HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

~~~~~~~*~~~~~~~

SOFTWARE ENGINEERING

Wearable Applications For iRestaurant

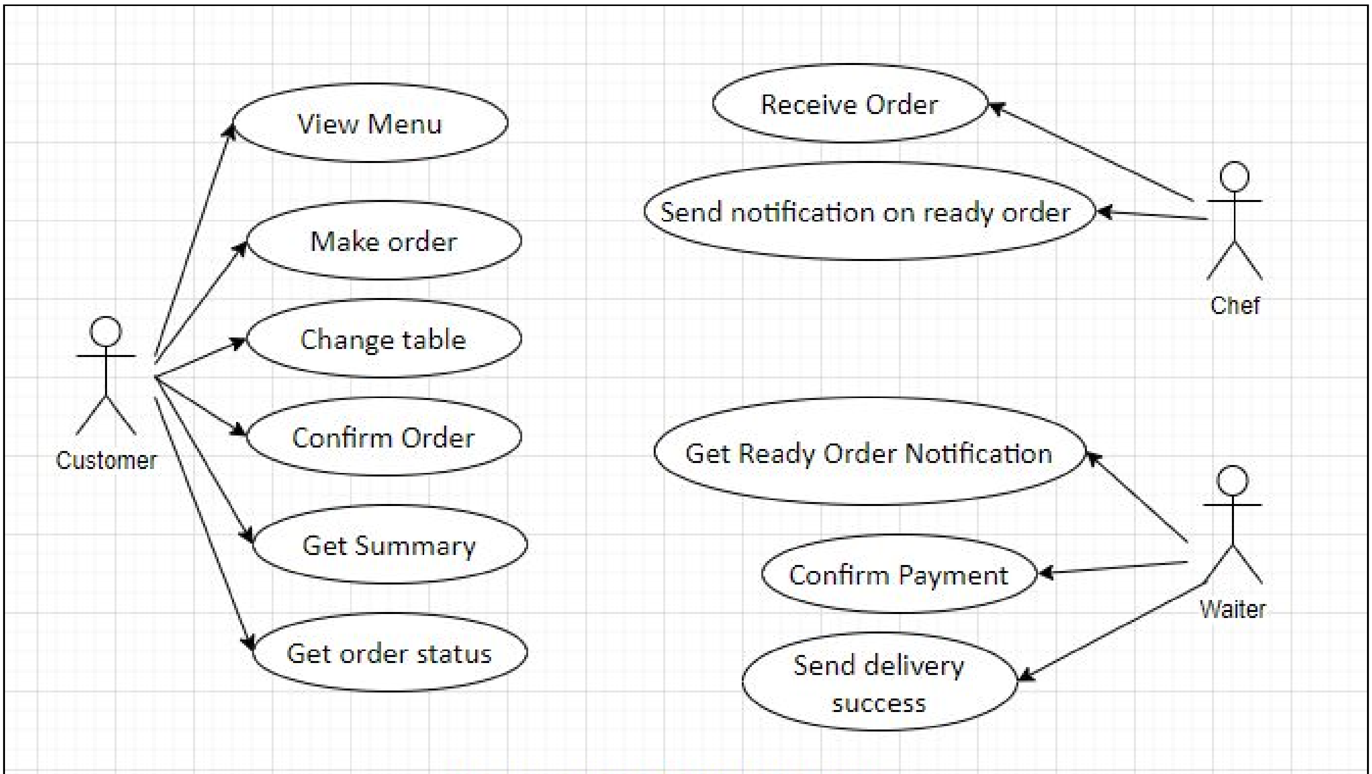
Lecturer: **Dr. Lê Lam Sơn**

Lý Quốc Hải	**1852348**
Nguyễn Hoàng Long	**1852532**
Dương Đình Trung	**1814498**
Đặng Gia Lệ	**1812791**
~~~~~~~

Take a look at the attached figure!

It depicts the overall architecture of the iRestaurant system that consists of a server and two wearable devices. The server allows the restaurant chef to manage all meal orders placed by the clients. This module, being run on a computer in the restaurant kitchen, serves the two wearable devices described further below. The clients may place their order using a wearable that is of mobility, i.e. can be moved around on their restaurant tables. This food-ordering device manages a list of food and drink items. The client should be able to browse/view them and later on add them to their meal order. Each food item is described having a name, a price and a short textual description. The clients get the summary of their order including the total amount of the meal and an estimated period of time they have to wait for their meal. Once the clients confirms of their order, this wearable communicates with chef's module to inform the restaurant chef of the newly-placed order. The second wearable of iRestaurant is designed to serve restaurant waiters. A restaurant waiter is supposed to use his wearable to communicate with the system twice: when the client's order is ready for delivery and when he has delivered it. The clients may pay for their order using their credit card or in cash upon meal delivery. In addition to LCD used for displaying menu, the two wearable comes with controls such as buttons and lightings for showing the status of the client's order (e.g., confirmed, cooking, delivering, delivered).

