```
"""
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 = "studen
t", passwd = "jchs", database = "RyanKennedyAndGabrielWaldner")
        # self.conn = mysql.connector.connect(host = "127.0.0.1", user = "root", pas
swd = "#Whalez17", database = "RyanKennedyAndGabrielWaldner")
        # self.conn = mysql.connector.connect(host = "127.0.0.1", user = "root", pas
swd = "mysqlpassword", database = "RyanKennedyAndGabrielWaldner")
        self.conn = mysql.connector.connect(host = "127.0.0.1", user = "root", passw
d = "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 W
HERE 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]), ha
shed_password=str(record[3]))
        return rec

    def users_get_records_with_username_and_hashed_password(self, username, hashed_p
assword):
        # get raw data
        self.cursor.execute("SELECT id, name, username, hashed_password FROM users W
HERE 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(
record.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
WHERE 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(recor
d[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) VA
LUES ({}, '{}', '{}');".format(str(record.user_id), record.short_name, record.descri
ption))
        self.conn.commit()

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

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