

**HO CHI MINH UNIVERSITY OF TECHNOLOGY**  
**FACULTY OF COMPUTER SCIENCE AND ENGINEERING**



**SOFTWARE ENGINEERING ASSIGNMENT**

# **RESTAURANT POS**

Instructor: Lê Đình Thuận

Group: 15

**Member List**

Nguyễn Kế Đạt - 1913048

Nguyễn Diệu Ái - 1910032

Huỳnh Đức Thịnh - 1910563

Đặng Quốc Thanh - 1915083

Nguyễn Phúc Thịnh - 1910565

Trương Hoàng Phúc - 1914720

Nguyễn Ngô Thanh Trúc - 1910650

## Table of Contents

|  |    |
|--|----|
| Table of Contents .....  | 2  |
| Introduction .....   | 4  |
| Stakeholders .....   | 4  |
| Project scope.....   | 5  |
| General Feature of the Project.....                            | 6  |
| Project Context .....  | 6  |
| Functional Requirements.....                                   | 8  |
| Non-functional Requirements .....                              | 9  |
| Project's Use-Case Diagram .....                               | 10 |
| Detailed Use-case .....  | 11 |
| Feature 1: Table Management .....                              | 11 |
| Feature 2: Order System .....                                  | 14 |
| Feature 3: Making Payment.....                                 | 18 |
| Feature 4: View Order History .....                            | 23 |
| Activity Diagram.....  | 26 |
| Feature 1: Table Management .....                              | 26 |
| Feature 2: Order System .....                                  | 28 |
| Feature 3: Making Payment.....                                 | 30 |
| Feature 4: View Order History .....                            | 32 |
| Sequence Diagram.....  | 34 |
| Feature 1: Table Management .....                              | 34 |
| Feature 2: Order System (Make Order - Customer View) .....     | 36 |
| Feature 2: Order System (Confirm Order - Clerk View).....      | 38 |
| Feature 3: Making Payment (Make Payment - Customer View) ..... | 40 |
| Feature 3: Making Payment (Confirm Payment - Clerk View).....  | 42 |
| Feature 4: View Order History .....                            | 44 |
| Class Diagram .....  | 46 |

|                                       |    |
|---------------------------------------|----|
| Architecture design.....              | 48 |
| 1.    Introduction to MVVM .....      | 48 |
| 2.    Applying MVVM to our model..... | 49 |
| 3.    Component Diagram .....         | 51 |
| Implementation Technology.....        | 52 |
| 1.    Front-end Implementation .....  | 52 |
| 2.    Back End Implementation .....   | 53 |
| 3.    Database Management System..... | 54 |
| Implementation.....                   | 55 |
| 1.    Customer View .....             | 55 |
| 2.    Clerk View .....                | 60 |
| 3.    Receptionist View .....         | 63 |
| 4.    Manager View .....              | 65 |
| Demo Video .....                      | 66 |
| GitHub repository.....                | 67 |

## Introduction

Point of sale (POS) or point of purchase is the time and place where a retail transaction is completed. 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 indicates the options for the customer to make payment. In restaurant business, POS systems often include table reservation, ordering food, alerts, billing, credit card processing and customer management.

The new POS system is requested to be developed based on a web-based system and shall implement the current business flow as described below.

## Stakeholders

| Stakeholder             | Role                | Description   |
|-------------------------|---------------------|---|
| Restaurant Owner        | Product Owner       | Who owns the application  |
| Restaurant Manager      | Restaurant Staff    | Who is able to see the orders and payments recorded in the Database |
| Restaurant Clerk        | Restaurant Staff    | Who is responsible for handling customer's orders and payments.     |
| Restaurant Receptionist | Restaurant Staff    | Who is responsible for keeping track of table status                |
| Customer                | Restaurant Customer | Who will use the application to order food and make payment.        |

# Project scope

## Project Justification

The primary goal of this Restaurant POS project is to provide a web-based application that automates many restaurant's processes in order to increase business intelligence, reduce wasted manpower and opportunity to scale to a large business.

## User Story

As a customer of the restaurant:

- I can browse the restaurant menu and look at the various food options available in the restaurant along with the price for each item.
- I am able to select dishes from the menu and add / remove wanted dishes to my order
- I can view my order and change the quantity of products in my order
- I can submit my order to inform the restaurant about my request
- I can make payment for my order either by physical methods (cash, credit cards, ...) or online methods (online banking...)

As a receptionist of the restaurant

- I can view all of the table status, if they are currently occupied or not
- I can change a table's status, from available to occupied or vice versa

As a clerk of the restaurant:

- I can view the customer's submitted orders to inform the kitchen staff.
- I can confirm customer's submitted order to confirm that the order has been finished
- I can view the customer's pending physical payments requests
- I can confirm customer's physical payments to confirm the customers have paid for their order

As the manager of the restaurant:

- I can view the restaurant's record (orders and payments record)
- I can filter the restaurant's record (by date or payment type)

## General Feature of the Project

### **Feature 1: Table Management**

Allowing restaurant receptionists to keep track of which tables are occupied currently and change their status from occupied to unoccupied when customers have finished their meal and vice versa when new customers arrived

### **Feature 2: Order System**

Offering customers an interactive menu and indirect way of ordering food. Customers can view the menu, filter the menu by category and submit the order to the Clerk. Restaurant clerk can view the pending order and confirm the order when the order is finished.

### **Feature 3: Making payment**

Allowing customers to see their total bills and pay for their order, either by physical methods or online methods . If they choose physical methods, the system will send to the clerk the payment details and when the customers have finished their meal, customers will make physical payments with the clerk and the clerk confirms the physical payments

### **Feature 4: View Order History**

Daily orders and its payments are recorded into the Database and this feature allows the restaurant manager to view the restaurant order and payment information.

