**Amity International School**

**Mayur Vihar**

****

**Computer Science Project**

**‘El Acertijo-The Riddler’**

**By-**

**Name :**

**Class: XII**

**Roll no. :**

**INDEX**

1. Certificate
2. Acknowledgment
3. Introduction
4. Modules used
5. Source Code
6. Project Output
7. User Manual
8. Bibliography

**CERTIFICATE**

This is to certify that \_\_\_\_\_\_\_\_\_\_ of class XII-B, Amity International School, \_\_\_\_, roll number- \_\_\_\_ has successfully completed Sproject in computer practical for the AISSCE as prescribed by CBSE in the academic year 2021-22.

**ACKNOWLEDGMENT**

We, Tanmay Goyal, Shivansh Adlakha, and Aarya Mehrotra, would like to sincerely thank Amity International School, Mayur Vihar, for giving us the opportunity and resources to complete our Computer Science project, El Acertijo.

We extend our heartfelt gratitude to our teacher \_\_\_\_\_\_\_\_, for their guidance and support, which helped us learn the practical applications of Python and MySQL.

This project has been a valuable learning experience, and we are grateful to everyone who supported us throughout.

Tanmay Goyal, Shivansh Adlakha, and Aarya Mehrotra

Class 12

Amity International School, Mayur Vihar

**INTRODUCTION**

In the modern educational environment, technology plays a significant role in simplifying teaching and learning processes. This project is a Python-based application built using MySQL for database management and Tkinter for a graphical user interface (GUI). It provides a platform for students and teachers to interact through quizzes where student answers are restricted to one word, ensuring clarity and consistency in response collection.

The system is designed to handle quiz scenarios where one-word answers are sufficient, such as terminologies, definitions, or quick facts. This format allows for streamlined evaluation while encouraging students to focus on concise responses.

**Core Features**

1. **Teacher Module**:
   * Enables teachers to create, modify, and delete quiz questions.
   * Allows management of student records and monitoring of quiz performance.
2. **Student Module**:
   * Provides secure login access to quizzes assigned by teachers.
   * Ensures answers are submitted in a one-word format, promoting uniformity.
3. **Database Integration**:
   * MySQL is used to securely store and manage quiz data, student information, and performance metrics.
   * Ensures fast and reliable data processing for seamless user experience.
4. **User-Friendly Interface**:
   * Tkinter is utilized to design an intuitive GUI, making the system easy to use for both teachers and students.
   * Provides a clean and organized layout for efficient navigation.

By integrating Python’s programming capabilities with MySQL’s robust database functionality and Tkinter’s interactive interface, it creates an efficient system for enhancing student-teacher engagement through one-word answer quizzes.

**MODULES AND FUNCTIONS**

**MODULES**

1. tkinter

* Purpose: Provides a graphical user interface (GUI) for the application.
* Key Uses:
  + Creating windows (e.g., login screen, admin page, quiz page).
  + Adding widgets like buttons, labels, entry fields, etc.
  + Displaying message boxes for warnings, errors, or success notifications using messagebox.

2. mysql.connector

* Purpose: Handles communication between the application and the MySQL database.
* Key Uses:
  + Establishing a connection to the MySQL database (connect function).
  + Creating and using a database (execute function).
  + Inserting, updating, deleting, and retrieving data for storing quiz questions, answers, scores, and other details.

3. os

* Purpose: Allows interaction with the operating system.
* Key Uses:
  + Exiting the application (os.\_exit(0)) after the quiz is completed.

**USER DEFINED FUNCTIONS**

Here is a detailed description of each user-defined function in the application, including its purpose and how it works:

1. check()

* Purpose: Handles the login validation for both admin and user. It checks if the entered username and password match the predefined credentials.
* Functionality:
  + Gets the username and password entered by the user.
  + If the username and password match the admin credentials ("admin" / "pass"), it opens the admin page.
  + If the username and password match the user credentials ("user" / "play"), it opens the quiz page.
  + If neither set of credentials matches, it displays an "Access Denied" error message.

Admin Page Functions:

2. exit\_func()

* Purpose: Exits the admin panel after confirming the user’s intent.
* Functionality:
  + Prompts a confirmation dialog (askyesno) asking if the user is sure they want to exit.
  + If confirmed, it closes the admin page using admin\_page.destroy().

