VYDEHI SCHOOL OF EXCELLENCE

ACADEMIC SESSION: 2024-25

COMPUTER SCIENCE PROJECT

TOPIC: QUIZ COMPETITION

SUBMITTED BY:

Name: Navaneeth Krishna V

Class: 12 'A'

SUBMITTED TO:

Ms. Ranjeeta Shrivastava

Vydehi School of Excellence

SIGNATURE OF EXTERNAL EXAMINER **SIGNATURE OF INTERNAL EXAMINER**

EXTERNAL EXAMINER NO.: _____

VYDEHI SCHOOL OF EXCELLENCE

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that **Navaneeth Krishna V**, of class **XII** has successfully completed the project under the guidance of **Ms. Ranjeeta Shrivastava** during the academic year **2024-25** in partial fulfilment of Computer Science Practical Examination.

SIGNATURE OF EXTERNAL EXAMINER **SIGNATURE OF INTERNAL EXAMINER**

EXTERNAL EXAMINER NO.: _____ **SIGNATURE OF PRINCIPAL**

ACKNOWLEGMENT

I would like to express my special thanks of gratitude to my teacher Ms. Ranjeeta Shrivastava as well as our Principal of Senior School, Ms. Chavvi Garg, who gave me the golden opportunity to do this wonderful project on Quiz Competition, which also helped in doing lot of research and in gaining a deeper knowledge in this subject.

I am really thankful to them.

Secondly, I would like to thank my parents and friends who helped me a lot in finishing this project within the limited time.

THANKS AGAIN TO ALL WHO HELPED ME.

Name: **Navaneeth Krishna V**

Class: **12 'A'**

INDEX

INTRODUCTION

This program is an interactive graphical quiz management system built using Python, MySQL, and the Tkinter library. The system supports both administrator and player roles, allowing the administrator to manage quiz categories, add questions, and store user scores, while players can log in, attempt quizzes, and view leaderboards.

The program's core functionality revolves around two primary classes:

- **Quiz**: Manages database connections, handles category and question additions, maintains scores, and retrieves leaderboard data.
- **QuizGUI**: Provides the graphical interface through Tkinter, guiding users through login, question-answering, and leaderboard display.

Players earn points by answering questions, and their scores are recorded and ranked on a leaderboard for each quiz category.

TECHNOLOGY USED

Programming Language – Python

- ***Development***: Python programming language was initially designed by Guido Van Rossum in February 1991 and developed by Python Square Foundation.

- ***Easy to Use***: Python is compact and easy to use object-oriented language with very simple syntax rules.

- ***Expressive Language***: Python is an expressive language – fewer lines of code and simpler syntax as compared to other popular programming languages like C++, Java, etc.

- ***Versatile***: Python is versatile. It can be used for many different tasks, from web development to machine learning.

WHAT CAN PYTHON DO?

1. **Web Development:** Python is often used to develop the back end of a website or application – the parts that a user doesn't see.

2. **Software testing and prototyping:** In software development, Python can aid in tasks like bug tracking and testing. With Python, software developers can automate testing for new products or features.

3. **Everyday tasks:** Python can be used for everyday tasks such as keeping track of stock market, sending yourself a text reminder to carry an umbrella anytime it's raining, etc.

4. **Data Analysis and Machine learning:** Python helps data analysts and other professionals to conduct complex statistical calculations, create data visualizations, build machine learning algorithms, manipulate and analyse data, and complete other data-related tasks.

WHY PYTHON?

- **Cross-platform Language**: Python can run equally well on variety of platforms – Windows, Linus/UNIX, smartphones, etc.

- **Simple syntax:** Python has a simple syntax similar to the English language.

- **Expressive language:** Python is an expressive language with fewer lines of code and simpler syntax as compared to other popular programming languages like C++, Java, etc.

- **Quick prototyping:** Python runs on an interpreter system. This means that the code can be executed as soon as it is written. This means that prototyping can be very quick.

- **Procedural way treatment:** Python can be treated in a procedural way, an object-oriented way or a functional way.

HARDWARE USED

Operating System	Windows 11
Processor	Intel Core I5 @ 2.16Ghz
RAM	4GB
Hard disk	SSD 500GB

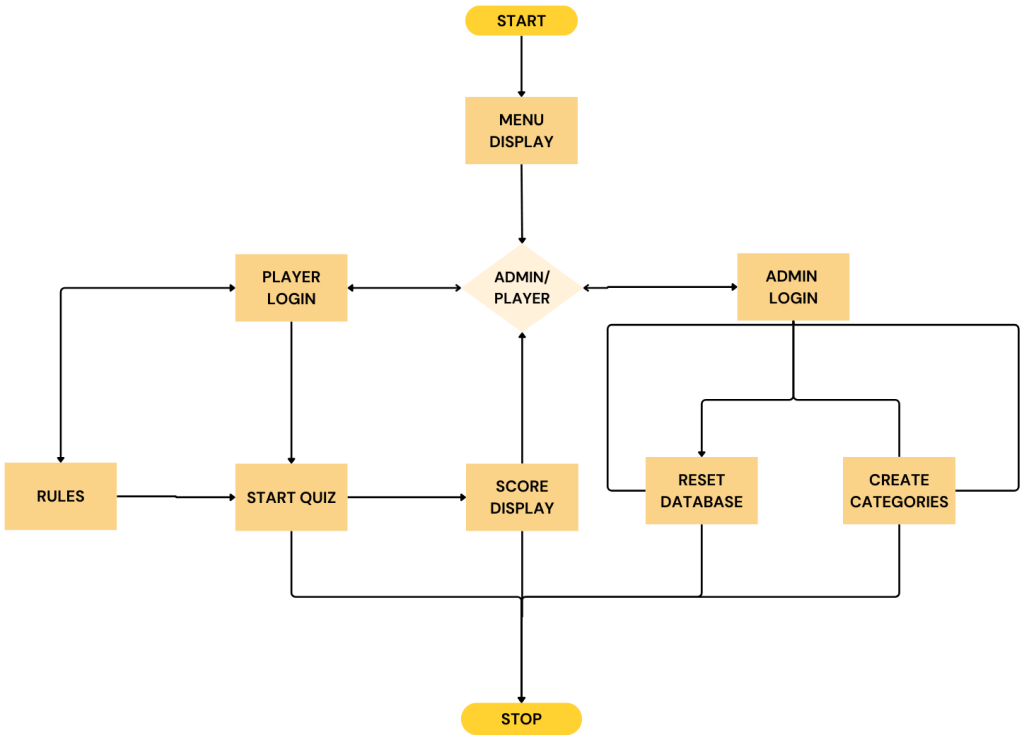
SOFTWARE USED

1. Windows and Vscode
2. Python and MYSQL

AIM

To create an interactive quiz game through which users can play a fun fill in the blanks quiz of topics/categories they prefer. The interactive elements include: A separate password-protected administrator system, A leaderboard system for players, and a GUI.

