A logo with a flower and text

Description automatically generated

COMPUTER SCIENCE PROJECT

CLASS: 12/A

TEAM MEMBERS:

AAWESH KUMAR

MANAM SINGH

VISHESH SINGH

This is to certify that , , and

, Students of **class XII** has successfully completed the research on the below mentioned project under the guidance of **Mrs. Cinderella A Augustine** (Teacher) during the year 2023-24 in the partial fulfilment of **Computer Science** practical examination conducted by CBSE , New Delhi

**AMITY INTERNATONAL SCHOOL**

**VIRAJ KHAND, LUCKNOW**

**CERTIFICATE**

**ACKNOWLEDGEMENT**

It is a great pleasure that I am penning down these lines to express my sincere thanks to all those people who helped me in completing this project.

The harmonious environment in our school provided the proper atmosphere for preparing this project. It was a privilege to have been guided by our Teacher Mrs. Cinderella A Augustine and our Principal Mrs. Rachna Mishra.

I am also grateful to my classmates who have helped me during the finalization of this project with their constructive criticism and advice.

**Introduction to Python**

Python is a powerful multi-purpose programming language created by Guido van Rossum.

Python is an easy to learn, powerful programming language. It has efficient high- level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

Python is a cross-platform programming language, meaning, it runs on multiple platforms like Windows, MacOS, Linux and has even been ported to the Java and

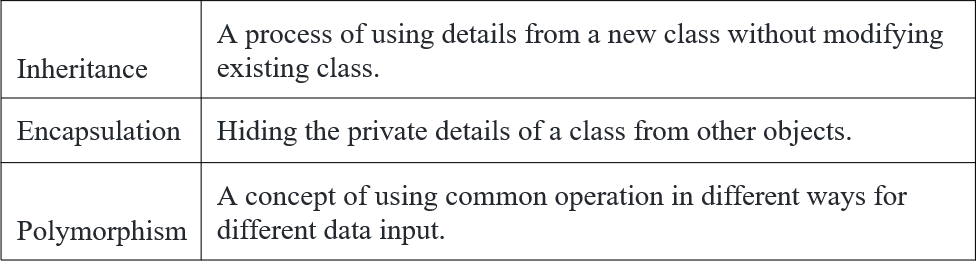
.NET virtual machines. It is free and open source.

**Features of Python Programming**

* A simple **language** which is easier to learn. **Python** has a very simple and elegant syntax. ...
* Free and open-source. ...
* Portability. ...
* Extensible and Embeddable. ...
* A high-level, interpreted **language**. ...
* Large standard libraries to solve common tasks. ...
* Object-oriented

3

QUIZ GAME



**Class**

A class is a blueprint for the object.

**Object**

An object (instance) is an instantiation of a class. When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.

Python is a multi-paradigm programming language. Meaning, it supports different programming approach.

One of the popular approach to solve a programming problem is by creating objects. This is known as Object-Oriented Programming (OOP).

An object has two characteristics:

* attributes
* behaviour

Let's take an example:

Person is an object,

* name, age, color are the attributes
* singing, dancing are the behaviour

The concept of OOP in Python focuses on creating reusable code. This concept is also known as DRY (Don't Repeat Yourself).

In Python, the concept of OOP follows some basic principles:



**Class**

A class is a blueprint for the object.

**Object**

4

QUIZ GAME

**Methods**

Methods are functions defined inside the body of a class. They are used to define the behaviours of an object.

**Inheritance**

Inheritance is a way of creating new class for using details of existing class without modifying it. The newly formed class is a derived class (or child class). Similarly, the existing class is a base class (or parent class).

**Encapsulation**

Using OOP in Python, we can restrict access to methods and variables. This prevent data from direct modification which is called encapsulation. In Python, we denote private attribute using underscore as prefix i.e single “ \_ “ or double “ “.

**Polymorphism**

Polymorphism is an ability (in OOP) to use common interface for multiple form (data types).

Suppose, we need to color a shape, there are multiple shape option (rectangle, square, circle). However we could use same method to color any shape. This concept is called Polymorphism.

