VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

SOFTWARE ENGINEERING

Assignment

Restaurant POS 2.0

Advisor: Dr. Quan Thanh Tho
Students: Dang Cao Cuong - 1952598
Tran Minh Vu - 1953107
Luu Chan Hung - 1952063
Pham Tan Phuoc - 1952406
Trinh Manh Hung - 1952740

HO CHI MINH CITY, AUGUST 2021

Contents

CHANGELOG

No	Date	Changes	Actors
1	24-Sep-21	"Introduction", "Stakeholders", "Project Scope" initialized	Dang Cao Cuong
		"Requirements" initialized	Pham Tan Phuoc
		"Use case for the whole system" initialized	Tran Minh Vu
		"Use case for 1.3" initialized	Luu Chan Hung
		Table for usecase initialized	Trinh Manh Hung
2	9-Oct-21	"Activity diagram" initialized	Tran Minh Vu
		"Class diagram" initialized	Luu Chan Hung
		"Sequence diagram : Order at table diagram" and "Sequence diagram: Login" initialized	Dang Cao Cuong
		"Sequence diagram: Place order" initialized	Trinh Manh Hung
		"Sequence diagram: Payment" initialized	Pham Tan Phuoc
3	10-Oct-21	"Context" added	Luu Chan Hung
		"Use case diagram for the whole system" updated	Dang Cao Cuong Tran Minh Vu
		"Table for use case" updated	Trinh Manh Hung Pham Tan Phuoc
4	24-Oct-21	"Package diagram" updated	Trinh Manh Hung
		"Architecture design " initialized	Luu Chan Hung Pham Tan Phuoc
		"Component diagram" updated	Dang Cao Cuong
		"Deployment diagram" updated	Tran Minh Vu
5	7-Nov-21	"Package diagram" updated	Trinh Manh Hung
		"Front end" and "Back end" initialized	Luu Chan Hung Pham Tan Phuoc
		"Overview of the project" and "Lesson learned" updated	Dang Cao Cuong
		"Deployment diagram" updated	Tran Minh Vu
		"Risks, challenges during the project" initialized	Trinh Manh Hung
6	21-Nov-21	"Front end" updated includes: Clerk, User and Manager interface	Luu Chan Hung
		"Chat feature between clerk and customer" initialized	Pham Tan Phuoc
		"Back end" for payment method initialized	Trinh Manh Hung
		"Back end" updated for Clerk	Dang Cao Cuong
		"Back end" update for Manager	Tran Minh Vu
		"Back end" synthesis	Luu Chan Hung

WORK ASSIGNMENT

Task 1

- Section 1.1: Dang Cao Cuong
- Requirements: Pham Tan Phuoc
- Use case diagram for the whole system: Tran Minh Vu
- Use case diagram for 1.3 : Luu Chan Hung

- Tabular format of use case 1.3 : Trinh Manh Hung

Task 2

- Activity diagram: Tran Minh Vu
- Class diagram : Luu Chan Hung
- Sequence diagram: Dang Cao Cuong, Pham Tan Phuoc, Trinh Manh Hung

Task 3

- Architecture design: Pham Tan Phuoc, Luu Chan Hung
- Deployment diagram: Tran Minh Vu
- Component diagram: Dang Cao Cuong
- Package diagram: Trinh Manh Hung

Task 4

- Design front end and database for foods: Luu Chan Hung, Pham Tan Phuoc
- Learn php for back-end implementation and do the back-end part: Dang Cao Cuong, Trinh Manh Hung, Tran Minh Vu

Task 5

- Improve front end and merge the hole code: Luu Chan Hung
- UI and back end for Manager: Tran Minh Vu
- UI and back end for Clerk: Tran Cao Cuong
- Handle Payment method: Trinh Manh Hung
- Update the database and code chat function between clerk and customer: Pham Tan Phuoc

1 Task 1: Requirement elicitation

1.1 Introduction

POS stands for Point Of Sale, a system that is used throughout the restaurant and retail industry. This computerized system allows business owners to track sales, cash flow, food inventory and can help simplify your bookkeeping enormously.

Why is POS Good for a Restaurant ?

The high volume of cash and credit cards that pass through a restaurant each day make a POS system a necessity. Not only does a POS system track every penny of your sales, many POS programs also act as credit card processors. This makes swiping credit cards more secure for both the customer and the business. Servers are accountable for all their sales, and it is impossible to alter checks in the computer unless you have the password. This helps cut down on employee theft.

Benefits:

One benefit of a POS system is that it simplifies communications between the kitchen and the wait staff. Orders go through the computer, directly to the kitchen printer. Another benefit of a restaurant POS program is that it can track everything from food usage to the most popular menu items. Because the POS system acts as a time clock, it can also help prepare payroll. This can save you a lot of money in your bookkeeping department. Along with the daily operations of running a restaurant, a POS system can organize profit and loss statement and sales tax.

1.2 Stakeholders

- End users: Restaurant customers; restaurant clerk; kitchen
- System managers: IT staff; Restaurant manager
- System owners: restaurant owners
- External stakeholders:

1.3 Project context: A traditional Restaurant

- Pay_first, , Payment Option: cash, credit, e-wallet
- Table: each table have a unique ID, customer access to the order page by the QR code on the table
- Reservation: Customers can reserve table on the restaurant website, they can also pre-order food
- Food: At the beginning of the day, we kitchen will update the ingredients quantity. After that the system auto calculate the quantity of a meal to show the availability. However, during the day, the kitchen can update the ingredients or the meal manually (because there may be some problems that led to reduction of the ingredients)
- Menu, meal and price : can only be defined by the manager or the restaurant owner

Business process:

- A customer can reserve a table before coming to the restaurant
- Customer then scan the QR code on the table to order food or to confirm the pre-order
- Customer choose payment method
- If it is pay by cash, the clerk will come to the table to receive the money
- After the clerk confirm each order, the kitchen will receive the order information to start cooking
- After the meal is ready, to clerk bring to the customer table

1.4 Project scope

PROJECT SCOPE				
Project Name	RESTAURANT POS 2.0			
Key deliverables of this projects	M	S	C	W
Table reservation	✓			
Ordering food	✓			
Alerts, billing	✓			
Credit card Processing			✓	
Customer management			✓	
Support take-away options	✓			
Non-direct contact between Clerks and Customers			✓	
Technical Requirements	M	S	C	W
QR code interaction	✓			
Extendable to use in multiple restaurants		✓		
Capable of 3000 orders per day			✓	
Use web technologies		✓		
Time Outline	Time start (week)		Duration (week)	
Task 1: Requirement Elitcitation	6		1	
Task 2: System modelling	7		1	
Task 3: Architecture design	8		2	
Task 4: Sprint 1	10		2	
Task 5: Sprint 2	12		1	

Table 2 continued from previous page

PROJECT SCOPE
Limitation and Exclusion: Finish in 12 weeks
Prioritisation, categorising funcitonality/features according to: Must Have(M), Should Have(S), Could Have(C), Won't Have(W)

1.5 Requirements

1.5.1 Functional requirements

1. Customers
    - Customers shall use QR Code which is placed on the tables for connecting to the website
    - Customers are allowed to reserve tables before coming to the restaurant
    - Customers can log in to earn points for vouchers and events
    - Customers shall select food from the menu
    - Customers can select takeaway options
    - Customers are recommended a suitable meal
    - Customers can confirm the order by a button on the app
    - Customers are able to choose to pay by cash/credit card or online payment
    - Customers receive information when the order is confirmed and ready
    - The application provides satisfaction level rating and feedback
    - Customers can indirectly contact the clerk
2. Clerk
    - Clerk will receive orders after the customers click on the confirm button
    - Clerk shall check order availability automatically by the app and confirm the order
    - Clerk should send notifications to the customers if some problems occur
    - Clerk can indirectly contact the customers
    - Clerk can receive feedbacks
3. Kitchen
    - Kitchen shall receive the information of orders to work with
    - Kitchen can communicate with the clerk
    - Kitchen should update the number of leftover ingredients for the app to check order availability
4. Restaurant Manager
    - Can manage daily/weekly/monthly report such as reports of feedbacks, order trends and income

- Should be given permissions to manage customer information to acknowledge how customers feedback and prevent some kinds of attacks

5. Restaurant Owner
   - Can manage daily/weekly/monthly report such as reports of feedbacks, order trends and income

1.5.2 Non-functional requirements

1. Usability
   - Have user-friendly interface and easy to use for customers. In details, customers just need 4-5 minutes to get used to the application
   - Need 2 training hours to use for clerks and managers
2. Performance
   - The capability of transactions is 300 orders per day
   - Be able to load over 100 orders in just 2 - 3 seconds
3. Security
   - Since customers may choose to pay online so banking information of customer must be also protected
   - Customers' profile information must be protected
   - Restaurant data must be protected
4. Portability and compatibility
   - Cross-platform
   - Be usable from a mobile device, a tablet device or a normal computer/laptop
5. Development
   - Be extendable to use in multiple restaurants in the future

1.6 Use case diagram for whole system

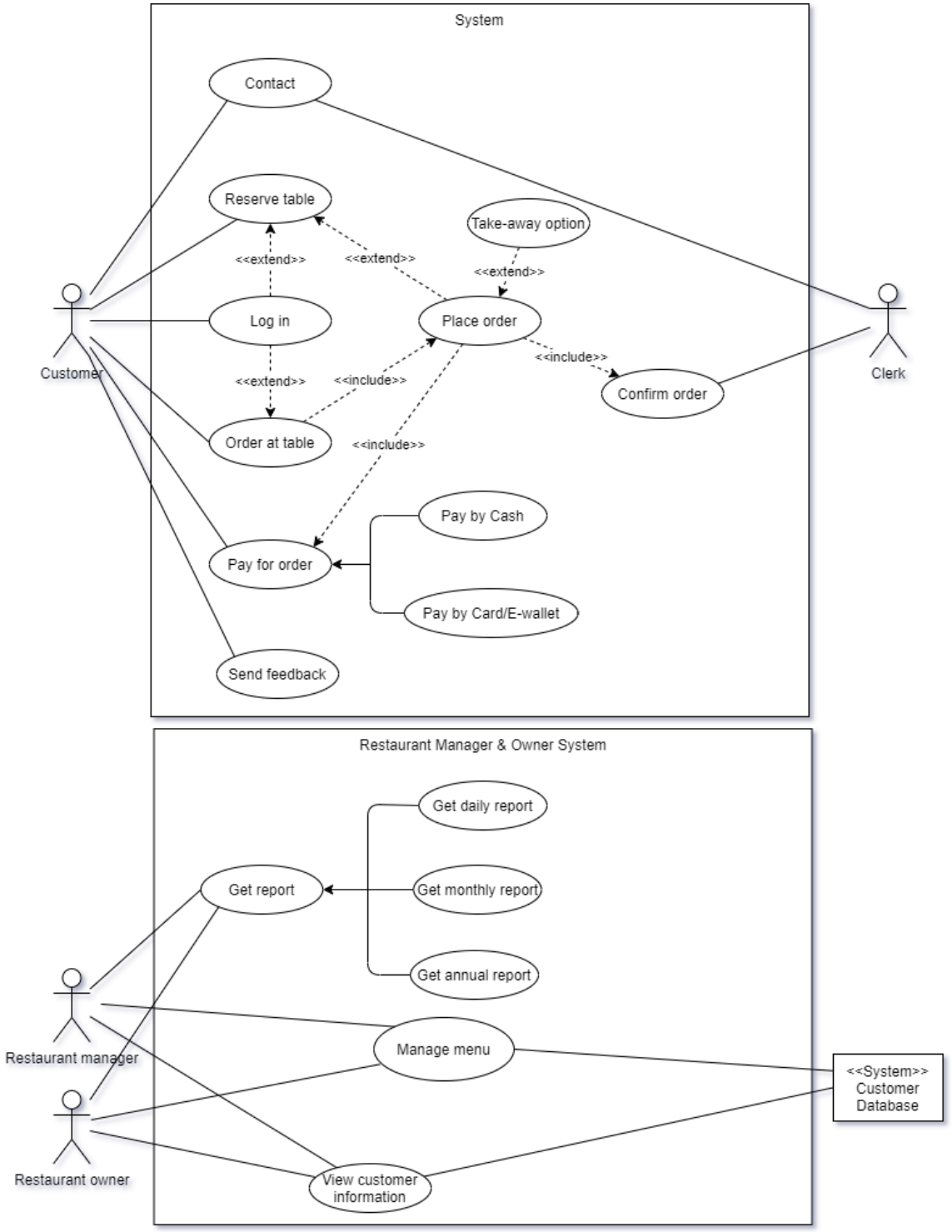


Figure 1: Use case diagram for Food Ordering

1.7 Food ordering feature

1.7.1 Use case diagram

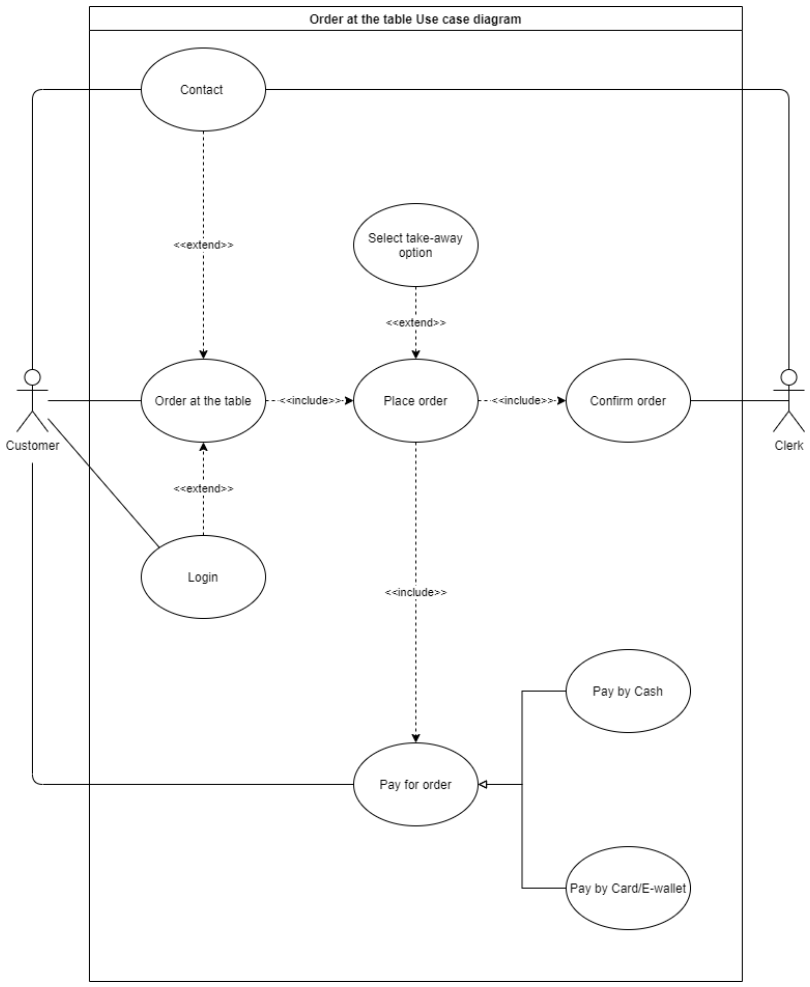


Figure 2: Use case diagram for Food Ordering

1.7.2 Table description

Use Case ID	1
Use Case Name	Order at table
Primary Actor	Customer
Secondary Actor	None
Description	Attempt to order a meal and send that order to the kitchen
Preconditions	Users has a smartphone that connects to internet
Successful End Condition	Kitchen receives order from user
Trigger	The event triggered by an actor that causes the use case to execute
Normal Flow	1. User scan QR code at the table to reach the website 2. System shows the main page 3. User logins 4. Include :: place order 5. System sent information of the order to GUI of kitchen
Exceptions	None
Alternative Flows	Alternative 1: At step 3 3a. User can login as a guest. 3b. User creates an account