3. clear\_func()

* Purpose: Clears the data from all input fields on the admin page.
* Functionality:
  + Clears the text in the "question", "answer", and "name" entry fields, allowing the user to reset them for new inputs.

4. add\_in\_data()

* Purpose: Adds a new question and its answer to the database.
* Functionality:
  + Retrieves the question and answer entered in the input fields.
  + Checks if either of the fields is empty and shows a warning if they are.
  + If both fields are filled, it establishes a connection to the MySQL database and inserts the question and answer into the datas table.
  + Displays a success message upon successful insertion or an error message if something goes wrong.

5. show\_func()

* Purpose: Displays all the questions and answers stored in the database.
* Functionality:
  + Establishes a connection to the database and fetches all entries from the datas table.
  + If no data is found, it shows an error message. Otherwise, it prints the data to the console and displays a success message.

6. del\_func()

* Purpose: Deletes a specific question from the database.
* Functionality:
  + Retrieves the question entered in the input field.
  + If the question field is empty, it shows a warning message.
  + It checks if the question exists in the database and, if found, deletes it.
  + If deletion is successful, it clears the input fields and shows a success message.

7. update\_func\_quest()

* Purpose: Updates the question in the database.
* Functionality:
  + Retrieves the new question and its answer from the input fields.
  + If the answer field is empty, it prompts the user to fill it.
  + Updates the question in the database based on the old answer.
  + Displays a success or error message based on the outcome.

8. update\_func\_ans()

* Purpose: Updates the answer to a specific question in the database.
* Functionality:
  + Retrieves the question and the new answer from the input fields.
  + If the question field is empty, it shows a warning message.
  + Updates the answer in the database based on the question.
  + Displays a success or error message based on the outcome.

9. show\_attempt()

* Purpose: Displays all the student scores stored in the database.
* Functionality:
  + Retrieves all the entries from the scores table in the database.
  + If there are no entries, it shows an error message. Otherwise, it prints the score data to the console and shows a success message.

10. search\_student()

* Purpose: Searches for a student’s score using the student’s name.
* Functionality:
  + Retrieves the student's name entered in the search field.
  + If the name is empty, it prompts the user to fill it.
  + It fetches the score for that specific student from the scores table in the database and displays the result, or shows a warning if no such student is found.

User Page Functions:

11. go\_in\_quiz()

* Purpose: Starts the quiz by fetching questions from the database and displaying the first question.
* Functionality:
  + Destroys the current user page and loads a new quiz page.
  + Establishes a connection to the MySQL database and retrieves all the questions and answers stored in the datas table.
  + Displays the first question and allows the user to enter an answer.
  + Handles the flow of the quiz by submitting answers, checking correctness, and moving to the next question.

12. submit\_answer()

* Purpose: Submits the user's answer for the current question and checks if it is correct.
* Functionality:
  + Retrieves the user’s answer from the input field.
  + If the user's answer matches the correct answer from the database, it displays a "CORRECT ANSWER" message and adds "True" to the score tracker (score\_calculator).
  + If the user's answer is incorrect, it displays a "WRONG ANSWER" message along with the correct answer.
  + Moves to the next question or finishes the quiz if it was the last question.
  + Once the quiz is complete, it calculates and displays the total score.
  + Sends the score and the user’s name to the admin page by inserting it into the scores table in the database.

13. submit()

* Purpose: Button function that allows users to submit their answers during the quiz.
* Functionality:
  + Calls the submit\_answer() function to check the answer, move to the next question, and eventually calculate the final score.

These user-defined functions are key to managing the flow of the application, whether it's for handling user login, admin tasks (like adding or deleting questions), or managing the quiz-taking process and storing results. Each function is responsible for a specific task, ensuring the overall smooth operation of the system.