**What are exceptions in Python?**

Python has many [built-in exceptions](https://www.programiz.com/python-programming/exceptions) which forces your program to output an error when something in it goes wrong.

When these exceptions occur, it causes the current process to stop and passes it to the calling process until it is handled. If not handled, our program will see.

In Python, exceptions can be handled using a try statement.

A critical operation which can raise exception is placed inside the try clause and the code that handles exception is written in except clause

**Introduction to MYSQL**

MySQL is an open-source, fast reliable, and flexible relational database management system.

* MySQL server design is multi-layered with independent modules.
* MySQL is fully multithreaded by using kernel threads. It can handle multiple CPUs if they are available.
* MySQL provides transactional and non-transactional storage engines.
* MySQL has a high-speed thread-based memory allocation system.
* MySQL supports in-memory heap table.
* MySQL Handles large databases.
* MySQL Server works in client/server or embedded systems.
* MySQL Works on many different platforms.

SQL commands are divided into four subgroups, DDL, DML, DCL, and TCL.

**DDL**

DDL is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

* [CREATE](https://www.w3schools.in/mysql/php-mysql-create/) - to create a database and its objects like (table, index, views, store procedure, function, and triggers)
* ALTER - alters the structure of the existing database
* DROP - delete objects from the database
* TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
* COMMENT - add comments to the data dictionary
* RENAME - rename an object

6

QUIZ GAME

**DML**

DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

* [SELECT](https://www.w3schools.in/mysql/php-mysql-select/) - retrieve data from a database
* [INSERT](https://www.w3schools.in/mysql/php-mysql-insert/) - insert data into a table
* [UPDATE](https://www.w3schools.in/mysql/php-mysql-update/) - updates existing data within a table
* [DELETE](https://www.w3schools.in/mysql/php-mysql-delete/) - Delete all records from a database table
* MERGE - UPSERT operation (insert or update)
* CALL - call a PL/SQL or Java subprogram
* EXPLAIN PLAN - interpretation of the data access path
* LOCK TABLE - concurrency Control

**DCL**

DCL is short name of Data Control Language which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

* GRANT - allow users access privileges to the database
* REVOKE - withdraw users access privileges given by using the GRANT command

**TCL**

TCL is short name of Transaction Control Language which deals with a transaction within a database.

7

QUIZ GAME

* COMMIT - commits a Transaction
* ROLLBACK - rollback a transaction in case of any error occurs
* SAVEPOINT - to rollback the transaction making points within groups
* SET TRANSACTION - specify characteristics of the transaction.

**Connecting MYSQL with Python**

There are the following steps to connect a python application to our database.

* 1. Import mysql.connector module
  2. Create the connection object.
  3. Create the cursor object
  4. Execute the query

**Creating the connection:**

To create a connection between the MySQL database and the python application, the connect() method of mysql.connector module is used.

Pass the database details like Hostname, username, and the database password in the method call. The method returns the connection object.

The syntax to use the connect() is given below.

Connection-Object= mysql.connector.connect(host = <host- name> , user = <username> , passwd = <password> )

**Creating a cursor object:**

The cursor object can be defined as an abstraction specified in the Python DB-API

2.0. It facilitates us to have multiple separate working environments through the same connection to the database. We can create the cursor object by calling the 'cursor' function of the connection object. The cursor object is an important aspect of executing queries to the databases.

The syntax to create the cursor object is given below.

<my\_cur> = conn.cursor()

8

QUIZ GAME

9

QUIZ GAME

* Microsoft Windows 11 as Operating System.
* Python 3.10
* VS code
* MySQL as Back-end Sever with Database for Testing.
* MS-Word 2016 for documentation.

