from tkinter import \*  
from tkinter import messagebox  
import sqlite3  
root = Tk()  
# Create PhotoImage objects for images with alternate text  
image1 = PhotoImage(file="C:/Users/Abimb/OneDrive/Desktop/Easylist-logo.png")  
image2 = PhotoImage(file="C:/Users/Abimb/OneDrive/Desktop/EasyList2.png")  
# Create Label widgets for images with alternate text  
image1\_label = Label(root, image=image1, bd=0)  
image1\_label.image = image1  
image1\_label.pack(pady=5)  
image2\_label = Label(root, image=image2, bd=0)  
image2\_label.image = image2  
image2\_label.pack(pady=5)  
root.mainloop()  
  
# declare the global variables  
global list\_name, priority, items\_listbox, current\_list\_id  
  
# create the main window  
root = Tk()  
root.title("EasyList")  
  
# connect to the database  
conn = sqlite3.connect('easylist.db')  
c = conn.cursor()  
  
# create the lists table if it doesn't exist  
c.execute('''CREATE TABLE IF NOT EXISTS lists  
 (id INTEGER PRIMARY KEY AUTOINCREMENT,  
 name TEXT,  
 priority TEXT)''')  
  
# create the items table if it doesn't exist  
c.execute('''CREATE TABLE IF NOT EXISTS items  
 (id INTEGER PRIMARY KEY AUTOINCREMENT,  
 list\_id INTEGER,  
 name TEXT,  
 finished INTEGER,  
 priority INTEGER,  
 FOREIGN KEY(list\_id) REFERENCES lists(id))''')  
  
# function to create a new list  
  
  
def new\_list():  
 global list\_name\_entry, priority\_entry, new\_list\_window  
  
 # create a new window for the new list  
 new\_list\_window = Toplevel(root)  
 new\_list\_window.title("New List")  
  
 # create and position the list name label and entry box  
 list\_name\_label = Label(new\_list\_window, text="List Name:")  
 list\_name\_label.grid(row=0, column=0, padx=5, pady=5)  
  
 list\_name\_entry = Entry(new\_list\_window)  
 list\_name\_entry.grid(row=0, column=1, padx=5, pady=5)  
  
 # create and position the priority label and entry box  
 priority\_label = Label(new\_list\_window, text="Priority Level (Optional):")  
 priority\_label.grid(row=1, column=0, padx=5, pady=5)  
  
 priority\_entry = Entry(new\_list\_window)  
 priority\_entry.grid(row=1, column=1, padx=5, pady=5)  
  
 # create and position the save button  
 save\_button = Button(new\_list\_window, text="Save", command=save\_list)  
 save\_button.grid(row=2, column=0, columnspan=2, padx=5, pady=5)  
  
# function to save the new list and close the new list window  
  
  
def save\_list():  
 global list\_name\_entry, priority\_entry, list\_name, priority, items\_listbox, new\_list\_window, current\_list\_id  
  
 # get the values from the entry boxes  
 list\_name = list\_name\_entry.get()  
 priority = priority\_entry.get()  
  
 # validate the input  
 if not list\_name:  
 messagebox.showerror("Error", "List name cannot be empty.")  
 return  
  
 # close the new list window  
 new\_list\_window.destroy()  
  
 # insert the list into the database  
 c.execute("INSERT INTO lists (name, priority) VALUES (?, ?)",  
 (list\_name, priority))  
 conn.commit()  
  
 # get the id of the new list  
 current\_list\_id = c.lastrowid  
  
 # create a new window for the list  
 list\_window = Toplevel(root)  
 list\_window.title(list\_name)  
  
 # create and position the list name label  
 list\_name\_label = Label(list\_window, text=list\_name)  
 list\_name\_label.grid(row=0, column=0, padx=5, pady=5)  
  
 # create and position the add item button and entry box  
 add\_item\_entry = Entry(list\_window)  
 add\_item\_entry.grid(row=1, column=0, padx=5, pady=5)  
  
 def add\_item():  
 global items\_listbox  
  
 # get the value from the entry box  
 item = add\_item\_entry.get()  
  
 # validate the input  
 if not item:  
 messagebox.showerror("Error", "Item cannot be empty.")  
 return  
  
 # add the item to the database  
 c.execute("INSERT INTO items (list\_id, name, finished, priority) VALUES (?, ?, ?, ?)",  
 (current\_list\_id, item, 0, 0))  
 conn.commit()  
  
 # add the item to the listbox  
 items\_listbox.insert(END, item)  
  
 # clear the entry box  
 add\_item\_entry.delete(0, END)  
  
 add\_item\_button = Button(list\_window, text="Add Item", command=add\_item)  
 add\_item\_button.grid(row=1, column=1, padx=5, pady=5)  
  
