```
"""
Ryan Kennedy, Gabriel Waldner
Cmdr. Schenk
Cloud Computing
7th Period
May 5, 2025
"""

import tkinter as tk
from tkinter import Canvas, ttk
from tkinter import messagebox
from PIL import ImageTk, Image

from user_record import UserRecord
from task_record import TaskRecord
from database import Database
from gui_states import GUIStates

class TaskBrowser:

    def __init__(self, root, db):
        self.frame = tk.Frame(root)
        self.db = db
        self.user = UserRecord()
        self.tasks = []
        self.selected_task = 0 # index into self.tasks
        self.filename= 'src/med.jpeg'
        self.is_inserting = False  # Track if we are in insert mode
        #defining the image so it can be placed later on:
        self.my_img = ImageTk.PhotoImage(Image.open(self.filename).resize((500, 550)
))

    def init_resources(self):
        self.canvasMain = Canvas(self.frame, width=1000, height=1000)
        #Places the image on the canvas:
        self.canvasMain.create_image(260, 100, image=self.my_img , anchor="nw") #ima
ge=self.my_img,

        #CREATING LABELS AND ENTIRES FOR CREATE A LOGIN SCREEN, DONE BY GABRIEL WALD
NER
        #Placing the canvas:

        # Style setup (once, in __init__ or init_resources)
        style = ttk.Style()
        style.theme_use('clam')  # Options: 'clam', 'alt', 'default', 'vista'
        style.configure("TButton", font=("Segoe UI", 10), padding=6)

        # Title Label
        style.configure("TitleLabel.TLabel", font=("Segoe UI", 20, "italic"))

        style.configure("HeaderLabel.TLabel", font=("Segoe UI", 12, "italic"))


        # Field Labels
        style.configure("FieldLabel.TLabel", font=("Segoe UI", 10), foreground="#444
")

        # Field Entry
        style.configure("Field.TEntry", font=("Segoe UI", 20), padding=10)

        # Info Buttons
        style.configure("Info.TButton", font=("Segoe UI", 7), padding=10)

        # Action Buttons
        style.configure("Action.TButton", font=("Segoe UI", 10), padding=10)

        # Delete Buttons
        style.configure("Delete.TButton", font=("Segoe UI", 20), padding=10, backgro
und="#d32f2f")

        ttk.Label(self.frame, text="Task Browser", style="TitleLabel.TLabel").place(
x=50, y = 20)
```

```
        # gui state change buttons

        ttk.Button(self.frame, text="Goto User Info", command=self.goto_user_info, s
tyle="Info.TButton").place(x=500, y=20)
        ttk.Button(self.frame, text="Logout", command=self.logout, style="Info.TButt
on").place(x=650, y=20)

        # entries and their respective labels

        self.task_num_lbl = tk.Label(self.frame)
        self.task_num_lbl.place(x=50, y=100)

        ttk.Label(self.frame, text="Short Name:", style="HeaderLabel.TLabel").place(
x=50, y=200)
        self.short_name_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.short_name_ent.place(x=50, y=250)

        ttk.Label(self.frame, text="Description:" , style="HeaderLabel.TLabel").plac
e(x=50, y=350)
        self.description_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.description_ent.place(x=50, y=400)

        # CRUD buttons
        ttk.Button(self.frame, text="New", command=self.add_record, style="Info.TBut
ton").place(x=75, y=500)
        ttk.Button(self.frame, text="Update", command=self.update_record, style="Inf
o.TButton").place(x=75, y=600)
        ttk.Button(self.frame, text="Delete", command=self.delete_record, style="Inf
o.TButton").place(x=75, y=700)

        # nav buttons
        ttk.Button(self.frame, text="<", command=lambda:self.increment_selected_task
(-1), style="Info.TButton").place(x=415, y=700)
        ttk.Button(self.frame, text="<<", command=lambda:self.increment_selected_tas
k(-3), style="Info.TButton").place(x=335, y=700)
        ttk.Button(self.frame, text="|<", command=lambda:self.increment_selected_tas
k(-1 * self.selected_task), style="Info.TButton").place(x=265, y=700)
        ttk.Button(self.frame, text=">", command=lambda:self.increment_selected_task
(1), style="Info.TButton").place(x=500, y=700)
        ttk.Button(self.frame, text=">>", command=lambda:self.increment_selected_tas
k(3), style="Info.TButton").place(x=575, y=700)
        ttk.Button(self.frame, text=">|", command=lambda:self.increment_selected_tas