**SOURCE CODE**

from tkinter import \*

from tkinter import messagebox

from PIL import ImageTk,Image

import mysql.connector

import os

score\_calculator = []

i = 0

solution = ""

name\_entry=""

# THE LOGIN SCREEN

home = Tk()

home.title("Login Page")

home.configure(background="darkgreen")

connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#")

cursor = connection.cursor()

cursor.execute("create database if not exists qa")

cursor.execute("use qa")

cursor.execute("create table if not exists datas(Name varchar(60),Answer varchar(60))")

cursor.execute("create table if not exists scores(Score int,Name varchar(30))")

connection.close()

play\_room1 = 0

# THE CHECK FUNCTION

def check():

    global name\_entry

    # GETTING DATA FROM THE ENTRIES

    username = user\_name.get()

    username.lower()

    userpass = user\_pass.get()

    userpass.lower()

    # FOR ADMIN

    if username == "admin" and userpass == "pass":

        home.destroy()  # DESTROYING THE LOGIN PAGE

        admin\_page = Tk()  # CREATING THE ADMIN PAGE

        admin\_page.title("Admin Page")  # SETTING THE TITLE OF THE PAGE

        admin\_page.configure(background="darkgreen")  # SETTING THE BACKGROUND COLOUR

        # SETTING THE LABEL OF THE PAGE(TITLE)

        # THE LABEL FOR THE QUESTIONS ENTRY

        Label\_1 = Label(admin\_page, text="Write Question here", font=('arial', 10), bd=9)

        Label\_1.grid(row=0, column=0)

        # THE ENTRY IN WHICH THE QUESTION WILL BE TAKEN

        question = Entry(admin\_page, width=15)

        question.grid(row=0, column=1)

        # THE LABEL FOR THE ANSWER ENTRY

        answer = Label(admin\_page, text="Enter the Answer:", font=('arial', 10), bd=9)

        answer.grid(row=1, column=0)

        name\_search = Label(admin\_page, text="Name", font=('arial', 10), bd=9)

        name\_search.grid(row=0, column=3)

        name\_en = Entry(admin\_page, width=15)

        name\_en.grid(row=0, column=4)

        # THE ENTRY FOR THE ANSWER

        answer\_entry = Entry(admin\_page, width=15)

        answer\_entry.grid(row=1, column=1)

        # THE FUNCTION FOR CLOSING THE WINDOW

        def exit\_func():

            try:

                confirm = messagebox.askyesno("Confirm", "Are you sure you want to exit")

                if confirm > 0:

                    admin\_page.destroy()

            except Exception as e:

                messagebox.showerror("Error", "Something Went wrong\nCheck the entries again or try again")

        # THE FUNCTION FOR CLEARING THE DATA IN THE ENTRIES

        def clear\_func():

            try:

                question.delete(0, END)

                answer\_entry.delete(0, END)

                name\_en.delete(0, END)

            except Exception as e:

                messagebox.showerror("Error", "Something Went wrong\nCheck the entries again or try again")

        # THE FUNCTION FOR ADDING TO THE DATABASE

        def add\_in\_data():

            try:

                # GETTING THE DATA FROM ENTRIES

                qdata = question.get()

                adata = answer\_entry.get()

                # ADDING THE IF ELSE FOR IF THE DATA FEILDS ARE EMPTY

                if qdata == "" or adata == "":

                    messagebox.showwarning("Incomplete Feilds",

                                           "Please make sure that you have filled both the entries")

                else:

                    connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#",

                                                         db="qa")

                    cursor = connection.cursor()

                    cursor.execute("insert into datas(name, answer)values(%s, %s)", (qdata, adata))

                    connection.commit()

                    question.delete(0, END)

                    answer\_entry.delete(0, END)

                    messagebox.\_show("Insertion Complete", "Your data has been succesfully added to the database")

                    connection.close()

                    connection.close()

            except Exception as e:

                messagebox.showerror("Error", "Something Went wrong\nCheck the entries again or try again")

        # THE SHOW DATA FUNCTION

        def show\_func():

            try:

                connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#", db="qa")

                cursor = connection.cursor()

                cursor.execute("select \* from datas")

                data = cursor.fetchall()

                cursor.close()

                if data == []:

                    messagebox.showerror("No Data", "There is no data yet")

                else:

                    messagebox.\_show("Questions",data)

                connection.close()

            except Exception as e:

                messagebox.showerror("Error", "Something Went wrong\nCheck the entries again or try again")

