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

from enum import Enum

class GUIStates(Enum):
    REGISTER_USER = 0
    LOGIN_USER = 1
    USER_INFO = 2
    TASK_BROWSER = 3
```

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

class TaskRecord:

    def __init__(self):
        self.id = -1
        self.user_id = -1
        self.short_name = "blank"
        self.description = "blank"

    def fill(self, id, user_id, short_name, description):
        self.id = id
        self.user_id = user_id
        self.short_name = short_name
        self.description = description

    def to_string(self):
        result = "id: {}".format(self.id)
        result += "\nuser_id: {}".format(self.user_id)
        result += "\nshort_name: {}".format(self.short_name)
        result += "\ndescription: {}".format(self.description)
        return result
```

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


# CREATE TABLE users (
#        id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL UNIQUE,
#        name TEXT NOT NULL,
#        username TEXT NOT NULL,
#        hashed_password TEXT NOT NULL,
# );

class UserRecord:

    def __init__(self):
        self.id = -1
        self.name = "blank"
        self.username = "blank"
        self.hashed_password = "blank"

    def fill(self, id, name, username, hashed_password):
        self.id = id
        self.name = name
        self.username = username
        self.hashed_password = hashed_password

    def to_string(self):
        result = "id: {}".format(self.id)
        result += "\nname: {}".format(self.name)
        result += "\nusername: {}".format(self.username)
        result += "\nhashed_password: {}".format(self.hashed_password)
        return result
```

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

import mysql.connector
import random

from user_record import UserRecord
from task_record import TaskRecord

# CREATE DATABASE RyanKennedyAndGabrielWaldner;
# USE RyanKennedyAndGabrielWaldner;
# CREATE TABLE users (
#         id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL UNIQUE,
#         name TEXT NOT NULL,
#         username TEXT NOT NULL,
#         hashed_password TEXT NOT NULL
# );

# CREATE TABLE tasks (
#         id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL UNIQUE,
#         user_id INTEGER NOT NULL,
#         short_name TEXT NOT NULL,
#         description TEXT,
#         FOREIGN KEY(user_id) REFERENCES users(id)
# );

class Database():

    def __init__(self):
        # self.conn = mysql.connector.connect(host = "192.168.0.100", user = "student"
, passwd = "jchs", database = "RyanKennedyAndGabrielWaldner")
        # self.conn = mysql.connector.connect(host = "127.0.0.1", user = "root", passw
d = "#Whalez17", database = "RyanKennedyAndGabrielWaldner")
        # self.conn = mysql.connector.connect(host = "127.0.0.1", user = "root", passw
d = "mysqlpassword", database = "RyanKennedyAndGabrielWaldner")
        self.conn = mysql.connector.connect(host = "127.0.0.1", user = "root", passwd
= "ryansmiles", database = "RyanKennedyAndGabrielWaldner")
        self.cursor = self.conn.cursor()

    def close(self):
        self.conn.close()

    def users_get_all_records(self):
        # get raw data
        self.cursor.execute("SELECT id, name, username, hashed_password FROM users;");
        arr_data = self.cursor.fetchall()

        result = []

        # return empty array if there is no data to pack into array
        if(len(arr_data) == 0):
            return result

        # pack into array of BookRecord for easier access
        for record in arr_data:
            rec = UserRecord()
            rec.fill(id=int(record[0]), name=str(record[1]), username=str(record[2]),
hashed_password=str(record[3]))
            result.append(rec)

        return result

    def users_get_record_by_id(self, id):
        self.cursor.execute("SELECT id, name, username, hashed_password FROM users WHE
RE id = {};".format(id));
        arr_data = self.cursor.fetchall()

        if (len(arr_data) == 0):
            return UserRecord()

        record = arr_data[0] # db record
        rec = UserRecord() # function result of type UserRecord
        rec.fill(id=int(record[0]), name=str(record[1]), username=str(record[2]), hash
ed_password=str(record[3]))
        return rec

    def users_get_records_with_username_and_hashed_password(self, username, hashed_pas
sword):
        # get raw data
        self.cursor.execute("SELECT id, name, username, hashed_password FROM users WHE
RE username = '{}' AND hashed_password = '{}';".format(username, hashed_password))
        arr_data = self.cursor.fetchall()

        result = []

        # return empty array if there is no data to pack into array
        if(len(arr_data) == 0):
            return result

        # pack into array of BookRecord for easier access
        for record in arr_data:
            rec = UserRecord()
            rec.fill(id=int(record[0]), name=str(record[1]), username=str(record[2]),
hashed_password=str(record[3]))
            result.append(rec)

        return result


    def users_insert(self, record):
        query = "INSERT INTO users (name, username, hashed_password) VALUES ('{}', '{}
', '{}');".format(record.name, record.username, record.hashed_password)
        self.cursor.execute(query)
        self.conn.commit()

    def users_update(self, record):
        query = "UPDATE users SET name = '{}', username = '{}', hashed_password = '{}'
 WHERE id = {};".format(record.name, record.username, record.hashed_password, str(reco
rd.id))
        self.cursor.execute(query)
        self.conn.commit()

    def users_delete(self, id):
        # delete books that reference the author then delete the author
        self.cursor.execute("DELETE FROM tasks WHERE user_id = {};".format(str(id)))
        self.cursor.execute("DELETE FROM users WHERE id = {};".format(str(id)))
        self.conn.commit()

    # ================== TASKS STUFF =======================

    def tasks_get_all_records_with_user_id(self, user_id):
        # get raw data
        self.cursor.execute("SELECT id, user_id, short_name, description FROM tasks WH
ERE user_id = {};".format(user_id))
        arr_data = self.cursor.fetchall()

        result = []

        # return empty array if there is no data to pack into array
        if(len(arr_data) == 0):
            return result

        # pack into array of BookRecord for easier access
        for record in arr_data:
            rec = TaskRecord()
            rec.fill(id=int(record[0]), user_id=int(record[1]), short_name=str(record[
2]), description=str(record[3]));
            result.append(rec)

        return result

    def tasks_insert(self, record):
        self.cursor.execute("INSERT INTO tasks (user_id, short_name, description) VALU
ES ({}, '{}', '{}');".format(str(record.user_id), record.short_name, record.descriptio
n))
        self.conn.commit()

    def tasks_update(self, record):
        self.cursor.execute("UPDATE tasks SET user_id = {}, short_name = '{}', descrip
tion = '{}' WHERE id = {};".format(str(record.user_id), record.short_name, record.desc
ription, str(record.id)))
        self.conn.commit()

    def tasks_delete(self, id):
        self.cursor.execute("DELETE FROM tasks WHERE id = {};".format(id))
        self.conn.commit()
```

