HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

**FACULTY OF COMPUTER SCIENCE AND ENGINEERING**

****

PROJECT REPORT

PRACTICE ON SOFTWARE ENGINEERING

**RESTAURANT POS 2.0**

Lecturer: Assoc. Prof. Quan Thanh Tho

| Tran Duc Thien | 1952998 |
| --- | --- |
| Ly Kim Phong | 1952916 |
| Tran Quoc Hoan | 1952051 |
| Nguyen Thanh Chuong | 1952595 |
| Le Thanh Dinh | 1952027 |

Contents

[***Task 1***: 3](#_heading=h.30j0zll)

[Changelog 8](#_heading=h.1fob9te)

## ***Task 1***:

##### *Context:*

* The POS or point of sale is when and where customers select something, add to cart and then check out for payment.
* At the point of sale, the merchant calculates the amount owed by the customer, indicates that amount, may prepare an invoice for the customer, and allows the customer to make payment(cash/credit card/apps).
* In the Sar-Cov-2 pandemic, indirect contact is advised, PoS application should be promoted to reduce the number of times a waiter has to face to face communicate with the customers, which is beneficial for both sides.
* PoS system allows for ordering process, feedback, status notification when the restaurant has menu updates or other information and credit card payment transaction (online/offline).
* Such systems are expected to increase business intelligence, reduce wasted effort and the opportunity to scale as the business grows.

=> So we decided to build a responsive Web app to serve the largest number of guests, ease of use and easy to maintain.

Relevant stakeholders: Customers, Res Owner, Clerks, Chefs.

Jobs: Display menu, meal select, remove from stock, order (placing, confirm, process, reject, deliver), waiting queue(food is done or not), feedback, logging order, paying methods (online(credit, third-party app) & cash).

Scope: 300 customers/day, serve 8am-10pm, 21 cus/hour, 1-2 orders at the same time, 1 restaurant.

##### Requirements:

Functional:

View list of restaurant: customer can see the list of available restaurants of the owner as well as the address(if to be scaled).

View menu: customer can see the menu of the restaurant, updated to the minute.

Placing order: customer can ordering food include search food they want and select it

View list of order: customers can preview the food they have chosen before ordering and payment.

Notifications of order: customers can know whether the course is being made or cancelled.

Feedback: After eating food at the restaurant or at home, customer can feedback about the food on the website

Non functional:

Usability: The system should allow non-direct contact between Clerks and Customers. Moreover, right on the menu page will have information on the status of food reviews.

User interface: The system should be implemented using Web technology and QR code, so customers will not have to install apps. The website should be responsive so that it can be displayed on different devices.

Scalability: The system should be extendable to use in multiple restaurants in the future. Capacity: The current transactions are about 300 orders per day.

Serviceability: The quality of being able to provide good services, high responsive interaction, and customers can give feedback on their experience to the clerk for better customer service.

Feedback: Multiple choice form for customers to check the quality of services, additional feedback and complaints. May update the system to post food reviews and critics later if time allows.

