Restaurant OOS Report

# Team member

Chai Yi (NUID:1226889 [chai.yi@husky.neu.edu](mailto:chai.yi@husky.neu.edu))

Sun Rui (NUID:1239997 [sun.rui@husky.neu.edu](mailto:sun.rui@husky.neu.edu))

Zhao Shengsi (NUID: 1239078 [zhao.shen@husky.neu.edu](mailto:zhao.shen@husky.neu.edu))

# MISSION STATEMENT

To build a Restaurant Online Ordering System with B/S structure which strictly followed by the software design pattern.

The system is aimed at offering a more convenient way for people to get yummy food around them. The system also provides restaurants one more efficient way to deliver food to customers.

# Problem Addressed:

How to divide the role and responsibility for each role

How to form an order according to the restaurant menu

How to connect the relationship between menu, restaurant and customers

What the relationship between dispatcher and customer

How to calculate the dispatcher’s wages according to their working time

# Entities and its groups

Customer Side

**1) Customer Entity**

**PK: CustomerID  
FK: UserID  
Customer:** Save the information for the customer like address, name, phone number, e-mail, birthday etc.

**Relationship:** Its relationship constructed with UserID is 1 to 1 identifying relationship (Though considering that some customers will create two accounts if they want, the customers are different if one customer has two accounts).

**2) UserAccount Entity**

**PK:UserID**

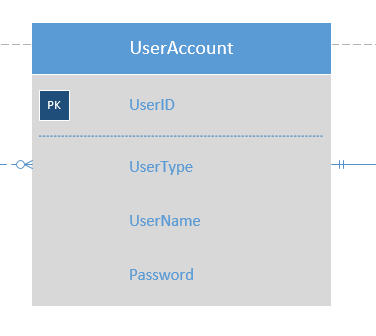
Figure 1 Customer Entity

**FK: CustomerID**

**User account:** Save the information about Customer ID, User name and user’s password.

**Relationship**: Its relationship is constructed with UserID is 1 to 1 identifying relationship;

Its relationship constructed with Card is 1 and only 1 to 1 or more identifying relationship (required the customer/ Restaurant/ Dispatcher must connect at least one bank card to the account on consideration of the payment security).

****Its relationship constructed with Order is 1 and only 1 to 0 or more not identifying relationship.

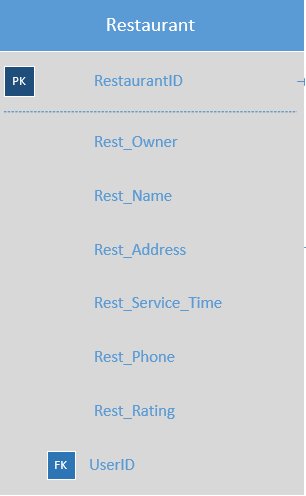
**3) Card Entity**

**PK: CardID**

**FK: UserID**

Figure 2 User Account Entity

**Card:** Save the card information, which is used when the system forms the order.

**Relationship:** 1 and only 1 to 1 or more identifying relationship (as mentioned previously).

Restaurant Side:

**1) Restaurant Entity**

**PK: RestaurantID**

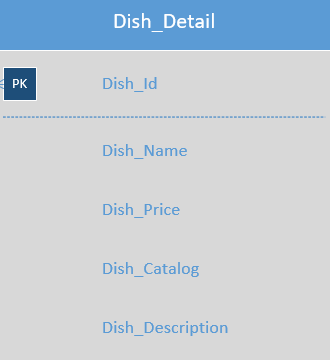
**Restaurant:** Save the restaurant information which has authority to edit the menu and deal with the order. The restaurant also could have several menus for different conditions such as seasonal menus, event special.

**Relationship:** It has one-to-many relationship with comments and menu.

Figure 3 Restaurant Entity

**2) Menu Entity**

**PK: MenuID**

**FK: RestaurantID**

**Menu:** Stored with valid time, menu name and restaurant ID.

**Relationship:** It has one-to-many relationship with Dish\_detail.

**3)** **Dish\_Detail Entity**

**PK: Dish\_Id**

**Dish\_Detail:** Save the information for each dish like Beef hamburger or green salad, which includes the content like price, name.

Figure 4 Dish Detail Entity

**Relationship:** It has many-to-one relationship with Order\_Detail.

**4) Comment Entity**

**PK: CommentID**

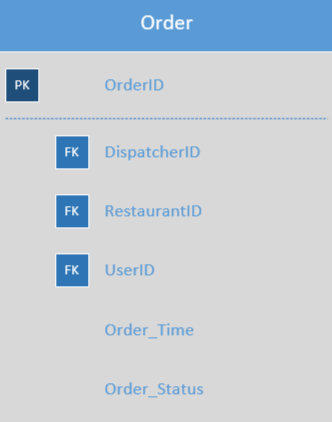
**FK: OrderID**

**Comment:** When the order has been placed, the messages that customer noted will be stored.

When the order has been completed, the rating-level that customers would rate will be stored.

