**Master Data**

**Food Category**

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Description** |
| Food category name | Yes |  |
| Food category description | No |  |

**Food**

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Description** |
| Food name | Yes |  |
| Price | Yes |  |
| Image | No |  |
| Food description | No | Describe ingredient, formula, usage |
| Food category | Yes |  |

**Table**

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Description** |
| Table name | Yes |  |
| Session | Yes | The session it belong to |
| Table type | Yes | There are 2 type of table type: normal and VIP |

**Position**

For this version, we currently hardcode 4 position:

* Waiter
* Boss
* Cashier
* Controller

**Session**

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Description** |
| Session name | Yes |  |
| Session belong to | Yes | The session it belong to |
| Session capacity | No | The capacity of session about the table |

**Functional Requirement Requirement**

**Feature**

**Upload logo and Change Skin**

Administrator can change their logo and skin due to their business.

**Session Management**

This feature use to show current structure of restaurant including Session and the table belong to that session.

User can add/edit/delete table in each session.

This feature is included in to Session master data.

**Password Confirmation**

With important action, system will require Admin user input their password as confirmation:

|  |  |
| --- | --- |
| **List of action** | **Note** |
| Delete user account |  |
| Modify entry in invoice |  |

**Reset Password**

Incase user forgot their password, admin will reset their password. The password reset will be fixed value ‘0123456789’, after reset password, user has to change their password for the first time login.

**Notification**

When Controller inform about the status of food, in case the ordered food is out of but still in waiting list order. System will auto poppulate notification to waiter about the food out of.

The format for this notification is that:

*No of Table*

*Food Name Quantity Not Available*

Sample of notification

*Table No.2*

*Fried Chicken 2 Not Available*

**Reserved Table**

Each table has 3 status:

* Available: the table is available to server.
* In Use: the table is used to server customer.
* Reserved: the table is reserved.

The table can be reserved. The reseved table cannot be order food till it transfer to ‘Available’ status.

Note that: the reserved table only can be changed to ‘Available’ status by the user booked it.

**Undo**

Incase the waiter take wrong merge, split, change table operation, they can undo it.

**Process Order**

**UC: Payment**

|  |  |
| --- | --- |
| **UC ID** | PAYMENT |
| **Brief Description** | Process payment for order |
| **Actor** | Cashier, Manager, Waiter |
| **User Goal** |  |
| **Precondition** |  |
| **Trigger** |  |
| **Main Success Scenario** |  |
| **Alternative Flows** |  |
| **Exception** |  |
| **UC Relationship** |  |
| **Page/Screen** |  |
| **Business Rule** |  |
| **Technical Note** |  |

**UC: Process Order**

|  |  |
| --- | --- |
| **UC ID** | PROCESSORDER |
| **Brief Description** | Use to create order for customer |
| **Actor** | Waiter |
| **User Goal** | Create order by Waiter |
| **Precondition** |  |
| **Trigger** | Click to order |
| **Main Success Scenario** | 1. Customer will view menu and choose their own meals. 2. Waiter check the available of food and choose in app. 3. After choosing, waiter will confirm to customer and click ‘Order’ button to send order to Controller. All food will be send to Controller screen. 4. Order will be generated together with OrderID. OrderID will be generated follow format. 5. When the customer finish their meal, waiter will click to ‘Finish’ button to transfer to payment process. |
| **Alternative Flows** | 4a. Waiter can cancel order with the agreement between restaurant and customer.  With the food has status ‘cooking’ or ‘cooked’, controller has to confirm that this food is not served to customer and still in restaurant.  The ‘Tracking Order’ will be saved after ‘Actual Order’ be canceled.  Data of ‘Tracking Order’ will be stored like in **UC PAYMENT.**  4b. If customer requires more foods, waiter will check the its available on system and add more food.  4c. If customer want to change table, refer to **UC CHANGETABLE**.  4d. If customer want to merge table, refer to **UC MERGETABLE**.  4e. If customer want to split table, refer to **UC SPLITTABLE**. |
| **Exception** |  |
| **UC Relationship** | **PAYMENT, CHANGETABLE, SPITTABLE.** |
| **Page/Screen** |  |
| **Business Rule** |  |
| **Technical Note** | **Order ID format**:  DDMMYY-Table No-WaiterID |