FLOWCHART


START
MENU DISPLAY
PLAYER LOGIN
ADMIN/ PLAYER
ADMIN LOGIN
RULES
START QUIZ
SCORE DISPLAY
RESET DATABASE
CREATE CATEGORIES
STOP

PROGRAM CODE

```
import mysql.connector as s
import tkinter as tk
from tkinter import messagebox, simpledialog
from tkinter import ttk
Label_color_fg = '#F4EBFB'
Label_color_bg = '#978bc4'
Button_color_fg = '#F4EBFB'
Button_color_bg = '#655b89'
Button_color_fg_selected = '#655b89'
Button_color_bg_selected = '#F4EBFB'
Entry_color_fg = '#F4EBFB'
Entry_color_bg = '#7465b1'
Background = '#978bc4'
table_color_text='#000000'
table_color_bg = '#7465b1'
table_color_inner_bg ='000000'
database_name = 'quiz'
mainnames=[]

def get_db_connection():
return s.connect(host="localhost", user='root',
passwd="VSE@2022", database=database_name)

def create_sql_db():
conn = get_db_connection()
cursor = conn.cursor()
cursor.execute(f"SHOW DATABASES LIKE '{database_name}'")
result = cursor.fetchone()
if result:
print(f"Database '{database_name}' already exists.")
else:
cursor.execute(f"CREATE DATABASE {database_name}")
print(f"Database '{database_name}' created successfully.")
cursor.close()
conn.close()

def reset_database():
conn = get_db_connection()
cursor=conn.cursor()
cursor.execute("SELECT table_name FROM
information_schema.tables WHERE table_schema = 'quiz';")
tables = cursor.fetchall()
for (table_name,) in tables:
cursor.execute(f"DROP TABLE IF EXISTS `{table_name}`;")
```

```
print(f"Dropped table: {table_name}")
conn.commit()
cursor.close()

class Quiz:

def __init__(self):
self.admin_u = "admin07"
self.admin_p = "admin@123"
self.categories = []

def category(self):
cat = int(input("Enter the number of categories: "))
for i in range(cat):
n = input(f'Enter the name of the category {i + 1}: ')
q = []
a = []
qa = (n, q, a)
self.categories.append(qa)

def add_qa_sql(self):
conn = get_db_connection()
cursor = conn.cursor()
for i in self.categories:
q = i[1]
a = i[2]
data = [(q[j], a[j]) for j in range(len(q))]
query = f"INSERT INTO {i[0]} (q, a) VALUES (%s, %s)"
cursor.executemany(query, data)
conn.commit()
cursor.close()
conn.close()

def add_question(self, category_name, question, answer):
for i in self.categories:
if i[0] == category_name:
i[1].append(question)
i[2].append(answer)
break
else:
messagebox.showerror("Error", f"Category '{category_name}' not
found.")

def create_tables(self):
conn = get_db_connection()
cursor = conn.cursor()
for i in self.categories:
create_table_query = f"CREATE TABLE {i[0]} (q VARCHAR(1000), a
VARCHAR(1000))"
```

```
cursor.execute(f"SHOW TABLES LIKE '{i[0]}'")
result = cursor.fetchone()
if result:
print(f"Table '{i[0]}' already exists.")
else:
cursor.execute(create_table_query)
print(f"Table '{i[0]}' created successfully.")
create_score_table_query = f"CREATE TABLE IF NOT EXISTS
{i[0].replace(' ', '_')}_scores (username VARCHAR(100), score
INT)"
cursor.execute(create_score_table_query)
cursor.close()
conn.close()

def add_score(self, username, category, score):
conn = get_db_connection()
cursor = conn.cursor()
query = f"INSERT INTO {category.replace(' ', '_')}_scores
(username, score) VALUES (%s, %s)"
cursor.execute(query, (username, score))
conn.commit()
cursor.close()
conn.close()

def get_leaderboard(self, category):
conn = get_db_connection()
cursor = conn.cursor()
cursor.execute(f"SELECT username, score FROM
{category.replace(' ', '_')}_scores ORDER BY score DESC")
leaderboard = cursor.fetchall()
cursor.close()
conn.close()
return leaderboard

class QuizGUI:

def __init__(self, root, quiz):
self.root = root
self.quiz = quiz
self.current_question_index = 0
self.current_category = ''
self.questions = []
self.root.title("Quiz Admin")
self.root.geometry("1000x1000")
self.score = 0
self.username=''
self.mainnames=[]
self.create_login_mode_screen()
```

```
def create_login_mode_screen(self):
self.clear_screen()
tk.Label(self.root, text="WELCOME TO THE QUIZ COMPETITION !",
font=("courier new", 30, "bold"), bg=Label_color_bg,
fg=Label_color_fg).pack(pady=20)
tk.Label(self.root, text="Do you want to login as
admin/player:", font=("courier new", 32,'bold'),
bg=Label_color_bg, fg=Label_color_fg).pack(pady=20)
tk.Button(self.root, text="Admin",
command=self.create_login_screen_admin, bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
16,'bold'),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=5)
tk.Button(self.root, text="Player",
command=self.create_login_screen_user, bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
16,'bold'),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=10)

def create_login_screen_admin(self):
self.clear_screen()
tk.Label(self.root, text="Login", font=("courier new", 32),
bg=Label_color_bg, fg=Label_color_fg).pack(pady=20)
tk.Label(self.root, text="Username", bg=Label_color_bg,
font=("courier new", 16), fg=Label_color_fg).pack()
self.admin_user_entry =
tk.Entry(self.root,fg=Entry_color_fg,bg=Entry_color_bg,
font=("courier new", 16))
self.admin_user_entry.pack(pady=20)
tk.Label(self.root, text="Password", bg=Label_color_bg,
fg=Label_color_fg, font=("courier new", 16)).pack()
self.admin_pass_entry = tk.Entry(self.root,
show="*",fg=Entry_color_fg,bg=Entry_color_bg, font=("courier
new", 16))
self.admin_pass_entry.pack(pady=20)
tk.Button(self.root, text="Login", command=self.login,
bg=Button_color_bg, fg=Button_color_fg, font=("courier new",
