Description of HungryApp.

HungryApp should let you search for and discover restaurants to eat out at or order from. The app lets user to browse through restaurant menus to decide where one wants to eat. A restaurant is identified by name as well as address which can be treated as a unique key. A user is identified by either User Id or Phone No**.** The food-delivery agent is identified by his agent-id or his phone number.

Right now, we will like to develop the App not as a mobile app but just as a desktop application. To develop the application, maintain the following databases.

1. Restaurant database – Records of restaurants with details like address/area, no-of-seats, special facilities, menu etc,

2. Agents-database - Each agent identified by agent-id, name, phone No., currently-accumulated-commission etc.

3. User-database, each user identified by user-id, name, address, phone no, etc. For every user, a record of all previous orders is also kept.

Following functions are required to be provided.

1. **Different Search() functions to search for eating locations;**
   - Based on category (restaurant, cafe, pub etc)
   - Based on category of food or cuisine (north Indian, south Indian, continental etc)
   - Based on area (Name of area and its nearby areas). Note that you can keep information which tells which other areas of city are near the given area.

2. **Order (customer name, address, Phone No., Eating-joint, Menu)**; Each order should be added to the global database of orders which maintain all the orders yet to be delivered. This function allocates an available agent to this particular order. The output of this function should be order details along with agent details. You can decide the sorted order you want to store the global database of orders so thatit is easy for different operations.

3. **Delivery()** : This function is to be executed on the actual delivery and the order from global order database is removed which also frees the respective agent. It also adds 10% commission to the particular agent's currently-accumulated commission.

4. **CancelOrder()** : It cancels the order, removes it from global database of orders and frees the agent.

5. **FindFavoriteFoodsOfUser(user-id) :** For the given user-id, find top 3 favorite food items based on previous orders.
6. **FindFavoriteRestaurants(int N) :** Finds the top-3 restaurants that have got maximum orders.
7. **FindFavoriteFoodsAcrossRestaurants(int N) :** Finds the top-3 ordered most favorite food items.

8. **Range-Search –** Create/print a list of current orders for users having user-id between *user-id1* and *user-id2*.