: Intel Pentium Gold G5400 CPU @ 3.70 GHz

:32 GB

:1 GB RAM

:64-bit OS

**The Hardware used:**

* Processor
* Hard Disk Space
* RAM
* System Type

**The Software used:**

**Hardware and Software Specification**

A screenshot of a computer program

Description automatically generated

**Question-Answer Table:**

A screenshot of a computer

Description automatically generated

**User Table :**

**Data Dictionary SQL Tables**

**MY SQL Queries**

**CREATING THE USERTABLE:**

CREATE TABLE UserTable (

ID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

Username VARCHAR(20) NOT NULL,

Password VARCHAR(8) NOT NULL,

Points INT DEFAULT 0);

**CREATING QUESTIONANSWER TABLE:**

CREATE TABLE IF NOT EXISTS QuestionAnswer

( Q\_no INT PRIMARY KEY,

Question TEXT NOT NULL,

Answer TEXT NOT NULL,

QuestionType VARCHAR(20) DEFAULT NULL);

**PROJECT DESCRIPTION**

**Aim :**

To create a simple quiz game using python and SQL connectivity

**Objective:**

-->Develop a very simple quiz game which starts with user information for authorization.

-->Create an easy point system to be updated after each session/play.

**DATABASE MANAGEMENT:**

MySQL is a server-baseddatabase management system. One server might contain multiple databases. To interact with a database, a connection should first be established.

1. Connecting to the MySQL server.
2. Creating a new database.
3. Connecting to the newly created or an existing database.
4. Executing a SQL query and fetch results.
5. Informing the database if any changes are made to a table.
6. Closing the connection to the MySQL server

**SYNTAX USED:**

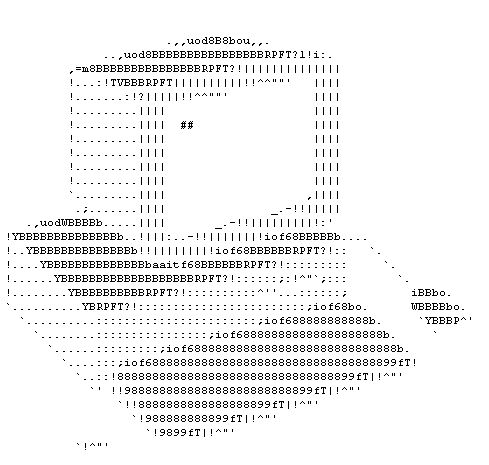
import mysql.connector:

user\_db\_connection = mysql.connector.connect(host="localhost",user="root", password="1234", database="CSproject")

11

QUIZ GAME

< SOURCE CODE >



import mysql.connector

import tkinter as tk

from tkinter import messagebox

import random

import difflib

from tkinter import ttk

def game\_code(username):

    # Function to handle the "Add New User" button click

    def add\_new\_user():

        # Function to save new user data to the UserTable

        def save\_user\_to\_database(username,password):

            try:

                # Establish a connection to the database for user

                user\_db\_connection = mysql.connector.connect(host="localhost",user="root", password="1234", database="CSproject")

                # Create a cursor to interact with the user database

                user\_db\_cursor = user\_db\_connection.cursor()

                # Insert data into the UserTable

                insert\_query = "INSERT INTO UserTable (Username, Password) VALUES (%s,%s)"

                values = (username, password)

                user\_db\_cursor.execute(insert\_query, values)

                user\_db\_connection.commit()

                messagebox.showinfo("Success","Useraddedsuccessfully!ID:{}".format(user\_db\_cursor.lastrowid))

            except mysql.connector.Error as err:

                messagebox.showerror("Error", f"An error occurred: {err}")

            finally:

                if user\_db\_cursor:

                    user\_db\_cursor.close()

                if user\_db\_connection:

                    user\_db\_connection.close()

        user\_window = tk.Toplevel()

        user\_window.title("Add New User")

        user\_window.geometry("300x200")

        username\_label = tk.Label(user\_window, text="Username:")

        username\_label.pack()

        username\_entry = tk.Entry(user\_window)

        username\_entry.pack()

        password\_label = tk.Label(user\_window, text="Password (not greater than 8 characters):")

        password\_label.pack()

        password\_entry = tk.Entry(user\_window, show="\*")

        password\_entry.pack()

        def save\_user():

            new\_username = username\_entry.get()

            new\_password = password\_entry.get()

            if len(new\_password) <= 8:

                save\_user\_to\_database(new\_username, new\_password)

                user\_window.destroy()

            else:

                messagebox.showerror("Error", "Password should not be greater than 8 characters.")