### **Assumptions**

- Both the restaurant and the customers have access to the Internet when using the application
- Online Payment Transaction are handled by the online payment service

## Project Context

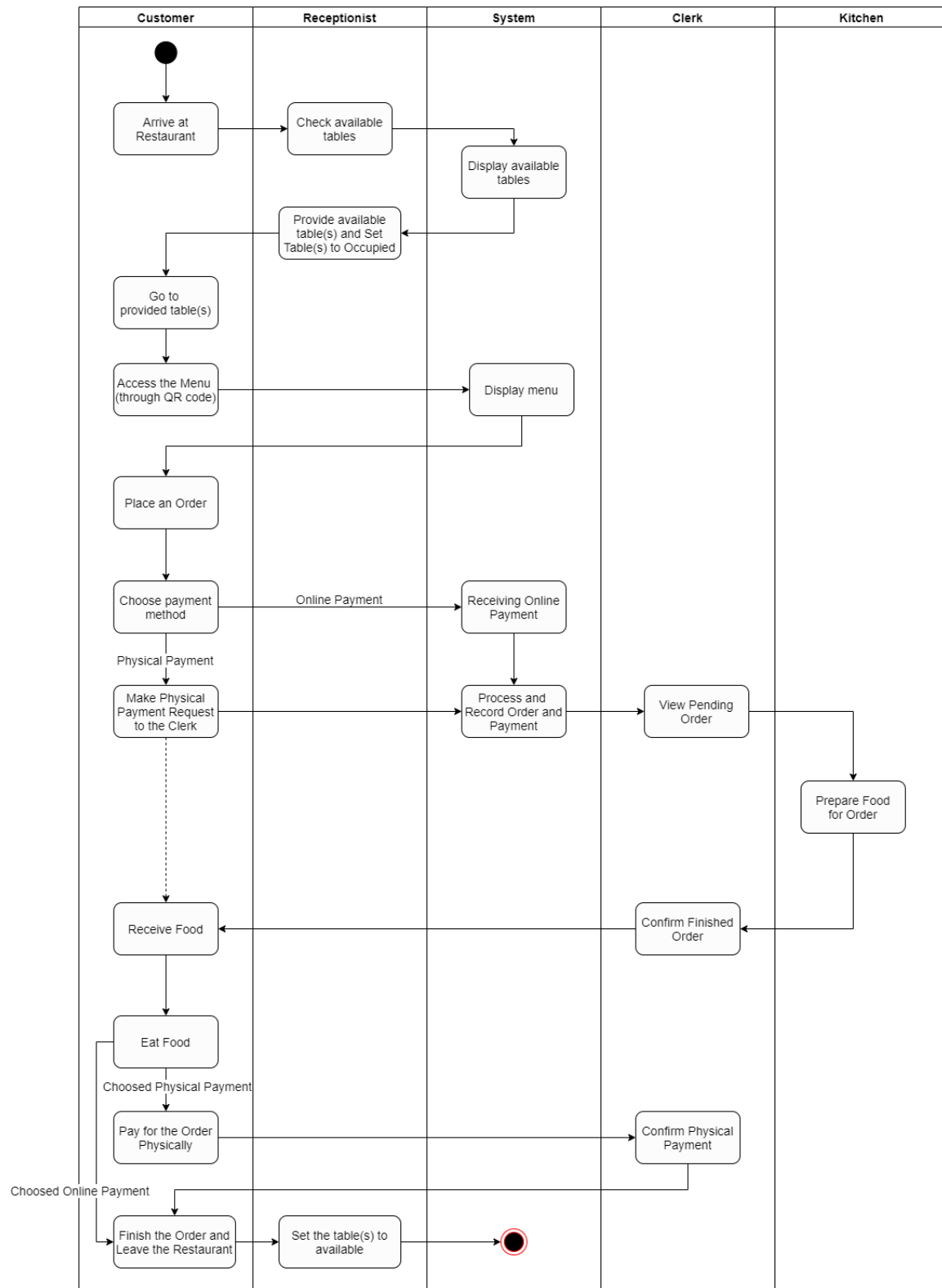
Business Model: Fast food restaurant

Restaurant Customer: Middle Class Customer

Payment: Support physical payment (cash, credit card) and online payment (e-wallet)