16),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=20)

def create_login_screen_user(self):
conn = get_db_connection()
cursor = conn.cursor()
cursor.execute(f"SHOW tables LIKE 'general_1'")
r = cursor.fetchone()
print(r)
if not r:
cursor.execute('CREATE TABLE General_1 (q VARCHAR(255), a
VARCHAR(255))')
```

```
cursor.execute("INSERT INTO General_1 (q, a) VALUES ('Which
animal is known as the ''Ship of the Desert''?', 'Camel'),
('How many days are there in a week?', '7 days'), ('How many
hours are there in a day?', '24 hours'), ('How many letters
are there in the English alphabet?', '26 letters'), ('Rainbow
consist of how many colours?', '7 colours');")
create_score_table_query = f"CREATE TABLE IF NOT EXISTS
general_1_scores (username VARCHAR(100), score INT)"
cursor.execute(create_score_table_query)
conn.commit()
cursor.close()
conn.close()
self.clear_screen()
tk.Label(self.root, text="Login", font=("courier new", 32),
bg=Label_color_bg, fg=Label_color_fg).pack(pady=20)
tk.Label(self.root, text="Username", bg=Label_color_bg,
font=("courier new", 16), fg=Label_color_fg).pack(pady=10)
self.user_entry =
tk.Entry(self.root,fg=Entry_color_fg,bg=Entry_color_bg,
font=("courier new", 16))
self.user_entry.pack()
tk.Button(self.root, text="Login",
command=self.user_login,bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
16),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=20)

def clear_screen(self):
for widget in self.root.winfo_children():
widget.destroy()

def user_login(self):
self.username = self.user_entry.get()
con=get_db_connection()
cursor=con.cursor()
cursor.execute("SHOW TABLES LIKE '%scores'")
tablesname=cursor.fetchall()
for i in tablesname:
table=i[0]
query = f"SELECT * FROM `{table}`"
cursor.execute(query)
scoresname=cursor.fetchall()
for j in scoresname:
self.mainnames.append(j[0])
print(self.mainnames)
if self.username in self.mainnames:
messagebox.showerror("ERROR", "Username already taken, try
again!")
else:
```

```
messagebox.showinfo("Login Success", f"Welcome
{self.username}!")
self.create_main_screen_u()

def create_rules_screen(self):
self.clear_screen()
tk.Label(self.root, text="1: It is a fill in the blank type
quiz\n2: Each correct answer gets -> 2 POINTS\n3: Each Wrong
answers gives -> 0 POINTS\n4:A leaderboard displayed based on
category", font=("courier new", 20), bg=Label_color_bg,
fg=Label_color_fg).pack(pady=20)
tk.Button(self.root, text="BACK",
command=self.create_login_mode_screen, bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
20),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).place(relx=0.5,rely=0
.9,anchor='center')

def create_main_screen_u(self):
self.clear_screen()
tk.Label(self.root, text="WELCOME TO THE QUIZ COMPETITION !",
font=("courier new", 40, "bold"), bg=Label_color_bg,
fg=Label_color_fg).pack(pady=50)
tk.Button(self.root, text='START QUIZ', font=("courier new",
20, "bold"), command=self.start_quiz, bg=Button_color_bg,
fg=Button_color_fg,highlightbackground=
Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).place(relx=0.5,
rely=0.35, anchor='center')
tk.Button(self.root, text="  RULES  ",
command=self.create_rules_screen, bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
20,"bold"),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).place(relx=0.5,rely=0
.45,anchor='center')
tk.Button(self.root, text="   QUIT   ",
command=self.create_login_mode_screen, bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
20,"bold"),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).place(relx=0.5,rely=0
.55,anchor='center')


def start_quiz(self):
self.clear_screen()
conn = get_db_connection()
cursor = conn.cursor()
cursor.execute("SHOW TABLES WHERE `Tables_in_quiz` NOT LIKE
'%_scores'")
```

```
tables = cursor.fetchall()
a=0
for table in tables:
table_name = table[0]
button = tk.Button(self.root, text=table_name, command=lambda
category=table_name:
self.cat_display(category),bg=Button_color_bg,
fg=Button_color_fg, font=("courier new", 20,
"bold"),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected)
button.place(relx=0.5, rely=0.5, anchor='center', y=a)
a+=100
cursor.close()
conn.close()

def cat_display(self, category):
self.current_category = category
self.current_question_index = 0
self.score=0
conn = get_db_connection()
cursor = conn.cursor()
cursor.execute(f'SELECT * FROM {category}')
self.questions = cursor.fetchall()
cursor.close()
conn.close()
self.display_question()

def display_question(self):
if self.current_question_index < len(self.questions):
self.clear_screen()
question = self.questions[self.current_question_index]
q_text = question[0]
self.correct_answer = question[1]
tk.Label(self.root, text=q_text, font=("courier new", 20,
"bold"), bg=Label_color_bg, fg=Label_color_fg).pack(pady=20)
self.answer_entry =
tk.Entry(self.root,fg=Entry_color_fg,bg=Entry_color_bg,
font=("courier new", 20))
self.answer_entry.pack(pady=10)
tk.Button(self.root, text="Submit",
command=self.check_answer,bg=Button_color_bg,font=("courier
new",20),fg=Button_color_fg,highlightbackground=
Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=0)
else:
self.clear_screen()
tk.Label(self.root, text=f"Quiz Completed!, Your Total score
was {self.score}", font=("courier new", 20, "bold"),
bg=Label_color_bg, fg=Label_color_fg).pack(pady=20)
```

