# Restaurant Management System

## Restaurant Part

### Basic Information

Give the following information of the restaurant: name, company contact information (address, telephone, wechat, qq, e-mail), manager, comments, icon, photos

### Asset Management

(This functionality will be ignored, since it is not significant for a small or middle level restaurant)

Manage the asset owned by this restaurant, like tables, chairs, pots... Each entity generally contain the following information: serial number, name, icon, pictures, price, quantity. We also need to record the event of each asset, like asset buying, asset destruction. Each asset event might contain the following information: asset entity, event quantity (quantity before event, delta quantity, quantity after event), event time, event type (add event), comment.

### Employee Management

Manage employees of the restaurant. For each employee we store the following information: employee number, name, person contact information(address, telephone, wechat, qq, email, id, birthday, age), emergency person contact info, on board date, leave date, leave reason, salary rules, start working time, end working time. Since this part also manage the salary of employee. So here we need to define salary rules as entity. It should contain the following information: salary type(base, deduce, award, workload, hours), amount, employee. To calculate the salary of each employee, we have to log the salary event. The salary event can contain the following fields: employee, salary rule, event start time, event end time, comment.

## Sales Part

### Dish Management

Restaurant needs to maintain all dishes they sell. The following operations should be supported by the dish management system:

* Add new dishes for selling
* Modify information for some dishes, like price, ingredients, descriptions, pictures…
* Delete dishes no longer being sold

#### Dish Ingredient

To integrate with the inventory management, dishes need to main the ingredient. The ingredient table of one kind of dish should at least contain the following information (refer to Supplier Management Part)

Since the ingredient unit price was cached in the dishes, ingredient also need to update the dish ingredient’s price. So that manager could refresh the ingredient unit price and amount manually. Here we don't support automatically updating dish object if one of the ingredient has changed its price. Otherwise, the price changing operation would be with low performance.

Also, here ingredient doesn’t record its supplier. Because one ingredient might have many supplier. Instead, the supplier would contain information about the ingredients it supplied.

#### Dish table

The dish itself is an atomic selling unit. The dish can contain the following information (id ignored here):

* Dish name
* Dish barcode
* Dish picture uris
* Dish description
* Dish salesPrice
* Dish ingredient cost (Use virtual property)
* Dish profit (Use virtual property)
* Dish profit percentage (Use virtual property)
* Dish ingredient table

#### Relationship between Dish and Dish ingredient

Dish owns many DishIngredients. If Dish was removed, then DishIngredient identity must be removed too. And if the dish doesn’t need the ingredient, then the ingredient must also be removed from the database.

When dish was loaded from database, its ingredient cost, profit and profit percentage should be calculated automatically. Here we could use the virtual property of JPA to implement this, instead of calculating in application logic.

### Customer Management

Customer management system focuses on solving the customer related issues. It should contain the following the management unit:

* Record basic information for each customer
* Record simple membership points for each customer

#### Customer basic information

Information for each customer was contained in Customer table.

* Customer name (mandatory)
* Cell phone
* Wechat, QQ
* Membership point
* Address

The membership point can be used to pay customer’s order. And if customer used membership point to pay their orders, then they would gain no membership point for this time. Since they’ve got discount by using membership point for this time.

The system has to predefine a default customer, so that customers who don’t want to register can also place order.

### Sales Order Management

This part mainly focus on the sales orders. This system doesn’t introduce payment, since this is for small restaurant. User can directly pay the order. All is an order. The functionality of order is as follows:

* Select customer, dishes as the input of sales order
* User can freely change the customer, dishes as long as the order is not closed, canceled or paid.
* Order can be paid with cash or membership points. If membership point was used, then don’t generate points for this order.

To implement above features, order needs to have the following fields:

* Customer
* Sales order line
* Status (unpaid, paid, canceled)
* Order total
* Membership paid amount
* Other paid amount
* Order time
* Payment time
* Order cost

Each line of sales order represents one sales unit of one dish. Of course, we can add quantity for each line. The sales order needs to calculate the cost from sales order line, so line has to store the cost of the dish it represented. What’s more, the node has to be deleted/modified/created as long as the order was deleted/modified/created. According to above words, sales order line can contain the following fields:

* Sales order
* Dish
* Quantity
* Cost
* Total

## Purchase Part

This part mainly focus on the purchase events. Like manage the inventory, ingredient purchase and supplier management.

### Inventory Management

This sub part manage all the ingredient. Restaurant manager can query the amount of each kind of ingredient. And also list all ingredient’s inventory amount. Since the sales orders and purchase orders can all modify the ingredients, so here we need a gateway for ingredient’s increase or deduce. This gateway needs to be thread safe.

For now, the gateway can only need to implement the following interfaces:

* changeIngredientAmount(Long ingredientId, BigDecimal deltaAmount)

### Purchase Order Management

Purchase order mainly is responsible for buying ingredients. The logical of using purchase order can be as follows: Manager select ingredient and corresponding supplier, specifying the posting date and address. The address is loaded from restaurant by default. Of course, manager can input the address. The purchase order can contain the following information:

* manager (non null)
* posting time (non null)
* ingredient table (non null)
* order time
* status
* address (non null)
* contact phone number, by default the restaurant number (non null)

Purchasing order also needs to be paid. Once being paid, then the purchased amount must be updated into the supplier ingredient table line.

### Supplier Management

This part is responsible for managing supplier information. Manager can log in the supplier information. Should have the following functionality: Add supplier, delete supplier, add/remove supplier ingredient and manage the price of the ingredient. Supplier ingredient should be different entity of inventory ingredient and sales ingredient.

The supplier ingredient should contain the following fields:

* ingredient code (mandatory)
* ingredient name
* ingredient uom (mandatory)
* supply price (mandatory)

The inventory ingredient should contain the following fields:

* ingredient code (mandatory)
* ingredient name
* ingredient UoM (mandatory)
* inventory qantity (mandatory)
* buying price (mandatory) (Average value of each purchased)

The dish ingredient should contain the following fields:

* ingredient(mandatory)
* ingredient UoM(mandatory)
* ingredient dish quantity (mandatory)
* buying cost

The supplier table contain the following information:

* name
* contact detail
* supplier ingredient table line

## UoM Management

This part defines the uom functionality. Uom can be converted between any other uom which is in the same group. Thus before defining uom, user has to define uom group or selecting existing uom group. And also filling the conversion rate to the base uom. So the Uom group contain the following information:

* group name
* group id
* default uom

The uom contain the following information:

* Uom group
* Rate
* Uom name
* Is default

Also need a service to convert uom from one type to another. The service contain the following interface:

* BigDecimal convertUomQuantity(Uom from, Uom to, BigDecimal amount);

## Finance Part

TBD…