from pymysql import TIMESTAMP, cursors

import pymysql as sql

import os

import time

while True:  # Connecting to the databases

    try:

        # Password of the sql server

        passcode = input('Enter the password of your sql server: ')

        # Connecting to the user databases

        user\_db = sql.connect(host='localhost', user='root',

                              passwd=passcode, db='users')

        # Creating a user cursor to handle all queries in the user database

        user\_cursor = user\_db.cursor()

        try:  # Creating the table if it doesn't exist

            user\_cursor.execute("""create table users(

                                username char(20) primary key,

                                password char(20)

                            );""")

        except:

            pass

        try:

            user\_cursor.execute("""create table messaginggroups(

                                groupname char(20) primary key,

                                password char(20)

                            );""")

        except:

            pass

        # Once all the setup is done and the programming will be executed properly

        print("Successfully connected to the database")

        time.sleep(1.5)

        break

    except sql.err.OperationalError:

        print('INCORRECT PASSWORD')

def hazh(x):  # Hashing function that is used to store passowrds

    sume = sumo = suma = pr4 = pr3 = 0

    ods = evs = prall = 1

    for i in range(0, len(x), 2):

        sume += ord(x[i])

    for i in range(1, len(x), 2):

        sumo += ord(x[i])

    for i in range(len(x)):

        prall \*= ord(x[i])+i

    for i in range(0, len(x), 4):

        pr4 += (ord(x[i]) \* ord(x[i-1]) \* ord(x[i-2]) \* ord(x[i-3]))

    for i in range(0, len(x), 3):

        pr3 += (ord(x[i]) \* ord(x[i-1]) \* ord(x[i-2]))

    for i in range(1, len(x), 2):

        ods \*= ord(x[i])

    for i in range(0, len(x), 2):

        evs \*= ord(x[i])

    # for i in range(0,len(x),2):

    prs = hex(sume\*sumo)[-1:-3:-1]

    suma = (sume+sumo)\*len(x) - sume

    suma = hex(suma)[-1:-3:-1]

    if ods > evs:

        oediff = ods-evs

    else:

        oediff = ods + evs

    while not oediff % 16:

        oediff //= 16

    oediff = hex(oediff)[-1:-3:-1]

    while not prall % 16:

        prall //= 16

    prall = hex(prall)[-1:-5:-1]

    pr43 = hex(pr4 % pr3)[-1:-3:-1]

    result = prall+suma+oediff+prs+pr43

    return result

def clear\_shell():  # A function that clears the terminal

    if os.name == 'nt':

        os.system('cls')

    elif os.name == 'posix':

        os.system('clear')

    return None

def retrieve\_user\_list():  # This is used to retrieve the users every time they are updated

    record\_tuples = user\_cursor.execute('select \* from users;')

    record\_tuples = user\_cursor.fetchall()

    \_user = {}

    for i in record\_tuples:

        \_user[i[0]] = i[1]

    return \_user

def retrieve\_group\_list():  # This is used to retrieve the groups every time they are updated

    record\_tuples = user\_cursor.execute('select \* from messaginggroups;')

    record\_tuples = user\_cursor.fetchall()

    \_group = {}

    for i in record\_tuples:

        \_group[i[0]] = i[1]

    return \_group

def create\_account():  # Creating an account ie tables for the inbox and outbox

    global user\_list

    username = input("Enter the username: ")

    if username in user\_list:

        print('This username already exists')

        return create\_account()

    else:

        password = hazh(input("Enter your password: "))

        user\_cursor.execute(

            f"insert into users values('{username}','{password}')")

        user\_cursor.execute(f"""create table {username}\_inbox (

                            sender char(20) not null,

                            message text

                            );""")

        user\_cursor.execute(f"""create table {username}\_outbox (

                            sender char(20) not null,

                            message text

                            );""")

        user\_db.commit()

        user\_list = retrieve\_user\_list()

        print("Account created successfully")

        time.sleep(1.5)

    return username

def login(username):  # Auth function

    password = hazh(input("Enter your password: "))

    if not password == user\_list[username]:

        print("Incorrect password...")