 # create and position the items listbox  
 items\_listbox = Listbox(list\_window)  
 items\_listbox.grid(row=2, column=0, columnspan=2, padx=5, pady=5)  
  
 # get the items from the database  
 c.execute(  
 "SELECT name, finished, priority FROM items WHERE list\_id=?", (current\_list\_id,))  
 items = c.fetchall()  
  
 # add the items to the listbox  
 for item in items:  
 name = item[0]  
 finished = item[1]  
 priority = item[2]  
 if finished:  
 items\_listbox.insert(END, name + " (finished)")  
 else:  
 items\_listbox.insert(END, name)  
  
 # create and position the remove item button  
 def remove\_item():  
 global items\_listbox  
  
 # get the selected item  
 selected\_item = items\_listbox.curselection()  
 if not selected\_item:  
 messagebox.showerror("Error", "Please select an item to remove.")  
 return  
 item = items\_listbox.get(selected\_item[0])  
  
 # remove the item from the database  
 c.execute("DELETE FROM items WHERE list\_id=? AND name=?",  
 (current\_list\_id, item))  
 conn.commit()  
  
 # remove the item from the listbox  
 items\_listbox.delete(selected\_item[0])  
  
 remove\_item\_button = Button(  
 list\_window, text="Remove Item", command=remove\_item)  
 remove\_item\_button.grid(row=3, column=0, padx=5, pady=5)  
  
 # create and position the mark finished button  
 def mark\_finished():  
 global items\_listbox  
  
 # get the selected item  
 selected\_item = items\_listbox.curselection()  
 if not selected\_item:  
 messagebox.showerror(  
 "Error", "Please select an item to mark as finished.")  
 return  
 item = items\_listbox.get(selected\_item[0])  
  
 # mark the item as finished in the database  
 c.execute("UPDATE items SET finished=1 WHERE list\_id=? AND name=?",  
 (current\_list\_id, item))  
 conn.commit()  
  
 # add the "finished" tag to the item in the listbox  
 items\_listbox.delete(selected\_item[0])  
 items\_listbox.insert(selected\_item[0], item + " (finished)")  
  
 mark\_finished\_button = Button(  
 list\_window, text="Mark Finished", command=mark\_finished)  
 mark\_finished\_button.grid(row=3, column=1, padx=5, pady=5)  
  
 # create and position the mark unfinished button  
 def mark\_unfinished():  
 mark\_unfinished\_button = Button(  
 list\_window, text="Mark Unfinished", command=mark\_unfinished)  
 mark\_unfinished\_button.grid(row=4, column=0, padx=5, pady=5)  
  
 # create and position the sort by name button  
 def sort\_by\_name():  
 global items\_listbox  
  
 # get the items from the database, sorted by name  
 c.execute(  
 "SELECT name, finished, priority FROM items WHERE list\_id=? ORDER BY name", (current\_list\_id,))  
 items = c.fetchall()  
  
 # clear the listbox  
 items\_listbox.delete(0, END)  
  
 # add the items to the listbox  
 for item in items:  
 name = item[0]  
 finished = item[1]  
 priority = item[2]  
 if finished:  
 items\_listbox.insert(END, name + " (finished)")  
 else:  
 items\_listbox.insert(END, name)  
  
 sort\_by\_name\_button = Button(  
 list\_window, text="Sort by Name", command=sort\_by\_name)  
 sort\_by\_name\_button.grid(row=4, column=1, padx=5, pady=5)  
  
 # create and position the back to menu button  
  
 def back\_to\_menu():  
 list\_window.destroy()  
  
 back\_to\_menu\_button = Button(  
 list\_window, text="Back to Menu", command=back\_to\_menu)  
 back\_to\_menu\_button.grid(row=6, column=0, columnspan=2, padx=5, pady=5)  
  
  
# function to show the items in the selected list  
def show\_items(list\_id):  
 global items\_listbox  
  
 # create a new window for the list items  
 list\_window = Toplevel(root)  
 list\_window.title("List Items")  
  
 # # create and position the listbox  
 # items\_listbox = Listbox(list\_window)  
 # items\_listbox.grid(row=1, column=0, padx=5, pady=5)  
  
 # get the items from the database for the selected list  
 c.execute("SELECT name, finished FROM items WHERE list\_id=?", (list\_id,))  
 items = c.fetchall()  
  
