CS353 Spring 2024
Homework 4
Programming Assignment
Due: April 4, Thursday till midnight
You will use the Moodle course page for submission of this assignment

Introduction

In this assignment, you will implement a web application using Python, Flask, MySQL, and Docker Compose. This application will be a simple banking database system.

Description

In this web application, customers can perform some operations on their account in a banking system. A customer can have any number of accounts on different branches of the bank. An account can be owned by one or more customers. You should provide a **user registration page** for the customers. Upon registration, the customers can login via the **login page** by providing their username and password. We assume that customer names serve as login names and customer id's serve as passwords (for instance, customer "Ali" can login by entering "Ali" and "10002" as his login name and password). Give an appropriate error message if the login operation fails.

After successful login, in the **main page**, you should list all the accounts (aid, branch, balance and openDate fields) of that customer. Next to each of these accounts, display a link, namely "Close", so that the customer can close this account. When the customer clicks on the "Close" link, display either an "error message" or a "successful deletion" message, and allow the customer to return to the main page, again (which, of course, doesn't display the tuple(s) deleted from the owns table).

At the bottom of this page, also provide a link called "Money Transfer". When this link is clicked, open a new page. At this page, show the accounts of this customer again and also all the other accounts in your bank database so that the customer can choose which account to make transfer (you can use combo boxes and/or radio buttons for a more sophisticated GUI). Note that the customer should be able to make a transfer to one of his/her accounts as well. Provide three input fields (FromAccount, ToAccount and TransferAmount) and a "Submit" button, so that the customer can specify the accounts for the transfer (e.g., by typing the account ids) and the transfer amount. Money transfer operation is initiated by clicking the "Submit" button. You should first check that the "FromAccount" belongs to the current customer and if it does not give an appropriate error message. Also give a warning if any of these three input fields is blank. Then, check that the FromAccount balance is enough for the transfer amount given. Otherwise, give an appropriate error message. If all of these conditions are satisfied, allow the transfer by making account balance updates of the corresponding accounts and give an appropriate message indicating the successful completion of the money transfer. Allow the customer to return to the previous page that displays money transfer screen, to allow him/her to make more money transfers. At every page, remember to put a link to an appropriate previous page, so that the customer can go back without doing any modifications. Also, you may need to keep track of the current customer id through pages, as well.

Create the following relations in for the banking database. You should setup the primary and foreign keys yourself. You will create a file called **schema.sql** containing CREATE TABLE statements.

- **customer**(*cid*: CHAR(5), *name*: VARCHAR(30), *bdate*: DATE, *city*: VARCHAR(20), *nationality*: VARCHAR(20))
- **account**(*aid*: CHAR(8), *branch*: VARCHAR(20), *balance*: FLOAT, *openDate*: DATE, *city*: VARCHAR(20))
- **owns**(*cid*: CHAR(5), *aid*: CHAR(8))

Also add the following entries to the database using INSERT queries in the **schema.sql** file.

customer				
cid	**name**	**bdate**	**city**	**nationality**
10001	Ayse	08.09.1990	Ankara	TC
10002	Ali	16.10.1985	Ankara	TC
10003	Ahmet	15.02.1997	İzmir	TC
10004	John	26.04.2003	İstanbul	UK

account				
aid	**branch**	**balance**	**openDate**	**city**
A0000001	Kızılay	40,000.00	01.11.2019	Ankara
A0000002	Kadıköy	228,000.00	05.01.2011	İstanbul
A0000003	Çankaya	432,000.00	14.05.2016	Ankara
A0000004	Bilkent	100,500.00	01.06.2023	Ankara
A0000005	Tandogan	77,800.00	20.03.2013	Ankara
A0000006	Konak	25,000.00	22.01.2022	İzmir
A0000007	Bakırköy	6,000.00	21.04.2017	İstanbul

owns	
cid	**aid**
10001	A0000001
10001	A0000002
10001	A0000003
10001	A0000004
10002	A0000001
10002	A0000003
10002	A0000005
10003	A0000006
10003	A0000007
10004	A0000006

You should also create an **account analysis page** that could be reached from the main page, that lists various statistics about the accounts of the customer. You should use SQL queries (i.e., *not* python logic) to formulate the following (query results will form the account summary page). The SQL queries should be constructed based on cid of the customer.