Use Case ID	2
Use Case name	Place order
Primary Actor	Customer
Secondary Actor	None
Description	With placing order, customers can order food via an external website
Preconditions	Users is at the food-ordering website
Successful End Condition	Successful ordering notification is sent to the customer
Trigger	The event triggered by an actor that causes the use case to execute

Normal Flow	1. User clicks on the "Menu" button 2. System shows the menu page 3. User skims menu 4. User clicks on a meal 5. System shows a table of information of that meal 6. User changes the amount or specific requirements of the dish. 7. User clicks on "ADD TO CART" button 8. System adds that meal into "YOUR CART" and change the information of "YOUR CART" 9. Include :: Payment 10. Include :: Confirm order 11. System sends successful ordering notification to customer
Exceptions	Exception 1: at step 3 If the user needs help choosing meals, he can contact the clerk indirectly. Exception 2: at step 6 If the amount of the dishes excesses the amount of leftover, the system will inform the user about that, then return to step 5.
Extension	None

Use Case ID	3
Use Case Name	Payment
Primary Actor	Customer
Secondary Actor	Clerk
Description	Attempt to pay for the order after choosing the meals.
Preconditions	Users is at the payment page when clicking on the button Payment
Successful End Condition	Users can see successful payment message sent to the GUI and system records payment
Trigger	The event triggered by an actor that causes the use case to execute

Normal Flow	1. System displays three options, which are “Pay by cash”, “Pay by credit card” or “Pay by e-wallet” 2. User choose one option 3. System records the payment and sends successful payment message to the GUI of the user
Exceptions	Exception 1: at step 6 of step 2c If user inputs wrong signature, return to step 5

Alternative Flows	Alternative 1: at step 2 2a. User chooses the option “Pay by e-wallet” 1. System presents payment page 2. User logins payment page 3. System sends request to mobile banking app 4. User confirms payment 5. Mobile banking app approves the request 2b. User chooses the option “Pay by cash” 1. System sends message about the amount of money to clerk and payment method 2. Clerk receives money directly from customer 3. Clerk deposits cash tendered and returns balance in cash to User 2c. User chooses the option “Pay by credit card” 1. System sends message to clerk about the amount of money and asks clerk to bring POS(Point of sale) 2. System sends payment authorization request to an external Payment Authorization Service System, and requests payment approval 3. System receives payment approval and signals approval to Cashier 4. System presents credit payment signature input mechanism 5. System asks Customer for a credit payment signature 6. User enters signature

Use Case ID	4
Use Case Name	Confirm order
Primary Actor	Clerk
Secondary Actor	None
Description	Attempt to make sure that the order has no problem and the order is recorded
Preconditions	Users have already paid for the order

Successful End Condition	System sends successful ordering message to GUI of user
Trigger	The event triggered by an actor that causes the use case to execute
Normal Flow	1. Clerk checks order information 2. Clerk presses the button “Confirm” 3. System records order 4. System sends confirmed notification to the customer
Exceptions	Exception 1: at step 2 Confirmation is not ready, so clerk solves the problem manually. Then returns step 1.
Alternative Flows	None

2 Task 2: System modelling

2.1 Activity diagram

The activity of ordering and paying for order has two actors **Customer** and **Clerk**, and one class **Restaurant Management System**. After successfully logging in the system, **Customer** will select an order for use. The **Restaurant Management System** will tell the customer whether that order is available or not. If it is not available, the **Customer** will have to pick again until it is possible. Then, the order will be placed and the restaurant will serve it for the customer. After eating the food, the **Customer** has to pay for order. If the **Customer** pays for that order with the app, the **Restaurant Management System** will get that payment, and then record it for use in the future. Otherwise, the **Clerk** will take the money, and then tells the **Restaurant Management System** that the order has been paid. Lastly, **Customer** will send feedback about the restaurant and leave.

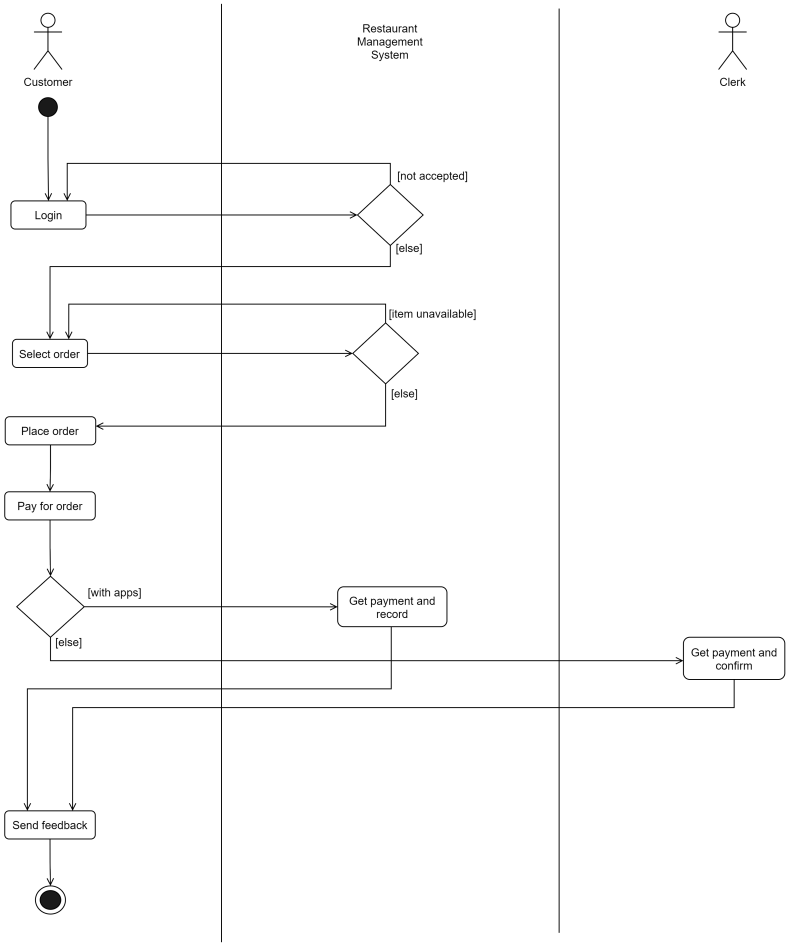


Figure 3: Activity Diagram

2.2 Class diagram

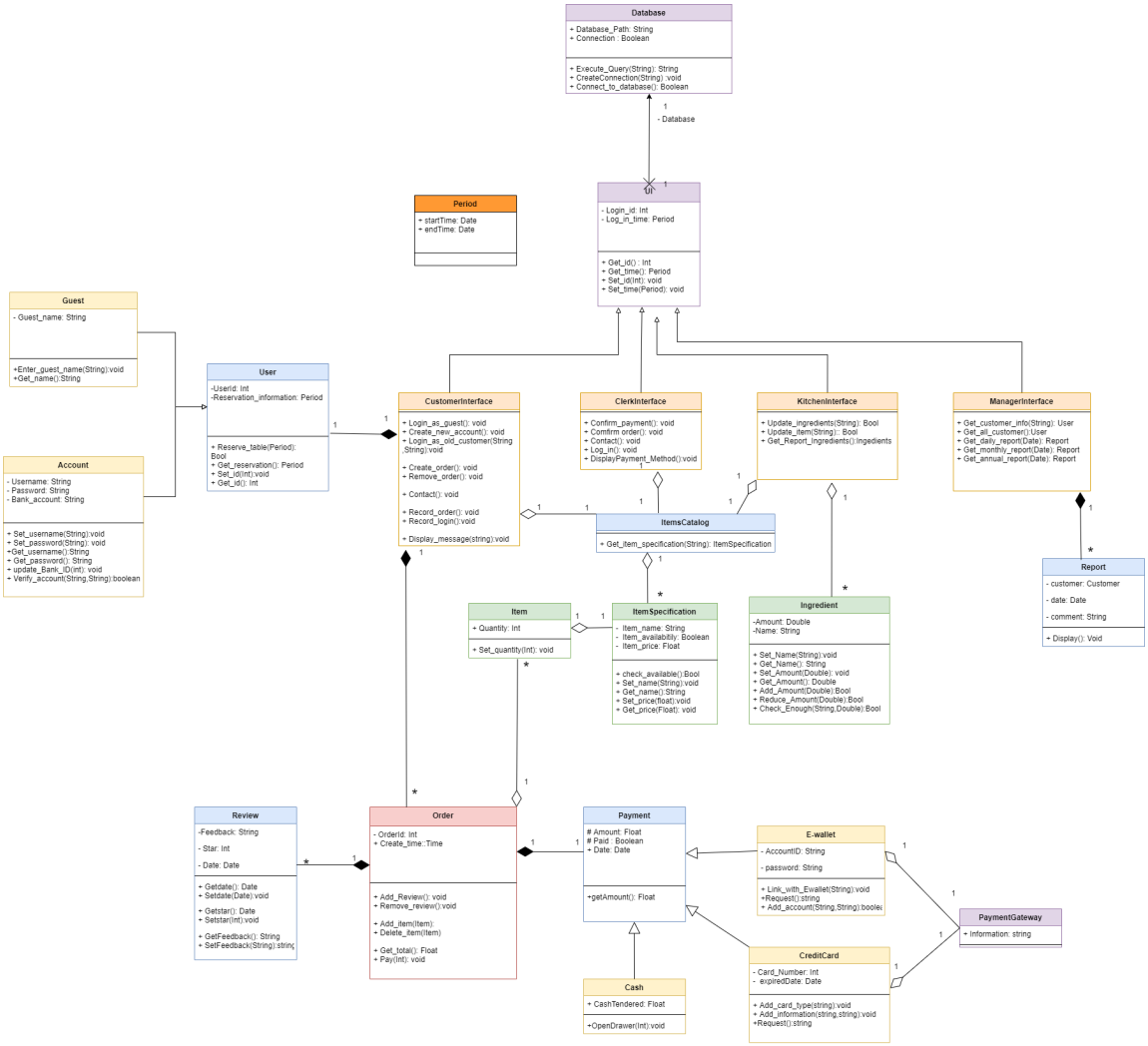


Figure 4: Class diagram

Our System is built around the different interfaces of the system.
Firstly, we will start with **"Database" class** which is the class for connecting and sending request to the database from the system.

- It has two public attributes: a String type Database_Path to store the path connect to the database and a boolean type attribute Connection to show that our system has connect to the database yet
- CreateConnection(String) :void: to create a connection path to data base
- Connect_to_database: Boolean: to connect to the database and return a boolean value to show the connection is success or not
- Execute_Query(String): String: to execute SQL script and send to the database, it receive the retrieval data ?. It has a String input indicate the SQL script

Secondly, we have the **"UI" class**, which is the parent class for all interface.

- It has 2 private attribute: Login_id and Log_in_time. While Login_id is an integer, Log_in_time has the data type of "Period", which is a class that have startTime and end Time
- The UI class also has 2 Get-Set operations for the private attributes
- UI class is inherited into "CustomerInterface", "ClerkInterface", "KitchenInterface", "ManagerInterface.
- We have the association relationship with navigability cross between UI and Database: it means that 1 UI has 1 Database and don't have the reverse side (the cross symbol). it means that UI class contains Database class to connect to the Database.

"CustomerInterface" class: this is the class ot present a system at the user side, the main system class when an customer connect to our system. The "CustomerInterface" class have 9 public methods for general operations of customers:

- Login_as_guest(): void: The user choose to login as guest
- Create_new_account(): void: The user choose to create an account of our restaurant
- Login_as_old_customer(String,String):void: The system will check the login of information of the user from the database to get the account of that customer
- Create_order():void : The system will create an "Order" class to begin an order from the customer
- Remove_order(): void: The system will remove the already created Order
- Contact():void: The customer want to contact to the clerk, so this method will send message to the ClerkInterface through network sockets
- Record_order():void and Record_login(): void: are the two methods to record the information of the order and the login to the database
- Display_message(string):void : Display an output message to the customer (notification...)

- "CustomerInterface" has Aggregation with ItemsCatalog: which means the interface has 1 object of ItemsCatalog to display the menu of the restaurant
- It also have the Composition relationships with User and Order: Each CustomerInterface contains 1 User class and many Order

"User" class

- Is the parent class for two types of login method: login as guest and login with an account
- It has two private attribute: UserId, which is an integer, and Reservation_information, which is an Period type to record the reservation information of the customer if the customer has reserve before on the web
- Reserve_table(Period):Bool to make a new reservation with a boolean return type indicates the creation is success or not.
- Get_reservation(): Period: to get the reservation information
- Set_id(Int): void and Get_id():Int is a Set-Get operation for the UserId

"Guest" class

- The class is create if the customer choose to login as guest
- It as an private attribute Guest_name, which is a string to store the login name
- It has 2 method Enter_guest_name(String):void and Get_name():String to set or get the username

"Account" class

- Is the class store the information of a customer account
- It has 3 private attribute: Username, Password and Bank_account. They are all String datatype
- Set_username and Get_username are Set-Get method for username
- Set_password and Get_password are Set-Get methods for password
- Update_Bank_Id(Int): void to set bank account information
- Verify_account(String,String):boolean to verify the account from the database

"Order" class

- Represent an order of a customer
- It contains an private Orderid and Create_time
- A customer can add or remove an review through Add_review():void and Remove_remove():void. The method Add_review will create a "Review" class, which contains 4 attributes (Feedback, Star, Date and Food) with String, Int, Date and String datatype respectively. A review class also has 3 pairs of Set-Get methods for the 3 attributes. An order can contains 0 or many reviews.

- Orders also has 1 to 1 composition relationships with payment, which represents the bill of 1 order. Payment has 3 attributes: Amount(Float), Paid(Boolean) and Date(Date) with 1 methods : getAmount(): Float to return the amount of money.
- Payment class is inherited into 3 subclasses
- Cash: a class for the case that the user choose to pay by cash. It has a CashTendered attributes with OpenDrawer(Int):void method to open the money drawer
- CreditCard class is created when the user choose to pay by card. It has 2 attributes CardNumber and ExpiredDate with 3 methods: Add_card_type(string):void; Add_information(String,String):void and Request():string to connect to the payment gateway
- E-wallet class is created when the customer want to pay by e-wallet. The class has 3 attributes: AccountID and password with 3 methods: Link_with_Ewallet(String):void and Request():string; Add_account(String,String):boolean
- CreditCard class and E-wallet class both contains 1 PaymentGateway class to stored the Gateway information

"ItemCatalog" class

- This class represent the menu of the restaurant and belongs to CustomerInterface; ClerkInterface and KitchenInterface
- It has 1 method : Get_item_specification(String): ItemSpecification that return ItemSpecification class

"ItemSpecification" class

- Represent the information of an item
- Has 3 attribute: Item_name(String), Item_availability(Boolean) and Item_price: Float all are private
- It has 4 methods to set_get the name and the price with and 1 method to check the availability of the item

"Item" class

- Represent 1 item in the cart: it has Quantity attribute and Set_quantity(int): void method

"ClerkInterface" class

- It is the class the the interface of the clerk
- It has 5 methods: Comfirm_payment():void; Confirm_order():void; Contact():void; Login():void and DisplayPayment_Method():void

"KitchenInterface" class

- It is the interface for the kitchen to update the ingredients of an item

- It can use 3 methods: Update_ingredients(String): Bool that take the name of an item and return the success message
- Update_item(String): Bool is to update the item availability manually
- Additionally, the Kitchenterface also contains many Ingredient to display at the interface
- Finally, the Get_Report_Ingredients(): Ingredients to get the report of 1 specific ingredients

"Ingredient" class

- A class for 1 type of ingredients with Amount and Name
- Besides Set-Get method, we also ave Add_Amount and Get_Amount method to adjust the ingredients quantity
- Lastly is the Check_Enough() method to check if the ingredients is enough for 1 item.