        return login(username)

    else:

        return True

def check\_user\_in\_group(user, group):

    user\_cursor.execute(f"select \* from {group}")

    records = user\_cursor.fetchall()

    for i in records:

        if i[0] == user:

            return True

    return False

def send\_message(username):  # Sends a message from {username} to {receiever}

    receiver = input("To(username/groupname): ")

    if receiver not in user\_list and receiver not in group\_list:

        print("This receiver does not exist")

        return send\_message()

    else:

        if receiver in group\_list and not check\_user\_in\_group(username, receiver):

            print("You are not in this group, so you can't send messages in it.")

            time.sleep(1.5)

            return None

        message = input("Message: ")

        user\_cursor.execute(

            f"insert into {receiver}\_inbox values('{username}','{message}');")

        user\_cursor.execute(

            f"insert into {username}\_outbox values('{receiver}','{message}');")

        user\_db.commit()

        print("Message sent successfully")

        time.sleep(1.5)

        return True

def check\_inbox(username):  # To check the inbox of {username}

    # If it is not an individual user and it is a group

    if not username in user\_list and username in group\_list:

        # Then ask for a password

        password = hazh(input("Enter the password of the group: "))

        # Return the function if the password is incorrect

        if not password == group\_list[username]:

            print("Incorrect password...")

            return check\_individual\_outbox(username)

    messages = user\_cursor.execute(f"select \* from {username}\_inbox;")

    messages = user\_cursor.fetchall()

    if not len(messages):

        print("NO MESSAGES FOUND")

        return None

    for i in messages:

        print(f"{i[0]}: {i[1]}")

        print("\n")

        print("----------------------")

    return None

def check\_individual\_outbox(username):  # To check the outbox of {username}

    messages = user\_cursor.execute(f"select \* from {username}\_outbox;")

    messages = user\_cursor.fetchall()

    for i in messages:

        print(f"{i[0]}: {i[1]}")

        print("\n")

        print("----------------------")

    return None

def erase\_inbox(username):  # Erasing all the messages from the inbox of a user

    user\_cursor.execute(f"drop table {username}\_inbox;")

    user\_cursor.execute(f"""create table {username}\_inbox (

                            sender char(20) not null,

                            message text

                            );""")

    user\_db.commit()

    return None

def create\_group(username):  # Creating a group

    global group\_list

    groupname = input("Enter the name of the group: ")

    if groupname in group\_list:

        print("This group name already exists")

        return create\_group(username)

    else:

        # The variable admin isn't really required but I still made it coz why not when a statement makes more sense?

        # That statement is the one where the adming is automatically added to the group no matter what

        admin = username

        grouppassword = input("Enter the group password: ")

        user\_cursor.execute(

            f"insert into messaginggroups values('{groupname}','{grouppassword}')")

        # Creating a table for the list of users

        user\_cursor.execute(f"""create table {groupname}(

                                        group\_member char(20) primary key

                                        );

                                        """)

        # Creating a table for the messages

        user\_cursor.execute(f"""create table {groupname}\_inbox(

                                        sender char(20) not null,

                                        message text

                                        );

                                        """)

        # Adding the admin to thr group no matter what

        user\_cursor.execute(f"insert into {groupname} values('{admin}');")

        # In case the user doesn't input a number for the no\_of\_group\_members

        while True:

            try:

                no\_of\_group\_members = int(

                    input("How many group\_members do you want in the group? "))

                print("If you want to stop entering usernames at a point, type 'exit'")

                break

            except:

                print("Enter an integer")

        i = 1

        while i <= no\_of\_group\_members:

            group\_member = input(f"Enter usename {i}")

            if group\_member == 'exit':

                break

            elif not group\_member in user\_list:

                print("This user does not exist")

            else:

                try:  # Checking if the user has entered a duplicate username

                    user\_cursor.execute(

                        f"insert into {groupname} values('{group\_member}');")

                    i += 1

                except:

                    print("This user is already in the group")

                else:  # If the user has been added to the table successfully:

                    user\_cursor.execute(

                        f"insert into {group\_member}\_inbox values('{admin}','You have been added to a group called {groupname} and the password is {grouppassword}');")

        user\_db.commit()