Dining service: Eat-in restaurant

## Business Flow Diagram:



## Functional Requirements

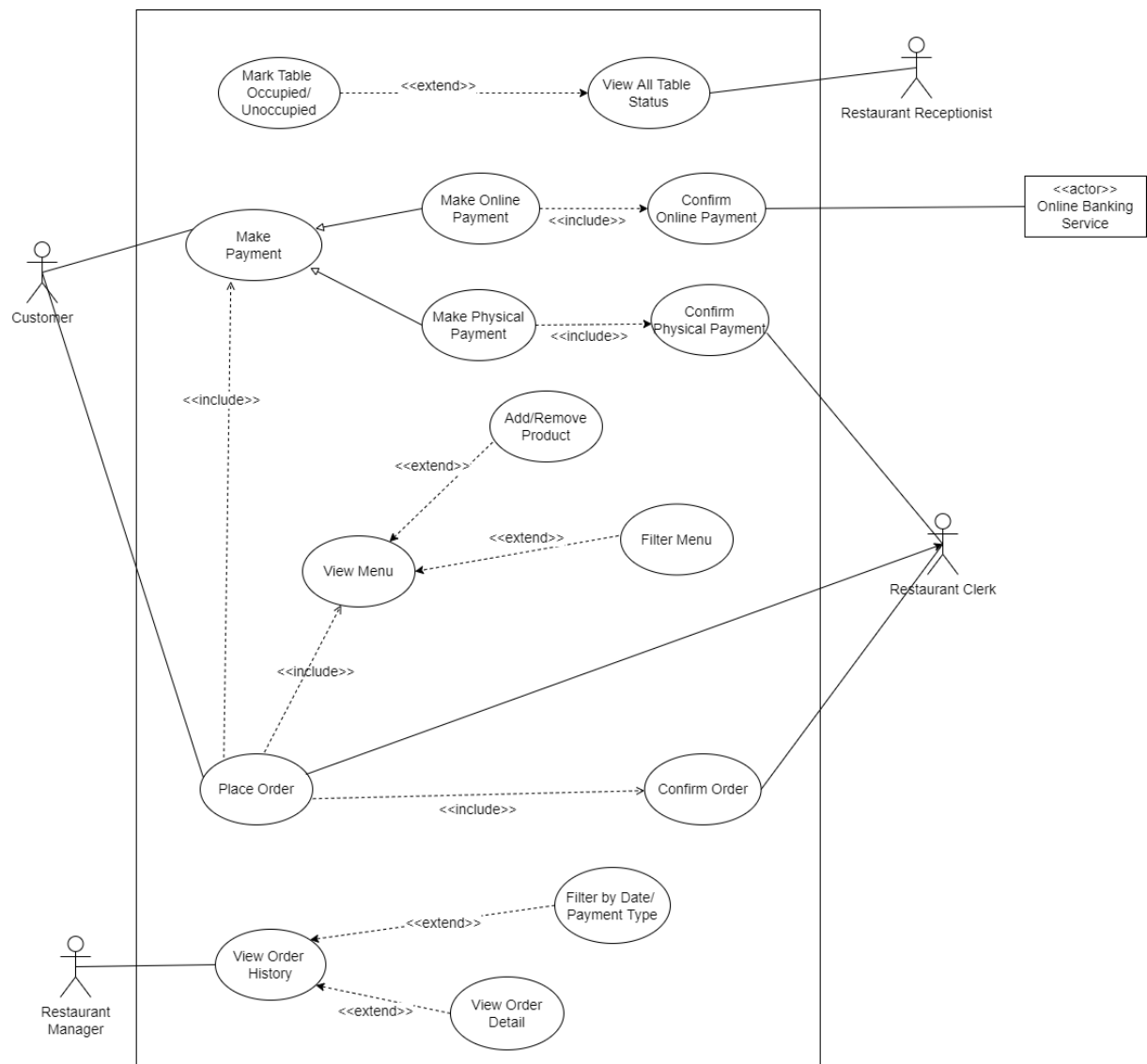
| Requirement ID | Requirement   | Priority | Comments |
|----------------|---|----------|----------|
| FR001          | Restaurant Receptionist can view all table status and update table status                   | Medium   |          |
| FR002          | Customer can view the menu, make food orders and Clerk can confirm the food order           | High     |          |
| FR003          | Customer can make payment for their meal either by direct payment or online banking service | High     |          |
| FR004          | Restaurant Manager can view all the record recorded in the Database by the System           | Low      |          |



## Non-functional Requirements

| Requirement ID | Requirement  | Comments |
|----------------|--|----------|
| NFR001         | The system is implemented using Web technology and QR code               |          |
| NFR002         | The system should be used on mobile, table devices and computer/laptop   |          |
| NFR003         | The system can handle at least 300 orders per day                        |          |
| NFR004         | The system should allow non-direct contact between Clerks and Customers. |          |

## Project's Use-Case Diagram



Use case diagram of the whole system ([Link image](#))

## Detailed Use-case

### Feature 1: Table Management

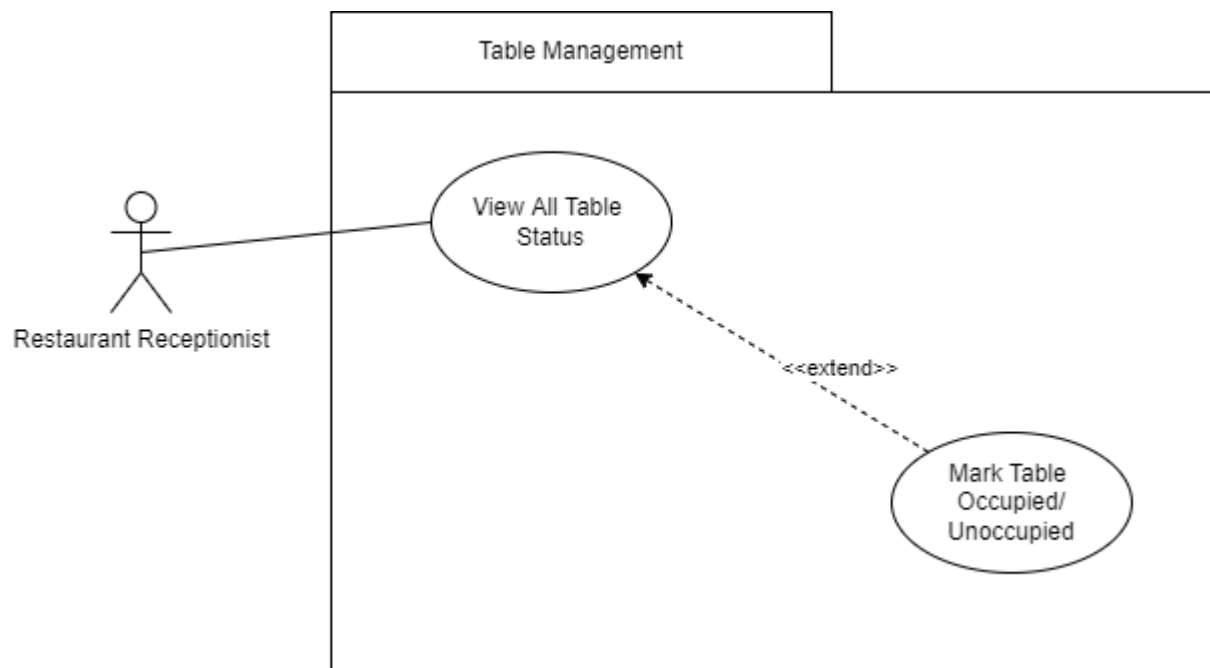


Table Management Use Case Diagram ([Link image](#))

To further investigate Table Management Feature, we make Use Case Tables for that use case as demonstrated by Table UC-1.1, Table UC-1.2.