        save\_button = tk.Button(user\_window, text="Save User", command=save\_user)

        save\_button.pack(padx=20, pady=10)

    # Function to add new question to the database

    def add\_question():

        # Establish a connection to the database

        db\_connection = mysql.connector.connect(host="localhost", user="root", password="1234", database="CSproject")

        cursor = db\_connection.cursor()

        qno\_var = tk.StringVar()

        # Display the automatically generated question number

        last\_question\_query = "SELECT MAX(Q\_no) FROM QuestionAnswer"

        cursor.execute(last\_question\_query)

        last\_question\_number = cursor.fetchone()[0]

        next\_question\_number = last\_question\_number + 1

        qno\_var.set(str(next\_question\_number))

        # Function to save data to the database

        def save\_to\_database():

            try:

                qno\_str = qno\_var.get().strip()

                if not qno\_str:

                    raise ValueError("Question number cannot be empty")

                q\_no = int(qno\_str)

            except ValueError as ve:

                messagebox.showerror("Error", f"Invalid input: {ve}")

            except mysql.connector.Error as err:

                messagebox.showerror("Error", f"An error occurred: {err}")

            try:

                # Establish a connection to the database

                db\_connection = mysql.connector.connect(host="localhost", user="root", password="1234", database="CSproject")

                # Create a cursor to interact with the database

                db\_cursor = db\_connection.cursor()

                # Create the QuestionAnswer table if it doesn't exist

                create\_table\_query = """

                CREATE TABLE IF NOT EXISTS QuestionAnswer (

                    Q\_no INT PRIMARY KEY,

                    Question TEXT NOT NULL,

                    Answer TEXT NOT NULL,

                    QuestionType VARCHAR(20) DEFAULT NULL

                );

                """

                db\_cursor.execute(create\_table\_query)

                # Get user input

                question = question\_text.get("1.0", tk.END).strip()

                answer = answer\_text.get("1.0", tk.END).strip()

                question\_type\_input = question\_type\_text.get("1.0", tk.END).strip()  # Retrieve question type from text widget

                # Insert data into the QuestionAnswer table

                insert\_query = "INSERT INTO QuestionAnswer (Q\_no, Question, Answer, QuestionType) VALUES (%s, %s, %s, %s)"

                values = (q\_no, question, answer, question\_type\_input)

                db\_cursor.execute(insert\_query, values)

                db\_connection.commit()

                messagebox.showinfo("Success", "Question and answer saved successfully!")

            except mysql.connector.Error as err:

                messagebox.showerror("Error", f"An error occurred: {err}")

            finally:

                if db\_cursor:

                    db\_cursor.close()

                if db\_connection:

                    db\_connection.close()

        # Create the GUI window

        root = tk.Tk()

        root.title("Question and Answer Entry")

        root.geometry("400x300")

        def cancel():

            root.destroy()  # Close the Add Question window

        # Create and place GUI widgets

        qno\_var = tk.StringVar()

        # Display the automatically generated question number

        last\_question\_query = "SELECT MAX(Q\_no) FROM QuestionAnswer"

        cursor.execute(last\_question\_query)

        last\_question\_number = cursor.fetchone()[0]

        next\_question\_number = last\_question\_number + 1

        qno\_var.set(str(next\_question\_number))

        #Quetion No label

        qno\_label\_text = f"Question Number: {next\_question\_number}"

        qno\_label = tk.Label(root, text=qno\_label\_text)

        qno\_label.pack()

        question\_label = tk.Label(root, text="Question:")

        question\_label.pack()

        question\_text = tk.Text(root, height=3, width=40)

        question\_text.pack(padx=10)

        answer\_label = tk.Label(root, text="Answer:")

        answer\_label.pack()

        answer\_text = tk.Text(root, height=3, width=40)

        answer\_text.pack(padx=10)

        question\_type\_label = tk.Label(root, text="Question Type (Easy, Medium, Hard):")

        question\_type\_label.pack()

        question\_type\_text = tk.Text(root, height=1, width=10)

        question\_type\_text.pack(padx=30)

        save\_button = tk.Button(root, text="Save", command=save\_to\_database)

        save\_button.pack(padx=20, pady=10)

        cancel\_button = tk.Button(root, text="Cancel/Close", command=cancel)

        cancel\_button.pack(padx=20, pady=10)

        root.mainloop()