"ManagerInterface" class

- The "ManagerInterface" class is the class for the manager to get report
- A manager can get daily, monthly or annual report through 3 method that all return a Report class that contain customer information; the date its was recorded and some comments with Display() method
- Otherwise, a manager can get the information of a specific customer through Get_customer_info(String): User method
- A "ManagerInterface" class can have many Report

2.3 Sequence diagram

2.3.1 Order at table diagram

In the activity of ordering food at table, it is separated into two phases, namely login and placing order. There are four classes and one use case to illustrate the sequence diagram, including **Customer**, **Customer Interface**, **Database**, **Clerk Interface** and **Kitchen Interface**. The next section will dive into details of the login phase.

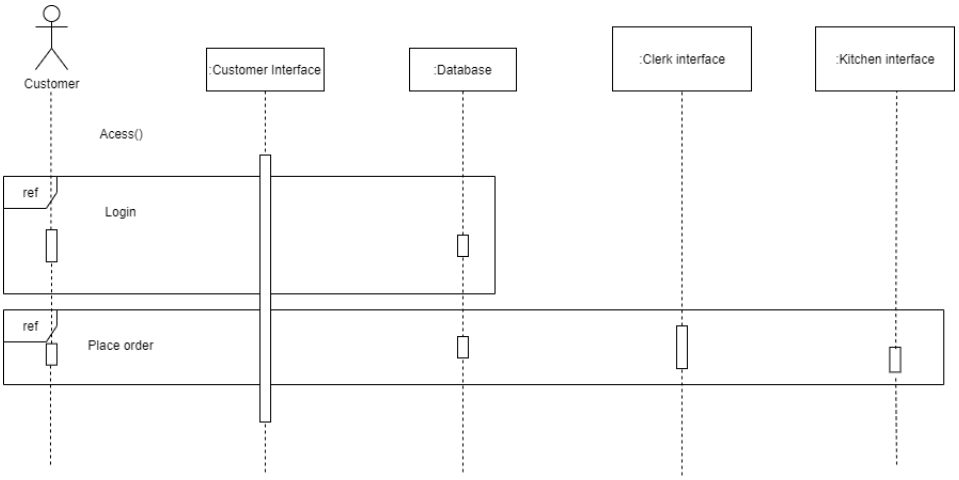


Figure 5: Order at table-sequence diagram

2.3.2 Login diagram

In the login diagram, **Figure 4**, there are three main class and use cases, customer, customer interface and database. At the beginning, the customer login the website which is represented by the interaction between the use case **Customer** and class **Customer Interface** via **Login_as_guest** or **Login_as_old_customer** functions. When the customer logout, the class **Guest** is destroyed by the **Customer Interface**

In the case the customer login as a guest, the **Customer Interface** will create a class named **Guest**. After that, the customer need to fill their basic personal information like name, age or mobile phone number, etc, described by **Enter_guest_info** function, then displayed by the customer interface.

In the case the customer login as a customer, the **Customer Interface** will firstly create a class named **Account**. Secondly, the **Account** send a signal to **Database** to verify the user agent of the customer. Then the database return data of the user agent displayed by the **Account**. Similarly, when the customer logout, the class **Account** will be deleted.

Finally, all of the login information will be recorded by the database.

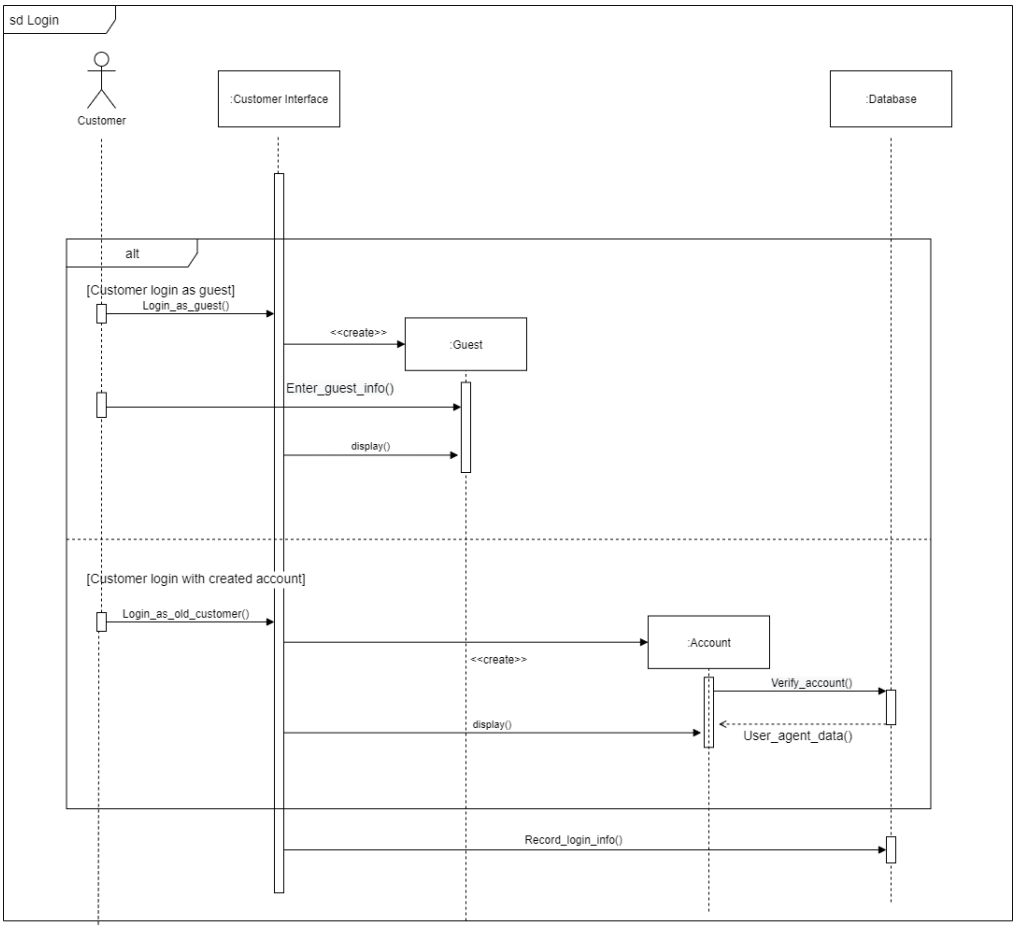


Figure 6: Login-sequence diagram

2.3.3 Place order diagram

In the place order diagram, **Figure 5**, there are six main class and use cases, including **Customer**, **Customer Interface**, **Order**, **Item**, **Clerk Interface** and **Clerk**.
At the beginning, the customer creates an order represented by the interaction between the use case **Customer** and class **Customer Interface** via Create_order function. Then, the class **customer interface** creates a class named **order**.
Next, the customer can add dishes, and the class **order** creates a class named **Item**. Customer can change the quantity of meals via function **Set_quantity** between use-case **Customer** and class **Item**.
After finish choosing dishes, the **Customer** will pay for his meal via reference Payment, which we will dive into in the next section.
Next, the **Clerk** confirms the order through **Clerk Interface**, then it sends back the **Order_status_message** to the **Customer Interface**. Finally, the **Customer interface** displays to itself the message, a return notification is sent to the **Customer** so he can be informed about his order's status.

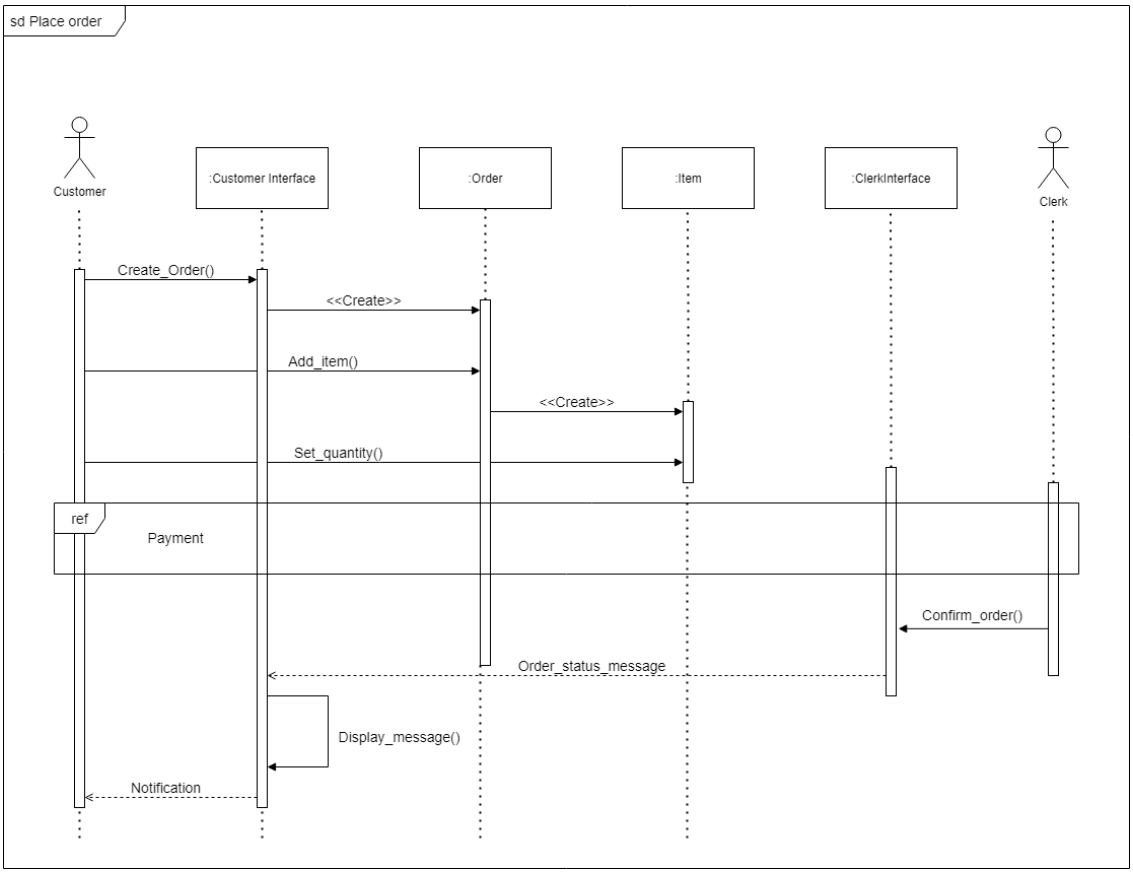


Figure 7: Place order-sequence diagram

2.3.4 Payment diagram

In the payment diagram, Figure 6, we have five main classes and use cases including Customer, CustomerInterface, Order, ClerkInterface and Clerk.

To begin with, the customer clicks on “Pay” button and then the CustomerInterface will call “Pay()” function, which allows the customer to choose a method for payment. There are three ways for payment, which are pay by cash, pay by card and pay by e-wallet. So now, we will go into detail about each method.

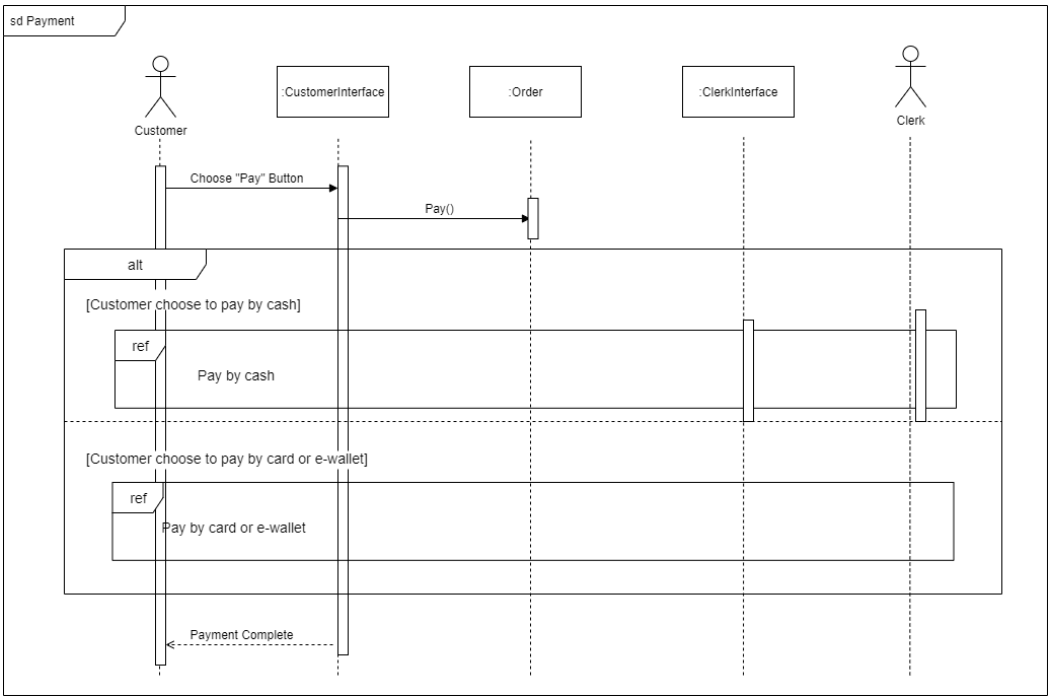


Figure 8: Payment sequence diagram

First, we will get into Pay by card or pay by e-wallet sequence diagram, Figure 7. For pay by card, the Order will first create an object of class Card. After that, the Customer is able to add card type and necessary information via Add_card_type() and Add_information() function. Following that, the Card object requests class PaymentGateway to process the customer’s payment and displays successful message to the Customer’s interface through Display_message() function.

For pay by e-wallet, the Order will begin with creating an object of class E-wallet. Then, the Customer can access their E-wallet account by Link_with_E_wallet() function. After successfully accessing, the E-wallet object requests class PaymentGateway to process the customer’s payment and displays successful message to the Customer’s interface through Display_message() function.

Now, we are going on with pay by cash method, which is a traditional one. Let’s look at Pay by cash sequence diagram below, Figure 8. The class Order will first create an object of class Cash. After that, the Customer will call function openDrawer() for paying. Then, the Clerk’s interface displays a notification to the clerk for confirming payment.

