# Design Service and Interfaces of Restaurant

1. Description

Online Restaurant system can be used by the customer to browse the menu. Customer can order items from the menu either for pickup or delivery. Customer pays for the order. Customer can also book for in-dining reservation with number of people for future timing. Customer can walk into the restaurant be seated or put on the waiting list with a call back number.

1. Functional Requirements
   1. When showing the next available delivery or pickup time, it should be based on the pending order workload on the kitchen staff.
   2. People waiting in the queue to be called back from the head of the queue to be fair.
   3. Customer payment should be secure and database ACID compliant.
2. Design consideration
   1. For simplicity we are assuming that all restaurant chain has same menu and price. Same menu for Lunch and dinner.
   2. Traffic could spike during lunch and dinner time so system should be able to handle it.

Discovery Server

1. System APIs

CookService

Web client

API Gateway

OrderService

PaymentService

ActiveMQ

Oracle Database

OrderService: Takes the order from the customer. Sends the order to Event Bus with MENUITEM\_ORDERED when SendOrders is called. When MENUITEM\_COOKED gets published on the event bus by CookService adds items to menuItems in the OrderService. When DeliverMenuItems is called publish MENUITEM\_DELIVERED to Event Bus. Then clears the menuItems.

CookService: When MENUITEM\_ORDERED gets published on the event bus by OrderService starts cooking the item and publish MENUITEM\_COOKED to Event Bus when done cooking.

PaymentService: When MENUITEM\_DELIVERED published on the event bus Processes payment for the item.

MenuService: This will give menu for the given restaurant. For simplicity menu is same for all restaurant.

Customer-Service: Will give information about the Customer.

Create each service as a SpringBoot web application. Then have them discover each other using the Eureka Discovery server. Make them send messages using ActiveMQ to interact between them. Expose all the services as REST/WebClient APIs. I tried doing this in <https://github.com/navadapps/mc> but did not complete.

Then in [https://github.com/navadapps/RestaurantMC I used net.engio.mbassador](https://github.com/navadapps/RestaurantMC%20I%20used%20net.engio.mbassador) to create an EventBus to which services could subscribe and publish events. Then act on the event as needed.

1. Database Design

|  |  |
| --- | --- |
| Menu | |
| PK | Menu\_id int |
|  | Name varchar2(256) |
|  | Description varchar2(512) |
|  | Price numeric |

|  |  |
| --- | --- |
| Customer | |
| PK | Customer\_id int |
|  | Name varchar2(256) |
|  | Password varchar2(200) |
|  | Email varchar2(256) |
|  | Phone varchar2(16) |
|  | Address varchar2(256) |

|  |  |
| --- | --- |
| Order | |
| PK | Order\_id int |
| FK | Customer\_id int |
|  | Delivery\_type tinyint (0 – dine-in, 1 – pick up, 2 – delivery) |
|  | Description varchar2(512) |
|  | Price numeric |
|  | Timestamp timestamp |

|  |  |
| --- | --- |
| Order\_item | |
| PK | Order\_id int |
| FK | Menu\_id int |
|  | Quantity int |

|  |  |
| --- | --- |
| Payment | |
| PK | Order\_id int |
|  | Credit\_card\_no numeric |
|  | Name\_on\_card varchar2(256) |
|  | Expiration Date |
|  | Cvn tinyint |
|  | Address varchar2(256) |
|  | Timestamp timestamp |

|  |  |
| --- | --- |
| Restaurant | |
| PK | Restaurant\_id int |
|  | Name varchar2(256) |
|  | description varchar2(512) |

1. Changes for the pandemic

Can add contactless delivery option. For dine-in can provide outside seating, social distancing between the table and number of people at a table. Have table with enclosed transparent box around the table. Put hand sanitizer in most places. Put notice to wear mask while not eating. During the payment we can add option to donate for Covid-19 charities