    # Function to view the question database

    def view\_database():

        # Establish a connection to the database

        db\_connection = mysql.connector.connect(host="localhost", user="root", password="1234", database="CSproject")

        # Create a cursor to interact with the database

        db\_cursor = db\_connection.cursor()

        try:

            def cancel():

                view\_window.destroy()

            # Query to fetch all rows from the QuestionAnswer table

            select\_query = "SELECT \* FROM QuestionAnswer"

            db\_cursor.execute(select\_query)

            rows = db\_cursor.fetchall()

            if not rows:

                messagebox.showinfo("Info", "No questions available in the database.")

                return

            # Create a new window to display the tabular data

            view\_window = tk.Toplevel(app)

            view\_window.title("View Database")

            view\_window.geometry("1500x600")

            # Create a Treeview widget to display the data in a tabular form

            tree = ttk.Treeview(view\_window, columns=("Q\_no", "Question", "Answer", "QuestionType"))

            tree.heading("#1", text="Question Number")

            tree.heading("#2", text="Question")

            tree.heading("#3", text="Answer")

            tree.heading("#4", text="Question Type")

            cancel\_button = tk.Button(view\_window, text="Close", command=cancel)

            cancel\_button.pack()

            # Insert the data into the Treeview

            for row in rows:

                tree.insert("", "end", values=row)

            tree.pack()

        except mysql.connector.Error as err:

            messagebox.showerror("Error", f"An error occurred: {err}")

    # Function to play the quiz game

    def play\_quiz():

        #Establish connection with database

        db\_connection = mysql.connector.connect(host="localhost", user="root", password="1234", database="CSproject")

        cursor = db\_connection.cursor()

         # Select a random question from the database

        cursor.execute("SELECT question FROM QuestionAnswer")

        questions = cursor.fetchall()

        if not questions:

            messagebox.showinfo("Info", "No questions available in the database.")

            return

        # Access the first element of the tuple

        random\_question = random.choice(questions)[0]

        # Create a new window for the quiz game

        quiz\_window = tk.Toplevel(app)

        quiz\_window.title("Quiz Game")

        quiz\_window.geometry("500x400")

        def cancel():

            quiz\_window.destroy()

        # Display the random question

        question\_label = tk.Label(quiz\_window, text=random\_question)

        question\_label.pack()

        # Entry for user's answer

        answer\_entry = tk.Entry(quiz\_window)

        answer\_entry.pack()

        cancel\_button = tk.Button(quiz\_window, text="Cancel", command=cancel)

        cancel\_button.pack(padx=20, pady=10)

            # Add a Help button

        help\_button = tk.Button(app, text="Help", command=display\_help)

        help\_button.pack()

        # Function to check the answer

        def check\_answer():

            db\_connection = mysql.connector.connect(host="localhost", user="root", password="1234", database="CSproject")

            db\_cursor = db\_connection.cursor()

            user\_answer = answer\_entry.get()

            correct\_answer = db\_cursor.execute("SELECT Answer FROM QuestionAnswer WHERE question = %s", (random\_question,))

            correct\_answer = db\_cursor.fetchone()[0]

            print(user\_answer)

            print(correct\_answer)

            # Compare the user's answer with the correct answer

            similarity\_ratio = difflib.SequenceMatcher(user\_answer, correct\_answer).ratio()

            try:

                if user\_answer:

                    # Compare the user's lowercase answer with the correct answer

                    similarity\_ratio = difflib.SequenceMatcher(None, correct\_answer,user\_answer.lower()).ratio()

                    print(similarity\_ratio)

                    if (similarity\_ratio == 1.0) or (similarity\_ratio>=0.90):

                        result = "Correct"

                        db\_connection = mysql.connector.connect(host="localhost", user="root", password="1234", database="CSproject")

