# DCDS LAB ESE PROJECT CODE – SOCIAL NETWORKING

import mysql.connector

import tkinter as tk

from tkinter import messagebox, Listbox, simpledialog

# MySQL connection

conn = mysql.connector.connect(host='localhost', user='root', password='123456', database='project\_data')

cursor = conn.cursor()

# Color scheme

bg\_color = "#FFF7D1"

button\_color = "#FFB0B0"

text\_color = "black"

# Offset for opening windows in new positions

window\_offset = 30

window\_count = 0

# Helper function to center windows with offset

def center\_window(window):

global window\_count

window.update\_idletasks()

width = window.winfo\_width()

height = window.winfo\_height()

x = (window.winfo\_screenwidth() // 2) - (width // 2) + (window\_offset \* window\_count)

y = (window.winfo\_screenheight() // 2) - (height // 2) + (window\_offset \* window\_count)

window.geometry(f"{width}x{height}+{x}+{y}")

window\_count += 1

def home\_page():

global window\_count

window\_count = 0

home = tk.Tk()

home.title("Social Media App")

home.geometry("400x400")

home.configure(bg=bg\_color)

tk.Label(home, text="Social Media CRUD App", font=("Arial", 16), bg=bg\_color, fg=text\_color).pack(pady=20)

# Buttons to navigate to each table's CRUD page

for page\_name, page\_func in [

("Users", users\_page),

("Photos", photos\_page),

("Tags", tags\_page),

("Follows", follows\_page),

("Comments", comments\_page)

]:

tk.Button(home, text=page\_name, command=page\_func, width=20, bg=button\_color).pack(pady=10)

center\_window(home)

home.mainloop()

def comments\_page():

def add\_comment():

comment\_text = comment\_text\_entry.get()

user\_id = user\_id\_entry.get()

photo\_id = photo\_id\_entry.get()

if comment\_text and user\_id and photo\_id:

try:

cursor.execute(

"INSERT INTO comments (comment\_text, user\_id, photo\_id, comment\_created\_at) VALUES (%s, %s, %s, NOW())",

(comment\_text, user\_id, photo\_id)

)

conn.commit()

messagebox.showinfo("Success", "Comment added successfully.")

refresh\_comments()

except Exception as e:

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

else:

messagebox.showerror("Input Error", "Comment text, User ID, and Photo ID are required.")

def delete\_comment():

selected\_items = comments\_listbox.curselection()

if not selected\_items:

messagebox.showerror("Selection Error", "Please select a comment to delete.")

return

selected = comments\_listbox.get(selected\_items[0])

comment\_id = int(selected.split(",")[0].split(": ")[1])

try:

cursor.execute("DELETE FROM comments WHERE id = %s", (comment\_id,))

conn.commit()

messagebox.showinfo("Success", "Comment deleted successfully.")

refresh\_comments()

except Exception as e:

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

def refresh\_comments():

comments\_listbox.delete(0, tk.END)

cursor.execute("SELECT id, comment\_text, user\_id, photo\_id, comment\_created\_at FROM comments")

for comment in cursor.fetchall():

comments\_listbox.insert(tk.END, f"ID: {comment[0]}, Text: {comment[1]}, User ID: {comment[2]}, "

f"Photo ID: {comment[3]}, Created At: {comment[4]}")

# Window for Comments Page

comments\_window = tk.Toplevel()

comments\_window.title("Comments Page")

comments\_window.geometry("500x500")

comments\_window.configure(bg=bg\_color)

# Input fields for Comment Text, User ID, and Photo ID

tk.Label(comments\_window, text="Comment Text:", bg=bg\_color, fg=text\_color).pack(pady=5)

comment\_text\_entry = tk.Entry(comments\_window, bg="white")

comment\_text\_entry.pack(pady=5)

tk.Label(comments\_window, text="User ID:", bg=bg\_color, fg=text\_color).pack(pady=5)

user\_id\_entry = tk.Entry(comments\_window, bg="white")

user\_id\_entry.pack(pady=5)

tk.Label(comments\_window, text="Photo ID:", bg=bg\_color, fg=text\_color).pack(pady=5)

photo\_id\_entry = tk.Entry(comments\_window, bg="white")

photo\_id\_entry.pack(pady=5)

# CRUD buttons

tk.Button(comments\_window, text="Add Comment", command=add\_comment, bg=button\_color).pack(pady=5)

tk.Button(comments\_window, text="Delete Comment", command=delete\_comment, bg=button\_color).pack(pady=5)