 # # add the items to the listbox  
 # for item in items:  
 # name = item[0]  
 # finished = item[1]  
 # items\_listbox.insert(END, name + (" (Finished)" if finished else ""))  
  
 add\_item\_entry = Entry(list\_window)  
 add\_item\_entry.grid(row=1, column=0, padx=5, pady=5)  
 # create and position the add item button  
  
 def add\_item():  
 global items\_listbox  
  
 # get the value from the entry box  
 item = add\_item\_entry.get()  
  
 # validate the input  
 if not item:  
 messagebox.showerror("Error", "Item cannot be empty.")  
 return  
  
 # add the item to the database  
 c.execute("INSERT INTO items (list\_id, name, finished, priority) VALUES (?, ?, ?, ?)",  
 (current\_list\_id, item, 0, 0))  
 conn.commit()  
  
 # add the item to the listbox  
 items\_listbox.insert(END, item)  
  
 # clear the entry box  
 add\_item\_entry.delete(0, END)  
  
 add\_item\_button = Button(list\_window, text="Add Item", command=add\_item)  
 add\_item\_button.grid(row=1, column=1, padx=5, pady=5)  
  
 # create and position the items listbox  
 items\_listbox = Listbox(list\_window)  
 items\_listbox.grid(row=2, column=0, columnspan=2, padx=5, pady=5)  
  
 # get the items from the database  
 c.execute(  
 "SELECT name, finished, priority FROM items WHERE list\_id=?", (current\_list\_id,))  
 items = c.fetchall()  
  
 # add the items to the listbox  
 for item in items:  
 name = item[0]  
 finished = item[1]  
 if finished:  
 items\_listbox.insert(END, name + " (finished)")  
 else:  
 items\_listbox.insert(END, name)  
  
 # create and position the remove item button  
 def remove\_item():  
 global items\_listbox  
  
 # get the selected item  
 selected\_item = items\_listbox.curselection()  
 if not selected\_item:  
 messagebox.showerror("Error", "Please select an item to remove.")  
 return  
 item = items\_listbox.get(selected\_item[0])  
  
 # remove the item from the database  
 c.execute("DELETE FROM items WHERE list\_id=? AND name=?",  
 (current\_list\_id, item))  
 conn.commit()  
  
 # remove the item from the listbox  
 items\_listbox.delete(selected\_item[0])  
  
 remove\_item\_button = Button(  
 list\_window, text="Remove Item", command=remove\_item)  
 remove\_item\_button.grid(row=3, column=0, padx=5, pady=5)  
  
 # create and position the mark finished button  
 def mark\_finished():  
 global items\_listbox  
  
 # get the selected item  
 selected\_item = items\_listbox.curselection()  
 if not selected\_item:  
 messagebox.showerror(  
 "Error", "Please select an item to mark as finished.")  
 return  
 item = items\_listbox.get(selected\_item[0])  
  
 # mark the item as finished in the database  
 c.execute("UPDATE items SET finished=1 WHERE list\_id=? AND name=?",  
 (current\_list\_id, item))  
 conn.commit()  
  
 # add the "finished" tag to the item in the listbox  
 items\_listbox.delete(selected\_item[0])  
 items\_listbox.insert(selected\_item[0], item + " (finished)")  
  
 mark\_finished\_button = Button(  
 list\_window, text="Mark Finished", command=mark\_finished)  
 mark\_finished\_button.grid(row=3, column=1, padx=5, pady=5)  
  
 # create and position the mark unfinished button  
 def mark\_unfinished():  
 global items\_listbox  
  
 # get the selected item  
 selected\_item = items\_listbox.curselection()  
 if not selected\_item:  
 messagebox.showerror(  
 "Error", "Please select an item to mark as unfinished.")  
 return  
 item = items\_listbox.get(selected\_item[0])  
  
 # mark the item as finished in the database  
 c.execute("UPDATE items SET finished=1 WHERE list\_id=? AND name=?",  
 (current\_list\_id, item))  
 conn.commit()  
  
 # add the "finished" tag to the item in the listbox  
 items\_listbox.delete(selected\_item[0])  
 items\_listbox.insert(selected\_item[0], item)  
  
 mark\_unfinished\_button = Button(  
 list\_window, text="Mark Unfinished", command=mark\_unfinished)  
 mark\_unfinished\_button.grid(row=4, column=0, padx=5, pady=5)  
  
 # create and position the sort by name button  
 def sort\_by\_name():  
 global items\_listbox  
  
 # get the items from the database, sorted by name  
 c.execute(  
 "SELECT name, finished, priority FROM items WHERE list\_id=? ORDER BY name", (current\_list\_id,))  
 items = c.fetchall()  
  
 # clear the listbox  
 items\_listbox.delete(0, END)  
  
 # add the items to the listbox  
 for item in items:  
 name = item[0]  
 finished = item[1]  
 if finished:  
 items\_listbox.insert(END, name + " (finished)")  
 else:  
 items\_listbox.insert(END, name)  
  
 sort\_by\_name\_button = Button(  
 list\_window, text="Sort by Name", command=sort\_by\_name)  
 sort\_by\_name\_button.grid(row=4, column=1, padx=5, pady=5)  
  