                        db\_cursor = db\_connection.cursor()

                        # Execute the update query

                        update\_points\_query = "UPDATE UserTable SET Points = Points + 1 WHERE Username = ","%s"

                        db\_cursor.execute(update\_points\_query, (current\_username))

                        db\_connection.commit()

                        # Update points for the user in the database

                        current\_points\_query = "SELECT Points FROM UserTable WHERE Username = %s"

                        db\_cursor.execute(current\_points\_query, (current\_username,))

                        current\_points = db\_cursor.fetchone()[0]

                        # Execute the update query

                        update\_points\_query = "UPDATE UserTable SET Points = Points + 1 WHERE Username = %s"

                        db\_cursor.execute(update\_points\_query, current\_username)

                        db\_connection.commit()

                        # Update current points and user info label

                        current\_points += 1

                        update\_user\_info(current\_username, current\_points)

                    # Check for spelling mistakes

                    elif (similarity\_ratio < 0.90) or (similarity\_ratio > 0.60):

                        result = "Almost Correct :( "

                        messagebox.showinfo("Result", "Your answer was correct but there was a spelling error or the anser was incomplete, No points will be given")

                    elif user\_answer == "":

                        result = "Please provide an answer!"

                    else :

                        result = "Incorrect!"

                        messagebox.showinfo("result")

            except:

                pass# mysql.connector.Error as err:

                #messagebox.showerror("Error", f"An error occurred: {err}")

            messagebox.showinfo("Result", f"Your answer is {result}.")

            # Close the quiz window

            quiz\_window.destroy()

        check\_button = tk.Button(quiz\_window, text="Check Answer", command=check\_answer)

        check\_button.pack()

    # Create the main application window

    app = tk.Tk()

    app.title("Quiz Game")

    app.geometry("669x450")

    def cancel():

        app.destroy()

        print("Game closed succesfully!")

    def display\_help():

        help\_content = """

        Welcome to the Quiz Game Help!

        This game allows users to play a quiz and manage questions.

        - To play the quiz, click the "Play Quiz Game" button and answer questions.

            >The answers are not case sensitive.

            >In numerical based questions, Don't forget to write the units!

        - To add a new question, click the "Add New Question" button and enter details.

            >If you want to add more than one ques, make sure to close and reopen the "add question" window after every successful attempt.

        - To view the question database, click the "View Question Database" button.

        - To add a new user, click the "Add New User" button and provide details.

            >The password must not exceed the lenght of 8 character(all special characters are allowed)

            >The password is case sensitive!

        - The Points displayed on the main menu will be updated only when the Game is closed.

            >The updated points will be displayed the next time you open the game :)

        For any further assistance, please contact our support team.

        """

        help\_window = tk.Toplevel()

        help\_window.title("Help")

        help\_text = tk.Label(help\_window, text=help\_content, padx=20, pady=20, anchor="w")

        help\_text.pack(anchor ="w")

    # Function to update and display user information

    def update\_user\_info(username, points):

        user\_info\_label.config(text=f"Username: {username}\nPoints: {points}")

    # Initial values for username and points

    current\_username = username

    db\_connection = mysql.connector.connect(host="localhost", user="root", password="1234", database="CSproject")

    info\_cursor = db\_connection.cursor()

    # Query to fetch the points of the current username

    points\_query = "SELECT Points FROM UserTable WHERE Username = %s"

    info\_cursor.execute(points\_query, (current\_username,))

    current\_points = info\_cursor.fetchone()[0]

    # Create a label to display user information

    user\_info\_label = tk.Label(app, text=f"Username: {current\_username}\nPoints: {current\_points}", anchor="w", padx=10)

    user\_info\_label.pack(side="top", fill="both")

    # Create and place the buttons horizontally

    #Add New question button

    add\_button = tk.Button(app, text="Add New Question", command=add\_question)

    add\_button.pack(side="left", padx=10, pady=10)

    #Play button

    play\_button = tk.Button(app, text="Play Quiz Game", command=play\_quiz)

    play\_button.pack(side="left", padx=10, pady=10)