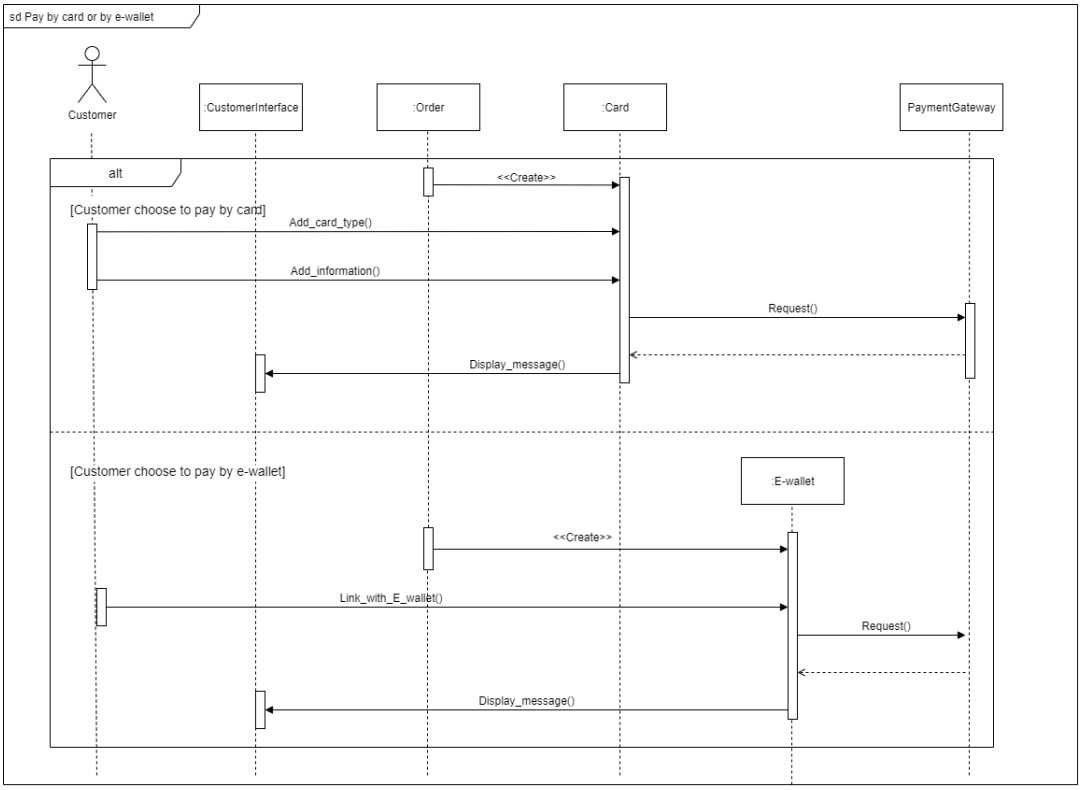


Figure 9: Pay by card/e-wallet sequence diagram

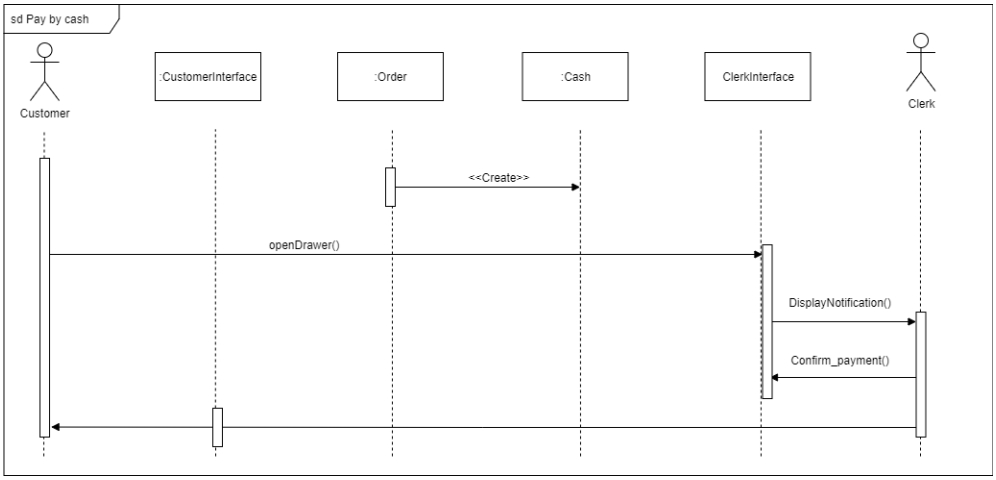


Figure 10: Pay by cash sequence diagram

After completing all steps for payment, the customer's interface will show a successful notification to the customer. That's all for our payment process.

3 Task 3:

3.1 Architecture design

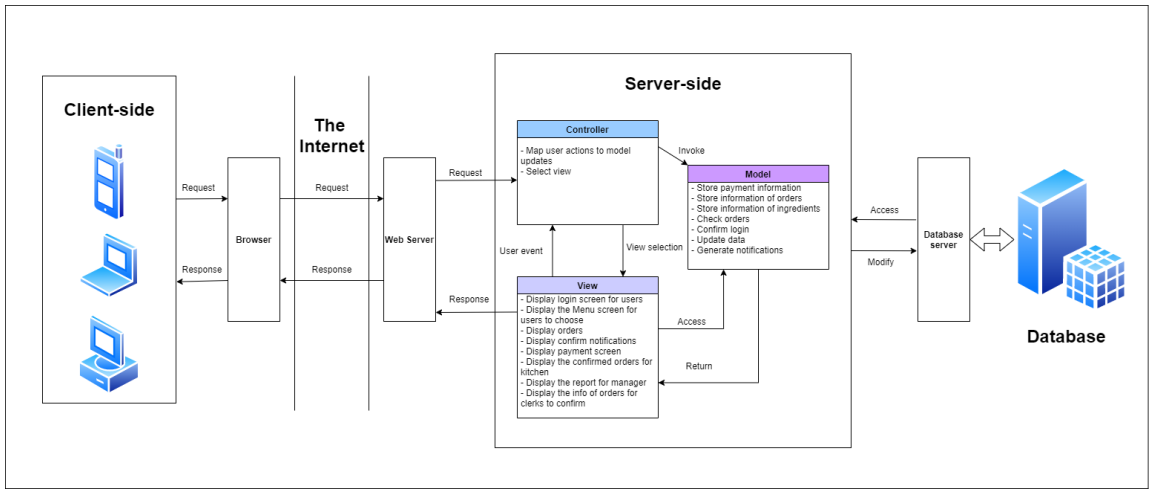


Figure 11: Architecture design

Our application architecture is based on MVC model inside a Client-Server model. First, On the Client-side, we have customers devices likes smartphones, laptops or PC that are used to connect to our website. After that, the client send request to the server side through internet layer. The server side will eventually response to the request through a MVC architecture. Moreover, the server-side connects to the database through a database server to retrieve or modify the data.

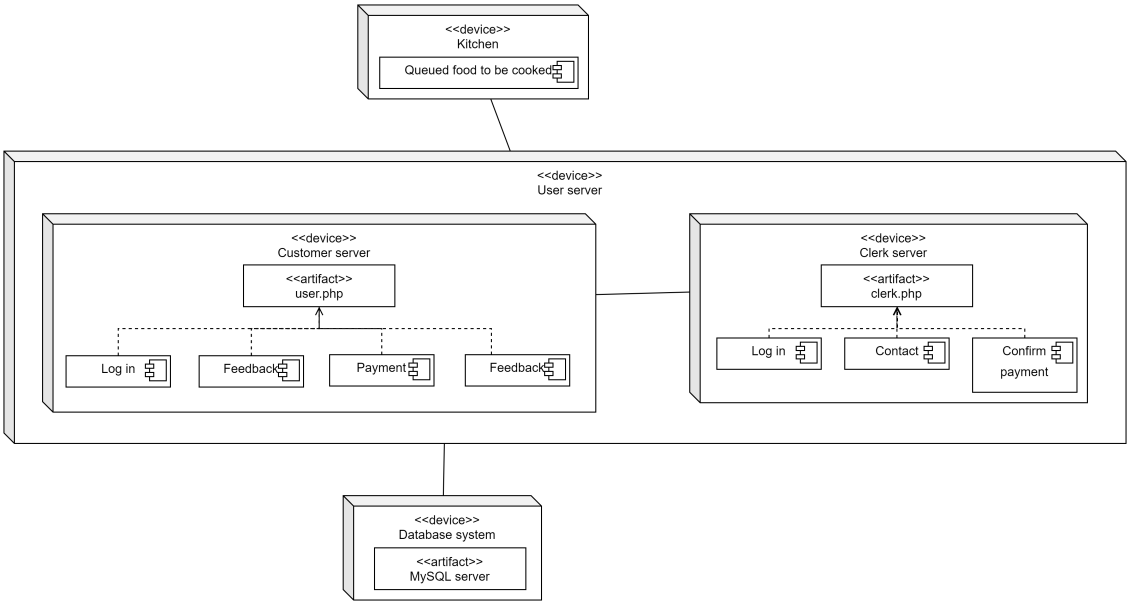
On the Server-side, we use MVC design pattern to illustrate our architecture design. It includes three major interconnected components, which are:

1. Controller: manages user interaction (e.g., key presses, mouse clicks, etc.) and passes these interactions to the View and the Model. Therefore, in our application, its main functions are mapping user actions to model updates and selecting views for displaying.

2. Model: is the central component of the pattern that directly manages the data, logic and rules of the application. In our application, it has several functions which consist of storing information, checking orders, confirming login, updating data and generating notifications.

3. View: defines and manages how the data is presented to the user. That is, in our application, used to displaying login screen, orders, notifications, etc. to users.

3.2 Deployment diagram



In this diagram, both **Customer**s and **Clerk**s are **User**s of this application. The **Kitchen** gets the queue of food ordered by **Customer**s and make all of it one by one. **User server** needs to gain access to the **Database system** in order to confirm the accounts, and save the history of orders and feedback.

3.3 Component diagram

Component diagrams are used to represent the how the physical components in a system have been organized. We use them for modelling implementation details. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development.

Component Diagrams become essential to use when we design and build complex systems. They are generally used for modeling subsystems. It represents how each and every component acts during execution and running of a system program. They are also used to show and represent structure and organization of all components.

This component diagram visualizes components of major functions, **Log in** and **Select and place order**.

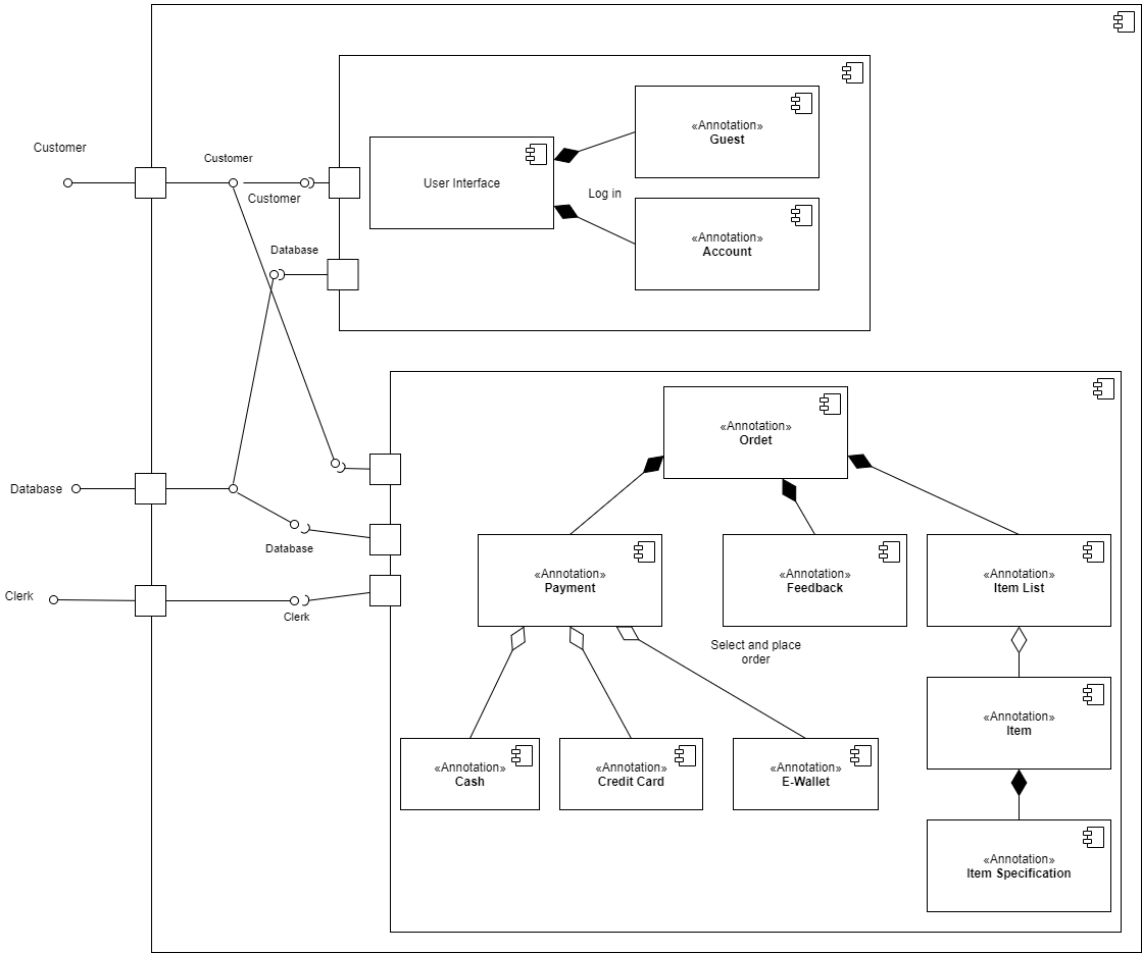


Figure 12: Component Diagram

3.4 Package diagram

Package diagram shows the dependencies between different packages in a system.

The purpose of package diagram is to structure high level system elements. Packages are used for organizing large system which contains diagrams, documents and other key deliverables.

The following package diagram shows the structure and dependencies between sub-systems **UI**, **Application Logic** and **Data**. We group classes into packages so as to simplify our class diagram in the figure 4.

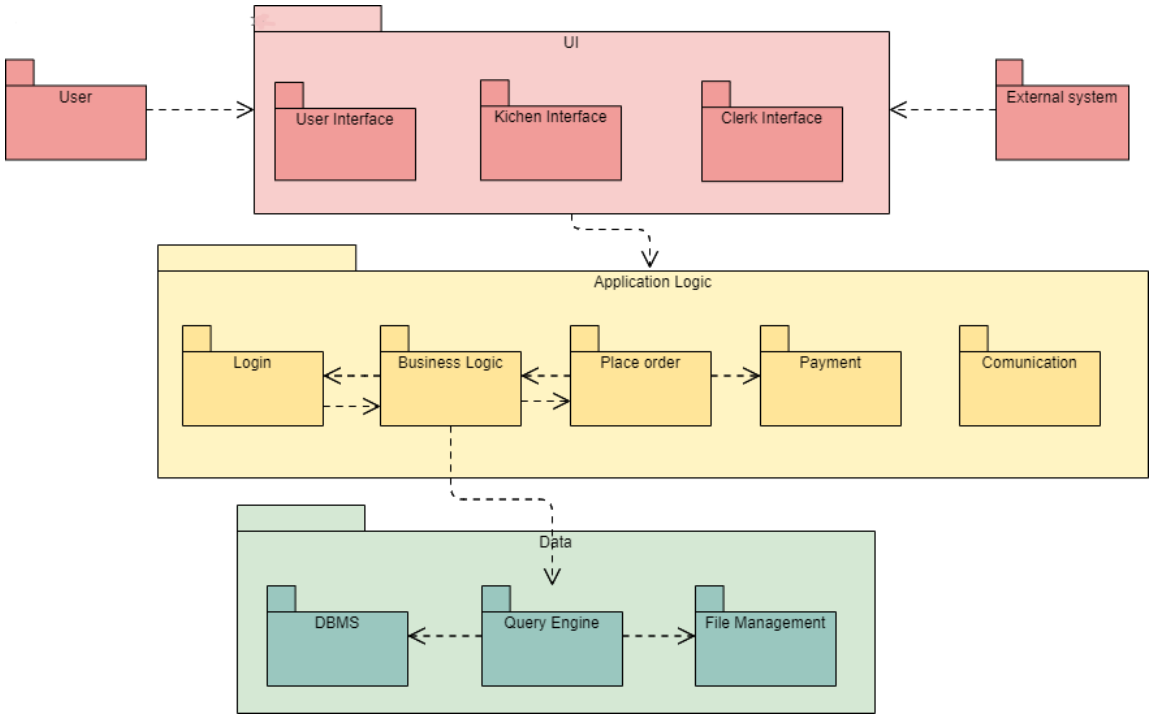


Figure 13: Package Diagram

4 Task 4: Implementation – Sprint 1

4.1 Overview of the project

In this project, our team implement an application for restaurant management with aforementioned requirements in Table 1.7.2. In order to complete the project successfully, we choose Agile methodology to govern the progress of our group. Additionally, we have learned and reviewed technologies supporting for the project. However, there are numerous risks and challenges our group have met mentioned in subsection 4.4.