        # THE DELETE ENTRY FUNCTION

        def del\_func():

            qdata = str(question.get())

            connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#", db="qa")

            cursor = connection.cursor()

            cursor.execute("SELECT \* FROM datas where name=%s", (qdata,))

            da = cursor.fetchall()

            cursor.close()

            if qdata == "":

                messagebox.showwarning("Delete Status", "The question is important for deletion")

            elif da == []:

                messagebox.showerror("Invalid Entry", "No such question is there ")

            else:

                try:

                    connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#",

                                                         db="qa")

                    cursor = connection.cursor()

                    cursor.execute("delete from datas where name='" + qdata + "'")

                    connection.commit()

                    question.delete(0, END)

                    answer\_entry.delete(0, END)

                    messagebox.\_show("Deletion Succesful", "The Question has been succesfully removed")

                    connection.close()

                except Exception as e:

                    messagebox.showerror("Error", "Something Went wrong\nCheck the entries again or try again\n")

        # THE UPDATE QUESTION FUNCTION

        def update\_func\_quest():

            qdata = question.get()

            adata = answer\_entry.get()

            if adata == "":

                messagebox.showwarning("Unfilled Entries", "For updating the questions the answer needs to be filled")

            else:

                try:

                    connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#",

                                                         db="qa")

                    cursor = connection.cursor()

                    cursor.execute("UPDATE datas SET name=%s where answer=%s;", (qdata, adata))

                    connection.commit()

                    question.delete(0, END)

                    answer\_entry.delete(0, END)

                    messagebox.\_show("Update Status", "The question is succesfully updated")

                except Exception as e:

                    messagebox.showerror("Error", "Something Went wrong\nCheck the entries again or try again")

        def update\_func\_ans():

            qdata = question.get()

            adata = answer\_entry.get()

            try:

                if qdata == "":

                    messagebox.showwarning("Unfilled Entries", "For updating the answer the question is neccasary")

                else:

                    connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#",

                                                         db="qa")

                    cursor = connection.cursor()

                    cursor.execute("UPDATE datas SET answer=%s where name=%s;", (adata, qdata))

                    connection.commit()

                    question.delete(0, END)

                    answer\_entry.delete(0, END)

                    messagebox.\_show("Update Status", "The answer is succesfully updated")

            except Exception as e:

                messagebox.showerror("Error", "Something Went wrong\nCheck the entries again or try again" + e)

        def show\_attempt():

            connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#", db="qa")

            cursor = connection.cursor()

            cursor.execute("SELECT \* FROM scores")

            score\_data = cursor.fetchall()

            cursor.close()

            if score\_data == []:

                messagebox.showerror("No Data", "There is no data yet")

            else:

                messagebox.\_show("Response data",score\_data)

            connection.close()

        def search\_student():

            name = name\_en.get()

            name = str(name)

            if name == "":

                messagebox.showerror("Data unfilled", "Name needs to be fiiled to search")

            else:

                connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#", db="qa")

                cursor = connection.cursor()

                cursor.execute("SELECT \* FROM scores WHERE name=%s", (name,))

                score\_data = cursor.fetchall()

                if score\_data == []:

                    messagebox.showwarning("No Data", "There is no such name")

                else:

                    messagebox.\_show("Student data",score\_data)

                connection.close()

        # THE BUTTONS

        # THE EXIT BUTTON

        search\_student = Button(admin\_page, text="Search by name", font=('arial', 18), width=20, bg="orange",

                                command=search\_student)

        search\_student.grid(row=1, column=4)

        exit\_bt = Button(admin\_page, text="Exit", font=('arial', 18), width=10, bg="orange", command=exit\_func)

        exit\_bt.grid(row=4, column=0)

        # THE CLEAR BUTTON

        clear\_bt = Button(admin\_page, text="Clear", font=('arial', 18), width=10, bg="orange", command=clear\_func)

        clear\_bt.grid(row=6, column=0)

        # THE ADD BUTTON

        add\_bt = Button(admin\_page, text="Insert question", font=('arial', 18), width=20, bg="orange", command=add\_in\_data)

        add\_bt.grid(row=4, column=1)

        # THE SHOW BUTTON

        show\_bt = Button(admin\_page, text="Show all questions", font=('arial', 18), width=20, height=1, bg="orange", command=show\_func)

        show\_bt.grid(row=6, column=1)