# Listbox to display comments

comments\_listbox = tk.Listbox(comments\_window)

comments\_listbox.pack(pady=5, fill=tk.BOTH, expand=True)

refresh\_comments()

# Back button

tk.Button(comments\_window, text="Back to Home", command=comments\_window.destroy, bg=button\_color).pack(pady=10)

center\_window(comments\_window)

# Remember to replace bg\_color, text\_color, button\_color, and center\_window function with your own configurations

def follows\_page():

def add\_follow():

follower\_id = follower\_id\_entry.get()

followee\_id = followee\_id\_entry.get()

if follower\_id and followee\_id:

try:

cursor.execute("INSERT INTO follows (follower\_id, followee\_id) VALUES (%s, %s)", (follower\_id, followee\_id))

conn.commit()

messagebox.showinfo("Success", "Follow relationship added successfully.")

refresh\_follows()

except Exception as e:

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

else:

messagebox.showerror("Input Error", "Both follower ID and followee ID are required.")

def delete\_follow():

selected\_items = follows\_listbox.curselection()

if not selected\_items:

messagebox.showerror("Selection Error", "Please select a follow relationship to delete.")

return

selected = follows\_listbox.get(selected\_items[0])

follower\_id, followee\_id = [int(part.split(": ")[1]) for part in selected.split(",")]

try:

cursor.execute("DELETE FROM follows WHERE follower\_id = %s AND followee\_id = %s", (follower\_id, followee\_id))

conn.commit()

messagebox.showinfo("Success", "Follow relationship deleted successfully.")

refresh\_follows()

except Exception as e:

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

def refresh\_follows():

follows\_listbox.delete(0, tk.END)

cursor.execute("SELECT follower\_id, followee\_id FROM follows")

for follow in cursor.fetchall():

follows\_listbox.insert(tk.END, f"Follower ID: {follow[0]}, Followee ID: {follow[1]}")

# Window for Follows Page

follows\_window = tk.Toplevel()

follows\_window.title("Follows Page")

follows\_window.geometry("500x500")

follows\_window.configure(bg=bg\_color)

# Input fields for Follower ID and Followee ID

tk.Label(follows\_window, text="Follower ID:", bg=bg\_color, fg=text\_color).pack(pady=5)

follower\_id\_entry = tk.Entry(follows\_window, bg="white")

follower\_id\_entry.pack(pady=5)

tk.Label(follows\_window, text="Followee ID:", bg=bg\_color, fg=text\_color).pack(pady=5)

followee\_id\_entry = tk.Entry(follows\_window, bg="white")

followee\_id\_entry.pack(pady=5)

# CRUD buttons

tk.Button(follows\_window, text="Add Follow", command=add\_follow, bg=button\_color).pack(pady=5)

tk.Button(follows\_window, text="Delete Follow", command=delete\_follow, bg=button\_color).pack(pady=5)

# Listbox to display follow relationships

follows\_listbox = tk.Listbox(follows\_window)

follows\_listbox.pack(pady=5, fill=tk.BOTH, expand=True)

refresh\_follows()

# Back button

tk.Button(follows\_window, text="Back to Home", command=follows\_window.destroy, bg=button\_color).pack(pady=10)

center\_window(follows\_window)

def users\_page():

def add\_user():

username = username\_entry.get()

if username:

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

conn.commit()

messagebox.showinfo("Success", "User added successfully.")

refresh\_users()

else:

messagebox.showerror("Input Error", "Username cannot be empty.")

def delete\_user():

selected\_items = users\_listbox.curselection()

if not selected\_items:

messagebox.showerror("Selection Error", "Please select a user to delete.")

return

selected = users\_listbox.get(selected\_items[0])

user\_id = int(selected.split(",")[0].split(": ")[1])

try:

cursor.execute("DELETE FROM users WHERE id = %s", (user\_id,))

conn.commit()

messagebox.showinfo("Success", "User deleted successfully.")

refresh\_users()

except mysql.connector.IntegrityError as err:

messagebox.showerror("Error", f"Cannot delete user due to foreign key constraint: {err}")

except Exception as e:

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

def update\_user():

selected\_items = users\_listbox.curselection()

if not selected\_items:

messagebox.showerror("Selection Error", "Please select a user to update.")

return

selected = users\_listbox.get(selected\_items[0])

user\_id = int(selected.split(",")[0].split(": ")[1])

new\_username = username\_entry.get()

if new\_username:

cursor.execute("UPDATE users SET username = %s WHERE id = %s", (new\_username, user\_id))

conn.commit()

messagebox.showinfo("Success", "User updated successfully.")

