Code:

TO DO LIST APP

import tkinter as tk

from PIL import Image, ImageTk

import sqlite3

import time

from plyer import notification

import datetime

import matplotlib.pyplot as plt

import os

import tkinter.messagebox

# ✅ Create the Tkinter root window FIRST

root = tk.Tk()

root.title("Smart To-Do List App")

root.geometry("900x600")  # Resize window

# ✅ Define Correct Background Image Path

bg\_image\_path = r"C:\Users\AKSHAYA\ToDoApp\def.jpeg"  # Make sure this file exists

# ✅ Load Background Image AFTER Creating Tkinter Window

try:

    bg\_image = Image.open(bg\_image\_path).resize((900, 600))

    bg\_photo = ImageTk.PhotoImage(bg\_image)

    bg\_label = tk.Label(root, image=bg\_photo)

    bg\_label.place(x=0, y=0, relwidth=1, relheight=1)

except FileNotFoundError:

    print("❌ Background image not found! Please check the file location.")

# ✅ Delete and Recreate Database to Fix `entry\_date` Issue

if os.path.exists("tasks.db"):

    os.remove("tasks.db")  # Delete old database to fix missing column

    print("✅ Old database deleted. A new one will be created!")

# ✅ Initialize Database for Tracking Entries

conn = sqlite3.connect("tasks.db")

cursor = conn.cursor()

cursor.execute("""CREATE TABLE IF NOT EXISTS tasks (

    id INTEGER PRIMARY KEY,

    task TEXT,

    completed BOOLEAN,

    entry\_date TEXT

)""")

cursor.execute("""CREATE TABLE IF NOT EXISTS user\_progress (

    id INTEGER PRIMARY KEY,

    date TEXT

)""")

conn.commit()

conn.close()

print("✅ New database created successfully!")

# Function to Show Badge Pop-up with Star Image

# Function to Show Badge Pop-up with Star Image & Motivational Message

# ✅ Updated function to center pop-up window correctly

def show\_badge\_popup(badge\_text):

    popup = tk.Toplevel(root)  # Create pop-up window

    popup.title("🏆 Badge Earned!")

    popup.geometry("400x300")  # Set window size

    # ✅ Center the window manually

    popup.update\_idletasks()

    width = popup.winfo\_width()

    height = popup.winfo\_height()

    x = (popup.winfo\_screenwidth() // 2) - (width // 2)

    y = (popup.winfo\_screenheight() // 2) - (height // 2)

    popup.geometry(f"+{x}+{y}")

    # Load star image

    try:

        star\_img = Image.open("star.png").resize((100, 100))

        star\_photo = ImageTk.PhotoImage(star\_img)

        star\_label = tk.Label(popup, image=star\_photo)

        star\_label.image = star\_photo

        star\_label.pack(pady=10)

    except FileNotFoundError:

        print("❌ Star image not found!")

    # ✅ Display badge message & motivational quote

    msg\_label = tk.Label(popup, text=badge\_text + "\n\nDon't lose your hope, move on!", font=("Arial", 14, "bold"), fg="black", justify="center")

    msg\_label.pack(pady=10)

    # Close Button

    close\_button = tk.Button(popup, text="OK", command=popup.destroy, width=10)

    close\_button.pack(pady=10)

# Attractive Welcome Message

welcome\_label = tk.Label(root, text="✨ Welcome to Your To-Do List! ✨",

                         font=("Rubik", 24, " bold"), fg="black", bg="white")

welcome\_label.pack(pady=10)

# Task Entry Box with Padding

task\_entry = tk.Text(root, width=40,height=1, font=("Times New Roman", 20, "bold"), bg="white", fg="black")

task\_entry.pack(pady=5)

# Left-Side Command Panel

button\_frame = tk.Frame(root)

button\_frame.pack(side=tk.LEFT, padx=10)

task\_list\_frame = tk.Frame(root)

task\_list\_frame.pack(pady=10)

# Store Task Checkboxes

task\_checkboxes = []

