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.

**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 APIs 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 API 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, the level is 0 for the customer, 1 for the 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, and 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“order\_id” as a parameter.

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 the admin can call this API. This API returns all the orders in the orders table.