        # Retrieving the group\_list again because a new group has just been created

        group\_list = retrieve\_group\_list()

        return True

def delete\_account(username):  # Deleting the account of the user from this messagin service completely

    global user\_list

    print('\n\n')

    print("THIS WILL PERMANENTLY DELETE YOUR ACCOUNT FROM THIS SERVICE")

    print("THIS CHANGE WILL BE IRREVERSIBLE")

    print('\n\n')

    print("Are you sure you want to delete the account ?")

    choice = input(

        f"Confirm by typing in '{username}:{user\_list[username]}' or type anything else to cancel: ")

    # If the user chooses to delete the account

    if choice == f"{username}:{user\_list[username]}":

        # Deleting the record from the users table

        user\_cursor.execute(

            f"delete from users where username = '{username}';")

        user\_cursor.execute(f"drop table {username}\_inbox;")

        user\_cursor.execute(f"drop table {username}\_outbox;")

        for i in group\_list:

            # Deletes the record of the user if they are in the group

            try:

                user\_cursor.execute(

                    f"delete from {i} where group\_member = '{username}';")

            except:

                pass

        user\_list = retrieve\_user\_list()

        user\_db.commit()

        return True

    # If the user chooses not to delete the account

    else:

        return False

def view\_all\_users(username):  # Viewing a list of all users except the current user themself

    print("\n\n")

    a = 1

    for i in user\_list:

        if not i == username:

            print(f"{a}. {i}")

            a+=1

    return None

def login\_menu(username):  # Menu of all tasks that a user can perform

    clear\_shell()

    print("These are all the tasks you can perform")

    print("1.Send a message")

    print("2.Check your individual inbox")

    print("3.Log out")

    print("4.Erase all messages in your inbox")

    print("5.Check your outbox")

    print("6.Create a group")

    print("7.Check your group inbox")

    print("8.View a list of all users")

    print("9.Delete account")

    choice = input("Enter your choice: ")

    if choice == '1':

        send\_message(username)

    elif choice == '2':

        check\_inbox(username)

    elif choice == '3':

        return None

    elif choice == '4':

        erase\_inbox(username)

    elif choice == '5':

        check\_individual\_outbox(username)

    elif choice == '6':

        create\_group(username)

    elif choice == '7':

        groupname = input("Enter the name of the group:")

        check\_inbox(groupname)

    elif choice == '8':

        view\_all\_users(username)

    elif choice == '9':

        if delete\_account(username):

            print("Account successfully deleted...")

            time.sleep(1.5)

            return None

        else:

            print("Deletion aborted")

    else:

        print("Enter a valid option")

    input("Press enter to continue...")

    return login\_menu(username)

def Menu():  # Main/Initial Menu

    clear\_shell()

    print("Choose what you want to do: ")

    print("1.Create an account")

    print("2.Login")

    print("3.Exit")

    response = input("Enter your response: ")

    if response == "1":

        login\_menu(create\_account())

    elif response == "2":

        username = input("Enter your username: ")

        if username not in user\_list:

            print("This username doesn't exist...")

            print("Try creating an account: ")

        else:

            if (login(username)):

                login\_menu(username)

    elif response == "3":

        user\_cursor.close()

        user\_db.close()

        exit()

    else:

        print("Invalid response")

    input("Press Enter to continue...\n\n")

    Menu()

if \_\_name\_\_ == "\_\_main\_\_":

    user\_list = retrieve\_user\_list()

    group\_list = retrieve\_group\_list()

    Menu()