 # create and position the back to menu button  
  
 def back\_to\_menu():  
 list\_window.destroy()  
  
 back\_to\_menu\_button = Button(  
 list\_window, text="Back to Menu", command=back\_to\_menu)  
 back\_to\_menu\_button.grid(row=6, column=0, columnspan=2, padx=5, pady=5)  
  
  
# function to show the user's lists  
def show\_lists():  
 global list\_select\_listbox  
  
 # create a new window for the list selection  
 lists\_window = Toplevel(root)  
 lists\_window.title("Lists")  
  
 # create and position the label  
 list\_select\_label = Label(lists\_window, text="Select a List:")  
 list\_select\_label.grid(row=0, column=0, padx=5, pady=5)  
  
 # create and position the listbox  
 list\_select\_listbox = Listbox(lists\_window)  
 list\_select\_listbox.grid(row=1, column=0, padx=5, pady=5)  
  
 # get the lists from the database  
 c.execute("SELECT name FROM lists")  
 lists = c.fetchall()  
  
 # add the lists to the listbox  
 for lst in lists:  
 list\_select\_listbox.insert(END, lst[0])  
  
 # create and position the open list button  
 def open\_list():  
 global current\_list\_id  
  
 # get the selected list  
 selected\_list = list\_select\_listbox.curselection()  
 if not selected\_list:  
 messagebox.showerror("Error", "Please select a list to open.")  
 return  
 list\_name = list\_select\_listbox.get(selected\_list[0])  
  
 # get the list id from the database  
 c.execute("SELECT id FROM lists WHERE name=?", (list\_name,))  
 row = c.fetchone()  
 if not row:  
 messagebox.showerror("Error", "List not found.")  
 return  
 current\_list\_id = row[0]  
  
 # destroy the lists window  
 lists\_window.destroy()  
  
 # show the items in the selected list  
 show\_items(current\_list\_id)  
  
 open\_list\_button = Button(  
 lists\_window, text="Open List", command=open\_list)  
 open\_list\_button.grid(row=2, column=0, padx=5, pady=5)  
  
 # create and position the new list button  
 def new\_list():  
 # create a new window for the new list  
 new\_list\_window = Toplevel(lists\_window)  
 new\_list\_window.title("New List")  
  
 # create and position the name label and entry box  
 new\_list\_label = Label(new\_list\_window, text="Name:")  
 new\_list\_label.grid(row=0, column=0, padx=5, pady=5)  
 new\_list\_entry = Entry(new\_list\_window)  
 new\_list\_entry.grid(row=0, column=1, padx=5, pady=5)  
  
 # create and position the priority label and entry box  
 new\_list\_priority\_label = Label(new\_list\_window, text="Priority:")  
 new\_list\_priority\_label.grid(row=1, column=0, padx=5, pady=5)  
 new\_list\_priority\_entry = Entry(new\_list\_window)  
 new\_list\_priority\_entry.grid(row=1, column=1, padx=5, pady=5)  
  
 # create and position the create button  
 def create\_list():  
 # get the values from the entry boxes  
 name = new\_list\_entry.get()  
 priority = new\_list\_priority\_entry.get()  
  
 # validate the input  
 if not name:  
 messagebox.showerror("Error", "Name cannot be empty.")  
 return  
  
 # add the list to the database  
 c.execute(  
 "INSERT INTO lists (name, priority) VALUES (?, ?)", (name, priority))  
 conn.commit()  
  
 # destroy the new list window  
 new\_list\_window.destroy()  
  
 # update the listbox  
 list\_select\_listbox.insert(END, name)  
 if priority:  
 list\_select\_listbox.insert(  
 END, f"{name} (priority {priority})")  
 else:  
 list\_select\_listbox.insert(END, name)  
  
 create\_list\_button = Button(  
 new\_list\_window, text="Create", command=create\_list)  
 create\_list\_button.grid(row=2, column=0, columnspan=2, padx=5, pady=5)  
  
 new\_list\_button = Button(lists\_window, text="New List", command=new\_list)  
 new\_list\_button.grid(row=3, column=0, padx=5, pady=5)  
  
  
# create and position the main menu buttons  
new\_list\_button = Button(root, text="New List", command=new\_list)  
new\_list\_button.grid(row=0, column=0, padx=5, pady=5)  
  
show\_lists\_button = Button(root, text="Show Lists", command=show\_lists)  
show\_lists\_button.grid(row=0, column=1, padx=5, pady=5)  
  
# create and position the exit button  
  
  
def exit\_program():  
 root.destroy()  
  
  
exit\_button = Button(root, text="Exit", command=exit\_program)  
exit\_button.grid(row=1, column=0, padx=5, pady=5)  
  
root.mainloop()