    #View database button

    view\_button = tk.Button(app, text="View Question Database", command=view\_database)

    view\_button.pack(side="left", padx=10, pady=10)

    #Add a user button

    add\_user\_button = tk.Button(app, text="Add New User", command=add\_new\_user)

    add\_user\_button.pack(side="left", padx=10, pady=10)

    #Add a Cancel button

    cancel\_button = tk.Button(app, text="Cancel/Close", command=cancel)

    cancel\_button.pack(side="left", anchor="sw", padx=10, pady=(1, 10))

    # Add a Help button

    help\_button = tk.Button(app, text="Help", command=display\_help)

    help\_button.pack(side="left", anchor="se", padx=10, pady=(1, 10))

    # Run the GUI application

    app.mainloop()

def authenticate\_and\_run\_game():

    # Function to handle the "Add New User" button click

    def add\_new\_user():

        # Function to save new user data to the UserTable

        def save\_user\_to\_database(username,password):

            try:

                # Establish a connection to the database for user

                user\_db\_connection = mysql.connector.connect(host="localhost", user="root", password="1234", database="CSproject")

                # Create a cursor to interact with the user database

                user\_db\_cursor = user\_db\_connection.cursor()

                # Insert data into the UserTable

                insert\_query = "INSERT INTO UserTable (Username, Password) VALUES (%s, %s)"

                values = (username, password)

                user\_db\_cursor.execute(insert\_query, values)

                user\_db\_connection.commit()

                messagebox.showinfo("Success", "User added successfully! ID: {}".format(user\_db\_cursor.lastrowid))

            except mysql.connector.Error as err:

                messagebox.showerror("Error", f"An error occurred: {err}")

            finally:

                if user\_db\_cursor:

                    user\_db\_cursor.close()

                if user\_db\_connection:

                    user\_db\_connection.close()

        user\_window = tk.Toplevel()

        user\_window.title("Add New User")

        user\_window.geometry("300x200")

        username\_label = tk.Label(user\_window, text="Username:")

        username\_label.pack()

        username\_entry = tk.Entry(user\_window)

        username\_entry.pack()

        password\_label = tk.Label(user\_window, text="Password (not greater than 8 characters):")

        password\_label.pack()

        password\_entry = tk.Entry(user\_window, show="\*")

        password\_entry.pack()

        def save\_user():

            new\_username = username\_entry.get()

            new\_password = password\_entry.get()

            if len(new\_password) <= 8:

                save\_user\_to\_database(new\_username, new\_password)

                user\_window.destroy()

            else:

                messagebox.showerror("Error", "Password should not be greater than 8 characters.")

        save\_button = tk.Button(user\_window, text="Save User", command=save\_user)

        save\_button.pack()

    def validate\_credentials():

        entered\_username = username\_entry.get()

        entered\_password = password\_entry.get()

        # Validate credentials against the database

        try:

            db\_connection = mysql.connector.connect(host="localhost", user="root", password="1234", database="CSproject")

            db\_cursor = db\_connection.cursor()

            query = "SELECT \* FROM UserTable WHERE Username = %s AND Password = %s"

            values = (entered\_username, entered\_password)

            db\_cursor.execute(query, values)

            user\_data = db\_cursor.fetchone()

            if user\_data:

                auth\_window.destroy()

                game\_code(entered\_username)  # Call the function to run the game

            else:

                messagebox.showerror("Error", "Invalid username or password!")

        except mysql.connector.Error as err:

            messagebox.showerror("Error", f"An error occurred: {err}")

        finally:

            if db\_cursor:

                db\_cursor.close()

            if db\_connection:

                db\_connection.close()

    auth\_window = tk.Tk()

    auth\_window.title("Authentication")

    auth\_window.geometry("300x200")