        # THE DELETE BUTTON

        del\_bt = Button(admin\_page, text="Delete question", font=('arial', 18), width=20, bg="orange", command=del\_func)

        del\_bt.grid(row=4, column=2)

        # THE UPDATE QUESTION BUTTON

        upd\_q = Button(admin\_page, text="Update Ques", font=('arial', 18), width=20, bg="orange",

                       command=update\_func\_quest)

        upd\_q.grid(row=6, column=2)

        # THE UPDATE ANSWER BUTTON

        upd\_a = Button(admin\_page, text="Update Ans", font=('arial', 18), width=10, bg="orange",

                       command=update\_func\_ans)

        upd\_a.grid(row=4, column=3)

        # THE SCORE ADDERS

        score\_button = Button(admin\_page, text="Show Marks", font=('arial', 18), width=10, bg="orange",

                              command=show\_attempt)

        score\_button.grid(row=6, column=3)

        # THE SEARCH STUDENT

        # THE ADMIN PAGE MAINLOOP

        admin\_page.mainloop()

    elif username == "user" and userpass == "play":

        home.destroy()

        user\_page = Tk()

        user\_page.title("Play Page")

        user\_page.configure(background="darkgreen")

        def go\_in\_quiz():

            global i

            global ans1

            global ans

            global play\_room1

            global name\_entry

            name = name\_entry.get()

            name = str(name)

            user\_page.destroy()

            user\_page.mainloop()

            connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#", db="qa")

            cursor = connection.cursor()

            cursor.execute("SELECT \* FROM datas")

            data = cursor.fetchall()

            cursor.close()

            connection.close()

            number = len(data)

            data\_num = number + 1

            play\_room = Tk()

            play\_room.title("Quiz Page")

            play\_room.configure(background="darkgreen")

            questi = data[i][0]

            user\_data = str(questi)

            ans = data[i][1]

            qlabel = Label(play\_room, text=f"Question: {user\_data}", font=('arial', 10), bd=18)

            qlabel.grid(row=0, column=0)

            ans1 = Entry(play\_room, width=15)

            ans1.grid(row=0, column=1)

            def submit\_answer():

                global score

                global userans

                global ans1

                global ans

                global i

                global play\_room1

                try:

                    userans = ans1.get()

                    try:

                        play\_room.destroy()

                    except:

                        pass

                    try:

                        play\_room1.destroy()

                    except:

                        pass

                    if i != len(data) -1:

                        if userans == str(ans):

                            messagebox.\_show(title = "CORRECT ANSWER", message=f'CORRECT ANSWER')

                            score\_calculator.append("True")

                            play\_room1 = Tk()

                            play\_room1.title("Quiz Page")

                            play\_room1.configure(background="yellow")

                            i += 1

                            questi = data[i][0]

                            user\_data = str(questi)

                            ans = data[i][1]

                            qlabel = Label(play\_room1, text=f"Question: {user\_data}", font=('arial', 10), bd=18)

                            qlabel.grid(row=0, column=0)

                            ans1 = Entry(play\_room1, width=15)

                            ans1.grid(row=0, column=1)

                            submit = Button(play\_room1, text="Submit", font=('arial', 18), width=10, bg="orange",

                                            command=submit\_answer)

                            submit.grid(row=3, column=0)

                            play\_room1.mainloop()

                        else:

                            messagebox.\_show(title = "WRONG ANSWER", message=f'{str(ans)} was the correct answer')

                            play\_room1 = Tk()

                            play\_room1.title("Quiz Page")

                            play\_room1.configure(background="darkgreen")

                            i += 1

                            questi = data[i][0]

                            user\_data = str(questi)

                            ans = data[i][1]

                            qlabel = Label(play\_room1, text=f"Question: {user\_data}", font=('arial', 10), bd=18)

                            qlabel.grid(row=0, column=0)

                            ans1 = Entry(play\_room1, width=15)

                            ans1.grid(row=0, column=1)

                            submit = Button(play\_room1, text="Submit", font=('arial', 18), width=10, bg="orange",

                                            command=submit\_answer)

                            submit.grid(row=3, column=0)