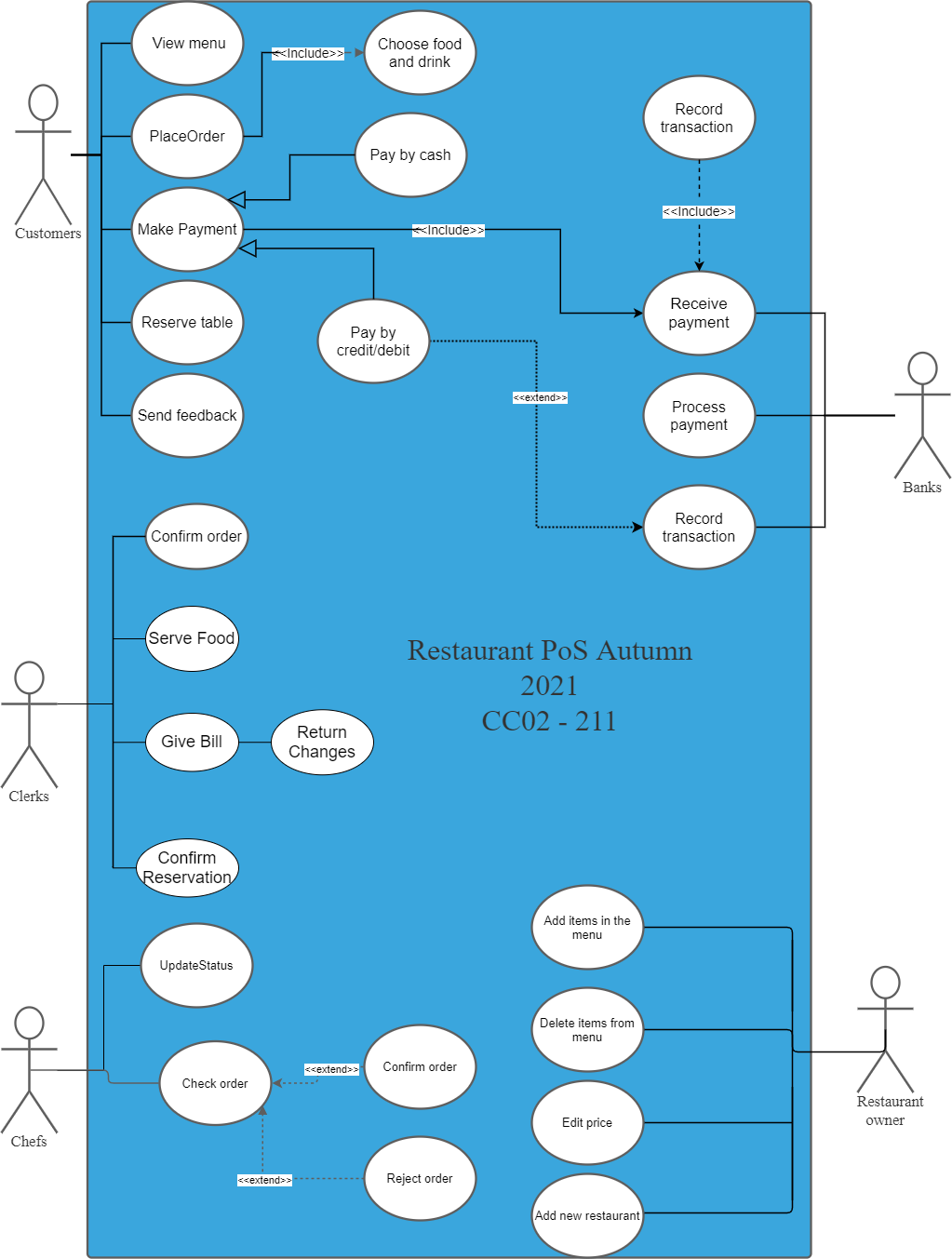
Security: When the user makes payment, the system is not allowed to store credit/debit card information of the user.

Maintainability: The web must be easy to maintenance, avoiding DDOS attacking from competitive services.

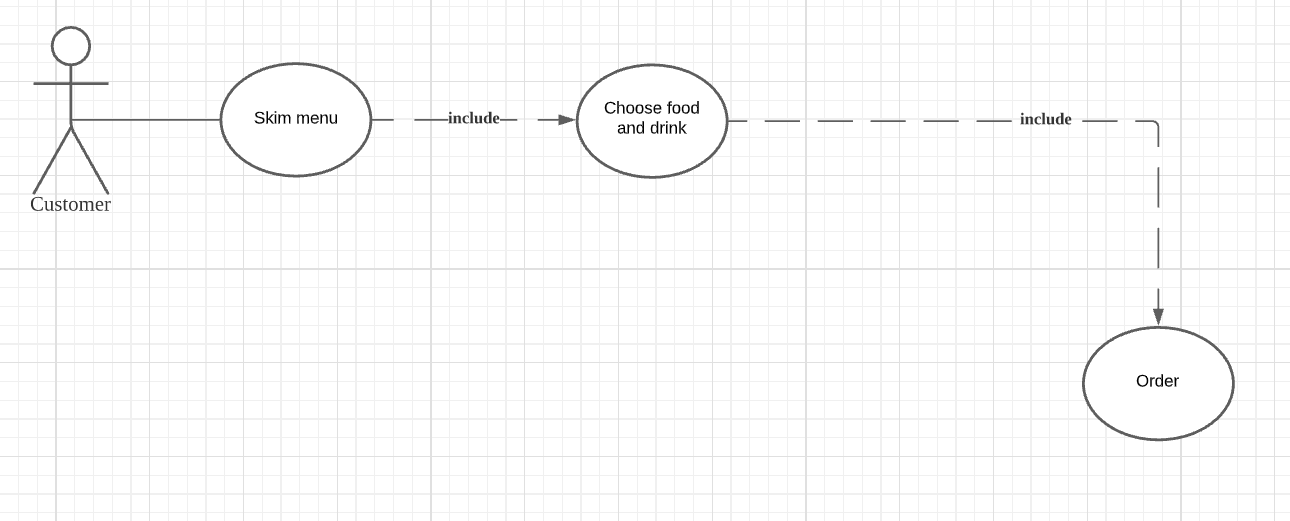
Ver 1: 18/9/2021(deleted)