# Function to Add Task

def add\_task():

    task = task\_entry.get("1.0", tk.END).strip()

    if task:

        var = tk.BooleanVar()

        checkbox = tk.Checkbutton(task\_list\_frame, text=task, variable=var, font=("Times New Roman", 20, "bold"))

        checkbox.pack(anchor="w")

        # ✅ Store actual widget reference in the list (instead of text)

        task\_checkboxes.append((checkbox, var))

        save\_task\_to\_db(task)

        task\_entry.delete("1.0", tk.END)  # ✅ Clears input box

def delete\_task():

    for checkbox, var in task\_checkboxes[:]:  # ✅ Iterate safely using a copy

        if var.get():  # ✅ Check if checkbox is selected

            checkbox.destroy()  # ✅ Remove checkbox widget from UI

            task\_checkboxes.remove((checkbox, var))  # ✅ Remove task from list

def save\_task\_to\_db(task):

    conn = sqlite3.connect("tasks.db")

    cursor = conn.cursor()

    cursor.execute("INSERT INTO tasks (task, completed, entry\_date) VALUES (?, ?, ?)",

                   (task, False, str(datetime.date.today())))

    conn.commit()

    conn.close()

# Badge System Based on Daily Entries

def calculate\_badge():

    conn = sqlite3.connect("tasks.db")

    cursor = conn.cursor()

    cursor.execute("SELECT COUNT(DISTINCT entry\_date) FROM tasks")

    days = cursor.fetchone()[0]

    conn.close()

    if days >= 30:

        return "🏆 Gold Star! You've been consistent for 30 days!"

    elif days >= 15:

        return "🥈 Silver Star! Keep up the great work!"

    elif days >= 3:

        return "🥉 Bronze Star! You're off to a great start!"

    else:

        return "Keep adding tasks daily to earn badges!"

# Function to Notify User Daily

def send\_reminder():

    notification.notify(

        title="Task Reminder!",

        message="You have tasks to complete today!",

        timeout=10

    )

# Function to Track Daily Progress

def update\_daily\_tracking():

    conn = sqlite3.connect("tasks.db")

    cursor = conn.cursor()

    today = str(datetime.date.today())

    cursor.execute("SELECT \* FROM user\_progress WHERE date = ?", (today,))

    result = cursor.fetchone()

    if not result:

        cursor.execute("INSERT INTO user\_progress (date) VALUES (?)", (today,))

    conn.commit()

    conn.close()

# Weekly & Monthly Progress Visualization

def show\_report():

    conn = sqlite3.connect("tasks.db")

    cursor = conn.cursor()

    cursor.execute("SELECT COUNT(DISTINCT entry\_date) FROM tasks")

    days\_completed = cursor.fetchone()[0]

    conn.close()

    plt.bar(["Weekly Progress", "Monthly Progress"], [days\_completed // 7, days\_completed])

    plt.title("Task Completion Report")

    plt.show()

# Add Buttons

add\_button = tk.Button(button\_frame, text="➕ Add Task", command=add\_task, width=15,height=2, font=("Times New Roman", 12, "bold"),bg="black",fg="white")

add\_button.pack(pady=5)

delete\_button = tk.Button(button\_frame, text="❌ Delete Task", command=delete\_task, width=15)

delete\_button.pack(pady=5)

reminder\_button = tk.Button(button\_frame, text="🔔 Send Reminder", command=send\_reminder, width=15)

reminder\_button.pack(pady=5)

report\_button = tk.Button(button\_frame, text="📊 Show Report", command=show\_report, width=15)

report\_button.pack(pady=5)

# ✅ Updated Badge Button to Show Pop-up with Star Image

badge\_button = tk.Button(button\_frame, text="🏆 Show Badge", command=lambda: show\_badge\_popup("🏆 No badge yet! Keep adding tasks daily!"), width=15)

badge\_button.pack(pady=5)

# Run Daily Tracking

update\_daily\_tracking()

# Run the GUI Loop

root.mainloop()