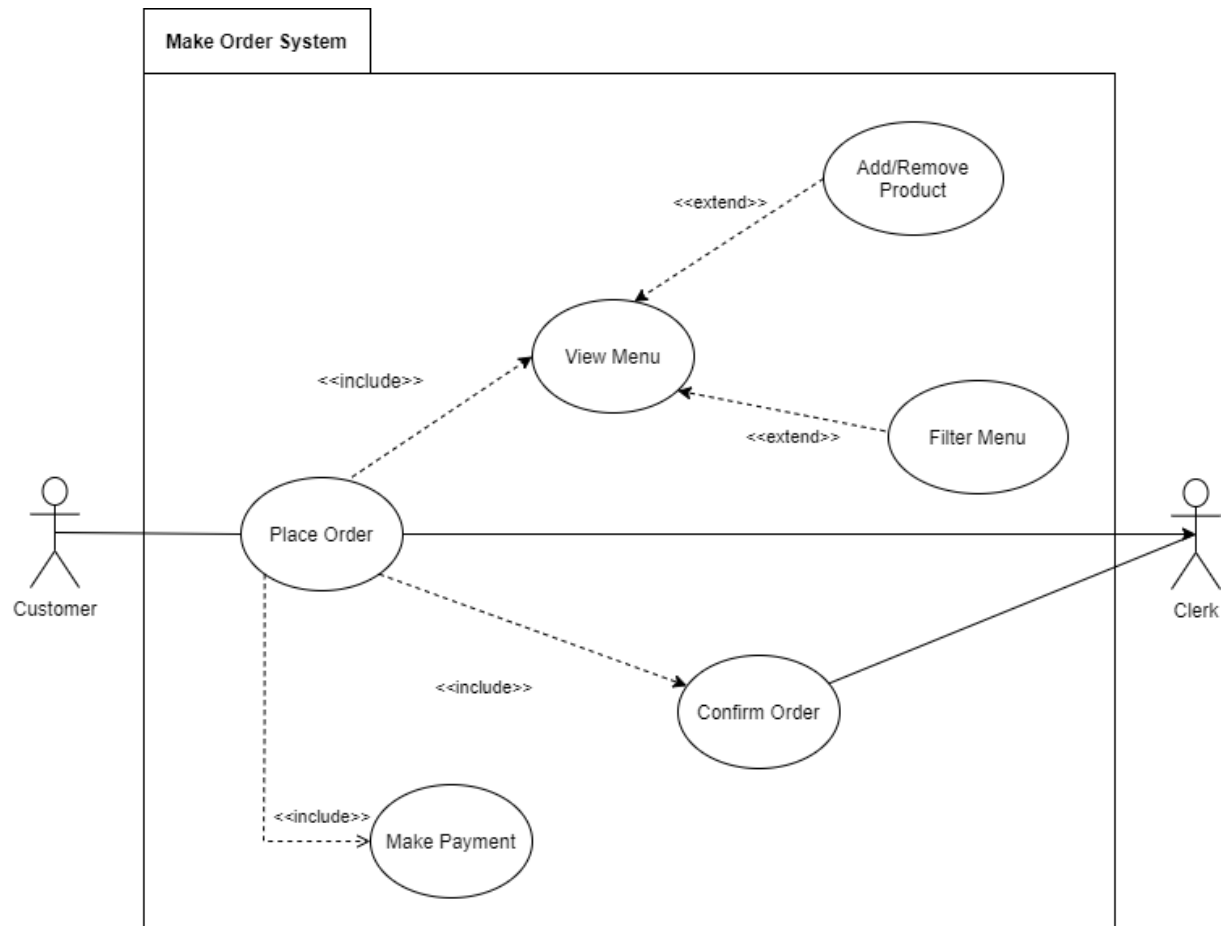
- Table UC-1.1: Table format for View All Table Status use case
- Table UC-1.2: Table format for Mark Table Occupied/ Unoccupied use case

|  |  |
|--|--|
| Use Case ID                                    | UC-1.1   |
| Use Case Name                                  | View All Table Status  |
| Description                                    | Receptionist can view all table status in the restaurant (occupied or unoccupied)  |
| Actor(s)                                       | Restaurant Receptionist  |
| Priority                                       | Medium   |
| Trigger  | Receptionist chooses to View All Table Status  |
| Precondition(s)                                | None   |
| Postcondition(s)                               | System displays all table status<br>System saves changes to table status to the Database (if the receptionist decides to save changes)   |
| Basic Flow                                     | <ol style="list-style-type: none"> <li>1. Receptionist chooses to View All Table Status</li> <li>2. System get the status of all table status from Database</li> <li>3. System displays the status of all table</li> <li>4. Extend:: Mark Table Occupied/Unoccupied</li> </ol> |
| Alternative Flow<br>(Save Changes to Database) | <ol style="list-style-type: none"> <li>5. Receptionist chooses to save</li> <li>6. System updates the change to Database</li> </ol>  |