                            play\_room1.mainloop()

                    else:

                        if userans == str(ans):

                            messagebox.\_show(title = "CORRECT ANSWER", message=f'CORRECT ANSWER')

                            score\_calculator.append("True")

                        else:

                            messagebox.\_show(title = "WRONG ANSWER", message=f'{str(ans)} was the correct answer')

                    play\_room.mainloop()

                except IndexError:

                    pass

                score = str(len(score\_calculator))

                messagebox.\_show("Your Score was", score)

                connection = mysql.connector.connect(host="localhost", user="root", password="@samysql1357#", db="qa")

                cursor = connection.cursor()

                cursor.execute("insert into scores(score, name)values(%s, %s)", (score, name))

                connection.commit()

                connection.close()

                messagebox.\_show("Succesful", "Your Data has been sent to the teacher")

                os.\_exit(0)

            submit = Button(play\_room, text="Submit", font=('arial', 18), width=10, bg="orange",

                            command=submit\_answer)

            submit.grid(row=3, column=0)

        play = Button(user\_page, text="Enter Quiz", font=('arial', 18), width=10, bg="orange", command=go\_in\_quiz)

        play.grid(row=5, column=5)

        name\_label = Label(user\_page, text="Enter your name:", font=('arial', 10), bd=18)

        name\_label.grid(row=2, column=0)

        name\_entry = Entry(user\_page, width=10)

        name\_entry.grid(row=2, column=1)

    else:

        messagebox.showerror("Access Denied", "Invalid Passoword or Username")

# THE MAINFRAME OF THE LOGIN PAGE

MainFrame = Frame(home, width=600, height=100, bd=1, relief=SOLID)

MainFrame.pack(side=TOP, pady=20)

# THE TITLE LABLE AND THE MIDDLE FRAME

w\_to = Label(MainFrame).pack(side="left")

lbl = Label(MainFrame, text="Login Screen", font=('arial', 38), width=600)

lbl.pack()

MidFrame = Frame(MainFrame, width=600)

MidFrame.pack(side=TOP, pady=50)

# THE USERNAME AND PASSWORD VARIABLES

USER\_NAME = StringVar()

USER\_PASS = StringVar()

# THE FRAME ON WHICH LABEL AND TEXT

lbl\_Name = Label(MidFrame, text="Name:", font=('arial', 25), bd=18)

lbl\_Name.grid(row=0)

lbl\_Pass = Label(MidFrame, text="Password:", font=('arial', 25), bd=18)

lbl\_Pass.grid(row=1)

# THE ENTRIES OF NAME AND PASSWORD

user\_name = Entry(MidFrame, textvariable=USER\_NAME, font=('arial', 25), width=15)

user\_name.grid(row=0, column=1)

user\_pass = Entry(MidFrame, textvariable=USER\_PASS, font=('arial', 25), width=15, show="\*")

user\_pass.grid(row=1, column=1)

# THE BOTTOM FRAME

BottomFrame = Frame(MainFrame, width=600)

BottomFrame.pack(side=TOP, pady=20)

# THE BUTTON WHICH WILL CHECK THE USERNAME AND PASSWORD

checker = Button(BottomFrame, text="Enter", font=("arial", 18), command=check, bg="orange")

checker.grid(row=2, column=3, pady=20)

#IMAGE

image=Image.open("riddler.jpeg")

resize\_image=image.resize((300,300))

img=ImageTk.PhotoImage(resize\_image)

image\_label=Label(image=img)