refresh\_users()

else:

messagebox.showerror("Input Error", "Username cannot be empty.")

def refresh\_users():

users\_listbox.delete(0, tk.END)

cursor.execute("SELECT id, username, user\_created\_at FROM users")

for user in cursor.fetchall():

users\_listbox.insert(tk.END, f"ID: {user[0]}, Username: {user[1]}, Created At: {user[2]}")

# Window for Users Page

try:

users\_window = tk.Toplevel()

users\_window.title("Users Page")

users\_window.geometry("500x500")

users\_window.configure(bg=bg\_color)

# User input fields

tk.Label(users\_window, text="Username:", bg=bg\_color, fg=text\_color).pack(pady=5)

username\_entry = tk.Entry(users\_window, bg="white")

username\_entry.pack(pady=5)

# CRUD buttons

tk.Button(users\_window, text="Add User", command=add\_user, bg=button\_color).pack(pady=5)

tk.Button(users\_window, text="Update User", command=update\_user, bg=button\_color).pack(pady=5)

tk.Button(users\_window, text="Delete User", command=delete\_user, bg=button\_color).pack(pady=5)

# Listbox to display users

users\_listbox = tk.Listbox(users\_window)

users\_listbox.pack(pady=5, fill=tk.BOTH, expand=True)

refresh\_users()

# Back button

tk.Button(users\_window, text="Back to Home", command=users\_window.destroy, bg=button\_color).pack(pady=10)

center\_window(users\_window)

except Exception as e:

messagebox.showerror("Error", f"Error opening users page: {e}")

# Photos Page

def photos\_page():

def add\_photo():

image\_url = image\_url\_entry.get()

username = username\_entry.get()

cursor.execute("SELECT id FROM users WHERE username = %s", (username,))

user\_id = cursor.fetchone()

if image\_url and user\_id:

cursor.execute("INSERT INTO photos (image\_url, user\_id) VALUES (%s, %s)", (image\_url, user\_id[0]))

conn.commit()

messagebox.showinfo("Success", "Photo added successfully.")

refresh\_photos()

else:

messagebox.showerror("Input Error", "Both fields must be filled with valid data.")

def delete\_photo():

selected\_items = photos\_listbox.curselection()

if not selected\_items:

messagebox.showerror("Selection Error", "Please select a photo to delete.")

return

selected = photos\_listbox.get(selected\_items[0])

photo\_id = int(selected.split(",")[0].split(": ")[1])

try:

cursor.execute("DELETE FROM photos WHERE id = %s", (photo\_id,))

conn.commit()

messagebox.showinfo("Success", "Photo deleted successfully.")

refresh\_photos()

except mysql.connector.IntegrityError as err:

messagebox.showerror("Error", f"Cannot delete photo due to foreign key constraint: {err}")

except Exception as e:

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

def refresh\_photos():

photos\_listbox.delete(0, tk.END)

cursor.execute(

"SELECT photos.id, photos.image\_url, users.username FROM photos JOIN users ON photos.user\_id = users.id"

)

for photo in cursor.fetchall():

photos\_listbox.insert(tk.END, f"ID: {photo[0]}, URL: {photo[1]}, User: {photo[2]}")

# Window for Photos Page

try:

photos\_window = tk.Toplevel()

photos\_window.title("Photos Page")

photos\_window.geometry("500x500")

photos\_window.configure(bg=bg\_color)

tk.Label(photos\_window, text="Image URL:", bg=bg\_color, fg=text\_color).pack(pady=5)

image\_url\_entry = tk.Entry(photos\_window, bg="white")

image\_url\_entry.pack(pady=5)

tk.Label(photos\_window, text="Username:", bg=bg\_color, fg=text\_color).pack(pady=5)

username\_entry = tk.Entry(photos\_window, bg="white")

username\_entry.pack(pady=5)

tk.Button(photos\_window, text="Add Photo", command=add\_photo, bg=button\_color).pack(pady=5)

tk.Button(photos\_window, text="Delete Photo", command=delete\_photo, bg=button\_color).pack(pady=5)

photos\_listbox = Listbox(photos\_window)

photos\_listbox.pack(pady=5, fill=tk.BOTH, expand=True)

refresh\_photos()

tk.Button(photos\_window, text="Back to Home", command=photos\_window.destroy, bg=button\_color).pack(pady=10)

center\_window(photos\_window)

except Exception as e:

messagebox.showerror("Error", f"Error opening photos page: {e}")