As we can see from the Table UC-1.1, the View All Table Status use case does extend Mark Table Occupied/Unoccupied use case. Therefore, we provide use case table for Mark Table Occupied/Unoccupied use case (Table UC-1.2)

|   |  |
|---|--|
| Use Case ID                                 | UC-1.2   |
| Use Case Name                               | Mark Table Occupied/Unoccupied   |
| Description                                 | Receptionist can mark a table occupied or unoccupied   |
| Actor(s)                                    | Restaurant Receptionist  |
| Priority                                    | Medium   |
| Trigger                                     | Receptionist chooses to update Table Status  |
| Precondition(s)                             | Receptionist has viewed Table Status List  |
| Postcondition(s)                            | System updates the table status (occupied or unoccupied) in the UI   |
| Basic Flow<br>(Update unoccupied table)     | <ol style="list-style-type: none"><li>1. Receptionist selects a table need to be updated</li><li>2. System checks the selected table status as unoccupied</li><li>3. System updates the table status as occupied in Database</li></ol> |
| Alternative Flow<br>(Update occupied table) | <ol style="list-style-type: none"><li>2a. System checks the selected table status as occupied<ol style="list-style-type: none"><li>1. System updates the table as unoccupied in Database</li></ol></li></ol>                           |

## Feature 2: Order System



Order System Use Case Diagram ([Link image](#))

To further investigate Order System Feature, we make Use Case Tables for that use case as demonstrated by Table UC-2.1, Table UC-2.2, Table UC-2.3.

- Table UC-2.1: Table format for Place Order use case
- Table UC-2.2: Table format for View Menu use case
- Table UC-2.2: Table format for Confirm Order use case

|                  |  |
|------------------|--|
| Use Case ID      | UC-2.1   |
| Use Case Name    | Place Order  |
| Description      | Customers can order the food on the website of the restaurant  |
| Actor(s)         | Customer, Clerk  |
| Priority         | High   |
| Trigger          | Customers ask the System to place the order  |
| Precondition(s)  | Customers accessed the Order System through QR code or Restaurant Website  |
| Postcondition(s) | The Order is processed by the System and is sent to Clerk to be confirm<br>Customers move to Payment Process for the Order |
| Basic Flow       | 1. include::ViewMenu<br>2. include::ConfirmOrder   |

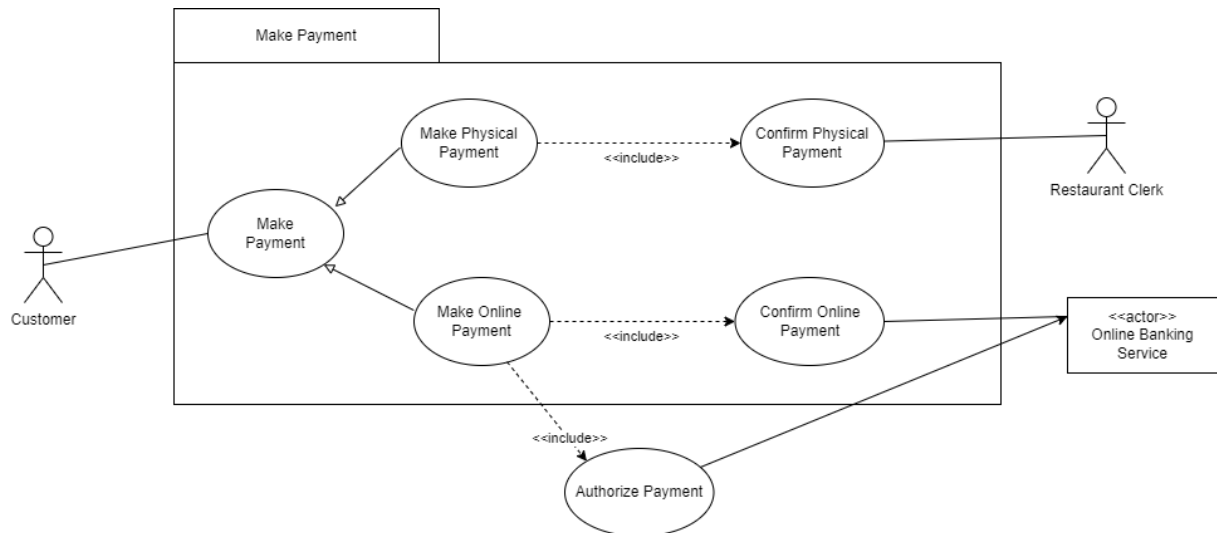
As we can see from the Table UC-2.1, the Place Order use case does include the View Menu use case with Confirm Order use case. Therefore, we provide two more use case tables for View Menu use case (Table UC-2.2) and Confirm Order use case (Table UC-2.3).

|                  |  |
|------------------|--|
| Use Case ID      | UC-2.2   |
| Use Case Name    | View Menu  |
| Description      | Customer views the Restaurant Menu and add /remove Product to their order  |
| Actor(s)         | Customer   |
| Priority         | High   |
| Trigger          | As the customer accesses the Customer's Order Tab, system displays the Menu and option to add product to Customer's Order  |
| Precondition(s)  | Customers accessed the Menu Page through QR code   |
| Postcondition(s) | Customers see the Restaurant Menu and are able to add / remove Product to their order, and submit the Order  |
| Basic Flow       | <ol style="list-style-type: none"> <li>1. Customers accessed the Menu Page</li> <li>2. System gets the list of products in the Menu</li> <li>3. System displays the Menu to Customer</li> <li>4. Customers add or remove the food to their Order</li> <li>5. Customers submit the Order</li> <li>6. System adds the Order Information with Pending status</li> <li>7. Use Case continues at Make Payment Use Case</li> </ol> <p>Extend Point:</p> <ol style="list-style-type: none"> <li>4a. Filter Menu: Customers choose a category in the Filter section and System filters all of the products in that category</li> </ol> |



|                  |  |
|------------------|--|
| Use Case ID      | UC-2.3   |
| Use Case Name    | Confirm Order  |
| Description      | Clerk confirms the customer's order  |
| Actor(s)         | Clerk  |
| Priority         | High   |
| Trigger          | Customer submits a Pending Order and the Kitchen has finished customer's Order   |
| Precondition(s)  | Customer placed an order and sent the Order information to Clerk   |
| Postcondition(s) | Clerk confirms the Pending Order<br>System changes the Order information to Confirmed Status   |
| Basic Flow       | <ol style="list-style-type: none"><li>1. Clerk clicks on Pending Order tab</li><li>2. System displays the Pending Order(s)</li><li>3. Clerk can views the Order's Detail</li><li>4. Clerk confirms the Order</li><li>5. System changes the Order status to Confirmed</li></ol> |