Customer could comment on each order they have been placed. The rating and comment will affect the restaurant’s overall rating.

**Relationship:** It has one-to-one relationship with order and many-to-one relationship with restaurant.

System side:

**1) Order Entity**

**PK: Order ID**

**FK: Dispatcher ID**

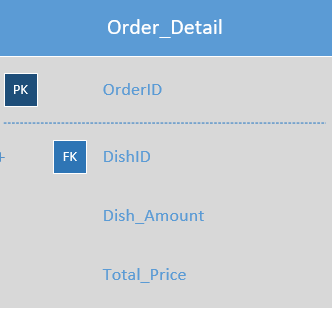
**Restaurant ID**

**User ID**

**Order:** The comprehensive information for each order like who order the food, who will deliver the food, where the food come from. Therefore, it is the most complicated entity in the whole system. It is connected with restaurant, dispatcher, Order\_detail and customer.

Figure 5 Order Entity

**Relationship:** It has many-to-one relationship with User account.

In addition, it has one to one relationship with Order Detail, Dispatcher and comment.

**2) Order\_Detail Entity**

**PK: OrderID**

**FK: DishID**

**Order\_detail:** This entity comprises of Dish\_detail and amount.

**Relationship**: It has one-to-one relationship with Dish\_detail.

**3) Payment Entity**

Figure 6 Order\_Detail Entity

**PK: PaymentID**

**FK: OrderID**

**CardID**

**Payment:** It stores the payment method the customer chooses in certain order.

**Relationship:** Hence it has one-to-one relationship with Card, coupon and Order.

**4) Coupon Entity**

**PK: CouponID**

**FK: OrderID**

**Coupon:** In order to increase the complexity of the whole system, we add this entity so that customer could use to reduce the expense.

**Relatipnship:** One-to-one relationship with Order entity.

Figure 7 Coupon Entity

Company side:

**1) Employee Entity**

**PK: EmployeeID**

**Employee**: Store the information on employees.

**Relationship**: It has one-to-one relationship with dispatcher.

**2) Dispatcher Entity**

**PK: DispatcherID**

**Dispatcher:** Record the information about deliver guy like wage, which has one-to-many relationship with order and one-to-one with employee.

**Relationship:** It has one-to-one relationship with Order.

The employee entity is designed to act liked a interface for compant’s further use to save the information for other staff like sales person.

**3) Wage Entity**

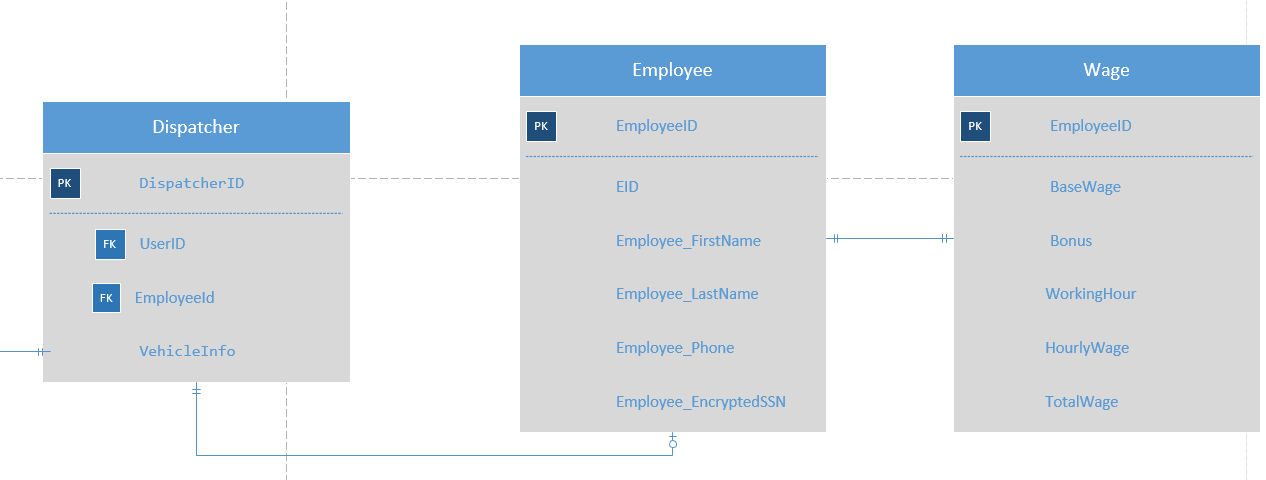
**PK**: **EmployeeID**

Figure 8 Dispatcher & Employee Entity & Wage Entity

**Wage:** Contain base wage, bonus, working hour, hourly wage and total wage of one employee.

**Relationship:** It has one-to-one relationship with Employee.