k(len(self.tasks) - 1 - self.selected_task), style="Info.TButton").place(x=655, y=70
0)

    def show(self, id):
        self.canvasMain.pack()
        self.user = self.db.users_get_record_by_id(id)
        self.selected_task = 0
        self.refresh()

        self.frame.place(x=0, y=0, width=1000, height=1000)

    def add_record(self):
        if not self.is_inserting:
            # Clear fields and change button text to "Insert"
            self.short_name_ent.delete(0, "end")
            self.description_ent.delete(0, "end")
            self.is_inserting = True
            self.createBtn = ttk.Button(self.frame, text="Insert", command=self.add_
record, style="Info.TButton").place(x=75, y=500)

        else:
            # Insert the record and refresh
            task = TaskRecord()
            task.fill(id=-1, user_id=self.user.id, short_name=self.short_name_ent.ge
t(), description=self.description_ent.get())
            self.db.tasks_insert(task)
            self.refresh()
            self.is_inserting = False
            self.createBtn = ttk.Button(self.frame, text="New", command=self.add_rec
ord, style="Info.TButton").place(x=75, y=500)
            # Select and display the newly added task
```

```
            self.selected_task = len(self.tasks) - 1
            self.display_record()


    def update_record(self):
        if (self.selected_task == -1):
            messagebox.showerror("Can't update a nonexistent record.", "Can't update
 a nonexistent record.");
            return

        task = TaskRecord()
        task.fill(id=self.tasks[self.selected_task].id, user_id=self.tasks[self.sele
cted_task].user_id, short_name=self.short_name_ent.get(), description=self.descripti
on_ent.get())

        result = messagebox.askquestion("Task Update Confirmation", "Are you sure th
at you would like to update this task?\n\n"+self.tasks[self.selected_task].to_string
()+"\n\nTo This Task\n\n"+task.to_string())
        if (result != "yes"):
            return


        self.db.tasks_update(task)
        self.refresh()

    def delete_record(self):
        if (self.selected_task == -1):
            messagebox.showerror("Can't delete a nonexistent record.", "Can't delete
 a nonexistent record.");
            return

        result = messagebox.askquestion("Task Delete Confirmation", "Are you sure th
at you would like to delete this task?\n\n"+self.tasks[self.selected_task].to_string
())
        if (result != "yes"):
            return

        self.db.tasks_delete(self.tasks[self.selected_task].id)

        self.refresh()

    def refresh(self):
        self.tasks = self.db.tasks_get_all_records_with_user_id(self.user.id)

        self.increment_selected_task(0) # bounds check self.selected_task

        self.display_record()

    def increment_selected_task(self, amt):
        self.selected_task += amt

        # check if there are no tasks
        if (len(self.tasks) == 0):
            self.selected_task = -1
            return

        # cap to within bounds

        if (self.selected_task >= len(self.tasks)):
            self.selected_task = len(self.tasks) - 1

        if (self.selected_task < 0):
            self.selected_task = 0

        self.display_record()


    def display_record(self):
        self.short_name_ent.delete(0, "end")
        self.description_ent.delete(0, "end")

        task = TaskRecord()
        if (self.selected_task == -1):
```

```
            task.fill(id = -1, user_id = self.user.id, short_name = "", description
= "")
            self.task_num_lbl["text"] = "Task #: {}".format("No Tasks")
        else:
            task = self.tasks[self.selected_task]
            self.task_num_lbl["text"] = "Task #: {}".format(self.selected_task + 1)

        self.short_name_ent.insert(0, task.short_name)
        self.description_ent.insert(0, task.description)

    def logout(self):
        self.goto_login()

    def goto_login(self):
        self.hide()
        self.show_map[GUIStates.LOGIN_USER]()

    def goto_user_info(self):
        self.hide()
        self.show_map[GUIStates.USER_INFO](self.user.id)

    def hide(self):
        self.frame.place_forget()
        self.canvasMain.pack_forget()

    def assign_show_map(self, show_map):
        self.show_map = show_map
```