Ver 2: 23/9/2021([link](https://viewer.diagrams.net/?page-id=Hcg7Q0vDsgE8VdpAob2g&highlight=0000ff&edit=_blank&layers=1&nav=1#G1Mt8L0EeDxM_pB8qi_fikuVrxQIUpPcJg))

Ver 3: 07/10/2021([link](https://viewer.diagrams.net/?page-id=pXoU4k3pK258AMGFvbBh&highlight=0000ff&edit=_blank&layers=1&nav=1&page-id=pXoU4k3pK258AMGFvbBh#G1Mt8L0EeDxM_pB8qi_fikuVrxQIUpPcJg), below)



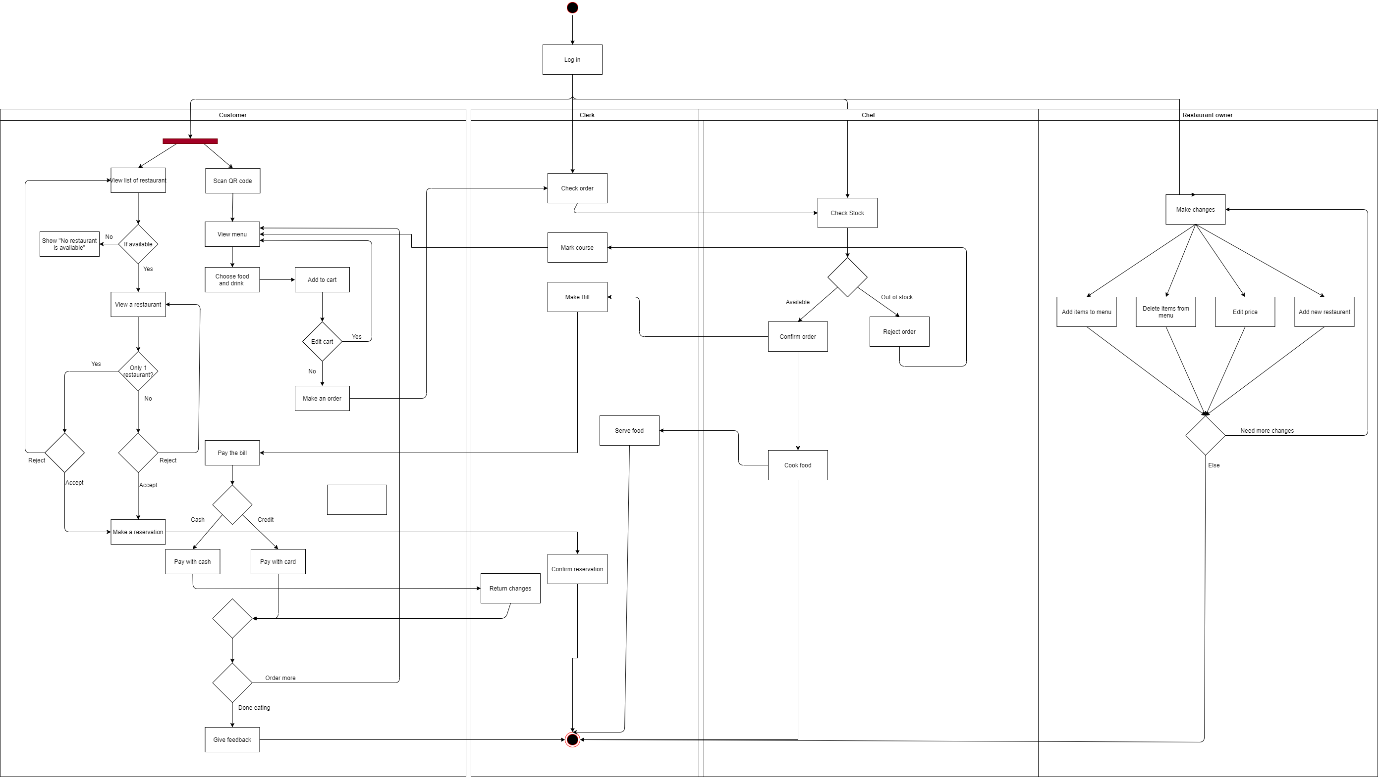
##### *Food ordering*



| Name | Food ordering |
| --- | --- |
| Actor | Customers |
| Description | When ordering food, people can skim through the menu choosing their food and drink. After that they can order the food and wait until the dishes are ready. They can return to the menu to order new food. |
| Preconditions | Customers need to access the home page. |
| Normal flow | 1. Customers go to the home page by URL or QR code.  2. Customers can search for the desired food and drink on the menu.  3. Customers put the food and drink they want in the cart.  4. Customers make the order so the restaurant can prepare the dish.  5. Customers are back to the menu to continue skimming. |
| Exceptions | Exception at step 4: Customers want to choose more food and drink before ordering. |
| Alternative flow | Customers can go back to step 2 |