4.2 Key technical decisions

4.2.1 HTML

The HyperText Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser.

- Easy to Learn and Use. HTML is very easy to learn and understand.
- Free.
- Supported by all Browsers.
- The Most Friendly Search Engine.
- Simple to Edit.
- Can Integrate Easily with Other Languages.
- Lightweight.
- Basic of all Programming Languages.

4.2.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML

- Easier to maintain and update.
- Greater consistency in design.
- More formatting options.
- Lightweight code.
- Faster download times.
- Search engine optimization benefits.
- Ease of presenting different styles to different viewers.
- Greater accessibility.

4.2.3 JavaScript

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.

- Speed. Client-side JavaScript is very fast because it can be run immediately within the client-side browser. Unless outside resources are required, JavaScript is unhindered by network calls to a backend server.
- Simplicity. JavaScript is relatively simple to learn and implement.
- Popularity. JavaScript is used everywhere on the web.
- Interoperability. JavaScript plays nicely with other languages and can be used in a huge variety of applications.
- Server Load. Being client-side reduces the demand on the website server.
- Gives the ability to create rich interfaces.

4.2.4 PHP

Hypertext Preprocessor (PHP) is a widely-used open-source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

- Easy and Simple to Learn
- Extremely Flexible
- Easy Integration and Compatibility
- Efficient Performance
- Cost-Efficient
- Gives Web Developer More Control

4.2.5 Web server: Apache

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows.

- Open-source and free, even for commercial use.
- Reliable, stable software.
- Frequently updated security patches.
- Flexible due to its module-based structure.
- Easy to configure, beginner-friendly.
- Cross-platform (works on both Unix and Windows servers).

- Optimal deliverability for static files and compatibility with any programming language (PHP, Python, etc)
- Huge community and easily available support in case of any problem.

4.2.6 Database: MySQL

MySQL is an open-source relational database management system

- Huge amount of online tutorial and documentation
- Free to install and use
- Security and reliability
- High performance
- Round-the-clock uptime
- Can run on multiple operating systems

4.3 Project outcome

4.3.1 Frontend

For this sprint of our project, we have finished the web appilcation for the customer side, which is the food ordering application.

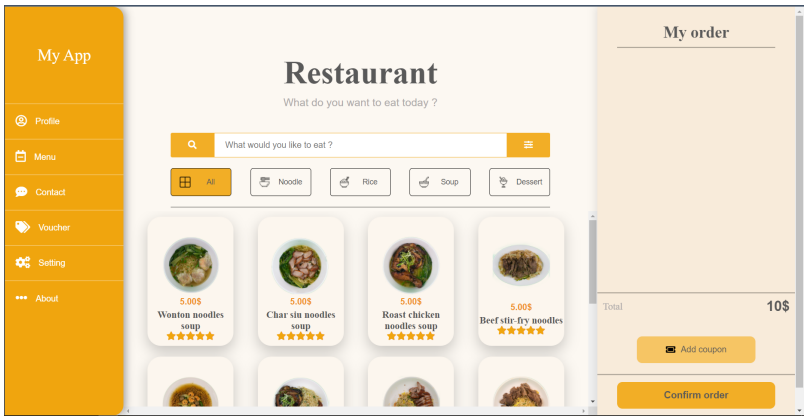


Figure 14: Window screen view

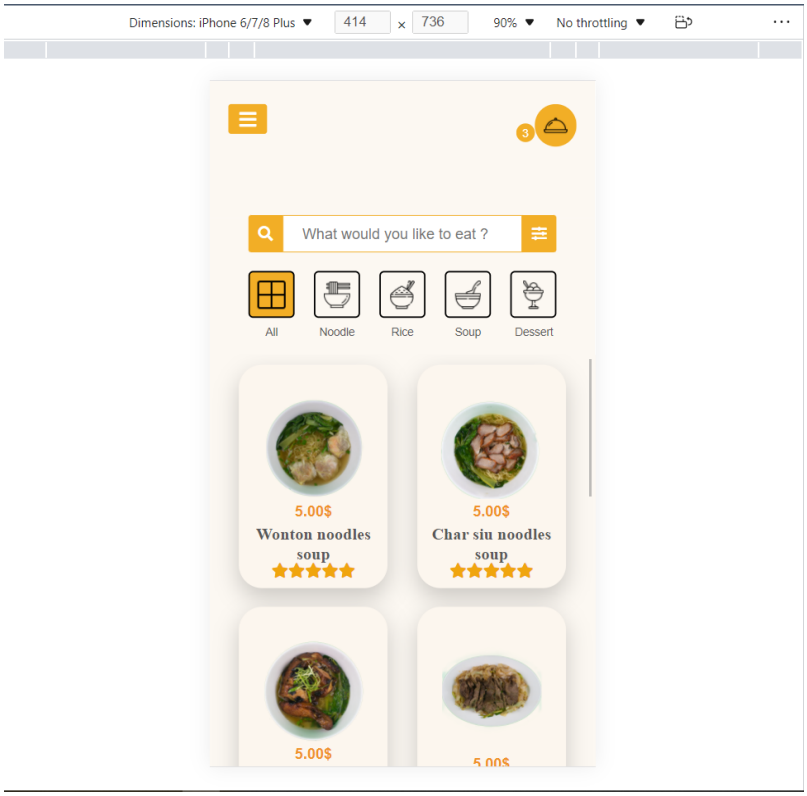


Figure 15: Mobile screen view

The basic of our design contains 3 pannel. On the left is the navigation bar, where we can access to different site like profile, setting.... On the right hand side, we have cart panel which show the items that we have selected. Finally, the main section in the middle contains 2 parts, some introduction information and search bar on the top, and the items list in the bottom.

From the above 2 figures, we have implemented a responsive website. The cool thing about our application is that, when it is reduce to mobile version, the navigation bar on the left and the cart panel on the right automatically disappear. After that, we could use the 2 button appeared on top to open those 2 panel.

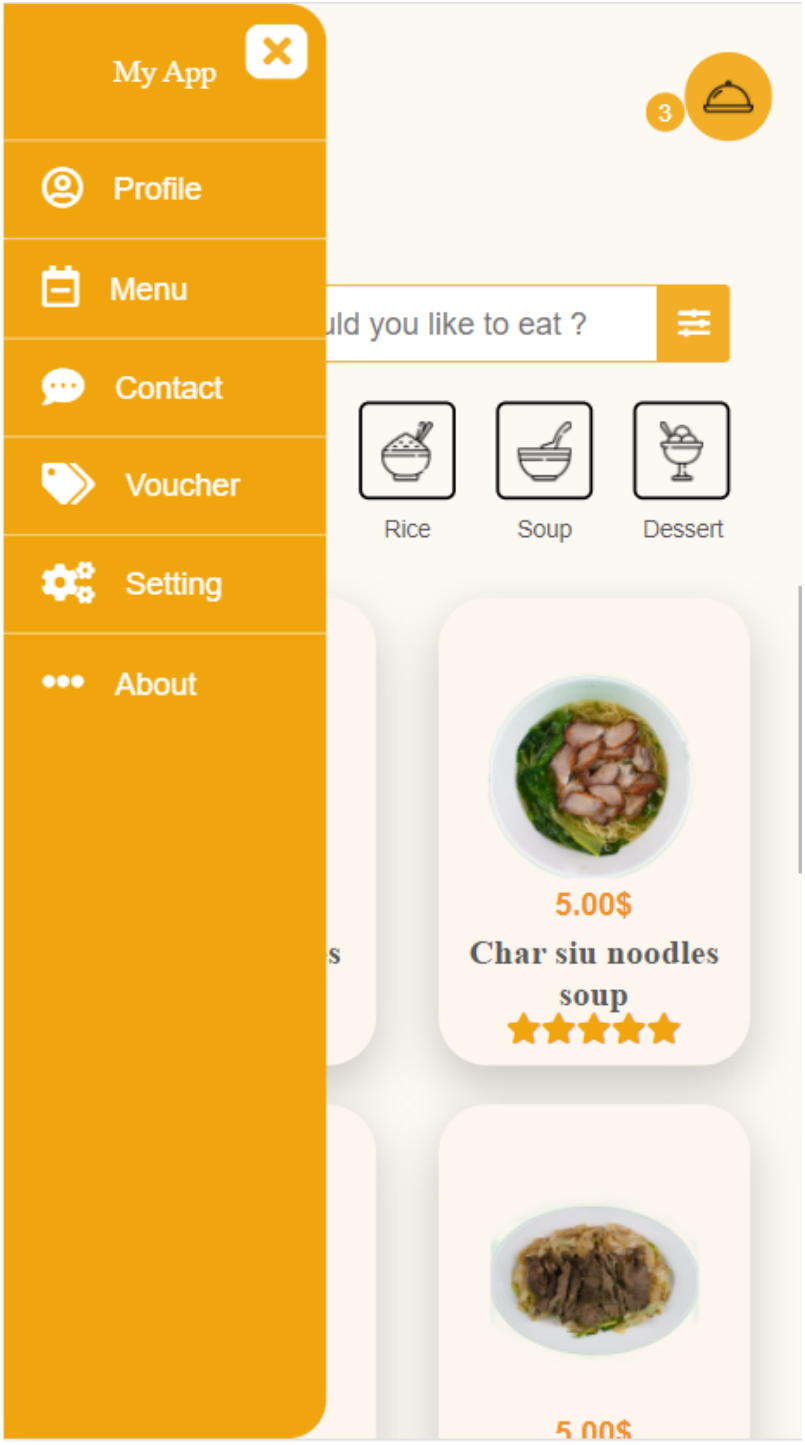


Figure 16: Mobile screen view opens navigation bar

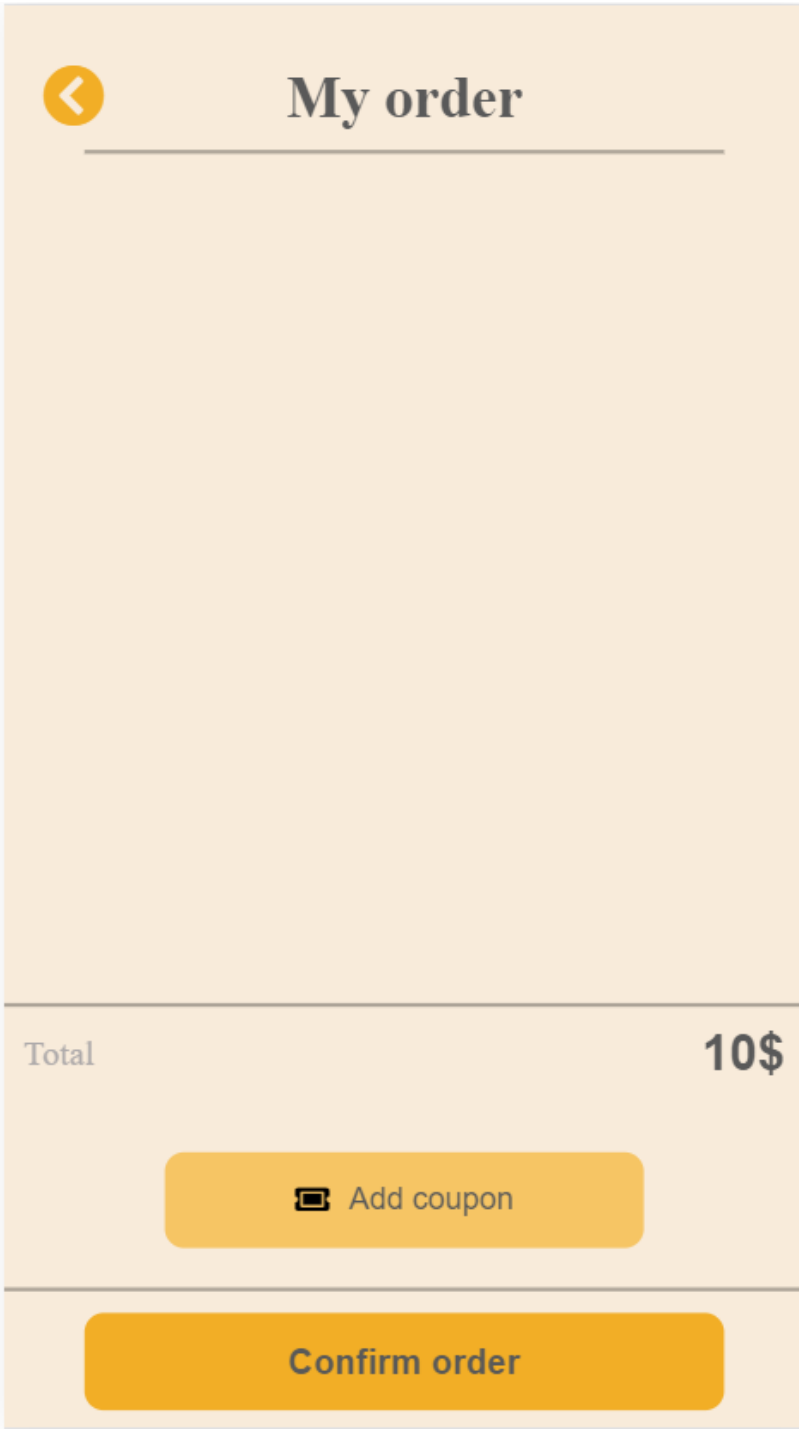


Figure 17: Mobile screen view opens cart panel

Furthermore, in this sprint, we have implemented 'filter' features that allow us to filter our meal by categories. In our data, we have 4 types of meal (Noodle, Rice, Soup and Dessert). After clicking a filter problem, suitable meal would appear.