## Feature 3: Making Payment



Making Payment Use Case Diagram ([Link image](#))

To further investigate Making Payment Feature, we make table format for that use case as demonstrated by Table UC-3.1, Table UC-3.2, Table UC-3.3, Table UC-3.4

- Table UC-3.1: Table format for Make Physical Payment use case
- Table UC-3.2: Table format for Confirm Physical Payment use case
- Table UC-3.3: Table format for Make Online Payment use case
- Table UC-3.4: Table format for Confirm Online Payment use case

|                  |  |
|------------------|--|
| Use Case ID      | UC-3.1   |
| Use Case Name    | Make Physical Payment  |
| Description      | After submitting their order, customers choose the physical payment option for their order using the payment feature (by Cash or Credit Card)  |
| Actor(s)         | Customer, Restaurant Clerk   |
| Priority         | High   |
| Trigger          | Customer selects 'Make Payment' Option and select 'Physical Payment' option  |
| Precondition(s)  | Customers have submitted their order   |
| Postcondition(s) | System displays their bills<br>System send the payment information to the Clerk  |
| Basic Flow       | <ol style="list-style-type: none"> <li>1. System gets the Customer's Orders's information</li> <li>2. System displays 2 option for payment: Physical and Online Payment</li> <li>3. Customer chooses Physical Payment</li> <li>4. System records the Payment as Pending Payment</li> <li>5. System displays the bill to customers</li> <li>6. include::Confirm Physical Payment</li> </ol> |

As we can see from the Table UC-3.1, the Make Physical Payment use case does include Confirm Physical Payment use case. Therefore, we provide the use case table for Confirm Physical Payment use case (Table UC-3.2).

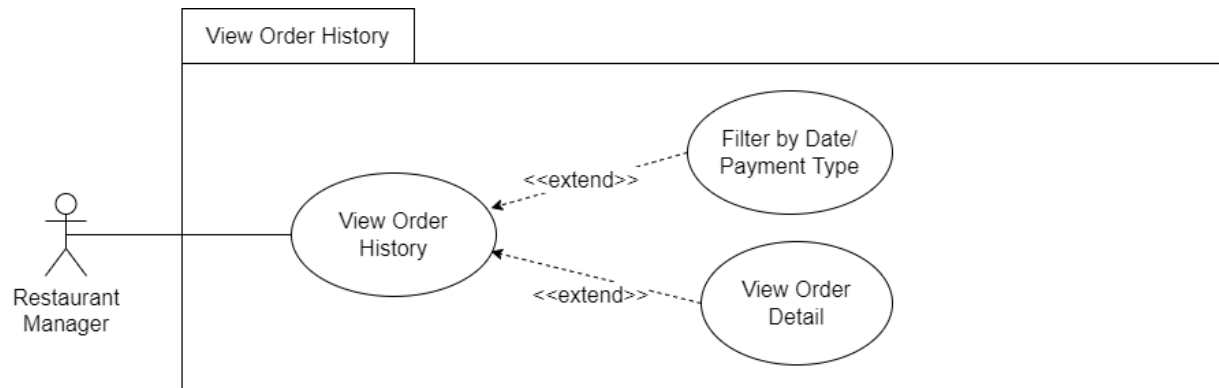
|                  |  |
|------------------|--|
| Use Case ID      | UC-3.2   |
| Use Case Name    | Confirm Physical Payment   |
| Description      | After the system sends a payment request to notify the Restaurant Clerk that a customer has made a payment request, when the customers have finished their meal, the Clerk receives physical payment from the customers and confirms the payment.  |
| Actor(s)         | Restaurant Clerk   |
| Priority         | High   |
| Trigger          | Clerk receives a Payment Request from customer and Customers request to process the physical payment   |
| Precondition(s)  | There are pending physical payment requests.   |
| Postcondition(s) | Customers pay off their bills by physical method.<br>Clerk confirms the Payment  |
| Basic Flow       | <ol style="list-style-type: none"><li>1. Clerk views the Payment Information and requests physical payment from Customer (by cash or by credit card) and go to the table</li><li>2. Customers pay for the Order</li><li>3. Clerk confirms the Payment</li><li>4. System changes payment status in Database from Pending to Confirmed</li></ol> |

|                  |  |
|------------------|--|
| Use Case ID      | UC-3.3   |
| Use Case Name    | Make Online Payment  |
| Description      | After submitting their order, customers choose the online payment option for their order using the payment feature (by online payment service)   |
| Actor(s)         | Customer, Online Payment Service   |
| Priority         | High   |
| Trigger          | Customer selects 'Make Payment' Option and select 'Online Payment' option  |
| Precondition(s)  | Customers have submitted their order   |
| Postcondition(s) | Online Payment Service is notified by the payment request  |
| Basic Flow       | <ol style="list-style-type: none"> <li>1. System gets the Orders's information</li> <li>2. System displays 2 option for payment: Physical and Online Payment</li> <li>3. Customer chooses Online Payment</li> <li>4. System sends Customer to the Online Payment Service to make payment</li> <li>5. System sends Payment Information to Online Payment Service</li> <li>6. Include::Confirm Online Payment</li> </ol> |

As we can see from the Table UC-3.3, the Make Physical Payment use case does include Confirm Online Payment use case. Therefore, we provide the use case table for Confirm Online Payment use case (Table UC-3.4).