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

import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

import bcrypt

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

class LoginUser:

    def __init__(self, root, db):
        self.frame = tk.Frame(root)
        self.db = db

    def init_resources(self):
        style = ttk.Style()
        style.theme_use('clam')  # Try also: 'alt', 'vista', etc.

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

        # Field Labels
        style.configure("FieldLabel.TLabel", font=("Segoe UI", 20), foreground="#444")

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

        # Action Buttons
        style.configure("Action.TButton", font=("Segoe UI", 20), padding=10)



        ttk.Label(self.frame, text="User Login", style="TitleLabel.TLabel").place(x=40
0, y=30, anchor="center")

        # Username label
        ttk.Label(self.frame, text="Username:", style="FieldLabel.TLabel").place(x=100
, y=150)
        self.username_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.username_ent.place(x=100, y=200, width=600, height=50)

        # Password
        ttk.Label(self.frame, text="Password:", style="FieldLabel.TLabel").place(x=100
, y=375)
        self.password_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.password_ent.place(x=100, y=425, width=600, height=50)

        #Buttonss
        ttk.Button(self.frame, text="Login", command=self.login).place(x=425, y=600, w
idth=225, height=75)
        ttk.Button(self.frame, text="Register", command=self.goto_register_user).place
(x=125, y=600, width=225, height=75)

    def login(self):
        record = UserRecord()
        password = self.password_ent.get()
        hashed_password = bcrypt.hashpw(password.encode(), b'$2b$12$zm4/D56Ntli/hWPKnm
LSgu')
```