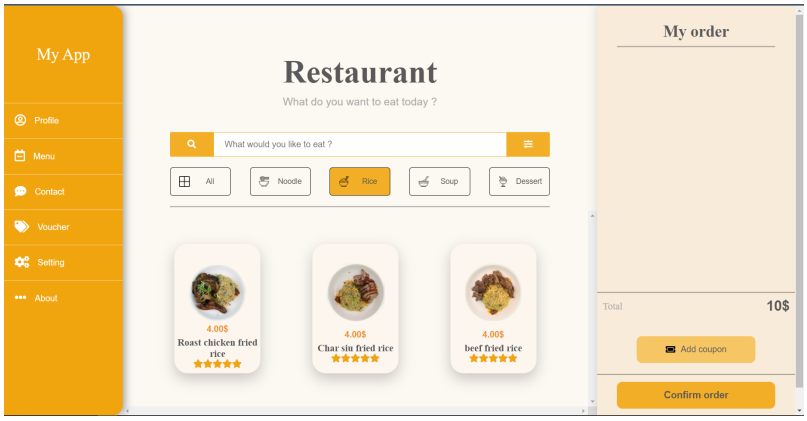


Figure 18: Filter features

Last but not least, our app allow us to see the information of the meal after clicking on the card itself. It will open a site like the figure below.

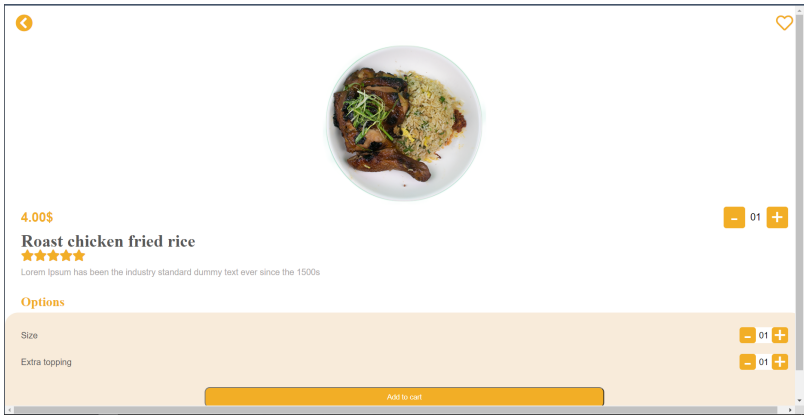


Figure 19: Filter features

4.3.2 Backend

For the backend side, our group simulate a server and also create a database. We use XAMPP software to simulate our server using Apache server and MySQL database.

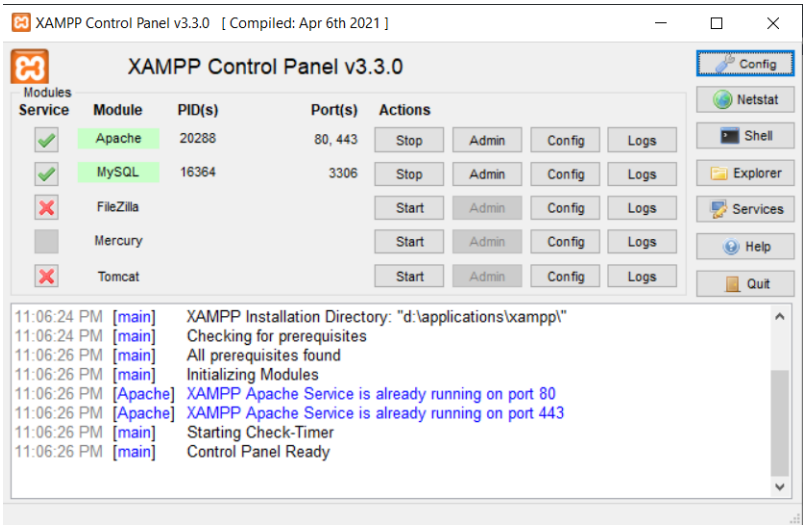


Figure 20: XAMPP software

After open XAMPP panel, we activate Apache server and MySQL database.

Next, we have to create our database. In this sprint, our group create a simple database to just represent the menu.

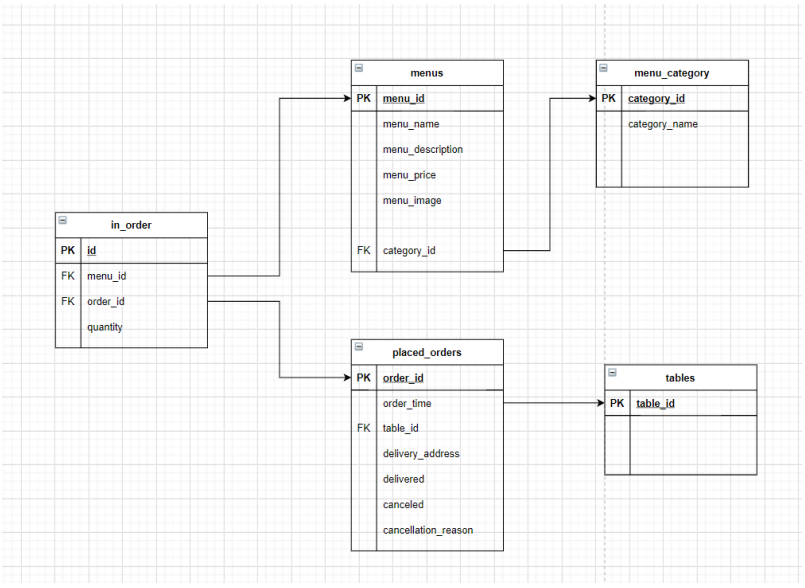


Figure 21: Logical database design

Our database contains 5 tables, menus to store the information about our restaurant menu. placed_order is used to represent our order and in_order is used as a many to many relationship

between order and menu. Additionally, we have a table to store item categories and a table to store actual table of our restaurant.After having a logical design, we create a SQL code and pass it to MyPHP Admin database server.

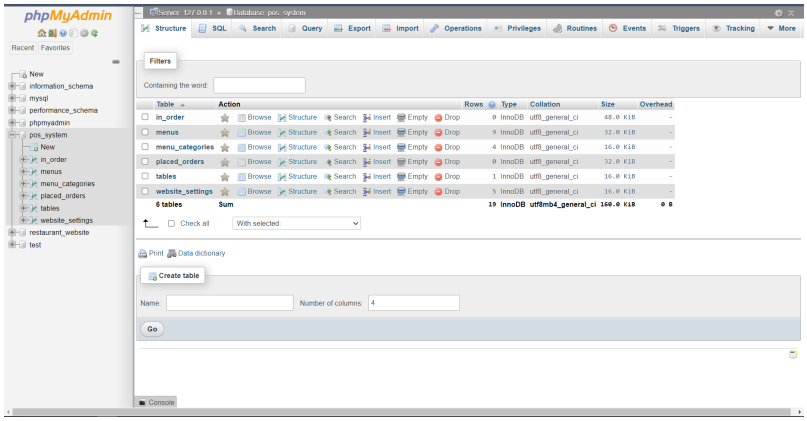


Figure 22: Database

Figure 23: Database

More than that, we also create some initial data on our menu.

Finally, to run our application, we just have to type in our web browser the command: loaclhost/¡Project Folder¿. After that, we got the website like we presented above.

4.4 Risks, challenges during the project

4.4.1 Risks

About application drawbacks:

- From the restaurant's owner aspect. This type of service is quite costly since the owners have to pay time and money for developing the apps, training clerks and upgradement for the application.
- Besides, from the side of customers. They may face with some security risks since all of the information such as phone number, bank account, etc are stored on the app. Moreover, some customers may find it hard to get used to with the app, especially old people. Some also feel uncomfortable because they have to have internet access at all the times.

About technology drawbacks:

- For PHP, due to its open-source property, we may face with some security issues. Also, using more features of PHP framework and tools can cause poor performance of online applications. Moreover, PHP do not allow change or modification in core behavior of online applications, this is also a challenge that we have to deal with while developing our app.
- Next, talk about MVC. Although it can help developers so much in developing an application; however, there are still some drawbacks that everyone may meet, that is, the increase of complexity, inefficiency of data in view, the need of multiple programmers and so on. But indeed, MVC is still a good framework to work on.

4.4.2 Challenges and solutions

We experienced numerous problems in implementing the project. One of them is UML, because we have little expertise with application design in the early stages of the project, it is difficult to understand the big picture and develop UML diagrams. Furthermore, in terms of technical issues, we confront numerous challenges in learning a new programming language such as PHP in a short period of time. For group management, some members may be unable to attend an urgent meeting because they are preoccupied or have not read the paper in advance.

We have a few remedies for these issues, such as reading the documents and doing the required job before to the meeting. When it comes to implementing the website, consulting previous projects is really beneficial because we can learn from them and program it accordingly. We are motivated to accomplish the project schedule since we have a weekly meeting. We outline the week's tasks and allocate duties to each member during that meeting. We also track member performance via an app like Notion or Hackmd.

4.5 Lesson learned

After this course, we have learned processes of making a software product. At the beginning of a project, we need to choose a software process models governing the activities of the team. After that, requirements is analyzed carefully followed by modelling the system from various perspectives, including context models, interactive models, structural models, behavioral models. Architectural

model facilitates discussion about the system design between stakeholder, large-scale reuse and system analysis.

In term of technology, we have acquired general knowledge about programming language supporting web development, front-end and back-end development. Regarding to front-end, we have reviewed HTML, CSS and learned further Bootstrap and Perl. About back-end, we have grasped the mechanisms of interaction between users and servers, namely database server, web server, file server, etc and learned to use PHP programming language.

Regrading to teamwork, we have learned interpersonal skills, such as how to share knowledges to other members, manage the timeline and assign the workload to other members in the team and how to have a better communication in the online environment.

5 Task 5: Implementation – Sprint 2

For this assignment, our group has improved our project significantly. We will start this sprint report by mentioning about the new database design which has been upgraded from the last sprint. After that, our group has decided to redo the User Interface of our application due to some design drawbacks in the previous version. For this part, we will point out the problems with the old design and show our new design. Moreover, in this sprint, our group has done many additional features. Regarding the main POS application, our group implemented "Add to cart", "Remove from cart", "Payment" and "Indirect Communication" features. Besides, we also completed 2 websites for restaurant clerks and restaurant manager to use with the purpose of complete our business context.

5.1 Rewind business context

In our system, it contains 3 part: Customer, Clerk and Manager. Regarding Customer, they will connect to each table POS page to order food. As for Clerk side, they will have responsibility of receiving the payment by cash, and then confirming each order before it is sent into the kitchen. Finally, Manager web page will help the restaurant manager to mange their restaurant, with some features like manage categories, manage item, manage clerk information...

5.2 Database design improvement

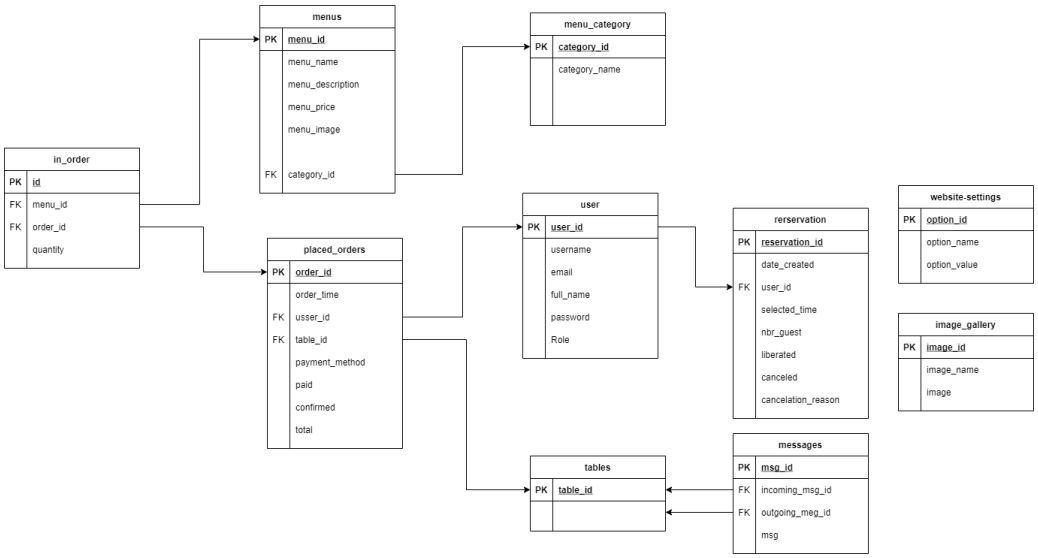


Figure 24: New database design

For this sprint, we have expanded the database design to not only store the information of menu items, but also contains customer information, as well as chatbox database.

user table : This is the table to store both information of customers and clerk, with the attributes "Role" to differentiate them. **reservation table**: This table is for the future development of table reservation features. **messages table**: The table for storing chat app data sent between customer and user.

5.3 New UI design of POS page

5.3.1 Old design problems

Low contrast The first problems with us is color pairing for the website. For the navigation bar, we chose yellow background with with text, this led to low-contrast design, Which is hard to see for a user.

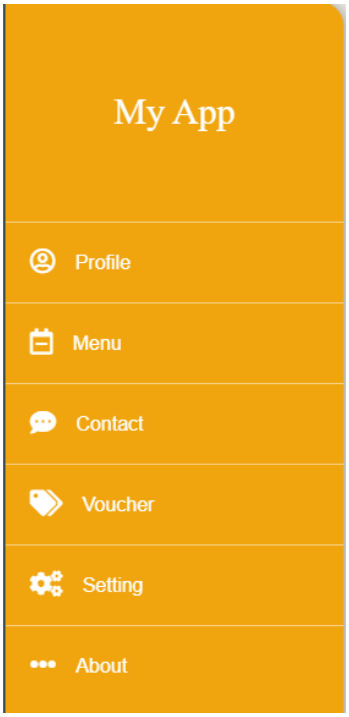


Figure 25: Low contrast between with text and yellow background

Too many colors For our first design, we selected many background color, which at first we thought that it would create a nice overall theme color. However, after re-evaluate the design, we finally realized that it lowers the visual level of our design.

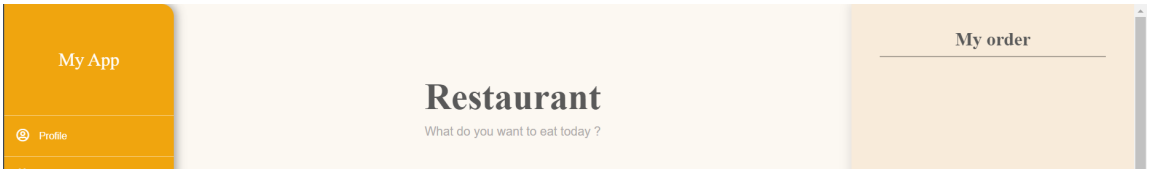


Figure 26: Too many background colors

Overflow problems in css Because its the first time we design a front end with css, so we may make some mistake that lead to page overflow. The consequence is that it appears the scrollbar on the right.

5.3.2 New mockup design

One lesson that we learnt is that we should draw a mockup of our design first before code the actual page. Because it will help us better visualize and adjust the design in early stages.