|  |  |
|--|--|
| Use Case ID  | UC-3.4   |
| Use Case Name  | Confirm Online Payment   |
| Description  | Online Payment Service sends a Payment Validation to confirm the Online Payment  |
| Actor(s)   | Online Payment Service   |
| Priority   | High   |
| Trigger  | System receives Payment Validation from Online Payment Service   |
| Precondition(s)                                      | A customer has made an online payment request for the Order  |
| Postcondition(s)                                     | Customer finishes the Online Payment Process<br>System records the Online Payment  |
| Basic Flow<br>(Online Payment)                       | <ol style="list-style-type: none"> <li>1. Online Payment Service sends Payment Validation as valid</li> <li>2. System displays the validation to the Customer</li> <li>3. System records the Payment in Database as Confirmed</li> <li>4. System display the bill to the Customer</li> </ol> |
| Exception Flow<br>(Online Payment is not successful) | <p>2a. The Validation that System receives from Online Payment Service is not valid</p> <ul style="list-style-type: none"> <li>● System return Customer back to Select Payment Step</li> </ul>   |

## Feature 4: View Order History



View Order History Use Case Diagram ([Link image](#))

To further investigate View Order History Feature, we make table formats for that use case as demonstrated by Table UC-4.1, Table UC-4.2

- Table UC-4.1: Table format for View Order History use case
- Table UC-4.2: Table format for View Order Detail use case

|                  |  |
|------------------|--|
| Use Case ID      | UC-4.1   |
| Use Case Name    | View Order History   |
| Description      | Restaurant Manager can view the restaurant order and payment record in Database, can filter the data by date or by payment type  |
| Actor(s)         | Restaurant Manager   |
| Priority         | Low  |
| Trigger          | None   |
| Precondition(s)  | The manager accesses to the manager's view to see the restaurant's Order History   |
| Postcondition(s) | System displays the Order Records or the Payment Records   |
| Basic Flow       | <ol style="list-style-type: none"> <li>1. Restaurant Manager gets access to the manager's view</li> <li>2. System gets list of Orders and Payments from Database</li> <li>3. System displays the list of Order and Payment</li> <li>4. Extend:: View Order Detail<br/>Extend:: Filter by Date / Payment Type</li> </ol> <p>Extend:: Filter by Date / Payment Type</p> <ul style="list-style-type: none"> <li>• The manager chooses the date or type to filter.</li> <li>• The system handles the filter request.</li> <li>• Use case continues at step 3.</li> </ul> |

As we can see from the Table UC-4.1, the View Order History use case does extend the View Order Detail use case. Therefore, we provide the use case table for View Order Detail use case (Table UC-4.2).



|                  |  |
|------------------|--|
| Use Case ID      | UC-4.2   |
| Use Case Name    | View Order Detail  |
| Description      | Restaurant Manager can view the restaurant Order's Detail of a specific order  |
| Actor(s)         | Restaurant Manager   |
| Priority         | Low  |
| Trigger          | Restaurant Manager clicks one of the orders displayed in Manager View  |
| Precondition(s)  | The manager access to the manager's view to see the restaurant Orders  |
| Postcondition(s) | System displays the Order's Detail, including the products in the Order  |
| Basic Flow       | <ol style="list-style-type: none"><li>1. Restaurant Manager clicks one of the orders displayed</li><li>2. System gets Order Detail with the Order ID in database</li><li>3. System displays the Order Detail</li></ol> |

