrity of user credentials and quiz data.

The data workflow (Fig. 2) begins with user input through the GUI (Graphical User Interface), which is processed by Python functions to execute CRUD (Create, Read, Update, Delete) operations on the database. The Users table stores user information like username, password, and scores, while the Quiz table contains questions and their answers. The relationship between these entities is defined in an Entity-Relationship (ER) diagram (Fig. 1), where the user_id serves as a primary key in the Users table. This modular architecture ensures seamless interaction between components, making the system efficient and user-friendly. For enhanced security and scalability, features like password hashing and cloud database integration could be incorporated in future interactions.

During the implementation of the project, multiple outside sources were consulted for extra help. Extra help on implementing the database in python, making a GUI, and getting the social engineering questions. Reference [4] was the first

source and used to retrieve social engineering questions and answers that were later implemented into the database. As previously stated, more social engineering questions were gathered using [5]. Using Reference [2], MySQL was imported into the python environment. [2] was also used to learn how to use python to access the database and execute the SQL Statements. Reference [1] gave the first glimpse into the GUI and gave basic instructions on how to implement the project. Reference [4] also provided implementation of the GUI and was the source of most of what was implemented into the GUI of the social engineering game.

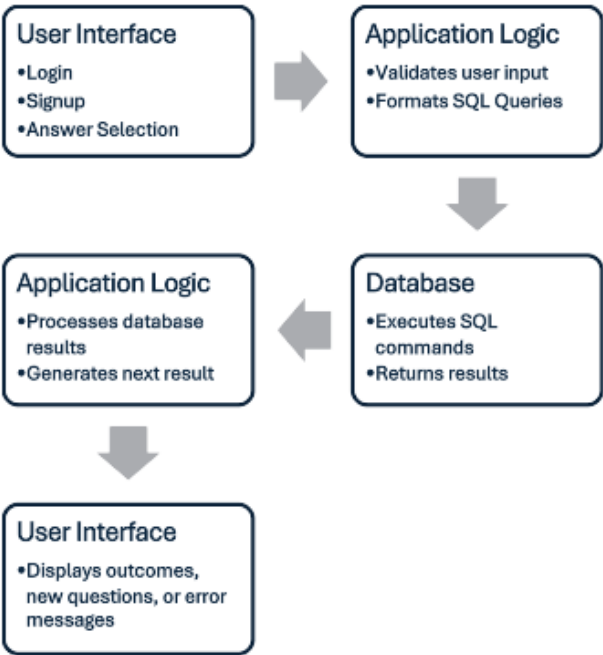


Fig. 1. Data Workflow Diagram

User	Quiz
user_id	question_nbr
username	question
password	a
high_score	b
avg_score	c
last_score	d
num_times_played	correct_answer

Fig. 2. Entity Relationship Diagram

IV. EVALUATION

To evaluate the solution, we focused on functionality, usability, and security to ensure the project met its original

objectives. Success was determined by testing whether the application allowed users to register, log in securely, and interact with quiz questions while maintaining data integrity. The first use case involved user registration and login, where a user could create an account with a unique username and password. I verified this functionality by attempting to register multiple users and ensuring no duplicate usernames were accepted and that passwords were stored securely in the database. The second use case involved playing the quiz, where users could answer questions, receive feedback on correctness, and have their scores updated and saved. This was tested by running through various scenarios to confirm that scores, high scores, and averages were calculated and displayed correctly.

The process began with gathering quiz questions, ensuring a diverse range of topics to make the game engaging. Devin Williams was in charge of this task and its deadline was November 1st. Devin Williams was also in charge of the next task to create the background for the database by creating the tables which had a deadline by November 8th. The next task was to research methods for creating the GUI and integrating the database with Python, which laid the groundwork for the game’s functionality. This had a deadline of November 12th and Devin Williams was in charge. Overall, the deadline for the pre planning of the project was November 12th which was when we planned a meeting to assign future tasks. Additionally, Devin Williams was in charge of this period of tasks.

After these foundational tasks, we assigned specific responsibilities for creating the deliverables and to ensure everyone had the necessary tools and applications installed. The deadline to have everything installed was November 15th and to have the finished game by the middle of thanksgiving break, November 25th. Knoelle Grassi was in charge of creating the code to connect the database to the Python application. Deborah Knowles was in charge of creating the initial GUI. Both these tasks had an initial deadline of November 19th. The connection was completed ahead of time, but the initial GUI had some problems, so the deadline was pushed back. The initial creation of the project was meant to be November 19th. This was moved back to November 21st due to difficulties in downloading the applications and tools.

The next group of tasks was improving the basic GUI to create the game the project required. Due to the difficulties stated before, Knoelle Grassi was in charge of this task. It was completed by November 22nd. Devin Williams and Debroah Knowles oversaw debugging the code which didn’t have a deadline. This included fixing problems such as running the game for infinite questions and figuring out what additional features should be added. Extra features were then added onto the GUI and by November 25th the GUI was finished. Finally, the work was consolidated into a PowerPoint presentation and a report. This task had a deadline of December 2nd.

The Gantt Chart (Fig. 3) shows the different tasks with the timeline for each task. It is color coded to show who primarily did each task. Orange represents Devin Williams, pink represents Knoelle Grassi, purple for Deborah Knowles, and blue represents everyone in the group.



Fig. 3. Gantt Chart

V. CONCLUSION

Overall, with this project, we achieved most of our original goals. The main goal we were not able to achieve that we originally listed in our software requirement specifications was an online hosted database. Rather, we utilized a local hosted database instead. Due to time and complexity restraints this was the most efficient solution for us. We were able to achieve the main function of the game which was to quiz users with social engineering questions. We achieved functionality for users to register an account a login, as well as performance statistics tracking for users. Also, we were able to implement guest user functionality that required no account to play the game. Another goal we were able to achieve was to provide a graphical user interface for the game. Throughout this project we learned how to set up MySQL on windows computers to host local database.

We also learned how to use MySQL Connector to perform database language binding to use MySQL with Python. We also learned how to use the Python library Tkinter to develop basic graphical user interfaces for python programs. This project provided a good experience for us to learn and gain skills with databases and python that we will be able to use in a professional setting. It also gave us experience working in a team, requiring us to work on time management to ensure deliverables are met.

References

- [1] GeeksforGeeks, "Create First GUI Application using PythonTkinter," GeeksforGeeks, Aug. 21, 2024. <https://www.geeksforgeeks.org/create-first-gui-application-using-python-tkinter/>
- [2] freeCodeCamp, "How to Create and Manipulate SQL Databases with Python," freeCodeCamp.org, Aug. 31, 2020. <https://www.freecodecamp.org/news/connect-python-with-sql/>
- [3] ExamCompass, "CompTIA Security+ SY0-701 Exam Social Engineering Quiz," ExamCompass | CompTIA Practice Exams. <https://www.examcompass.com/comptia-security-plus-sy0-701-exam-social-engineering-quiz>
- [4] "tkinter — Python interface to Tcl/Tk," Python Documentation. <https://docs.python.org/3/library/tkinter.html>
- [5] "Security+ SOCIAL ENGINEERING QUIZ," Quizlet. <https://quizlet.com/361105323/security-social-engineering-quiz-flash-cards/>