**UC: Change Table**

|  |  |
| --- | --- |
| **UC ID** | CHANGETABLE |
| **Brief Description** | Allow customer change their current table |
| **Actor** | Waiter, Customer |
| **User Goal** | Change the table and other linked items from source table to destination table |
| **Precondition** | UC Process Order already executed |
| **Trigger** | Click to ‘Change Table’ |
| **Main Success Scenario** | 1. Customer requires to change their table. 2. Waiter will click to ‘Change Table’ button. 3. All table with status ‘Available’ will be highlighted. Waiter can choose table. 4. Waiter click to ‘Confirm’ button. 5. All meals/item that customer order in this system will be transferred to new table. 6. Waiter has to choose the waiter in charge with new table. 7. New waiter has to confirm in system about this transaction. 8. The order of customer’s order still unchange in Controller’s screen. 9. The format of Order ID will be changed the Waiter ID value. 10. The ‘Tracking Invoice’ of this order is recored this transaction. |
| **Alternative Flows** |  |
| **Exception** |  |
| **UC Relationship** | **PROCESSORDER** |
| **Page/Screen** |  |
| **Business Rule** |  |
| **Technical Note** |  |

**UC: Merge Table**

|  |  |
| --- | --- |
| **UC ID** | MERGETABLE |
| **Brief Description** | Allow customer merge their table with current table |
| **Actor** | Customer, Waiter |
| **User Goal** | Merge 2 tables into 1 table |
| **Precondition** | UC Process Order already executed |
| **Trigger** | Click to ‘Merge Table’ button |
| **Main Success Scenario** | 1. Customer require to Merge their table. 2. Waiter will click to ‘Merge Table’ button, all In Use table will be highlighted. 3. Waiter will choose the destination table and confirm. 4. All meals and linked item will transferred to destination table. 5. The waiter of destination table has to confirm in system about this transaction. 6. The order of customer’s order still unchage in Controller’s screen. 7. The format of Order ID will be changed the Waiter ID value. 8. The ‘Tracking Invoice’ of this order is recorded this transaction. |
| **Alternative Flows** |  |
| **Exception** |  |
| **UC Relationship** | **PROCESSORDER** |
| **Page/Screen** |  |
| **Business Rule** |  |
| **Technical Note** |  |

**UC: Split Table**

|  |  |
| --- | --- |
| **UC ID** | SPLITTABLE |
| **Brief Description** | Allow customer split their table into 2 table |
| **Actor** | Customer, Waiter |
| **User Goal** | Split 1 table into 2 tables |
| **Precondition** | UC Process Order already executed |
| **Trigger** | Click to ‘Split Table’ button |
| **Main Success Scenario** | 1. Customer require to split 1 table into 2 tables. 2. Waiter click to ‘Split Table’. 3. All table are highlighted. Waiter choose the destination tables and the meal in current order. 4. The meal will be transferred to destination tables. 5. The waiter of destination tables has to confirm about this transaction. 6. The order of customer’s order still unchage in Controller’s screen. 7. The format of Order ID will be changed the Waiter ID value. 8. The ‘Tracking Invoice’ of this order is recorded this transaction. |
| **Alternative Flows** |  |
| **Exception** |  |
| **UC Relationship** | **PROCESSORDER** |
| **Page/Screen** |  |
| **Business Rule** |  |
| **Technical Note** |  |

**Security Requirement**

**Database Design of Security**

****

Security requirement will be developed in 2 statges:

**Stage 1**:

Role will be fixed, user cannot create new roles.

Each role in stage 1 will be pre defined in system with appropriate functions. The authorization of role will be described below:



**Stage 2:**

Role will be customize follow business of customers. They can configure by themselves about appropriate functionalities.

Apply Single Sign On.

*Stage 2 will be implemented in Phase 2.*