Restaurant Backend Web APP

Introduction

Suppose you are a Full-Stack Developer who is tasked to implement the backend for a restaurant website. You are supposed to implement this backend in Flask with your choice of database tool.

You are supposed to implement the complete backend system which allows you to signup customers and vendors, login them using their credentials, as a vendor add items in the database, as a customer place an order, and as an admin, I should be able to see all the orders placed etc.

Housekeeping points

- This is a minimal example and may not follow some standard practices.
- We focus on the main flow, and not much error handling.
- To avoid handling command-line arguments, config file paths are hard-coded in the source files - not a general practice.

Program Organization

The simple program is structured in various layers.

1. **Models:** In this package, we have different python files named Models.py. All these files are designed to do certain tasks.
    a. Models.py: This file has classes for different tables that will be created inside the database. The tables that are being created are:
        i. User(user_id, name, username, password, level), here level → 0 is for the customer, level → 1 for vendors and level→ 2 for admin.
        ii. Item(item_id, vendor_id, item_name, calories_per_gm, available_quantity,restaurant_name, unit_price)
        iii. Orders(order_id, user_id, total_amount, is_placed)
        iv. OrderItems(item_id, order_id, item_id, quantity)
2. **apis.py**
    a. This file is designed to call some of the implemented APIs such as signup_user, login, logout, add_vendor, list_all_vendors, add_item, list_all_items, create_order, place_order, list_orders_by_customer, list_all_orders

3. **main.py**
    a. This is a simple python script to start the application, once the application is started we should be able to call the API's and see the output accordingly.

Problem Statement

You are supposed to implement the following APIs to make this application work. These API methods exist inside the APIs package. More details are given below:

1. **apis.py: This file has all the APIs and methods related to the customer available in it.**

    a. **Signup**: This is a signup API. This should take, "name, username, password, level" as parameters. Here level is 0 for the customer, 1 for vendor and 2 for Admin.
    b. **Login**: This API should take the username and password of signed up users and successfully log them in.
    c. **Logout**: This API should log out the customer.

2. **apis.py: This python file will have implementations to add vendors to the database. This file has another method implemented in it to call and extract the list of all the vendors.**
    a. **Add_vendor**: Only added customers can be made vendors. This API should take "user_id" as a parameter.
    b. **Get_all_vendors**: Only logged in users can call this API. This should return all the vendor details with their store and item offerings.
3. **apis.py: This python file consists of the APIs relevant to orders. This will contain methods to create orders, add items in orders, get all the orders by customer, get_all_orders on the admin level.**
    a. **Add_item**: Only logged in vendors can add items. This API should take, "item_id, item_name, restaurant_name, available_quantity, unit_price, calories_per_gm".
    b. **Place_order**: Only logged in customers can place orders. This API should take, "customer_id, vendor_id, item_id, quantity" as parameters.
    c. **Get_all_orders_by_customer**: Only logged in users can call this API. This returns all the orders placed by that customer. This should take "customer_id" as a parameter.
    d. **Get_all_orders**: Only admin can call this API. This API returns all the orders in the orders table.

Evaluation Rubric

Total Project Points: **100**

- Basic compilation without errors (**10%**) : **10 Points**
- Correctness:
  Correctness of implementation
  - Problem statement - point 1.a (**12%**) : **12 Points**
  - Problem statement - point 1.b (**12%**) : **12 Points**
  - Problem statement - point 1.c (**12%**) : **12 Points**
  - Problem statement - point 2.a (**12%**) : **12 Points**
  - Problem statement - point 2.b (**12%**) : **12 Points**
  - Problem statement - point 3.a (**10%**) : **10 Points**
  - Problem statement - point 3.b (**10%**) : **10 Points**
  - Problem statement - point 3.c (**5%**) : **5 Points**
  - Problem statement - point 3.d (**5%**) : **5 Points**

Program Instructions

1. Download the zipped folder named **M01-Project04-Flask-Restaurant-App.zip**, unzip it on your local machine, and save it. Go into the directory named **M01-Project04-Flask-Restaurant-App.**
2. Before running the application make sure that you have a DB created in your system with the same name as it is mentioned in the model's python script. Do not forget to change the credentials based on your DB configuration

3. Make sure you have Python 3.9 or higher installed. At your command prompt, run:

   ```
   $ python --version
   Python 3.9.0
   ```
   If not installed, install the latest available version of Python 3.

4. You will be needing MySQL installed in your system to verify the changes that you are making. Please install **MySQL** and **MySQL Workbench**.

5. First, you need to make changes to the apis.py. Make sure that while running main.py you are not getting any errors. You can check the MySQL workbench to verify if the data is available in the database.

6. You can now examine and run **main.py**. This will currently run various simple calls. As you solve the problems, you'll be frequently running this file. You can comment or modify the initial code as needed.

```
$ python3 main.py (On many Linux platforms)
                        OR
$ python main.py  (On Windows platforms)
```

In any case, one of these two commands should work.

7. Alternatively, you could install a popular Python **IDE**, such as **PyCharm** or **Visual Studio Code**, and select a command to build the project from there.

8. Once the program is ready to submit, zip the parent folder **M01-Project04-Flask-Restaurant-App**, and upload the new zip file as **M01-Project04-Flask-Restaurant-App.zip**. It is now ready for evaluation.