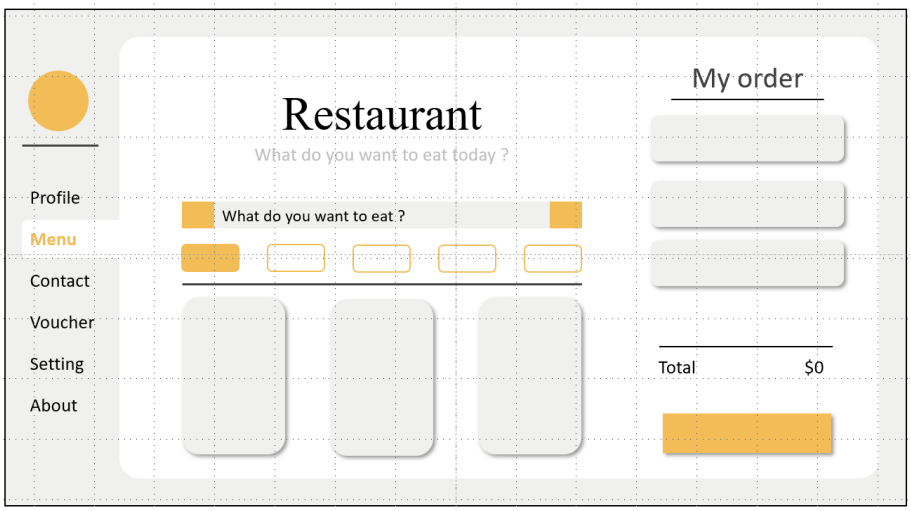


Figure 27: New POS main page mockup

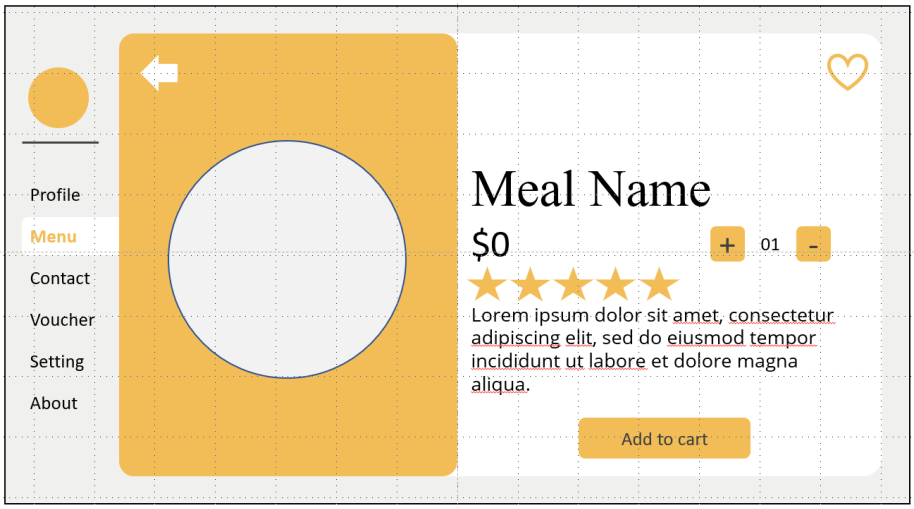


Figure 28: New POS item information page mockup

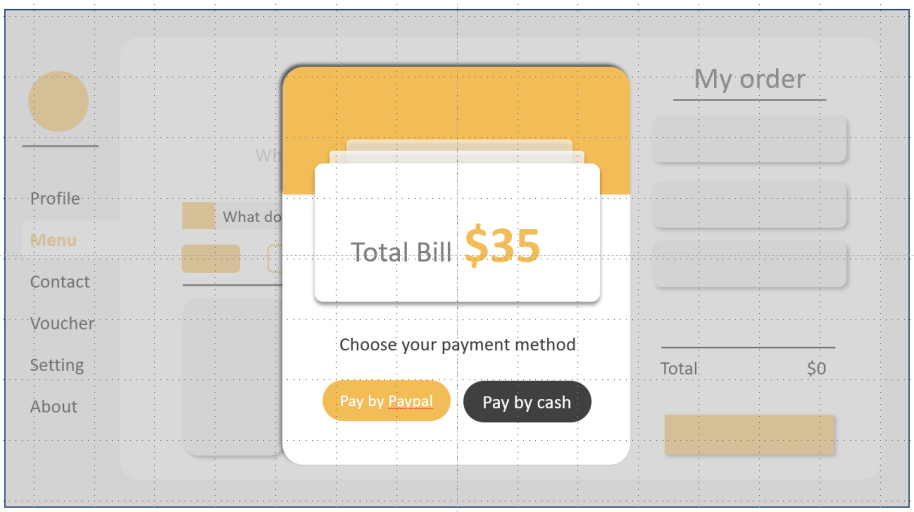


Figure 29: New payment page mockup

We have chosen a more modern design, with a concept of a dashboard. We also improve our color combination with with is the main theme, in addition with a lower contrast color on top of that.

5.4 POS implementation

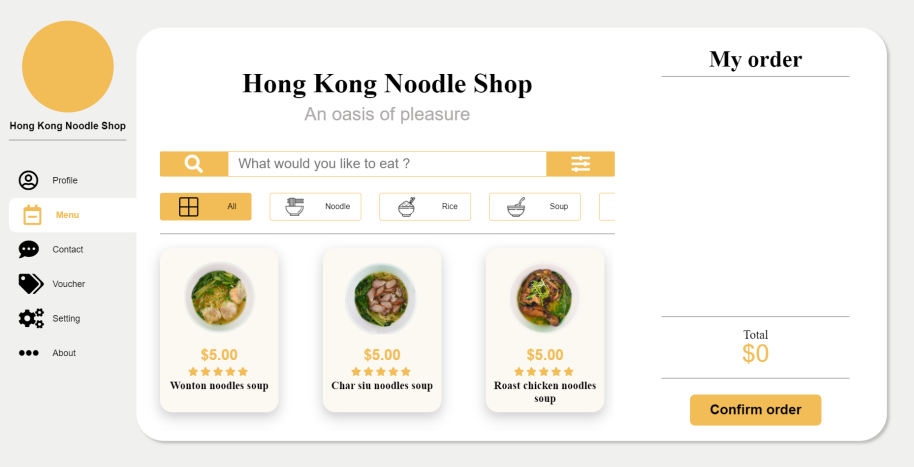


Figure 30: Full width main page

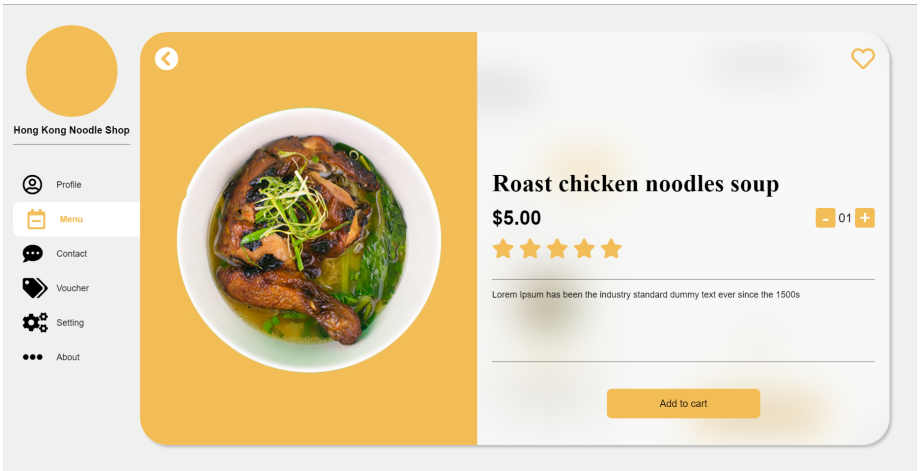


Figure 31: Full width information card before adding to cart

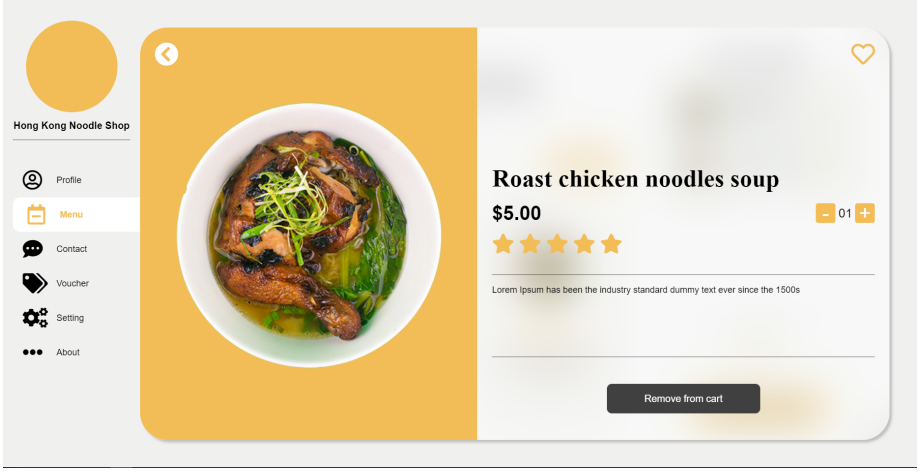


Figure 32: Full width information card after adding to cart

For the item information part, we use glassmorphism style to create a modern feel in our design. Moreover, after adding to cart, the button change to remove from cart. By that we can avoid having uncontrollable behaviour from the user.

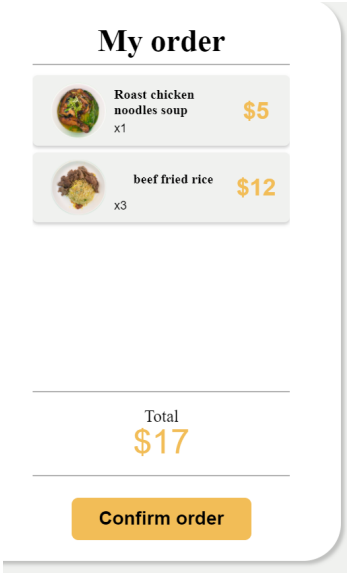


Figure 33: Cart section

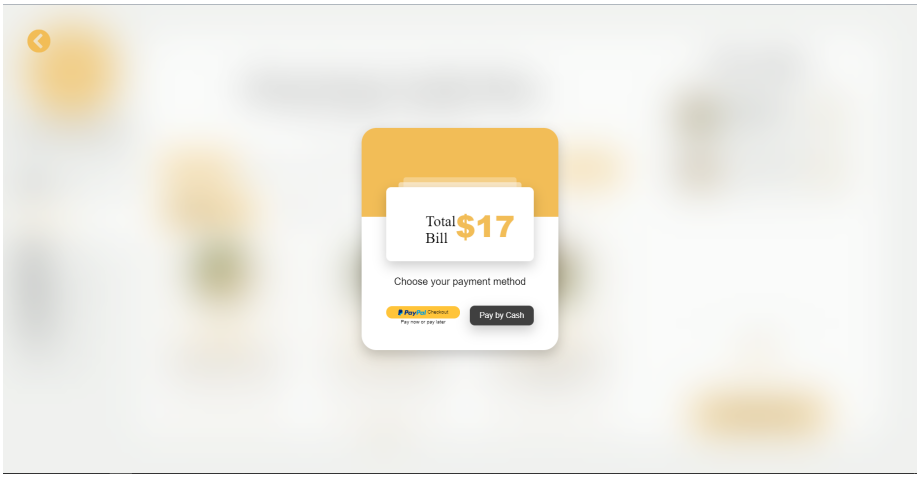


Figure 34: Payment section

We also use glassmorphism background for the payment section. In our implementation so far, a customer can pay by cash or pay by Paypal. If a customer choose to pay by cash, the system will sent a notification to the clerk panel for them to manually receive the cash. On the other hand, if a user decide to pay by Paypal, the POS system will open an Paypal box for them to sign-in Paypal account then pay for the meal.

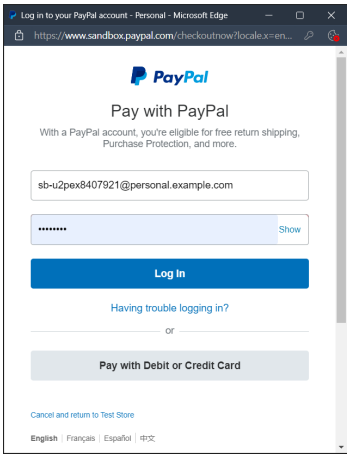


Figure 35: Paypal box

Mobile version Our web application also supports various platform, therefore we have to implement the responsiveness of features.

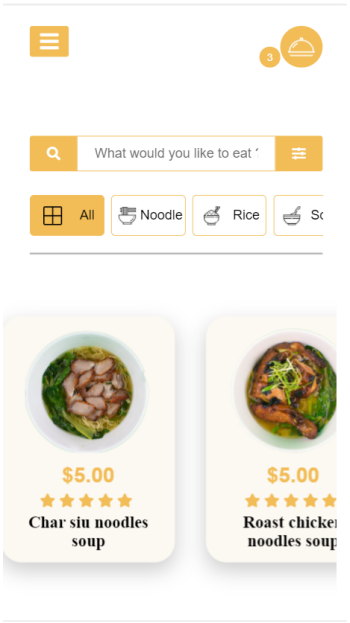


Figure 36: Mobile version main page

Regarding the main page in mobile version, we change the layout of the list of items. Instead of scrolling vertically, the user can scroll horizontally.

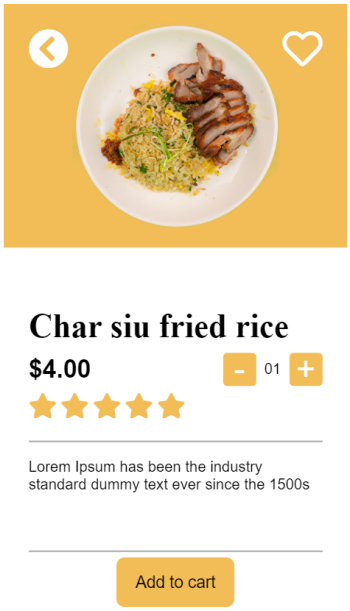


Figure 37: Mobile version information page

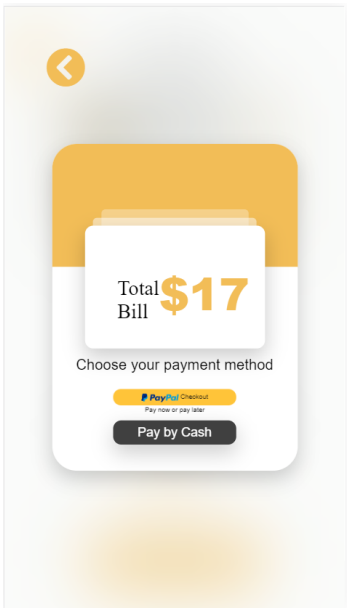


Figure 38: Mobile version payment page

Indirect communication For this features, we have implemented a simple chatbox for the customer to communicate with the clerk. Because each table have different id, therefore each

customer sit in each table will communicate with the clerk with table id.

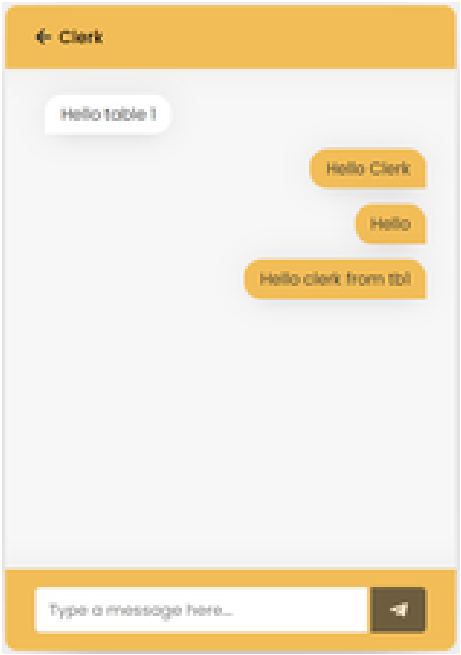


Figure 39: Customer chatbox to communicate with the clerk

5.5 Clerk page

In this version, we have updated some new features for a clerk panel. It offers a clerk to communicate to the customers indirectly via the **Chat** frame and manage the reservations. Besides, the main workflow of a clerk is to perform a final check before sending the order into the kitchen. Because of that, we have implement an easy to use panel for the clerks to see the incoming order, then decide whether it has any problem or not.