## Task 2:

#### Task 2.1: Draw an activity diagram to capture Major functional requirements of the desired system.



[Link to the diagram.](https://viewer.diagrams.net/?page-id=e7e014a7-5840-1c2e-5031-d8a46d1fe8dd&highlight=0000ff&edit=_blank&layers=1&nav=1&hide-pages=1#G1662ygfhhLBS8YC1OaAdpUHDNAeFx4APS)

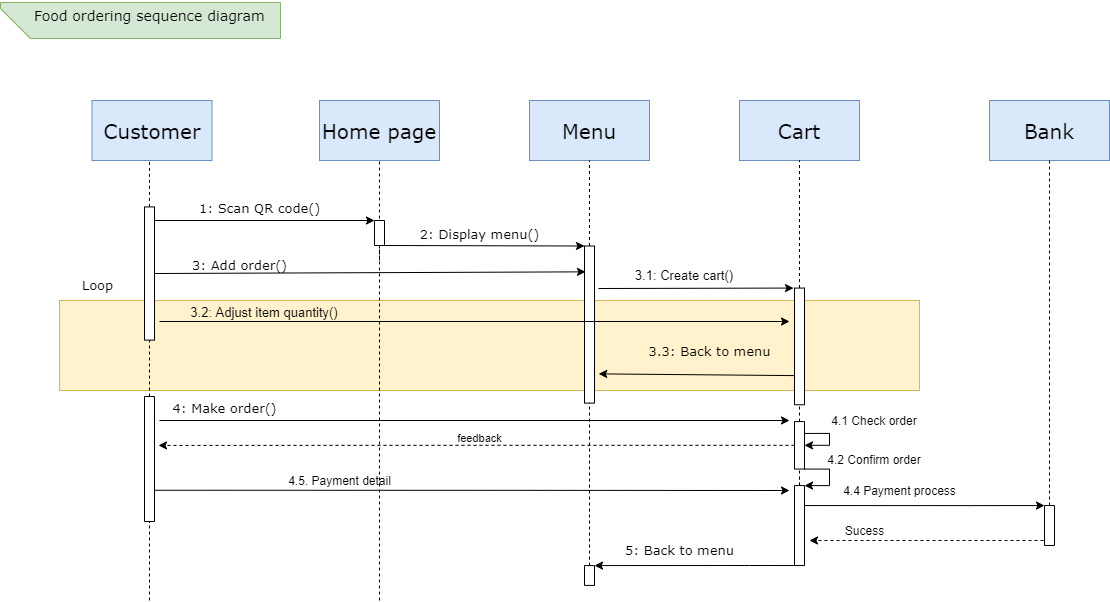
| Name | Reservation |
| --- | --- |
| Actor | Customers, Clerks |
| Description | Customers can view the list of available restaurants. They can pick a restaurant to make a reservation, and this will be confirmed by the clerk later. |
| Preconditions | Customers need to access the home page. |
| Normal flow | 1. Customers view the list of restaurants.  2. Customers choose to view a specific restaurant.  3. Customers make a reservation  4. The clerk will confirm the customer’s reservation |
| Exceptions | Exception 1: at step 1: If there is no restaurant .  Exception 2: at step 2: If there is only 1 restaurant .  Exception 3: at step 3: If there’re more than 1 restaurant and the customer get rejected. |
| Alternative flow | For exception 1, 2: the system will display “No restaurant is available.  For exception 3: customers can go back to step 2. |