# Tags Page (Tagging and Tags Management)

def tags\_page():

def add\_tag\_to\_photo():

photo\_id = simpledialog.askinteger("Photo ID", "Enter the Photo ID:")

tag\_name = simpledialog.askstring("Tag Name", "Enter the Tag Name:")

if photo\_id and tag\_name:

# Check if the tag exists; if not, create it

cursor.execute("SELECT id FROM tags WHERE tag\_name = %s", (tag\_name,))

tag = cursor.fetchone()

if tag:

tag\_id = tag[0]

else:

cursor.execute("INSERT INTO tags (tag\_name) VALUES (%s)", (tag\_name,))

conn.commit()

tag\_id = cursor.lastrowid # Get the ID of the new tag

# Add the tag to the photo

try:

cursor.execute("INSERT INTO photo\_tags (photo\_id, tag\_id) VALUES (%s, %s)", (photo\_id, tag\_id))

conn.commit()

messagebox.showinfo("Success", f"Tag '{tag\_name}' added to Photo ID {photo\_id}.")

except mysql.connector.Error as err:

messagebox.showerror("Error", f"Could not add tag: {err}")

else:

messagebox.showwarning("Input Error", "Both Photo ID and Tag Name are required.")

def view\_tags\_for\_photo():

photo\_id = simpledialog.askinteger("Photo ID", "Enter the Photo ID to view tags:")

if photo\_id:

cursor.execute("""

SELECT tags.tag\_name

FROM tags

JOIN photo\_tags ON tags.id = photo\_tags.tag\_id

WHERE photo\_tags.photo\_id = %s

""", (photo\_id,))

tags = cursor.fetchall()

tags\_text = "\n".join(tag[0] for tag in tags)

messagebox.showinfo("Tags for Photo", f"Tags for Photo ID {photo\_id}:\n{tags\_text}")

def update\_tag():

old\_tag\_name = simpledialog.askstring("Old Tag Name", "Enter the existing Tag Name to update:")

new\_tag\_name = simpledialog.askstring("New Tag Name", "Enter the new Tag Name:")

if old\_tag\_name and new\_tag\_name:

try:

cursor.execute("UPDATE tags SET tag\_name = %s WHERE tag\_name = %s", (new\_tag\_name, old\_tag\_name))

conn.commit()

messagebox.showinfo("Success", f"Tag '{old\_tag\_name}' updated to '{new\_tag\_name}'.")

except mysql.connector.Error as err:

messagebox.showerror("Error", f"Could not update tag: {err}")

else:

messagebox.showwarning("Input Error", "Both old and new tag names are required.")

def delete\_tag\_from\_photo():

photo\_id = simpledialog.askinteger("Photo ID", "Enter the Photo ID:")

tag\_name = simpledialog.askstring("Tag Name", "Enter the Tag Name to remove from photo:")

if photo\_id and tag\_name:

cursor.execute("SELECT id FROM tags WHERE tag\_name = %s", (tag\_name,))

tag = cursor.fetchone()

if tag:

tag\_id = tag[0]

cursor.execute("DELETE FROM photo\_tags WHERE photo\_id = %s AND tag\_id = %s", (photo\_id, tag\_id))

conn.commit()

messagebox.showinfo("Success", f"Tag '{tag\_name}' removed from Photo ID {photo\_id}.")

else:

messagebox.showerror("Error", "Tag not found.")

else:

messagebox.showwarning("Input Error", "Both Photo ID and Tag Name are required.")

# Window for Tags Page

tags\_window = tk.Toplevel()

tags\_window.title("Tags Management")

tags\_window.geometry("400x400")

tags\_window.configure(bg=bg\_color)

tk.Button(tags\_window, text="Add Tag to Photo", command=add\_tag\_to\_photo, bg=button\_color).pack(pady=5)

tk.Button(tags\_window, text="View Tags for Photo", command=view\_tags\_for\_photo, bg=button\_color).pack(pady=5)

tk.Button(tags\_window, text="Update Tag", command=update\_tag, bg=button\_color).pack(pady=5)

tk.Button(tags\_window, text="Delete Tag from Photo", command=delete\_tag\_from\_photo, bg=button\_color).pack(pady=5)

tk.Button(tags\_window, text="Back to Home", command=tags\_window.destroy, bg=button\_color).pack(pady=10)

center\_window(tags\_window)

# Run the app

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

home\_page()

conn.close()