Order service design

1. Database

A screenshot of a computer screen

Description automatically generated

* **Menu management**

|  |  |  |
| --- | --- | --- |
| **Table** | **Description** | **Relations** |
| Category | Category of each item(coffee, cakes…) |  |
| Item | Item information(coffee, cakes…) with price |
| ItemCategory | Many to many between Category and Item |  |
| Menu | Menu contains items, if location\_id is empty means menu is base menu |  |
| MenuItem | Many to many between Menu and Item |  |

* **Location management**

|  |  |  |
| --- | --- | --- |
| **Table** | **Description** | **Relations** |
| Chain | Shop information |  |
| Location | Location of chain | Many to one Chain table |
| LocationQueue | List queue of location | Many to one Location table |
| OperatingHour | Opening hourse of location | Many to one Location table |

* **Order management**

|  |  |  |
| --- | --- | --- |
| **Table** | **Description** | **Relations** |
| Customer | Customer information | Many to one Location table |
| Orders | Order information | Many to one Location, Customer, LocationQueue |
| OrderItem | List item inside of each Order Many to many between Orders and Item |  |

* **User management**

|  |  |  |
| --- | --- | --- |
| **Table** | **Description** | **Relations** |
| Users | Shop owner / shop operator information,  Role is enumeration | Many to one Location table |

1. **API**

* **Shop(Chain)/location apis**

|  |
| --- |
| Create/update/delete shop/chain |
| Create/update/delete/list location of shop(chain) |
| Create/update/delete/list queue of location |
| Find the nearest location by customer position (lat/long) |

* **Menu/item/queue apis**

|  |
| --- |
| Create/update/delete/list category and item of location |
| Create/update/delete/list menu of location |
| Find queue for new order |
| List order/customer of a queue |

* **user apis**

|  |
| --- |
| Create/update/delete/list users of location  System should create a default super admin user as a shop owner role |
| Login/logout/change password of user |

* **customer apis**

|  |
| --- |
| Create/update/delete/list customer of location |
| Update total order of customer after make order success |

* **order apis**

|  |
| --- |
| Create/update/delete/list order of customer |
| Put order to location queue |
| Pick order from location queue |
| Cancel order |
|  |

* **recommended/others module/apis**

|  |
| --- |
| Management membership program for customers |
| Management discount program |

1. **Security**

* User has a specific role: OWNER or OPERATOR
* System will create a super admin as shop owner with role “OWNER” when init database
* After OWNER login into system, he/she can do everything.
* After OPERATOR login into system he/she can do request with queue only
* To limit/detect the request belongs to OWNER or OPERATOR we need to split api requests to 2 domains:
  + /admin (for owner role only)
  + /operator(for owner or operator role)
* We define a java class UserDetailServiceImp which implements UserDetailsService to check the user role by username then we can define a SecurityConfig class to use this implementation.

1. **Testing**
   1. /login

|  |  |
| --- | --- |
| **Curl** | **Expected result** |
| curl -X POST -d "username=owner&password=owner" http://localhost:8080/login | Status=200  Message=Login success |
| curl -X POST -d "username=operator&password=operator" http://localhost:8080/login | Status=200  Message=Login success |
| UserSecurityTest.java | Define some unit test case to check security with user role by username |