1. List the account id, branch, balance, and opening date of the accounts owned by the customer, in ascending order of opening dates.
2. List the account id, balance, and opening date of the accounts owned by the customer, where the balance is higher than 50,000 and the opening date is after the end of 2015.
3. List the account id and balance of the accounts owned by the customer, where the accounts are from the same city as the customer.
4. Find the maximum and minimum balances of the accounts owned by the customer. Name the attributes of the resulting table as max-balance and min-balance, respectively.

Finally, implement the **logout** function.

Sample Application

To aid you in the coding process, **we give you a sample application** that uses Python, Flask-MySQL and Docker Compose.

Docker is a virtualization platform that helps developers to easily create, deploy, and run applications inside containers. Containers provide a consistent and isolated environment for applications, ensuring that the applications can be replicated across different environments.

Docker Compose is a tool for defining and running multi-container Docker applications. With Docker Compose, you can define the services (in our case web and database services) that make up your application in a YAML file, and then start and stop all services with a single command (we will show you how).

Flask is a micro web framework written in Python. It is designed to be simple and lightweight, allowing quick implementation and deployment of web applications with minimal setup.

In order to use Docker Compose, you first need to install it in your machine. In Ubuntu, you can use this link. On Windows, you can use this link (or any other resource if you are stuck, but it should be straightforward).

Once you install Docker Compose, you create the following file structure.

```
├── app
│   ├── app.py
│   └── templates
│       ├── login.html
│       └── register.html
├── docker-compose.yaml
├── Dockerfile
├── requirements.txt
└── schema.sql
```

We will provide you the Dockerfile, requirements.txt and docker-compose.yaml. Use these files in your implementation and do not change them. We will provide the files for simple user registeration and login logic (app.py and templates files) to aid you in the process.

The Dockerfile below uses an existing python image and adds the python modules defined in the requirements.txt file, which contains two modules Flask and flask_mysqldb, along with. It also points at the /app folder.

```
FROM python:3.9-slim-buster
RUN apt-get update
RUN apt-get install -y pkg-config
RUN apt-get install -y gcc
RUN apt-get install -y default-libmysqlclient-dev
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
```

You must build your own docker image tagged cs353hw4app using: **docker build -t cs353hw4app .**

The docker-compose.yaml file contains your webapp image cs353hw4app and your MySQL database.

```
version: '3'
services:
  web:
    image: cs353hw4app
    ports:
      - "5000:5000"
    volumes:
      - ./app:/app
    working_dir: /app
    command: python app.py
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: cs353hw4db
    ports:
      - "3307:3306"
    volumes:
      - ./schema.sql:/docker-entrypoint-initdb.d/schema.sql
```

Before running with this configuration, you need to prepare your schema.sql file that contains your CREATE TABLE statements. Once you prepare your database schema, you can fire up the system using **docker-compose up -d**. You can view your web and database services by running **docker-compose ps**. Depending on your IDE, you should see the changes you do in your app.py immediately without rebuilding your service.

What to Submit?

You **must** submit the following files (the same directory structure provided earlier) in a **zip** file named **surname_name_id_hw4.zip** (*not rar, tar, tar.gz etc.*) in the Moodle Course Page. Your code must be ready-to-run and without syntax errors. We are unable to do any debugging and fixing for you.

- Dockerfile (We already gave you this)
- docker-compose.yaml (We already gave you this)
- requirements.txt (We already gave you this)
- schema.sql file that contains all the database schema along with the tables (You need to fill this)
- Application files. You must submit at least the app.py and templates files. You can add extra files such as .css files, as you wish. The application should implement the requirements we specified in the Description section.

Grading

- Login (5 pts)
- Registration (5 pts)
- Money Transfer (25 pts)
- Close Account (15 pts)
- Logout (5 pts)
- Schema (15 pts)
- Account Summary/SQL Queries (20 pts)
- UI (10 pts)

Tips

- If you do not see your schema updating on your database service according to your updates in your schema.sql file, this might be a caching issue and you need to remove existing services by **docker-compose rm ,** than do **docker-compose up** or **docker-compose up --build --force-recreate –no-deps** again.
- You can connect to your database from your host machine using any mysql client such as from command line or MySQL Workbench. If you have the MySQL client on your Linux machine you can simply connect using:
  mysql --host=0.0.0.0 -P 3307 --user=root --password
- For Flask templates, you may visit this link.
- On Linux, you may need to run the docker or docker compose command with **sudo** (e.g., **sudo docker-compose up**). On Windows, you may need to open your command or Powershell prompt using "Run as Administrator".