| Name | Food ordering |
| --- | --- |
| Actor | Customers, Clerks, Chefs |
| Description | When ordering food, people can skim through the menu choosing their food and drink. After that they can order the food and wait until the dishes are ready. They can return to the menu to order new food. |
| Preconditions | Customers need to access the home page. |
| Normal flow | 1. Customers go to the home page by URL or QR code and the system will show the menu.  2. Customers can search for the desired food and drink on the menu.  3. Customers put the food and drink they want in the cart.  4. Customers make the order so the restaurant can prepare the dish.  5. The clerks will check the order and notify the chef.  6. The chefs will check for stock and prepare the dish.  7. The clerks will notify the food is ready and serve the dish.  8. The clerks make the bill.  9. The customers pay the bill.  10. The clerk will return the changes if any.  11. The customers give feedback. |
| Exceptions | Exception 1: at step 4: the customers want more food before ordering.  Exception 2: at step 6: the dish is out of stock. |
| Alternative flow | For exception 1: Customers can go back to step 2.  For exception 2:  the clerks will mark the course and customers go back to step 2. |

| Name | Make changes |
| --- | --- |
| Actor | Restaurant owner |
| Description | The owner wants to make some adjustments to the menu and include a new restaurant into the system. |
| Preconditions | The restaurant owner needs to access the home page. |
| Normal flow | 1. The restaurant owner logs in.  2. The restaurant owner will check if there are changes to be made.  3. The restaurant owner adds items to the menu, deletes items from menu, edits price, adds new restaurant. |
| Exceptions | At step 3, the owner is not satisfied with the change made. |
| Alternative flow | The restaurant owner can go back to step 2. |

#### Task 2.2: Draw a sequence diagram for use-case in Task 1.3

In this case, a food ordering diagram.



[Link to the diagram.](https://viewer.diagrams.net/?page-id=9361dd3d-8414-5efd-6122-117bd74ce7a7&highlight=0000ff&edit=_blank&layers=1&nav=1#G1pTpbitxA0Cyh5wxYQcxVh10cBTaO7Ns3)

| Name | Food ordering |
| --- | --- |
| Actor | Customers |
| Description | When ordering food, people can skim through the menu choosing their food and drink. After that they can order the food and wait until the dishes are ready. They can return to the menu to order new food. |
| Preconditions | Customers need to access the home page. |
| Normal flow | 1. Customers go to the home page by URL or QR code.  2. Customers can search for the desired food and drink on the menu.  3. Customers put the food and drink they want in the cart.  4. Customers make the order so the restaurant can prepare the dish.  5. Customers are back to the menu to continue skimming. |
| Exceptions | Exception at step 4: Customers want to choose more food and drink before ordering. |
| Alternative flow | Customers can go back to step 2 |

#### Task 2.3. Draw a class diagram

Customer’s interaction:

Customers attributes include:

* Name: protected, string, as customer name should not be accessible to outsider using the app
* ID: protected, int, an assigned number to that customer only, used for identify, and later implement the bonus point system.
* Phone: protected, string, to support the advertising department, they can use this field to inform customer of recent coupons and events.
* Reservation: protected, Boolean, whether there is a name in the system who reserve table, default is False.
* ReservationName: protected, string, name of the people called to reserve table, default is the same as the name.

Customers have the following methods:

* MakeOrder(), public, customers can add items to cart through this method.
* Makepayment(), public, customers pay through this method.
* ReserveTable(), private, customers, reserve table in advance using this.
* Feedback(), private, customers give feedback after
* CheckStatus(), public, check whether the dish is ready.

Menu:

Attributes:

* Name: public, string, name of the course.
* Available: public, Boolean, whether that disk is available or not at the time of checking.
* Price: public, string, price of that dish.