    # Change the background color to gray

    auth\_window.configure(bg="gray13")

    username\_label = tk.Label(auth\_window, text="Username:")

    username\_label.pack()

    username\_entry = tk.Entry(auth\_window)

    username\_entry.pack(pady=3)

    password\_label = tk.Label(auth\_window, text="Password:")

    password\_label.pack(pady=5)

    password\_entry = tk.Entry(auth\_window, show="\*")

    password\_entry.pack(pady=3)

    play\_button = tk.Button(auth\_window, text="Play", command=validate\_credentials)

    play\_button.pack(padx=20, pady=10)

    play\_button.configure(bg="light gray")

    add\_user\_button = tk.Button(auth\_window, text="Add New User", command=add\_new\_user)

    add\_user\_button.pack(padx=20, pady=10)

    add\_user\_button.configure(bg="light gray")

 auth\_window.mainloop()

# Call the authentication function to start the application

authenticate\_and\_run\_game()

OUTPUT SCREENS

A close-up of a game controller

Description automatically generated

Authentication Window:

A screenshot of a computer

Description automatically generated

Error Window:

A screenshot of a computer error

Description automatically generated

Adding a new User:

A screenshot of a computer

Description automatically generated

Main Menu:

A screenshot of a computer

Description automatically generated

**Options**:

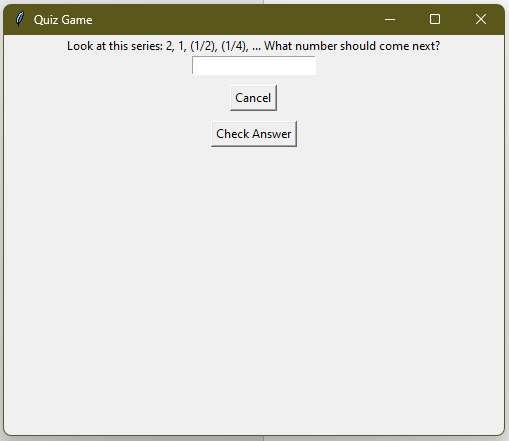
1. Add New User: *(add description)*
2. Play Quiz Game: *(add description)*
3. View Question Database: *(add description)*
4. Add New User: *(add description)*
5. Cancel: *(add description)*
6. Help: *(add description)*

Quick User Info:

*(add description)*

**Playing the Game!**

*Question Window:*

****

*Entering the answer*:

A screenshot of a computer

Description automatically generated

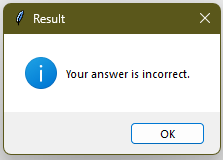
*Result:*

A screenshot of a computer error message

Description automatically generated

For correct Answer

For wrong Answers



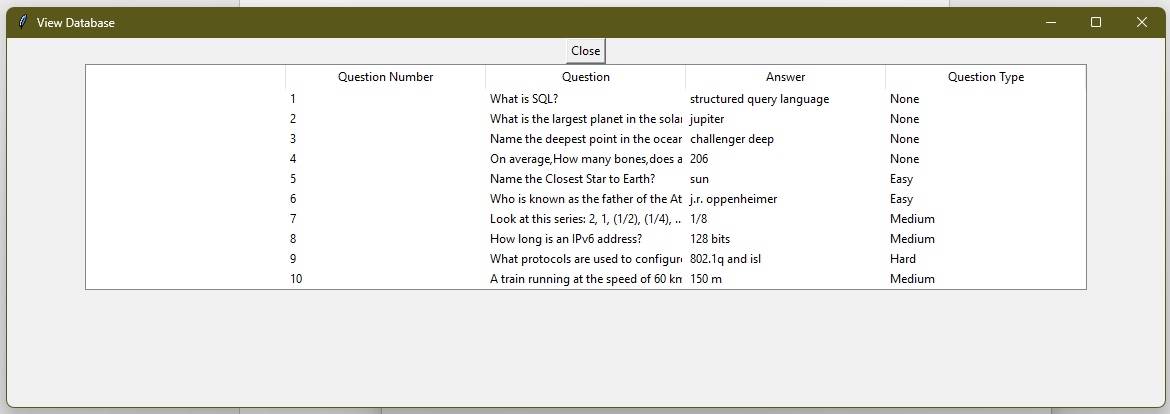
**Viewing The Question Database In-Game:**

A screenshot of a computer

Description automatically generated

Question Database option

*Database View:*



Adding Questions:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

*Help Window:*