1.Use case diagram



Use case diagam of iRestaurant system

Use-case name	**View Menu**
Actor	Customer
Description	Customer views the menu of each type of food and drink and their cost
Preconditions	
Normal Flow	1. Customer press “View Menu” on the device’s interface 2. Customer press on the interface “1” to select the food list or “2” to select drink list. 3. A list of item will be shown on the screen with each item item is described having a name, a price and a short textual description. 4. Customer press “View Menu” to hide the list.
Exceptions	Exception 1: At step 2: If an item is out of stock, the screen will not show the description of the item on the menu list.
Alternative Flows	

Use-case name	**Make Order**
Actor	Customer
Description	Customer make an order by adding and/or removing items in the order list
Preconditions	The Customer’s device has a food list or drink list shown.
Normal Flow	1. Customer chooses a food or drink based on their numerical order on the menu. 2. Customer may choose: a) Customer press “Add” on the device’s interface to add that item into the Customer’s order. b) Customer press “Remove” on the device’s interface to remove that item out of the Customer’s order. 3. If there is at least an item in the Customer’s order, the order’s list is shown to screen and will contain the item’s name, a name, a price and a short textual description 4. Then loop back step 2.
Exceptions	Exception 1: At step 2b: If there are no such item in the Customer’s order, the system will not remove anything from the order.
Alternative Flows	Alternative 1: At step 2b: Customer can press the button “delete all" to make the list of order empty

Use-case name	**Change table**
Actor	Customer
Description	Customer choose the index of the table they are sitting when they move to another table.
Preconditions	Customer had found a seat and know the table index.

Normal Flow	1. Customer press “Choose chable” on the device’s interface 2. The screen presents a page with choice of table index. 3. Customer select the table index they are sitting 4. Customer press “OK” to confirmed the chosen index 5. The order which is made by this customer’ wearable will be updated the table index info
Exceptions	Exceptions 1: At step 4: If the customer do not press OK button, the table index of customer’s wearable and order will not be updated.
Alternative Flows	Alternative 1: At step 3: If the table was booked or is not empty, the index will be inactive so that customer can not choose.

Use-case name	**Get Summary**
Actor	Customer
Description	Customer gets the summary of their order including the total amount of money of the foods and drinks as well as the estimated period of time they have to wait for their meal.
Preconditions	There is at least one item in the Customer’s order.
Normal Flow	1. Customer press “Summary” on the device’s interface. 2. The system will calculate the sum money of all the items in the order and the estimated amount of time when the order finish 3. The total time and amount of money will be shown on the screen
Exceptions	Exception 1: At step 2: If there are no items in the order, the sum of money will be zero.
Alternative Flows	

Use-case name	**Confirm Order**
Actor	Customer
Description	Customer confirms the order
Preconditions	There is at least one item in the Customer’s order
Normal Flow	1. Customer at the Customer’s order tab which show list of order with total time and money. 2. Customer pressed “Confirm” on the device’s interface to confirm their order. 3. The system will enter the Customer’s order into process. 4. The waiter’s wearable send a request to chef's module to inform the restaurant chef of the newly-placed order 5. The LED lighting changes color, update the status of order to “Confirmed”
Exceptions	Exception 1: At step 2: If there are no items in the Customer’s order, the system will not confirm the order.
Alternative Flows	Alternative 1: At step 2: If customer press “Cancel”, all the items in order list will be removed.