Methods:

* IsVegan()
* IsStarter()
* IsIntermediate()
* IsMainDish()
* IsDessert()
* IsSpecial()

Above methods are all public, to sort the menu in that category.

Menu Items:

Attributes:

* Name: public, string, name of the course.
* Available: public, Boolean, whether that disk is available or not at the time of checking.
* Description: public, string, describe the food shortly.
* Price: public, string, price of that dish.

Methods: None

Cart:

Attributes:

ID: public, int, id of the order.

+ Charge: public, int, amount of money the order cost.

+ Discount: public, int, any type of discount appliable.

+ Item: public, array of string, list of items in the cart.

+ NoItem: public, array of int, list of numbers of items.

+ FinalPrice: public, int, the amount of money the customer has to pay.

+ TypeOfPayment: public, Payment, the method of paying, refer to payment class.

+ Table: public, Table, anything relate to table, refer to Table class.

+ Status: public, string, status of the order, respectively pending, paid, serving, completed.

Methods:

Pay(): redirect to the payment class.

Cancel(): cancel and set the status back to pending.

AddMore(): call Cancel() method and return to the menu to add more items.

Table

Attributes:

+ ID: public, int, id of the table.

+ PairingOpt: public, bool, table for couples or not.

+ Reserved: public, bool, customer can see whether the table is reserved or not.

+ Capacity: public, int, the capacity of the table.

Chef:

Attributes:

+ Name: public, string, name of the chef.

+ ID: public, int, id of the chef.

+ Phone: protected, string, phone number of the chef.

+ Position: public, string, position of the chef.

Methods:

CheckOrder(): receive the order and call UpdateStatus() to check that order received

UpdateStatus(): update the status of order, respectively received, cooking, served

RejectOrder(): reject the items in the order

Owner:

Attributes:

Name: public, string, name of the owner.

Admin pass: private, string, password of admin to change the system.

Phone: private, string, phone number of the owner.

Methods:

Add\_course(): add a new dishes

Delete\_course(): delete an existing dishes

Update\_price(): change the price

Add\_newRes(): add new restaurant to the system

Update\_system(): close system to update

Close\_system(): shut down the system

Clerk:

Attributes:

+ Name: public, string, name of the clerk

+ ID: public, int, id of the clerk

+ Phone: public, string, phone number of the clerk

+ Position: public, string, position of the clerk

Methods:

CheckOrder(): check the order received or not.

UpdateStatus(): update the status to whether ready to serve or not.

RejectOrder(): reject if the ingredient is not enough to make the dishes.

Payment:

Attributes:

+ Type: public, string, type of payment.

+ Amount: public, int, amount of money need to be paid.

Methods:

ChoosePayment(): choose to pay by cash or credit card.

PrintBill(): print the bill with payment information like date, name, and part of credit no.

Changes(): the amount of money paid to return to customers, for later statistics.

Cash:

Attributes:

+Amount: public, int, amount of cash.

+Change: public, int, changes after paid.

Credit/debit:

Attributes:

+Cardno: private, int, credit/debit card number

+ OwnerName: private, string, name of the card owner

+SecretCombin : private, int, the 3 secret no of the credit card to confirm possession.

+ ExpireDate: private, long, expiry date written on the card, for MasterCard payment.

Methods:

CheckPIN(): confirm the PIN entered, if wrong, return the error message.

GetBalance(): check the balance, if unable to pay or card locked, etc, reject the card.

Payment(): subtract the amount with a note about the id of the bill.

Confirm(): print the bill and end connection.

Authorized users:

Attributes:

+ Name: public, string, name of the users

# Personal ID: protected, int, id of that user

- Phone: protected, string, phone number of that user

+ Position: public, string, position of that person.

Methods:

+ Record\_complaint(): record the complaint of customers

+ CheckStatus(): check the status of customer’s order

[Link to the diagram.](https://viewer.diagrams.net/?page-id=C5RBs43oDa-KdzZeNtuy&highlight=0000ff&edit=_blank&layers=1&nav=1&hide-pages=1#G137tRxU9NFXvCQelav6SM3Q9j6RO7ajNk)

## Changelog

18/9/2021: Context, requirement and use-case diagram first created.

23/9/2021: Update on the format and some relationship of use-case diagram.

7/10/2021: Update on the usecase diagram after the review.