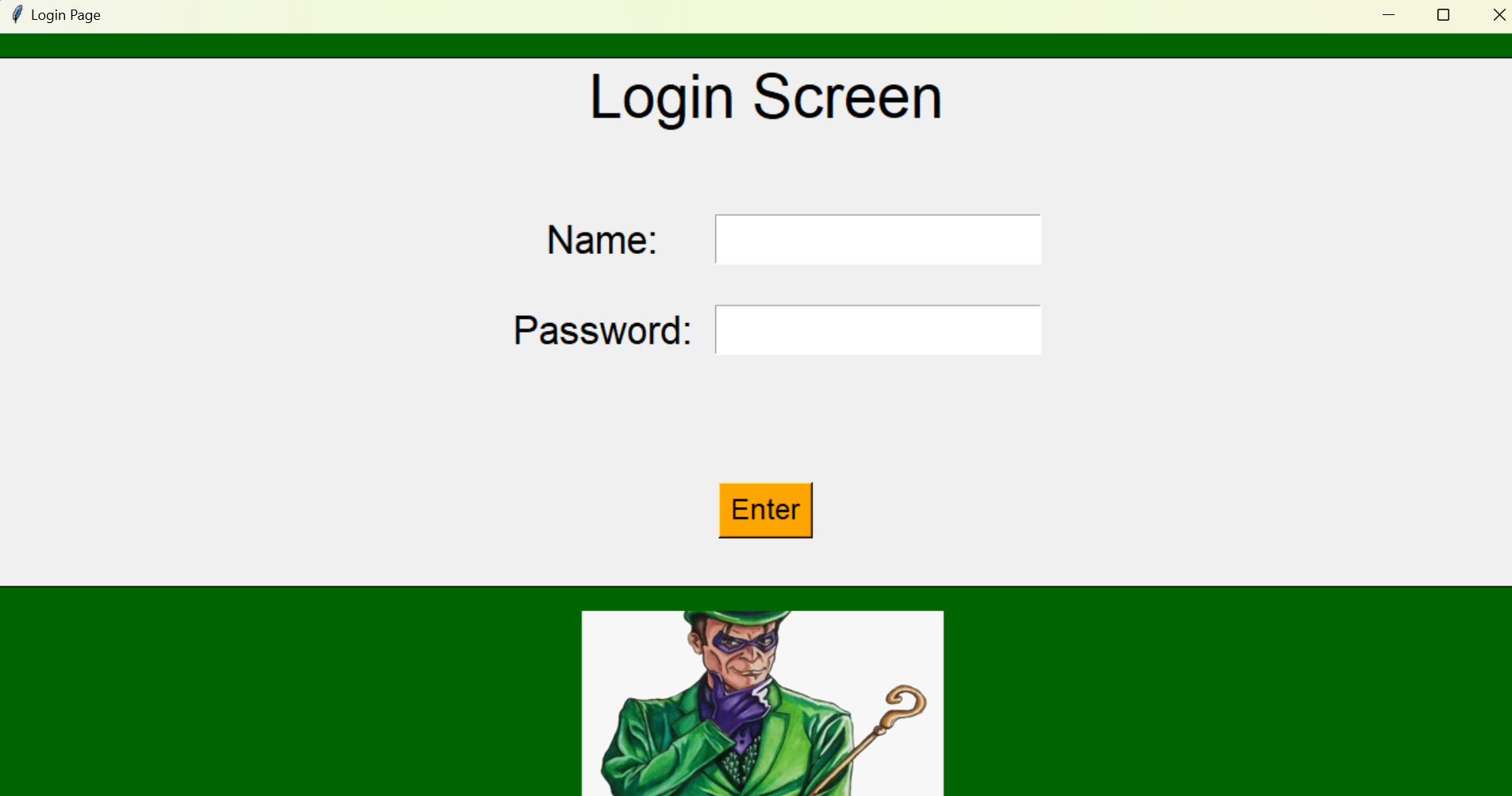
image\_label.image=img

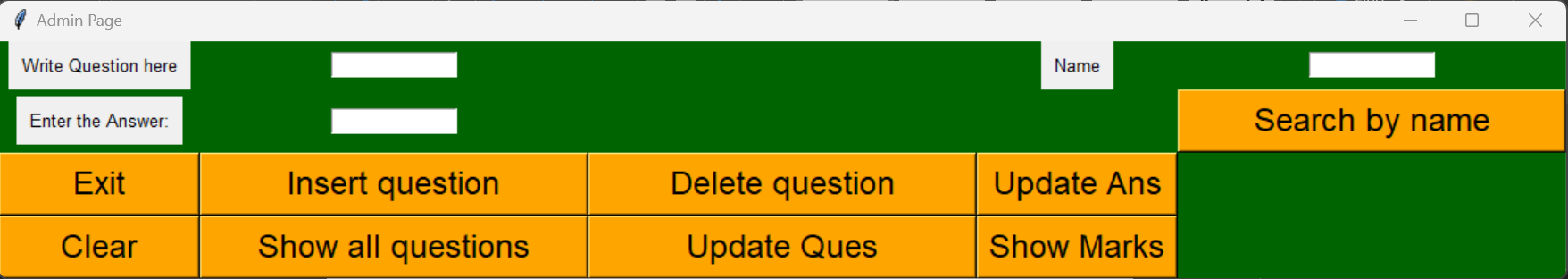
image\_label.pack()

