**Hangman Game**

A simple text-based hangman game that uses Python for the game logic and MySQL to store player data and high scores.

(In MySQL)

CREATE DATABASE hangman\_game;

USE hangman\_game;

CREATE TABLE players (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

high\_score INT DEFAULT 0

);

(In comm prompt)

pip install mysql-connector-python

(In python)

import tkinter as tk

from tkinter import messagebox

import mysql.connector

from mysql.connector import Error

import random

# Database configuration

db\_config = {

'host': 'localhost',

'user': 'root',

'password': '1234',

'database': 'hangman\_game'

}

class HangmanGUI:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Hangman Game")

self.username\_label = tk.Label(root, text="Username:")

self.username\_label.pack()

self.username\_entry = tk.Entry(root)

self.username\_entry.pack()

self.password\_label = tk.Label(root, text="Password:")

self.password\_label.pack()

self.password\_entry = tk.Entry(root, show='\*')

self.password\_entry.pack()

self.register\_button = tk.Button(root, text="Register", command=self.register)

self.register\_button.pack()

self.login\_button = tk.Button(root, text="Login", command=self.login)

self.login\_button.pack()

self.exit\_button = tk.Button(root, text="Exit", command=root.quit)

self.exit\_button.pack()

def create\_connection(self):

"""Create a database connection."""

try:

connection = mysql.connector.connect(\*\*db\_config)

if connection.is\_connected():

return connection

except Error as e:

messagebox.showerror("Error", f"Database connection error: {e}")

return None

def close\_connection(self, connection):

"""Close the database connection."""

if connection.is\_connected():

connection.close()

def register(self):

"""Register a new player."""

username = self.username\_entry.get()

password = self.password\_entry.get()

connection = self.create\_connection()

if connection:

cursor = connection.cursor()

try:

cursor.execute("INSERT INTO players (username, password) VALUES (%s, %s)", (username, password))

connection.commit()

messagebox.showinfo("Registration", "Registration successful.")

except Error as e:

messagebox.showerror("Error", f"Registration error: {e}")

finally:

self.close\_connection(connection)

def login(self):

"""Login an existing player."""

username = self.username\_entry.get()

password = self.password\_entry.get()

connection = self.create\_connection()

if connection:

cursor = connection.cursor()

cursor.execute("SELECT id, high\_score FROM players WHERE username = %s AND password = %s", (username, password))

result = cursor.fetchone()

self.close\_connection(connection)

if result:

player\_id, high\_score = result

messagebox.showinfo("Login", f"Login successful. Your high score is {high\_score}.")

self.play\_game(player\_id, high\_score)

else:

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

def play\_game(self, player\_id, high\_score):

"""Play the hangman game."""

words = ['python', 'mysql', 'database', 'programming', 'hangman']

word = random.choice(words)

guessed\_word = ['\_'] \* len(word)

guessed\_letters = set()

attempts = 7

hangman\_frame = tk.Frame(self.root)

hangman\_frame.pack()

hangman\_label = tk.Label(hangman\_frame, text=" ".join(guessed\_word))

hangman\_label.pack()

guessed\_word\_label = tk.Label(self.root, text="")

guessed\_word\_label.pack()

attempts\_label = tk.Label(self.root, text=f"Remaining attempts: {attempts}")

attempts\_label.pack()

hangman\_ascii\_label = tk.Label(self.root, text="")

hangman\_ascii\_label.pack()

def print\_hangman(attempts):

"""Print ASCII art of hangman."""

if attempts == 7:

hangman\_ascii\_label.config(text=

"+---+\n"

" |\n"

" |\n"

" |\n"

" ==="

)

elif attempts == 6:

hangman\_ascii\_label.config(text=

"+---+\n"

"O |\n"

" |\n"

" |\n"

" ==="

)

elif attempts == 5:

hangman\_ascii\_label.config(text=

"+---+\n"

"O |\n"

"| |\n"

" |\n"

" ==="

)

elif attempts == 4:

hangman\_ascii\_label.config(text=

"+---+\n"

" O |\n"

"/| |\n"

" |\n"

" ==="

)

elif attempts == 3:

hangman\_ascii\_label.config(text=

"+---+\n"

" O |\n"

"/|\ |\n"

" |\n"

" ==="

)

elif attempts == 2:

hangman\_ascii\_label.config(text=

"+---+\n"

" O |\n"

"/|\ |\n"

"/ |\n"

" ==="

)

elif attempts == 1:

hangman\_ascii\_label.config(text=

"+---+\n"

" O |\n"

"/|\ |\n"

"/ \ |\n"

" ==="

)

def make\_guess():

guess = guess\_entry.get().lower()

if guess in guessed\_letters:

messagebox.showinfo("Duplicate Guess", "You already guessed that letter. Try again.")

elif guess in word:

messagebox.showinfo("Good Guess", "Good guess!")

for i, letter in enumerate(word):

if letter == guess:

guessed\_word[i] = guess

hangman\_label.config(text=" ".join(guessed\_word))

else:

nonlocal attempts

attempts -= 1

attempts\_label.config(text=f"Remaining attempts: {attempts}")

print\_hangman(attempts)

if attempts == 0:

messagebox.showinfo("Game Over", f"Game over. The word was: {word}")

hangman\_frame.destroy()

else:

messagebox.showinfo("Wrong Guess", "Incorrect guess. Try again.")

guessed\_letters.add(guess)

guess\_entry.delete(0, tk.END)

if '\_' not in guessed\_word:

messagebox.showinfo("Congratulations", "Congratulations! You guessed the word.")

score = attempts

if score > high\_score:

messagebox.showinfo("New High Score", "New high score!")

self.update\_high\_score(player\_id, score)

guess\_label = tk.Label(self.root, text="Guess a letter:")

guess\_label.pack()

guess\_entry = tk.Entry(self.root)

guess\_entry.pack()

guess\_button = tk.Button(self.root, text="Guess", command=make\_guess)

guess\_button.pack()

def update\_high\_score(self, player\_id, score):

"""Update the high score of a player."""

connection = self.create\_connection()

if connection:

cursor = connection.cursor()

try:

cursor.execute("UPDATE players SET high\_score = %s WHERE id = %s", (score, player\_id))

connection.commit()

except Error as e:

messagebox.showerror("Error", f"Update high score error: {e}")

finally:

self.close\_connection(connection)

def main():

root = tk.Tk()

app = HangmanGUI(root)

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

To run go comm. Prompt n type :-

(After choosing correct address (jisme file save h))

python hangman\_game.py

(n play the game in comm. Prompt only)

• EXPLAINATION

- Database Configuration: The `db\_config` dictionary contains the database connection details.

- Create Connection: The `create\_connection` function establishes a connection to the MySQL database.

- Close Connection: The `close\_connection` function closes the database connection.

- Register and Login: The `register` and `login` functions handle player registration and login.

- Update High Score: The `update\_high\_score` function updates the player's high score in the database.

- Play Game: The `play\_game` function contains the logic for the hangman game.

- Main Loop: The `main` function provides a command-line interface for interacting with the game.