INSC 484 Homework 3 (100 points)

Instructions:

This homework will focus on building a complete application using Python and PostgreSQL. Just like Homework 2, you will turn in three things for this: 1. The project code, 2. Your database design with justification, and 3. Your software design description and justification. The main difference between this homework and homework 2 is that your **system must be object-oriented**.

Overview:

You will build a client/tax return system for a small accounting firm. Your customer (the accounting firm) requires the following from your application:

- You must store the following information:
    - Information about CPAs (Certified Public Accountants). Each CPA will have multiple clients.
    - Information about clients, including name, address, income, and if they have provided tax materials to the firm. Each client will have only one CPA.
    - Information about the tax return, including the status (is it filed or not), timestamp of filing (if it was filed), and if it has been checked by a CPA ('yes' automatically if the CPA has filed it, 'no' if it was filed by the assistant). Each client can only have one tax return.
- Your interface must have the following functions (they can be merged into less/nested options if you want):
    - Add clients
    - Add tax filing assistants
    - Add CPAs
    - Mark if a client has provided their required materials
    - Check if client has provided their required materials
    - Mark if tax return is filed
    - Check if a return for a client is filed
    - Mark if a CPA checked a return
    - Check if a CPA checked a return

Again, I am purposely being somewhat vague with my description of these requirements, as this is how most customers would be. You should be creative and try to uphold good design principles (both for the database and the software) in meeting the requirements.

Hint: Think about each of the entities and their actions for your object-oriented design. For example, you could have a class for Clients where a client can submit materials (example: client1.submit_matrials), which marks materials as submitted in the database.

Project Code and Database:

This part will be just like we have done in class: Build your code in PyCharm with a venv, zip up the completed project with a sample database created upload to Canvas.

For this project, please only use Python3 and PostgreSQL (using Elephant SQL).

Please write some comments in your code. You do not need to be extremely detailed (as your other documents will do that), but please be sure to write a brief comment of what each file is responsible for and what each function is supposed to do.

Database Design:

For your database design, please submit a text, word, excel, or pdf file that has your design for the database. You should show each of your tables and label where primary keys and foreign keys are. Write a few sentences explaining why you chose to design your database in the way you did.

Software Design:

For your software design, please submit a text, word, or pdf file that explains the overall architecture of your program. Explain your choices for each model (object). Again, think about the flow through your program. Remember in class we made Project 3e flow from the user side to the objects, then the objects handled the database function calls.

How you will be graded:

You will be graded based on three major facets:

1. The functionality of your application. Does it work? Does it meet the basic customer requirements? (**50 points**)
2. The quality of the software and database design. I will be looking for the principles we have talked about in class and implemented in the class projects. Remember things like: modularity in your code, one entity or action per table in your database, etc. (**30 points**)
3. My ability as the “code maintainer” to understand what is going on in your code. This means naming functions and variables well, writing meaningful comments in the code, and describing your software and database design well in the other documents. This also means using object-oriented design well. (**20 points**)

I will be sure to leave feedback if there is anything I take points off on.