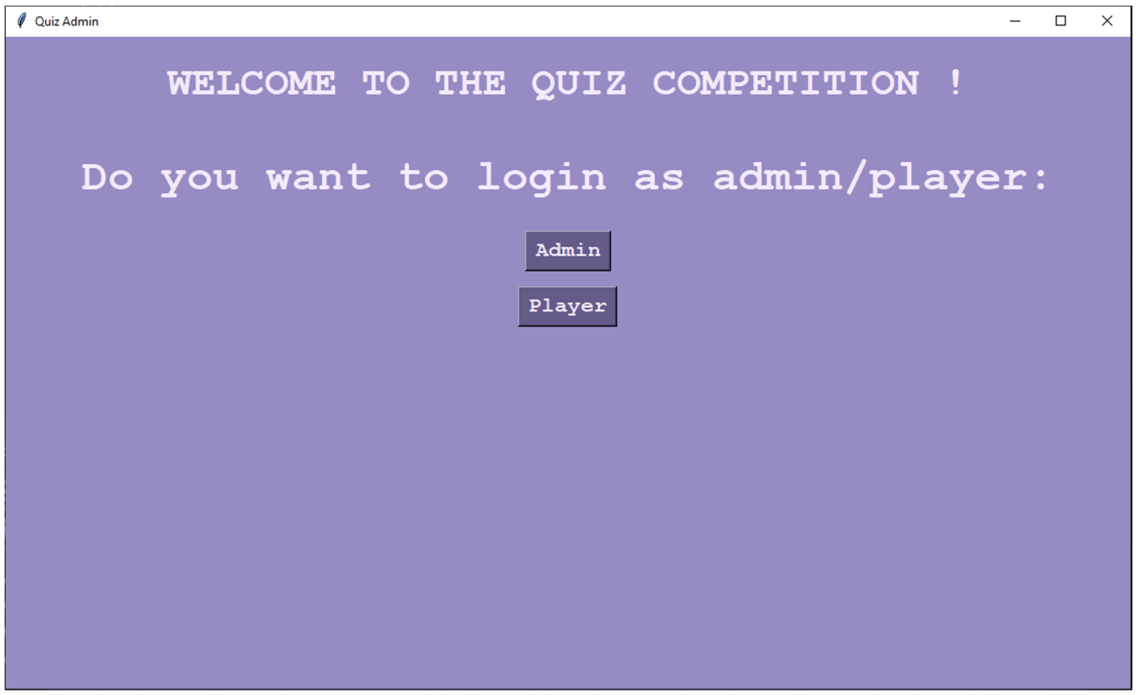
```
self.quiz.add_score(username=self.username,
category=self.current_category, score=self.score)
self.display_leaderboard()

def check_answer(self):
user_answer = self.answer_entry.get()
if user_answer.lower() == self.correct_answer.lower():
self.score += 2
messagebox.showinfo("Correct!", f"Correct Answer! Your current
score is {self.score}")
else:
messagebox.showerror("Wrong!", f"Wrong Answer! Correct answer
is {self.correct_answer}")
self.current_question_index += 1
self.display_question()

def login(self):
username = self.admin_user_entry.get()
password = self.admin_pass_entry.get()
if username == self.quiz.admin_u and password ==
self.quiz.admin_p:
messagebox.showinfo("Login Success", "Welcome admin!")
self.create_main_screen()
else:
messagebox.showerror("Login Failed", "Invalid username or
password.")

def create_main_screen(self):
self.clear_screen()
tk.Label(self.root, text="Admin Panel", font=("courier new",
32), bg=Label_color_bg, fg=Label_color_fg).pack(pady=20)
tk.Button(self.root, text="Add Category",
command=self.add_category, bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
16),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=10)
tk.Button(self.root, text="Add Question",
command=self.add_question, bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
16),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=10)
tk.Button(self.root, text="Create Tables",
command=self.quiz.create_tables, bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
16),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=10)
tk.Button(self.root, text="Save Questions to DB",
command=self.quiz.add_qa_sql, bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
```

```
16),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=10)
tk.Button(self.root, text="Reset database",command=lambda:
[reset_database(), self.create_login_mode_screen()],
bg=Button_color_bg, fg=Button_color_fg, font=("courier new",
16),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=10)
tk.Button(self.root, text="Quit",
command=self.create_login_mode_screen, bg=Button_color_bg,
fg=Button_color_fg, font=("courier new",
16),highlightbackground= Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=10)

def add_category(self):
category_name = simpledialog.askstring("Category", "Enter the
name of the category:")
if category_name:
self.quiz.categories.append((category_name, [], []))
messagebox.showinfo("Success", f"Category '{category_name}'
added.")

def add_question(self):
if not self.quiz.categories:
messagebox.showerror("Error", "No categories found. Please add
a category first.")
return
category_name = simpledialog.askstring("Category", "Enter the
name of the category:")
if category_name:
question = simpledialog.askstring("Question", "Enter the
question:")
answer = simpledialog.askstring("Answer", "Enter the answer:")
if question and answer:
self.quiz.add_question(category_name, question, answer)
messagebox.showinfo("Success", "Question and answer added.")

def display_leaderboard(self):
leaderboard = self.quiz.get_leaderboard(self.current_category)
self.clear_screen()
tk.Label(self.root, text=f"Leaderboard -
{self.current_category}", font=("courier new", 16),
bg=Label_color_bg, fg=Label_color_fg).pack(pady=20)
style = ttk.Style()
style.theme_use('clam')
style.configure("Treeview",
background=table_color_inner_bg,
foreground=table_color_text,
rowheight=25,
fieldbackground=table_color_bg,
```