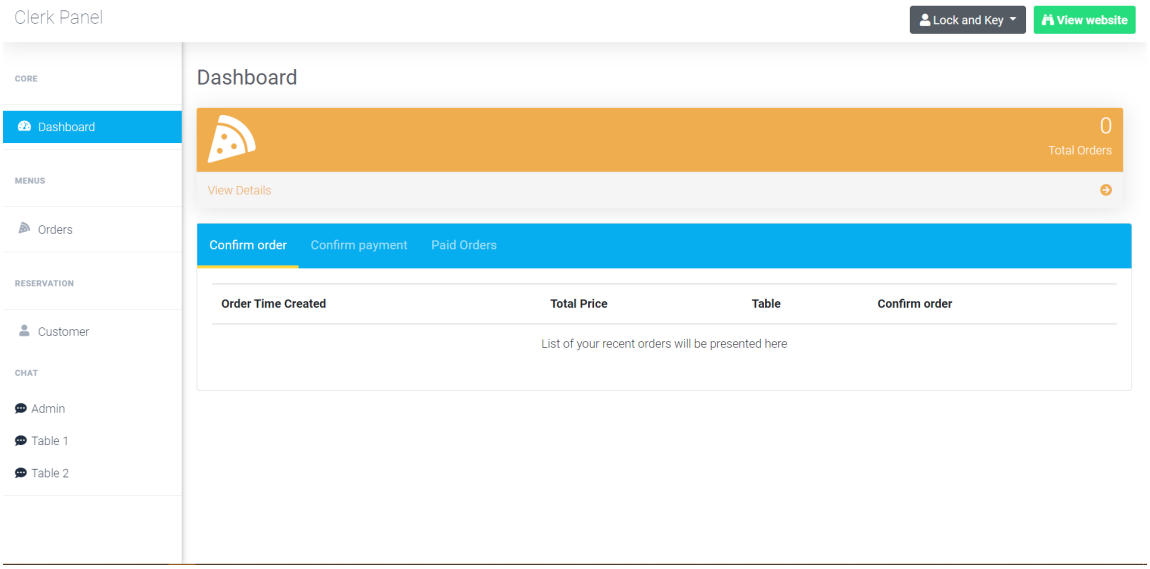


Figure 40: Clerk panel

Figure 41: Clerk chatbox to communicate with the customer

5.6 Admin page

This is the homepage for the admin of the website

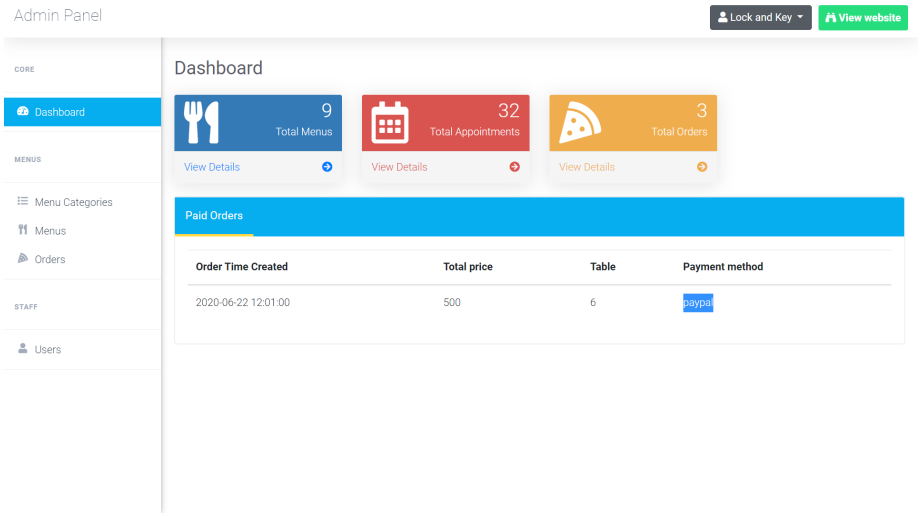


Figure 42: Admin panel

This homepage is quite similar to the clerk's homepage. However, admin has the authority to access in and change menu's data. In addition, admin cannot directly contact with clients.

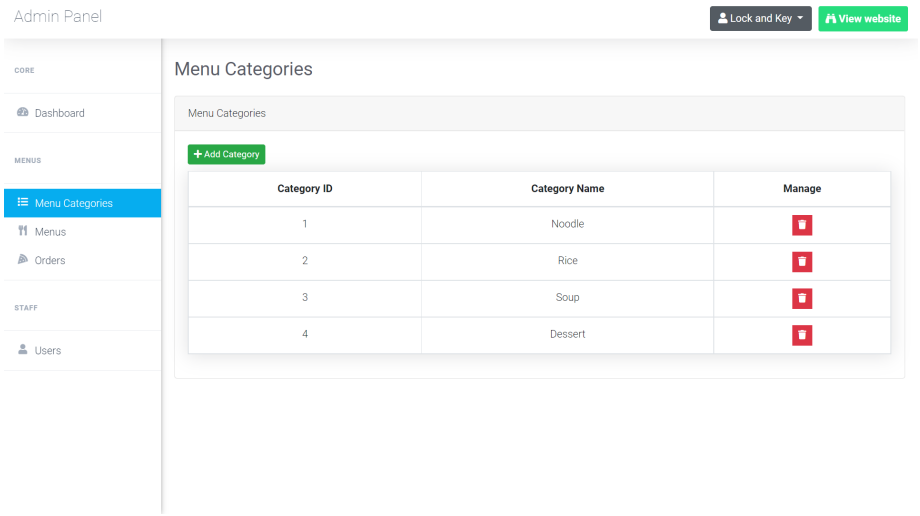


Figure 43: Menu categories

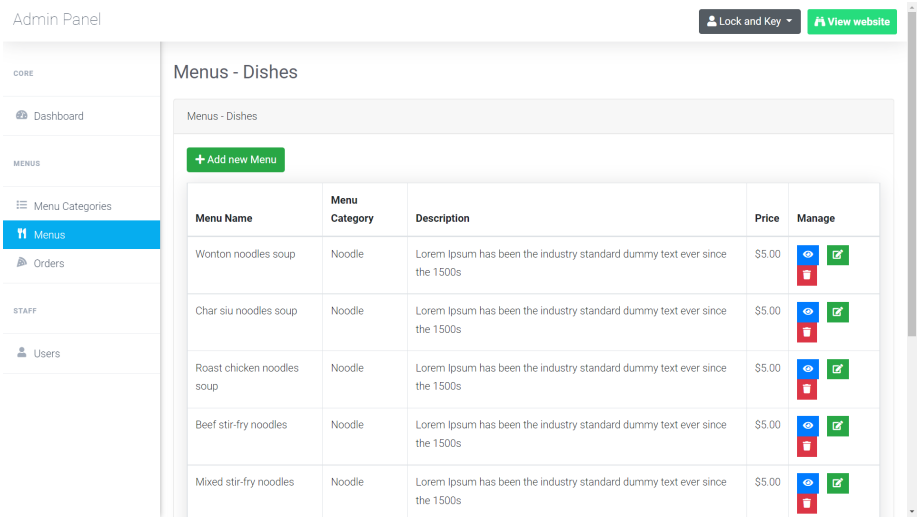


Figure 44: Menus

5.7 Testing

Responsiveness test As we have mentioned before, our application was create for multi platform usage, therefore we could test the responsiveness by opening developer tool in the browsers and reduce the size.

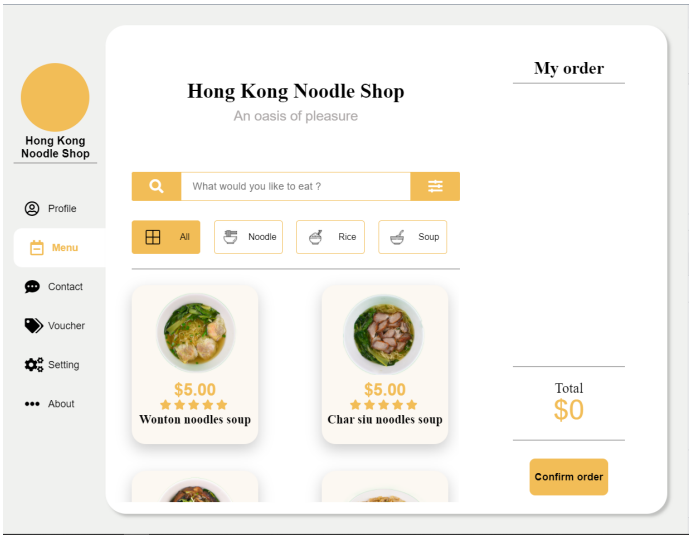


Figure 45: Size reduced test

The result we got shows well responsiveness in our website.

Cart function test One of the most complicated functions in our application is cart function, so we have to test it carefully before launching our application. Because one small bug in this section would cause severe damage to our business.

First we have to test the test case when we add an item to cart, the total number have to increase.

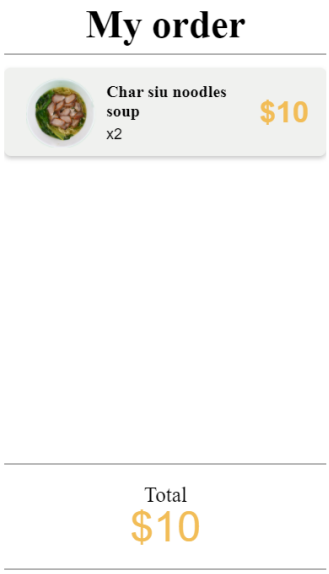


Figure 46: Add to cart test

After we add to cart successfully, the total number has accordingly increased. Next, we will test if changing the quantity in the information card will change the total number or not.

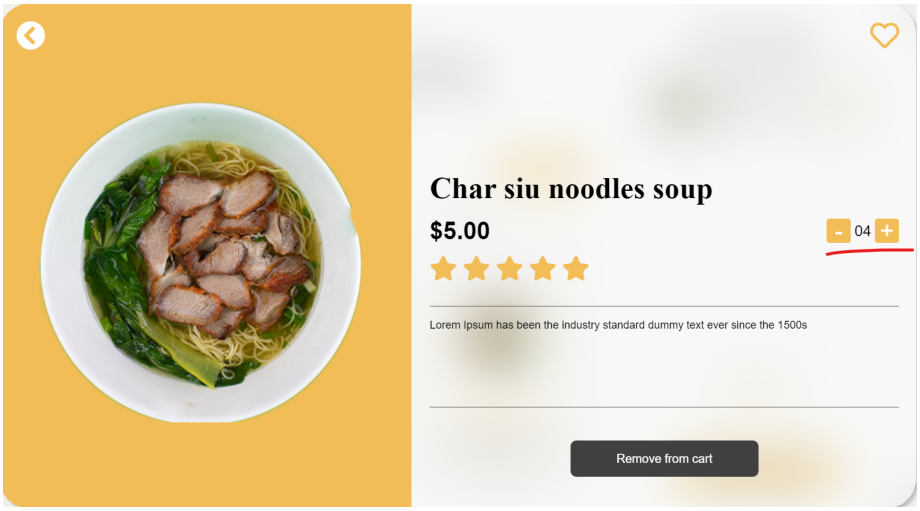


Figure 47: Quantity change test

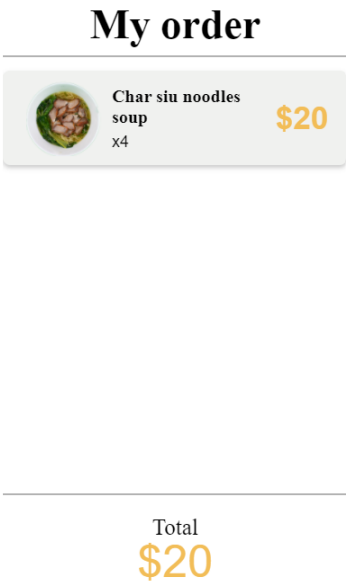


Figure 48: Quantity change result

As we can see, the total bill and the price on the bill section also increase. Finally, we will test the remove button works well or not.

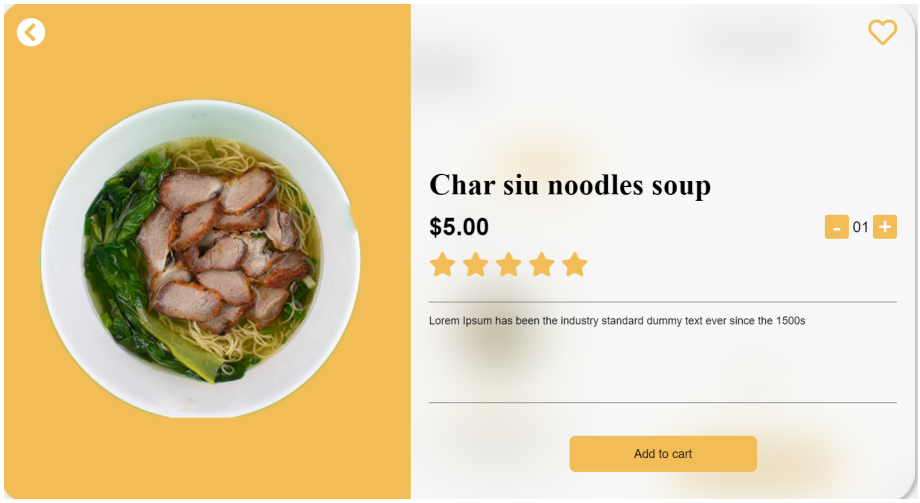


Figure 49: Remove from cart test

Figure 50: Remove from cart result

5.8 Final conclusion on the whole project

In this 12-week project, we have gone through many stages, faced lots of challenges. But in the end, we could proudly say that we have given our best effort to learn and to complete the assignment.
What we have done in this assignment

- **Identifying the requirement** The first step was one of the hardest task in the whole project. Not because the technical aspect but rather the mental aspect. We took a lot of time visualizing the whole project, set detailed timelime but unfortunately, that timeline is not practical at all.
- **Conceptual design the application through diagrams** For this part, we found that this task is like an exercising session for what we have learn. Although we have learn about various kinds of UML diagrams in the lecture time, but actual making a diagrams that fit our requirements is super challenging
- **Implement the project** Finally, we have implemented 2 sprints in our time span. Although we couldn't have the most perfect project, have all the features that we have mention, we have made a huge different in terms of team work efficiency between sprint 1 and sprint 2, such as better team assignment, better communication. Because of that, we have finished more than sprint 1.

What we have learnt through the whole projects The biggest lesson must be the ability to self-learn new languages. From someone that knows nothing about web development, we could say that until now, we could made a real website. From conceptual design to frontend then backend. More than that, team work is what really important to finish this project. Our team went through many discussions, arguing about the development path for our project, but eventually we have learn to believe in each others and work like a team. These are really valuable in our career path.

6 Github link

https://github.com/hungluu6453/BK_211_SE_Assignment.git