```
        record.fill(-1, "", self.username_ent.get(), hashed_password.decode("utf-8"))

        matched_users = self.db.users_get_records_with_username_and_hashed_password(re
cord.username, record.hashed_password)

        if (len(matched_users) == 0):
            tk.messagebox.showerror("Failed to Login", "Invalid Login Credentials");
            return

        self.hide()
        self.show_map[GUIStates.USER_INFO](matched_users[0].id)

    def goto_register_user(self):
        self.hide()
        self.show_map[GUIStates.REGISTER_USER]()

    def clear_entries(self):
        self.username_ent.delete(0, "end")
        self.password_ent.delete(0, "end")

    def show(self):
        self.clear_entries()
        self.frame.place(x=0,y=0,width=1000,height=1000)

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

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

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

import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

import bcrypt

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

class RegisterUser:

    def __init__(self, root, db):
        self.frame = tk.Frame(root)
        self.db = db

    def init_resources(self):
        style = ttk.Style()
        style.theme_use('clam')  # Try also: 'alt', 'vista', etc.

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

        # Feedback Label
        style.configure("FeedbackLabel.TLabel", font=("Segoe UI", 9, "italic"))

        # Field Labels
        style.configure("FieldLabel.TLabel", font=("Segoe UI", 20), foreground="#444")

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

        # Action Buttons
        style.configure("Action.TButton", font=("Segoe UI", 20), padding=10)



        ttk.Label(self.frame, text="User Registration", style="TitleLabel.TLabel").pla
ce(x=400, y=30, anchor="center")

        self.feedback_lbl = ttk.Label(self.frame, text="", style="FeedbackLabel.TLabel
")
        self.feedback_lbl.place(x=50, y=30)

        # name
        ttk.Label(self.frame, text="Name:", style="FieldLabel.TLabel").place(x=100, y=
125)
        self.name_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.name_ent.place(x=100, y=175, width=600, height=50)

        # Username label
        ttk.Label(self.frame, text="Username:", style="FieldLabel.TLabel").place(x=100
, y=260)
        self.username_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.username_ent.place(x=100, y=310, width=600, height=50)

        # Password
        ttk.Label(self.frame, text="Password:", style="FieldLabel.TLabel").place(x=100
, y=400)
        self.password_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.password_ent.place(x=100, y=450, width=600, height=50)

        # buttons
        ttk.Button(self.frame, text="Register", command=self.register_user, style="Act
ion.TButton").place(x=425, y=600, width=225, height=75)
        ttk.Button(self.frame, text="Goto User Login", command=self.goto_login_user, s
tyle="Action.TButton").place(x=125, y=600, width=225, height=75)

    def register_user(self):
        record = UserRecord()
        password = self.password_ent.get()
        hashed_password = bcrypt.hashpw(password.encode(), b'$2b$12$zm4/D56Ntli/hWPKnm
LSgu')
        record.fill(-1, self.name_ent.get(), self.username_ent.get(), hashed_password.
decode("utf-8"))
        self.db.users_insert(record)

        self.goto_login_user()

    def goto_login_user(self):
        self.frame.place_forget()
        self.show_map[GUIStates.LOGIN_USER]()

    def clear_entries(self):
        self.name_ent.delete(0, "end")
        self.username_ent.delete(0, "end")
        self.password_ent.delete(0, "end")

    def show(self):
        self.clear_entries()
        self.frame.place(x=0,y=0,width=1000,height=1000)