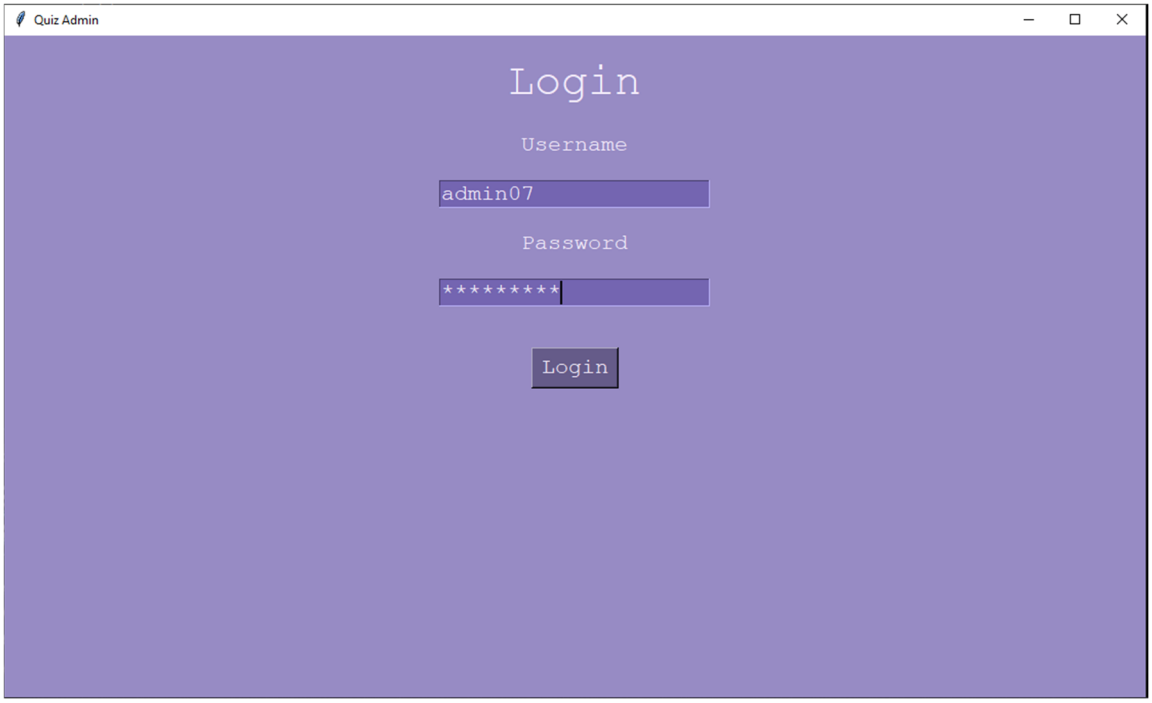
```
font=("courier new", 16))
style.configure("Treeview.Heading", background="#fbf5fd",
foreground="black",font=("courier new", 16))
style.map("Treeview",
background=[('selected', '#978bc4')],
foreground=[('selected', 'white')])
columns = ("Position", "Username", "Score")
tree = ttk.Treeview(self.root, columns=columns,
show='headings')
tree.heading("Position", text="Position")
tree.heading("Username", text="Username")
tree.heading("Score", text="Score")
tree.column("Position", width=130, anchor=tk.CENTER)
tree.column("Username", width=130, anchor=tk.CENTER)
tree.column("Score", width=130, anchor=tk.CENTER)
for i, (username, score) in enumerate(leaderboard, start=1):
tree.insert('', tk.END, values=(i, username, score))
tree.pack(pady=10)
tk.Button(self.root, text="Quit",
command=self.create_login_mode_screen,
bg=Button_color_bg, fg=Button_color_fg, font=("courier new",
16),
highlightbackground=Button_color_bg_selected,
highlightcolor=Button_color_fg_selected).pack(pady=10)

if __name__ == "__main__":

create_sql_db()
quiz = Quiz()
root = tk.Tk()
root.configure(bg=Background)
gui = QuizGUI(root, quiz)
root.mainloop()
```