Use-case name	**View Food Status**
Actor	Customer
Description	Customer may view the status of the order through the LED lighting on device
Preconditions	Customer already has an order that is processing
Normal Flow	1. Customer press “View Status” on the device’s interface. 2. Depends on the stage of the order (confirmed, cooking, delivering, delivered), the light on the device will turn on the respective color that indicate the stage.
Exceptions	
Alternative Flows	Alternative 1: At step 1: Customers can press the “turn off LED” to turn off the lighting if they don’t want to view the food status

Use-case name	**Receive Order**
Actor	Chef
Description	Chef receive newly created order from server after new order is made by Customer device.
Preconditions	- There is at least one Customer’s order that has not been cooked. - Server responses successfully to the chef query.
Normal Flow	1. The Chef computer query the server whether there is any new order made by customer. 2. The computer of Chef notifying the Chef by a pop-up in the window. 3. The new order is pushed to the rear of the queue and displayed at the list of orders that need to be made. 4. The order which is at cooking process will be updated the status to “Cooking”
Exceptions	Exception at step 1: If after 1 hour, there is no new order, the system notify the chef to check whether anything wrong happened to the system.
Alternative Flows	

Use-case name	**Notify Order**
Actor	Chef
Description	Chef notifies to the system that an order is finished and ready to be delivered.
Preconditions	Chef finished a Customer’s order
Normal Flow	1. Chef click the Finish button on his screen. 2. The system will update the order status to “Ready” 3. The order is then removed from the screen of the chef

	4. The system will send a signal to water's wearable that the order is ready to deliver
Exceptions	Exception at step 2: If the signal fail to update the order status in the server, then an error message should be shown up on the chef screen.
Alternative Flows	

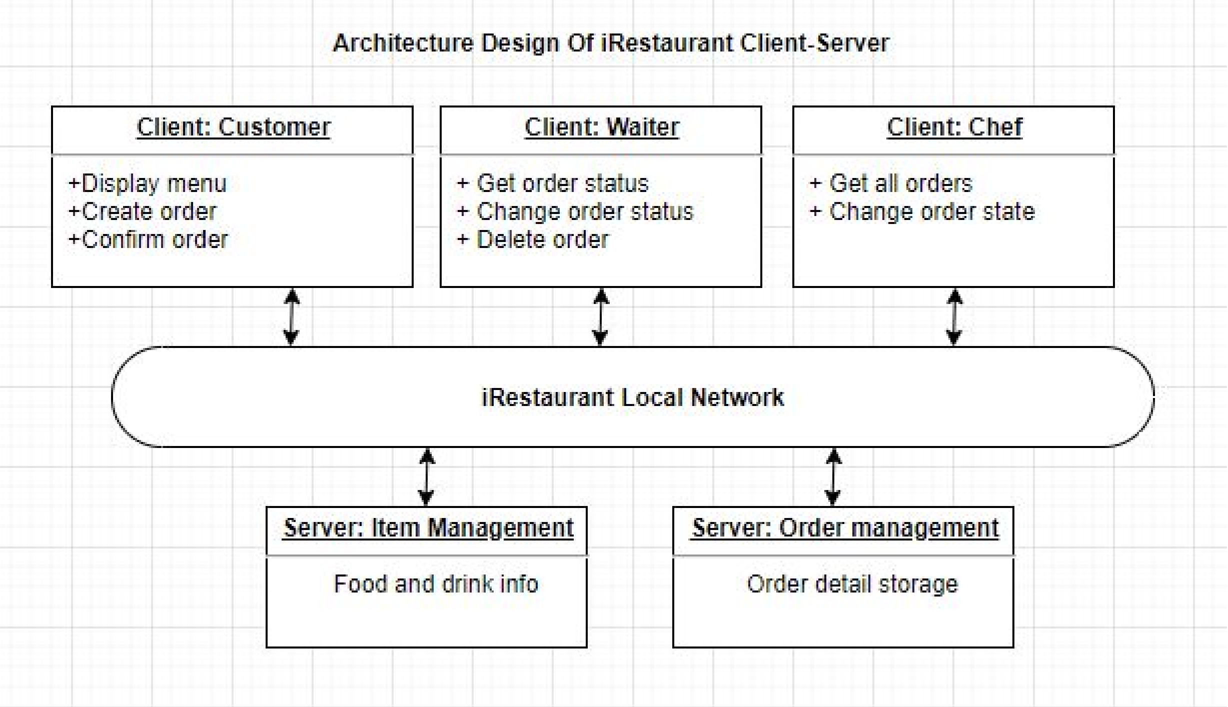
Use-case name	**Get Ready Order Notification**
Actor	Waiter
Description	Waiter 's wearable device receives notification from the system notifying that order has been done.
Preconditions	- There is at least one order on the serve table - The waiter is not serving any order - The chef notified the system that an order is finished
Normal Flow	1. Waiter's wearable devices receives a signal from the system . 2. The light on waiter's wearable device is changed from green into red. 3. The table ID of the order need to be delivered will be show on waiter's wearable
Exceptions	
Alternative Flows	