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

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

```python
"""
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") #image
=self.my_img,

        #CREATING LABELS AND ENTIRES FOR CREATE A LOGIN SCREEN, DONE BY GABRIEL WALDNE
R
        #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, backgroun
d="#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, sty
le="Info.TButton").place(x=500, y=20)
        ttk.Button(self.frame, text="Logout", command=self.logout, style="Info.TButton
").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").place(
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.TButto
n").place(x=75, y=500)
        ttk.Button(self.frame, text="Update", command=self.update_record, style="Info.
TButton").place(x=75, y=600)
        ttk.Button(self.frame, text="Delete", command=self.delete_record, style="Info.
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_task(
-3), style="Info.TButton").place(x=335, y=700)
        ttk.Button(self.frame, text="|<", command=lambda:self.increment_selected_task(
-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_task(
3), style="Info.TButton").place(x=575, y=700)
        ttk.Button(self.frame, text=">|", command=lambda:self.increment_selected_task(
len(self.tasks) - 1 - self.selected_task), style="Info.TButton").place(x=655, y=700)

    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_re
cord, 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.get(
), 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_recor
d, 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.select
ed_task].user_id, short_name=self.short_name_ent.get(), description=self.description_e
nt.get())

        result = messagebox.askquestion("Task Update Confirmation", "Are you sure that
 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 that
 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
```

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

import sys
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

import bcrypt

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

class UserInfo:

    def __init__(self, root, db):
        self.frame = tk.Frame(root)
        self.line_canvas = tk.Canvas(self.frame, width=800, height=800, bg="white", hi
ghlightthickness=0)
        self.db = db
        self.user = UserRecord() # stores the user that we are currently displaying th
e information of, currently blank but gets set in self.show(...)

    def init_resources(self):
        style = ttk.Style()
        style.theme_use('clam')

        # 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",
 background="white")

        # Field Entry
        style.configure("Field.TEntry", font=("Segoe UI", 7), 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, backgroun
d="#d32f2f")



        ttk.Label(self.frame, text="Account Info", style="TitleLabel.TLabel").place(x=
400, y=30, anchor="center")

        # change name

        ttk.Label(self.frame, text="Change Name", style="HeaderLabel.TLabel").place(x=
20,y=60)

        ttk.Label(self.frame, text="Name:", style="FieldLabel.TLabel").place(x=100, y=
125)
        self.name_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.name_ent.place(x=100, y=150)

        ttk.Button(self.frame, text="Save", command=self.change_name, style="Info.TBut
ton").place(x=100, y=225, width=150, height=50)

        # change credentials

        ttk.Label(self.frame, text="Change Credentials", style="HeaderLabel.TLabel").p
lace(x=20, y=350)

        ttk.Label(self.frame, text="Username:", style="FieldLabel.TLabel").place(x=100
, y=400)
        self.username_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.username_ent.place(x=100, y=425)

        ttk.Label(self.frame, text="New Password:", style="FieldLabel.TLabel").place(x
=100, y=500)
        self.password_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.password_ent.place(x=100, y=525)

        ttk.Label(self.frame, text="Old Password:", style="FieldLabel.TLabel").place(x
=100, y=600)
        self.old_password_ent = ttk.Entry(self.frame, style="Field.TEntry")
        self.old_password_ent.place(x=100, y=625)

        ttk.Button(self.frame, text="Save", command=self.change_credentials, style="In
fo.TButton").place(x=100, y=700, width=150, height=50)

        # buttons for action

        ttk.Button(self.frame, text="Goto Task Browser", command=self.goto_task_browse
r, style="Action.TButton").place(x=500, y=80, width=225, height=100)
        ttk.Button(self.frame, text="Logout", command=self.logout, style="Action.TButt
on").place(x=500,y=280, width=225, height=100)
        ttk.Button(self.frame, text="Quit App", style="Action.TButton", command=lambda
:sys.exit(0)).place(x=500, y=466,width=225, height=100)
        ttk.Button(self.frame, text="Delete Account", command=self.delete_user, style=
"Delete.TButton").place(x=500 - 8, y=650, width=240, height=100)


    def show(self, id):
        self.clear_entries()
        self.user = self.db.users_get_record_by_id(id)
        self.name_ent.insert(0, self.user.name)
        self.username_ent.insert(0, self.user.username)
        self.line_canvas.place(x=0,y=0)
        self.frame.place(x=0,y=0,width=800,height=800)

        self.line_canvas.create_line(400, 75, 400, 800, fill="black", width=5)

    def delete_user(self):
        result = messagebox.askquestion("Account Deletion Confirmation", "Are you sure
 that you would like to delete your account?")
        if (result != "yes"):
            return

        self.db.users_delete(self.user.id)
        self.goto_login()

    def change_credentials(self):
        result = messagebox.askquestion("Change Credentials Confirmation", "Are you su
re that you would like to update your credentials?")
        if (result != "yes"):
            return

        old_password = self.old_password_ent.get()
        hashed_old_password = bcrypt.hashpw(old_password.encode(), b'$2b$12$zm4/D56Ntl
i/hWPKnmLSgu').decode("utf-8")
```