SAMPLE OUTPUT

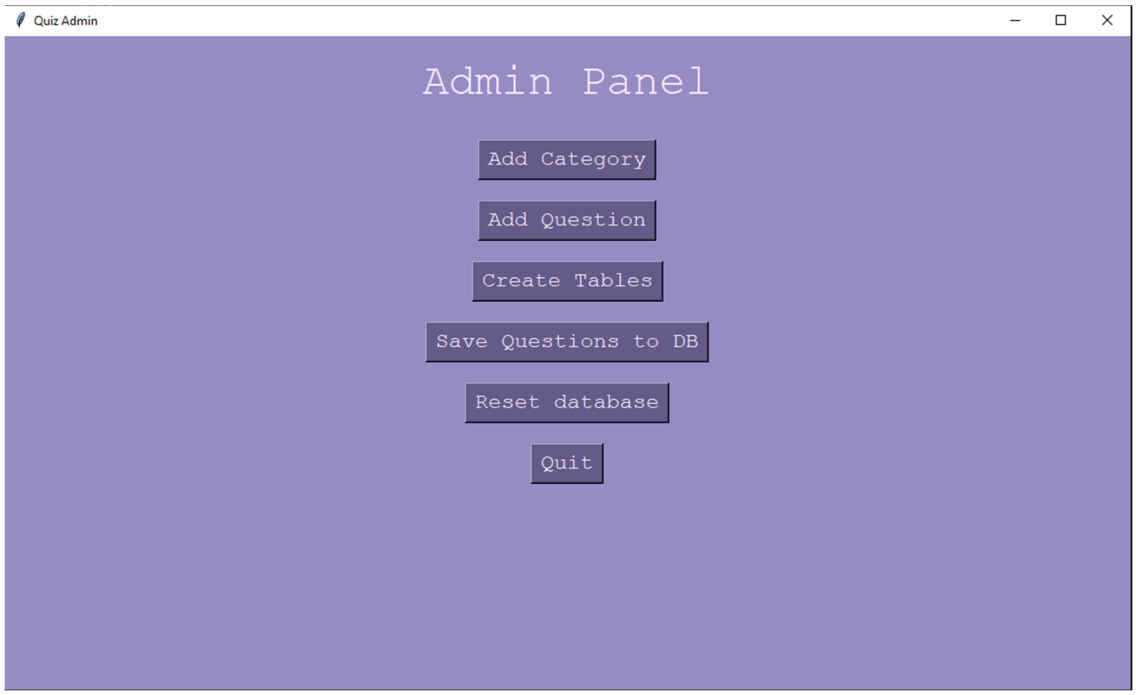
- ***MAIN MENU:***



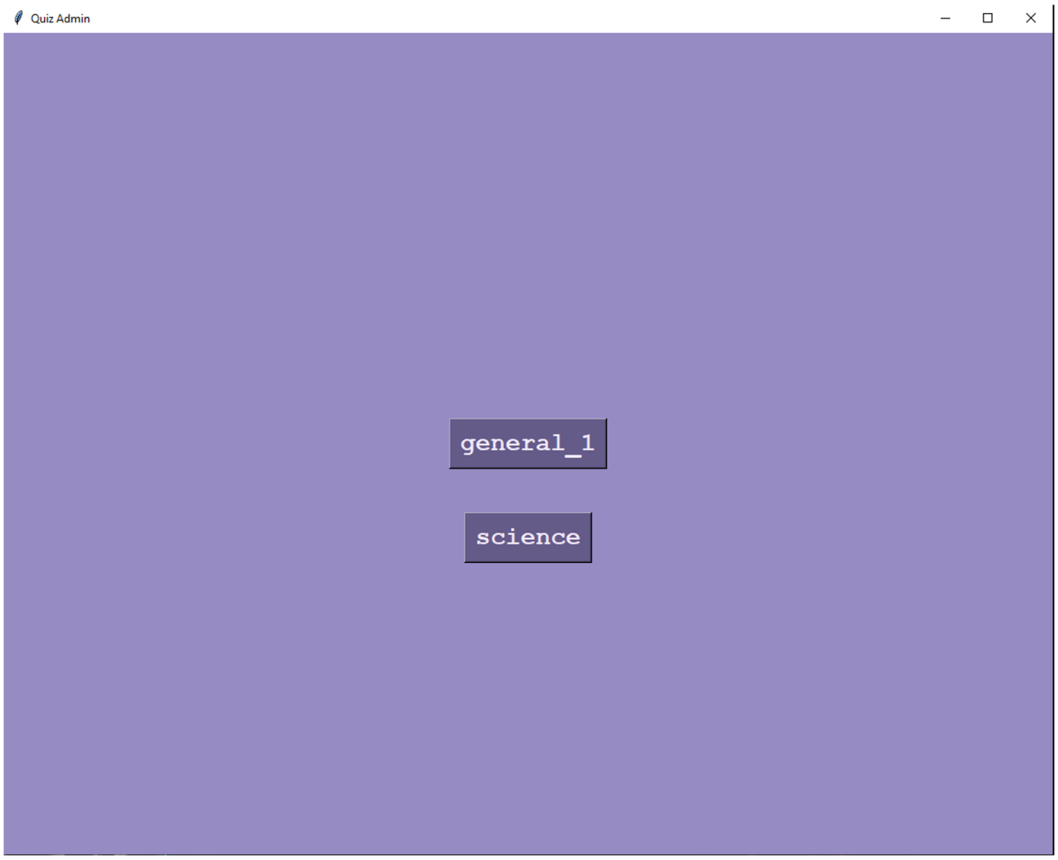
- **LOGIN MENU FOR ADMIN:**



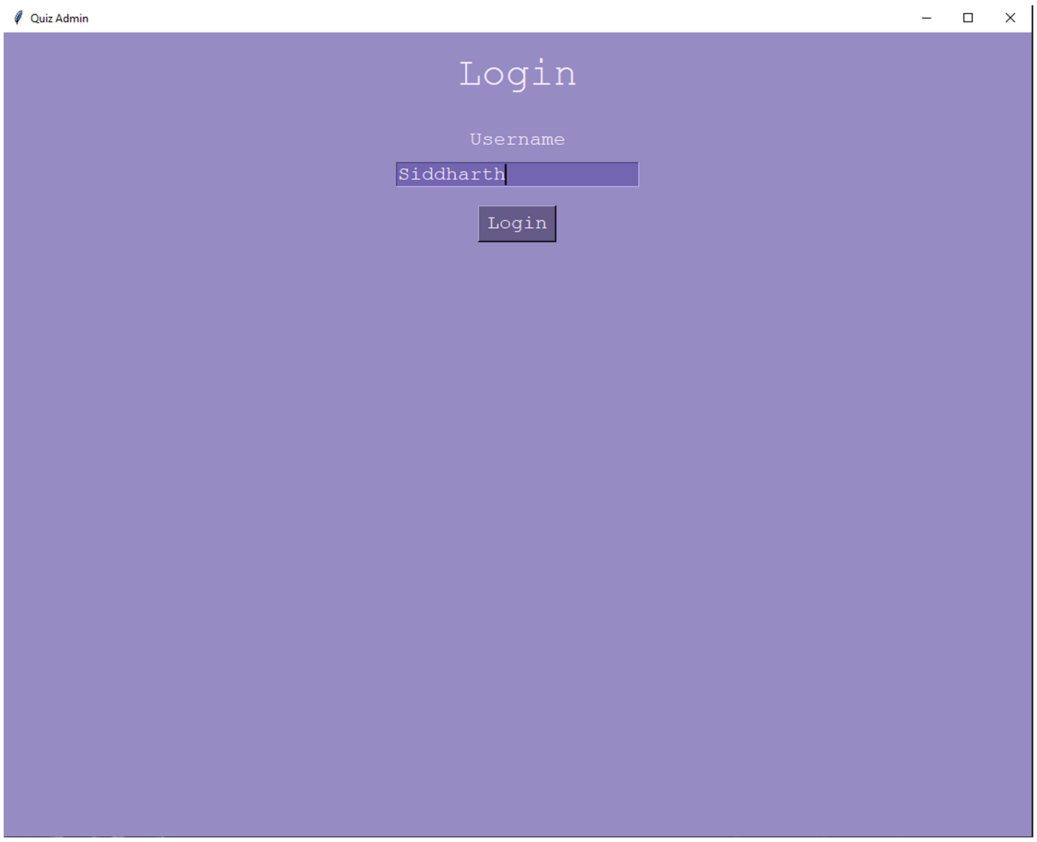
▪ ADMIN PANEL:



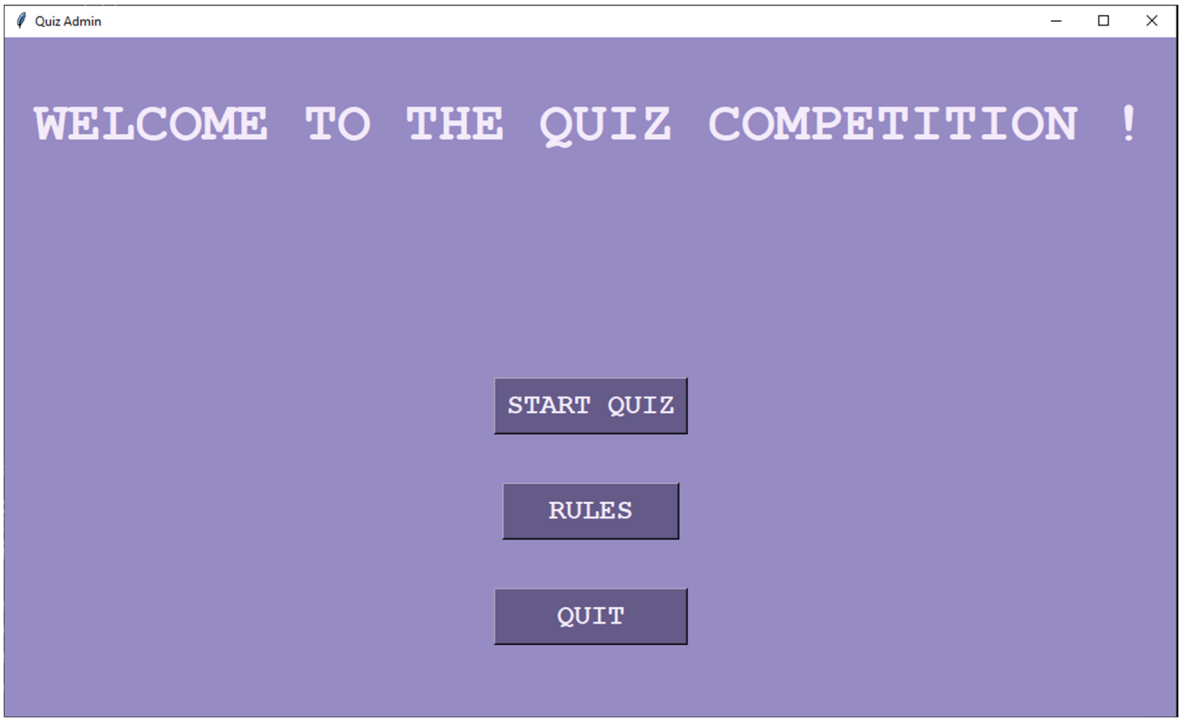
▪ CATEGORIES PANEL



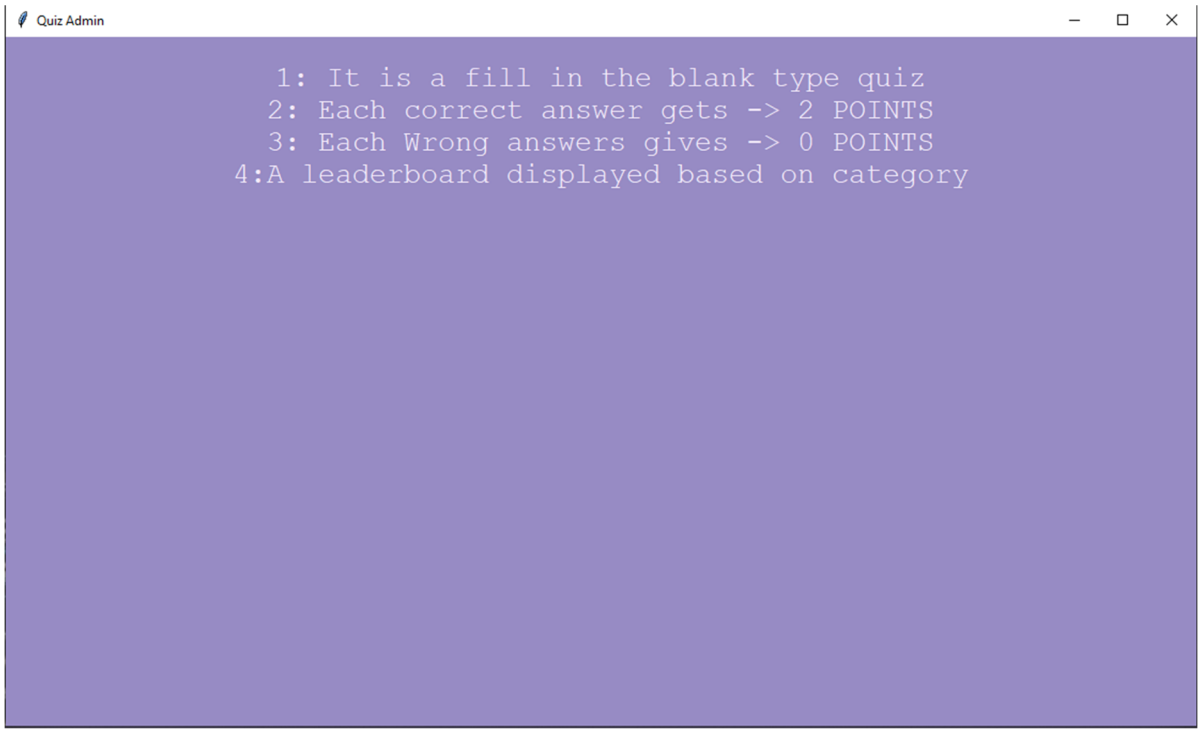
▪ LOGIN MENU FOR PLAYERS:



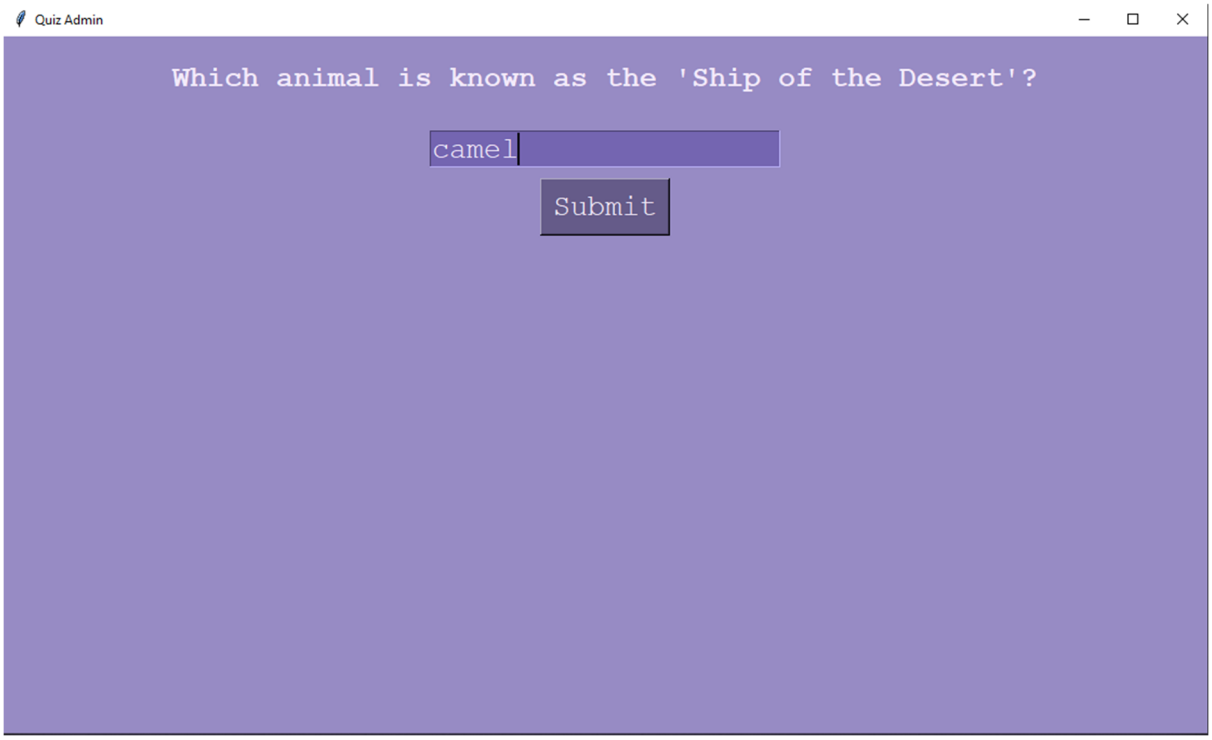
▪ PLAYER MENU:



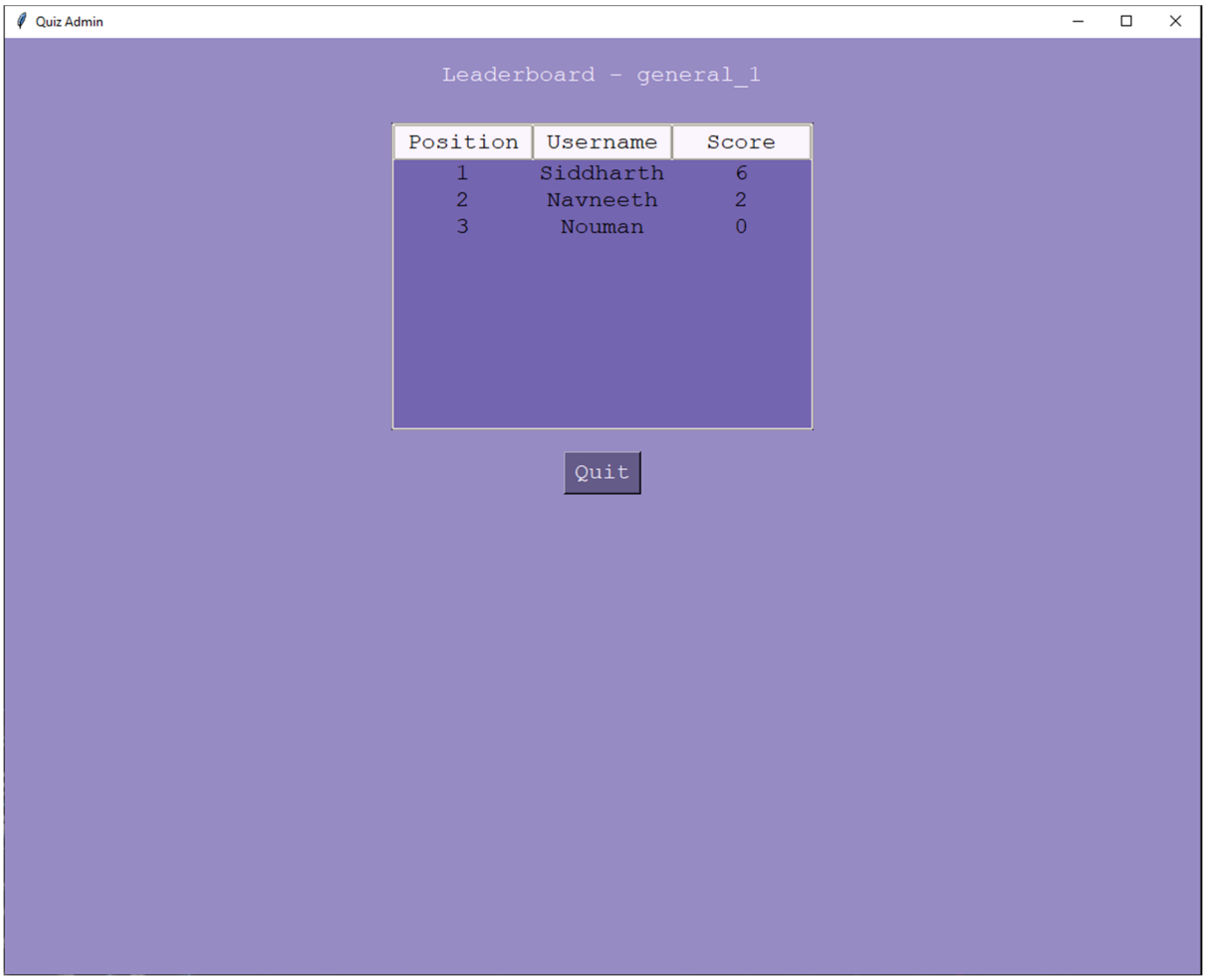
- **RULES PANEL:**



- **START QUIZ/ QUESTION PANEL:**



▪ LEADERBOARD WINDOW:



▪ SQL TABLES FROM DATABASE "QUIZ"

```
mysql> USE quiz
Database changed
mysql> SHOW TABLES;
+------------------+
| Tables_in_quiz   |
+------------------+
| general_1        |
| general_1_scores |
| science          |
| science_scores   |
+------------------+
4 rows in set (0.00 sec)
```

▪ TABLE GENERAL_1

```
mysql> SELECT * FROM general_1;
+-----------------------------------------------------+------------+
| q                                                   | a          |
+-----------------------------------------------------+------------+
| Which animal is known as the 'Ship of the Desert'?  | Camel      |
| How many days are there in a week?                  | 7 days     |
| How many hours are there in a day?                  | 24 hours   |
| How many letters are there in the English alphabet? | 26 letters |
| Rainbow consist of how many colours?                | 7 colours  |
+-----------------------------------------------------+------------+
5 rows in set (0.02 sec)
```

▪ TABLE GENERAL_1_SCORES

```
mysql> Select * from general_1_scores;
+-----------+-------+
| username  | score |
+-----------+-------+
| Nouman    |     0 |
| Navneeth  |     2 |
| Siddharth |     6 |
+-----------+-------+
3 rows in set (0.01 sec)
```

BIBLIOGRAPHY

- Computer Science with python Class XII Sumita Arora
- docs.python.org
- geeksforgeeks.org
- www.google.com