Use-case name	**Send Delivery Success**
Actor	Waiter
Description	Waiter sends the signal to the system informing that the order finishes delivery to the customer.
Preconditions	The waiter is serving an order.
Normal Flow	1. The lighting of the the device will be blue. 2. When the Waiter has served the order, Waiter press "Finish" to notify the system that the order is delivered to the Customer successfully. 3. The order status is updated to "Delivered"
Exceptions	
Alternative Flows	

Use-case name	**Confirm Payment**
Actor	Waiter
Description	Waiter confirms that Customer has paid the order's bill

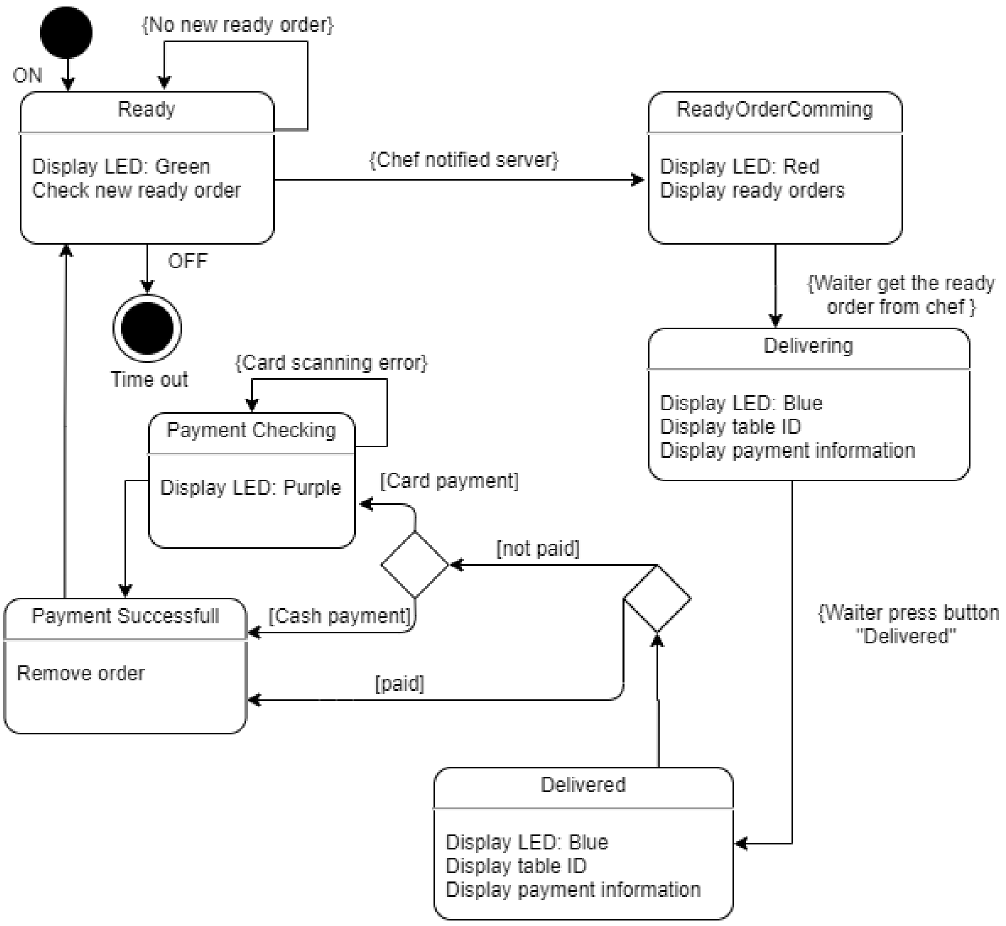
Preconditions	Customer receives the right order
Normal Flow	1. Waiter ‘s wearable device receives the payment signal. 2. Waiter chooses the table ID, the screen will show the total amount of money 3. Waiter chooses method of payment (credit card or in cash)which client using to pay for their bill. 4. When the payment is successful, the waiter press “ Successful payment” to confirms with the system that the bill is paid. 5. The order status is updated
Exceptions	
Alternative Flows	Alternative 1: At step 1: Customer can pay with credit card instead of cash. Alternative 2: At step 1: If the Customer had paid for the food before going to the restaurant, the system will automatically confirms with the Waiter that the bill is paid.