```
        if (hashed_old_password != self.user.hashed_password):
            tk.messagebox.showerror("Failed to change credentials", "Old Password is I
ncorrect")
            return

        password = self.password_ent.get()
        hashed_password = bcrypt.hashpw(password.encode(), b'$2b$12$zm4/D56Ntli/hWPKnm
LSgu').decode("utf-8")

        record = UserRecord()
        record.fill(id=self.user.id, name=self.name_ent.get(), username=self.username_
ent.get(), hashed_password=hashed_password)
        self.db.users_update(record)
        self.user = self.db.users_get_record_by_id(self.user.id)
        self.update_entries()
        tk.messagebox.showinfo("Successfully Updated Credentials", "Successfully Updat
ed Credentials")

    def change_name(self):
        record = UserRecord()
        record.fill(id=self.user.id, name=self.name_ent.get(), username=self.user.user
name, hashed_password=self.user.hashed_password)
        self.db.users_update(record)
        tk.messagebox.showinfo("Successfully Updated Name", "Successfully Updated Name
")

        self.user = self.db.users_get_record_by_id(self.user.id)
        self.update_entries()

    def update_entries(self):
        self.clear_entries()
        self.name_ent.insert(0, self.user.name)
        self.username_ent.insert(0, self.user.username)

    def clear_entries(self):
        self.name_ent.delete(0, "end")
        self.username_ent.delete(0, "end")
        self.password_ent.delete(0, "end")
        self.old_password_ent.delete(0, "end")

    def logout(self):
        self.goto_login()

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

    def goto_task_browser(self):
        self.hide()
        self.show_map[GUIStates.TASK_BROWSER](self.user.id)

    def hide(self):
        self.line_canvas.place_forget()
        self.frame.place_forget()

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

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

import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

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

from register_user import RegisterUser
from login_user import LoginUser
from user_info import UserInfo
from task_browser import TaskBrowser

class GUI():

    def __init__(self):

        # tkinter init
        self.root = tk.Tk()
        self.root.geometry("800x800")

        # db init
        self.db = Database()

        # where you can create a new user
        self.register_user = RegisterUser(self.root, self.db)

        # where you can login to the user
        self.login_user = LoginUser(self.root, self.db)

        # where you can update/delete your user
        self.user_info = UserInfo(self.root, self.db)

        # where you can view/CRUD on the logged in user's tasks
        self.task_browser = TaskBrowser(self.root, self.db)

        # a map that stores the functions to show the different states so that a butto
n in one state can change to another state
        show_map = {
            GUIStates.REGISTER_USER : self.register_user.show,
            GUIStates.LOGIN_USER : self.login_user.show,
            GUIStates.USER_INFO : self.user_info.show,
            GUIStates.TASK_BROWSER : self.task_browser.show
        }
        self.register_user.assign_show_map(show_map)
        self.login_user.assign_show_map(show_map)
        self.user_info.assign_show_map(show_map)
        self.task_browser.assign_show_map(show_map)

        # this is the actual init because now inside the states the self.show_map is v
alid
        self.register_user.init_resources()
        self.login_user.init_resources()
        self.user_info.init_resources()
        self.task_browser.init_resources()

    def __del__(self):
        self.db.close()

    def run(self):
        self.login_user.show()
        self.root.mainloop()
```

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

import sys
from gui import GUI

def main():

    # initializes the gui
    gui = GUI()

    # starts the gui
    gui.run()

    # exit successfully
    sys.exit(0)

if (__name__=="__main__"):
    main()
```