CS353 Spring 2023
Homework 4
Programming Assignment
Due: 6 April, 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. The web application you will develop is a generic Todo-list application where users can manage their todo list.

Description

In this web application, the user can create tasks in different categories such as health, job, lifestyle, family, hobbies. The tasks need to have a deadline. The user can finish the task by marking the task as Done. The user can also delete or edit the task. You should provide a **user registration page**. Upon registration, the users can login via the **login page** by providing their username and password. After successful login, in the **main page**, you list the tasks of the user sorted by their deadline (the topmost should be the one with the closest deadline). You also show the completed tasks separately in this page, sorted by their completion time. In this page, allow the user to add a task, delete a task, edit a task, and finish a task. The following is an example database schema you can use to create the application. **You should setup the primary and foreign keys yourself.** You will create a file called the **schema.sql**.

- User (id:int, password:varchar, username:varchar, email:varchar)
- Task (id:int, title:varchar, description:text, status:varchar, deadline:datetime, creation_time:datetime, done_time: datetime, user_id: int, task_type:varchar)
- TaskType (type: varchar)

In the schema.sql, use the following tables to initially populate the database.

User table

id	password	username	email
1	pass123	user1	user1@example.com
2	password	user2	user2@example.com

TaskType table

type
Health
Job
Lifestyle
Family
Hobbies

Task table

id	title	description	status	deadline	creation_time	done_time	user_id	task_type
1	Go for a walk	Walk for at least 30 mins	Done	2023-03-20 17:00:00	2023-03-15 10:00:00	2023-03-20 10:00:00	1	Health
2	Clean the house	Clean the whole house	Done	2023-03-18 12:00:00	2023-03-14 09:00:00	2023-03-18 17:00:00	1	Lifestyle
3	Submit report	Submit quarterly report	Todo	2023-04-12 17:00:00	2023-03-21 13:00:00	null	1	Job
4	Call Mom	Call Mom and wish her	Todo	2023-04-06 11:00:00	2023-03-23 12:00:00	null	1	Family
5	Gym workout	Do weight training for an hour	Done	2023-03-19 14:00:00	2023-03-12 10:00:00	2023-03-19 11:00:00	1	Health
6	Play guitar	Learn new song for an hour	Todo	2023-04-05 20:00:00	2023-03-20 14:00:00	null	2	Hobbies
7	Book flights	Book flights for summer vacation	Done	2023-03-16 09:00:00	2023-03-13 13:00:00	2023-03-16 11:00:00	2	Lifestyle
8	Write a blog post	Write about recent project	Todo	2023-04-11 17:00:00	2023-03-22 09:00:00	null	2	Job
9	Grocery shopping	Buy groceries for the week	Todo	2023-04-05 18:00:00	2023-03-31 10:00:00	null	2	Family
10	Painting	Paint a landscape for 2 hours	Done	2023-03-23 15:00:00	2023-03-18 14:00:00	2023-03-23 16:00:00	2	Hobbies

You should also create an **analysis page** that could be reached from the main page, that lists various statistics about the task history of the user. You should use SQL queries (i.e., *not* python logic) to solve the following (query results will form the analysis page).

1. List the title and latency of the tasks that were completed after their deadlines (for the user).
2. Give the average task completion time of the user.
3. List the number of the completed tasks per task type, in descending order (for the user).
4. List the title and deadline of uncompleted tasks in increasing order of deadlines (for the user).
5. List the title and task completion time of the top 2 completed tasks that took the most time, in descending order (for the user). (You can use the LIMIT clause).

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 db 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 registration and login and register 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. 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 (8 pts)
- Create Task (8 pts)
- Edit Task (8 pts)
- Delete Task (8 pts)
- Finish Task (8 pts)
- Logout (5 pts)
- Schema (15 pts)
- Analysis (25 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 commandline 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”.