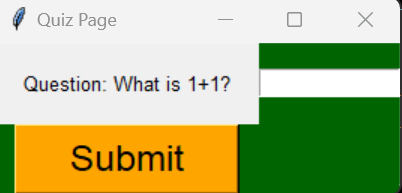
# THE LOGIN PAGE MAINLOOP

home.mainloop()

**PROJECT OUTPUT**

****

****

****

**USER MANUAL**

**Overview**

This application is a quiz platform with two user roles:

* **Admin:** Can manage quiz content (add, update, delete questions) and view user scores.
* **User:** Can log in, attempt quizzes, and have their scores automatically sent to the admin.

The application uses **MySQL** for database management and includes **GUI** elements for user interaction, built with **Tkinter**.

**System Requirements**

1. **Python 3.x** with the following libraries installed:
   * mysql-connector
   * tkinter
2. **MySQL** with credentials matching the application's configuration.

**Getting Started**

**1. Launch the Application**

Run the script quiz\_app.py in your Python environment. The login screen will appear.

**2. Login Credentials**

* **Admin Login**
  + Username: admin
  + Password: pass
* **User Login**
  + Username: user
  + Password: play

**Admin Role**

After logging in as **Admin**, you'll see the **Admin Panel**, which has the following functionalities:

**Admin Features and Buttons**

|  |  |
| --- | --- |
| Feature | Details |
| Add a Question | Enter a question in the "Write Question Here" field and its corresponding answer in the "Enter the Answer" field. Click **Insert** to save it. |
| View All Questions | Click **Show** to display all the questions stored in the database. Results are printed in the terminal. |
| Update Question | Provide the question text and its updated text or answer in the respective fields. Click **Update Ques** or **Update Ans** to update. |
| Delete Question | Enter the question text in the "Write Question Here" field and click **Delete** to remove it from the database. |
| Search User | Enter the user’s name in the "Name" field and click **Search** to fetch their quiz scores from the database. Results are shown in the terminal. |
| View All Scores | Click **Show Marks** to display all users' scores in the terminal. |
| Reset Fields | Clears all input fields in the panel. |
| Exit | Closes the Admin Panel. |

**User Role**

After logging in as **User**, you'll be taken to the **User Panel**, where you can enter your name and begin the quiz.

**User Features**

|  |  |
| --- | --- |
| Feature | Details |
| Enter Your Name | Input your name in the "Enter Your Name" field before starting the quiz. |
| Start Quiz | Click **Enter Quiz** to begin answering questions. |
| Answer Questions | For each question displayed, type your answer in the text box and click **Submit** to proceed. |
| Receive Feedback | Each answer submission is followed by feedback indicating whether the answer was correct or not. |
| View Score | After completing the quiz, your total score is displayed. |

**Quiz Workflow**

1. Questions are presented one at a time.
2. Enter your answer in the text field and click **Submit**.
3. Feedback is provided for each answer.
4. At the end of the quiz, your score is displayed, and it is automatically sent to the admin.

**Database Design**

1. **Tables**
   * datas: Stores questions and answers.
   * scores: Stores user scores and names.
2. **Schema**
   * datas:
     + Name (VARCHAR) - Question text
     + Answer (VARCHAR) - Correct answer
   * scores:
     + Score (INT) - User score
     + Name (VARCHAR) - User name

**Error Handling**

* The application includes exception handling for database errors and incomplete user input.
* Error messages guide users to correct their actions.

**BIBLIOGRAPHY**

<https://www.youtube.com/watch?v=VMP1oQOxfM0>

<https://www.youtube.com/watch?v=yQSEXcf6s2I&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV>

<https://ncert.nic.in/textbook.php>

<https://stackoverflow.com/>