CS353 Fall 2023
Homework 4
Programming Assignment
Due: November 6, Monday 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 summer internship application system in a university.

Description

In this web application, students can apply to companies for summer internship. Each company allocates a certain internship quota for students. Applying for an internship at a particular company requires students to meet the gpa threshold of that company. You should provide a **user registration page** for the students. Upon registration, the users (i.e., students) can login via the **login page** by providing their username and password. We assume that student names serve as login names and student id's serve as passwords (for instance, student "Ali" can login by entering "Ali" and "S101" 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 companies (cid, cname, quota and gpa-threshold fields) where the student has applied for internships. Students can apply up to 3 companies**.** Next to each of these internship applications, display a link, namely "Cancel", so that the student can cancel this application. When the student clicks on the "Cancel" link, display either an "error message" or a "successful deletion" message, and allow the student to return to main page again (which, of course, doesn't display the tuple(s) deleted from the apply table).

At the bottom of this page, also provide a link called "apply for new internship". If the student has already applied for 3 companies, give an appropriate error message (either at a new page or a dialog box) when (s)he clicks on this link. Otherwise, when this link is clicked, open **a new page**. At this page, show ids and names of those candidate companies that are not applied by this particular student (You should not show the company if all of its internship quota is filled or student's gpa is lower than the company threshold.). Also provide an input field and a "submit" button, so that the student can choose to apply in one of these displayed companies (e.g., by typing the company id and clicking on the submit button). Again, show an appropriate message after the addition and allow user to return to the previous page. At every page, remember to put a link to an appropriate previous page, so that the student can go back without doing any modifications. Also, you may need to keep track of the current student id through pages, as well.

While doing the above tasks, give meaningful error messages when necessary.

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

- **student**(*sid*: CHAR(6), s*name*: VARCHAR(50), *bdate*: DATE, *dept*: CHAR(2), *year*: VARCHAR(15), *gpa*: FLOAT)
- **company**(*cid*: CHAR(5), *cname*: VARCHAR(20), *quota*: INT, gpa-threshold: FLOAT, city: VARCHAR(20))
- **apply**(*sid*: CHAR(6), *cid*: CHAR(5))

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

student					
sid	**sname**	**bdate**	**dept**	**year**	**gpa**
S101	Ali	15.07.1999	CS	sophomore	2,92
S102	Veli	07.01.2002	EE	junior	3,96
S103	Ayşe	12.02.2004	IE	freshman	3,30
S104	Mehmet	23.05.2003	CS	junior	3,07

company				
cid	**cname**	**quota**	**gpa-threshold**	**city**
C101	tübitak	10	2,50	Ankara
C102	bist	2	2,80	Istanbul
C103	aselsan	3	3,00	Ankara
C104	thy	5	2,40	Istanbul
C105	milsoft	6	2,50	Ankara
C106	amazon	1	3,80	Palo Alto
C107	tai	4	3,00	Ankara

apply	
sid	**cid**
S101	C101
S101	C102
S101	C104
S102	C106
S103	C104
S103	C107
S104	C102
S104	C107

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

1. List the name, quota, and gpa-thresholds of the companies applied by the student, in descending order of quotas.
2. Find the maximum and minimum gpa-thresholds of the companies applied by the student. Name the attributes of the resulting table as max-gpa-threshold and min-gpa-threshold, respectively.
3. Find the number of companies applied by the student in each city. Name the attributes of the resulting table as city and application-count.
4. Give the names of the companies applied by the student with the maximum and minimum quotas. Name the attributes of the resulting table as company-with-max-quota and company-with-min-quota, 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 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 (10 pts)
- Internship Application (25 pts)
- Application Cancel (10 pts)
- Logout (5 pts)
- Schema (15 pts)
- Application 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”.