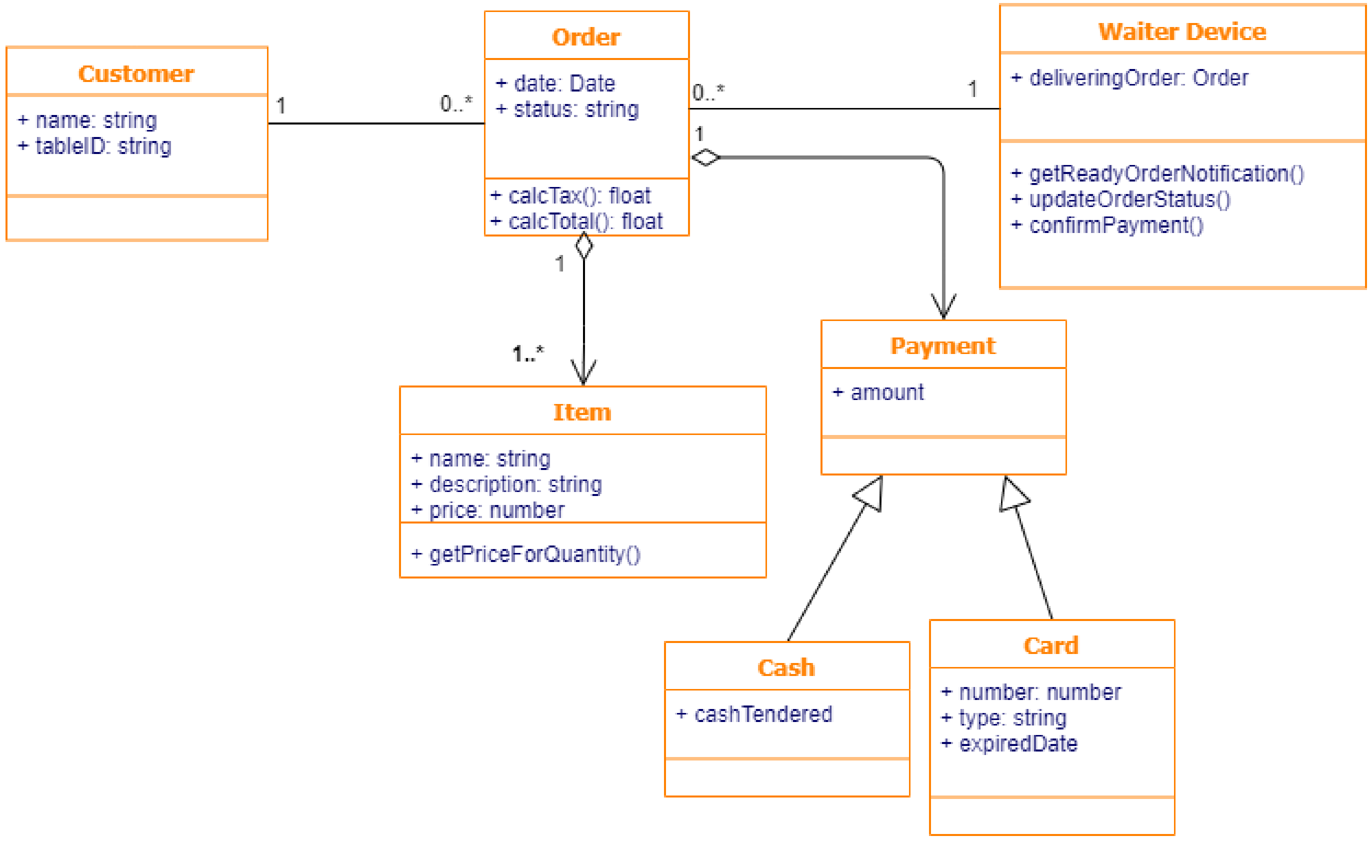
2. Architectural design



3. Statechart diagram and class diagram for the waiter's wearable

UML state diagram for Waiter wearable device





Customer
+ name: string
+ tableID: string
1
0..*
Order
+ date: Date
+ status: string
+ calcTax(): float
+ calcTotal(): float
0..*
1
Waiter Device
+ deliveringOrder: Order
+ getReadyOrderNotification()
+ updateOrderStatus()
+ confirmPayment()
1
1
1..*
Item
+ name: string
+ description: string
+ price: number
+ getPriceForQuantity()
Payment
+ amount
Cash
+ cashTendered
Card
+ number: number
+ type: string
+ expiredDate