# Activity Diagram

## Feature 1: Table Management

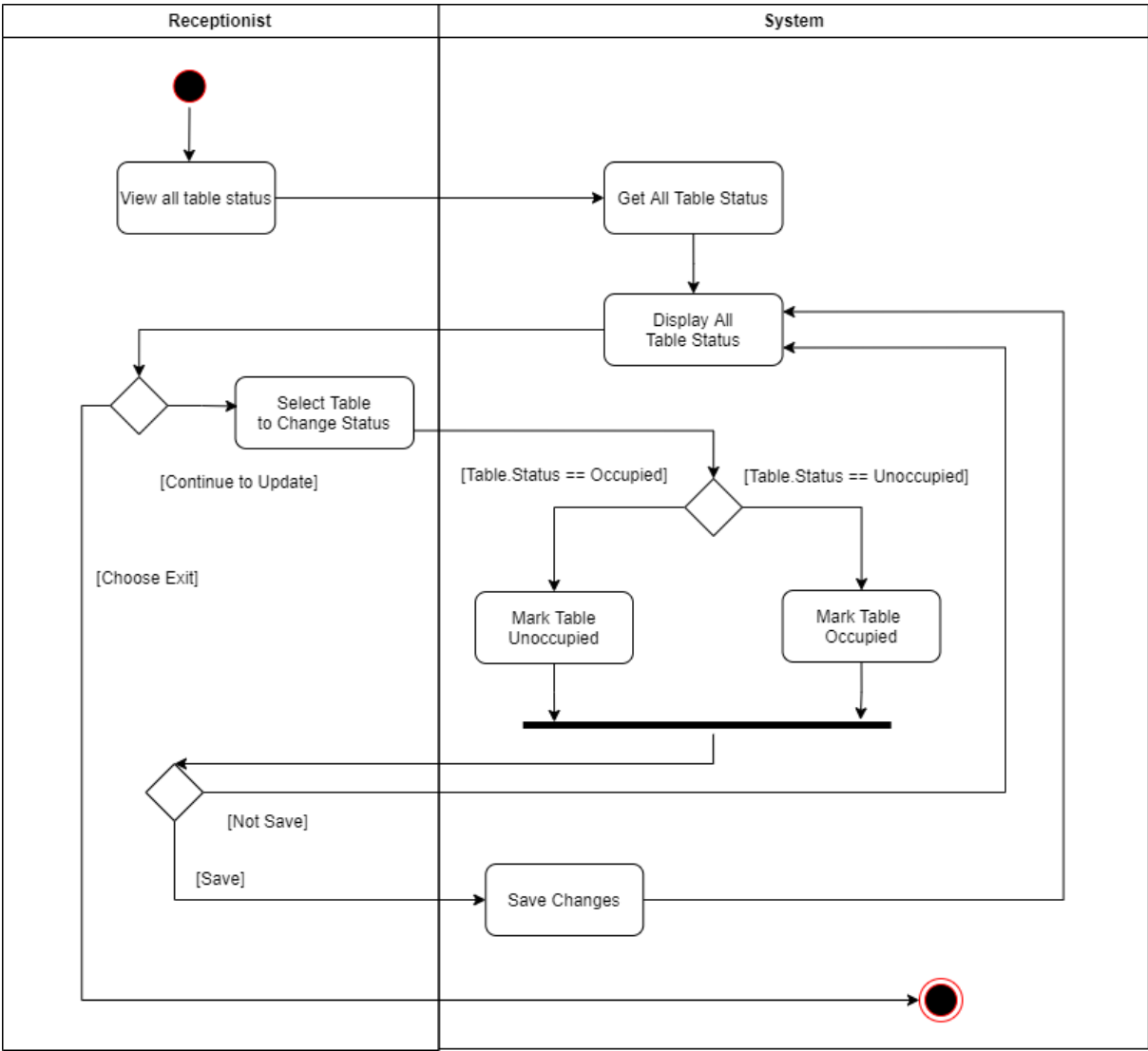
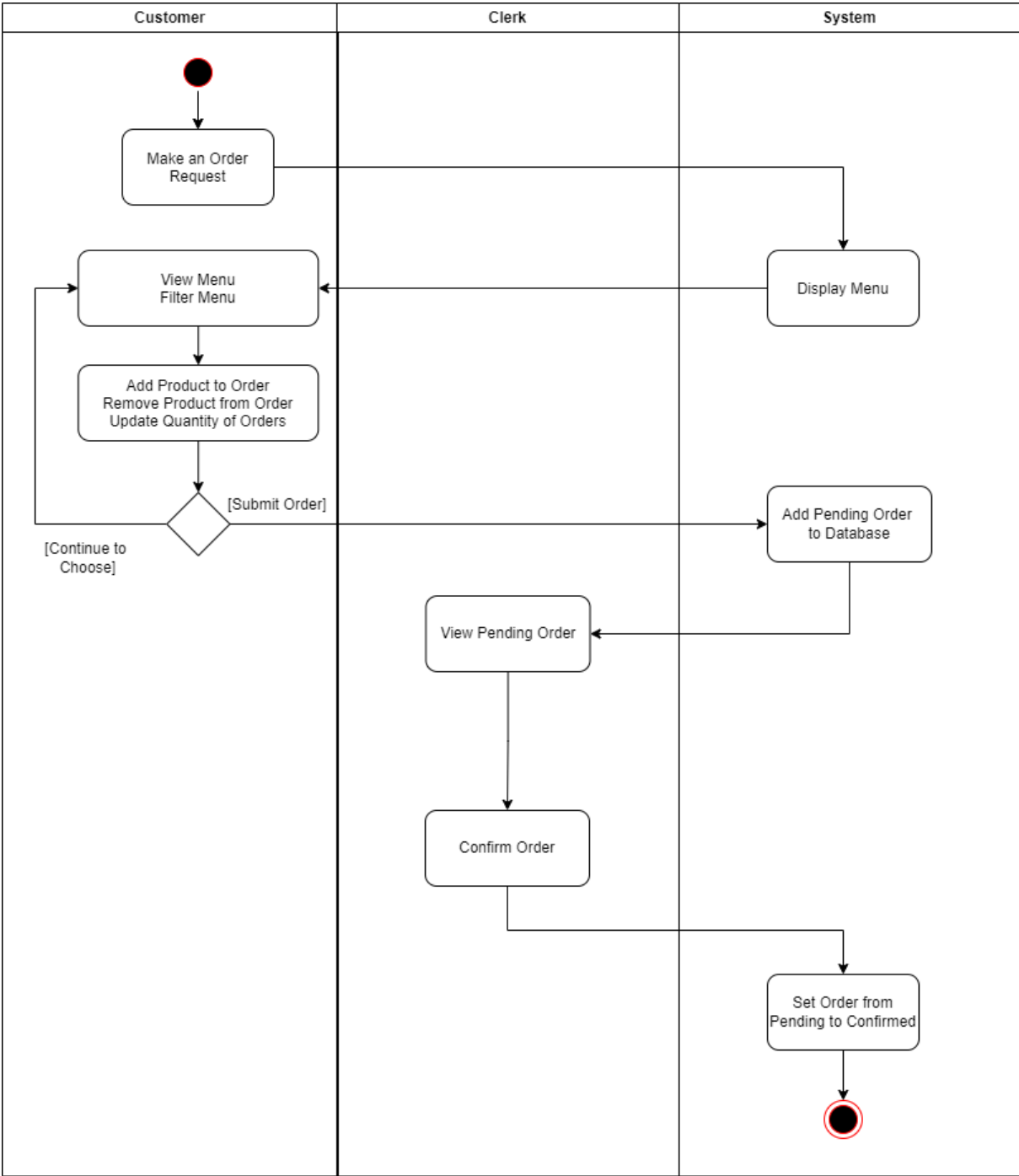


Table Management Activity Diagram ([Link image](#))

Above is the activity diagram and below is the description of Table management process:

1. The Process is divided into 2 swimlanes: Receptionist and the System
2. The Receptionist requests to view all table status in the Receptionist Tab
3. System request to get from database and display all table status on the screen.
4. Now, the Receptionist can choose to update one or many table's status by changing it from Occupied to Unoccupied or vice versa and save all changes to the database or not and exit, or the Receptionist just views the table status list and exit.

Feature 2: Order System

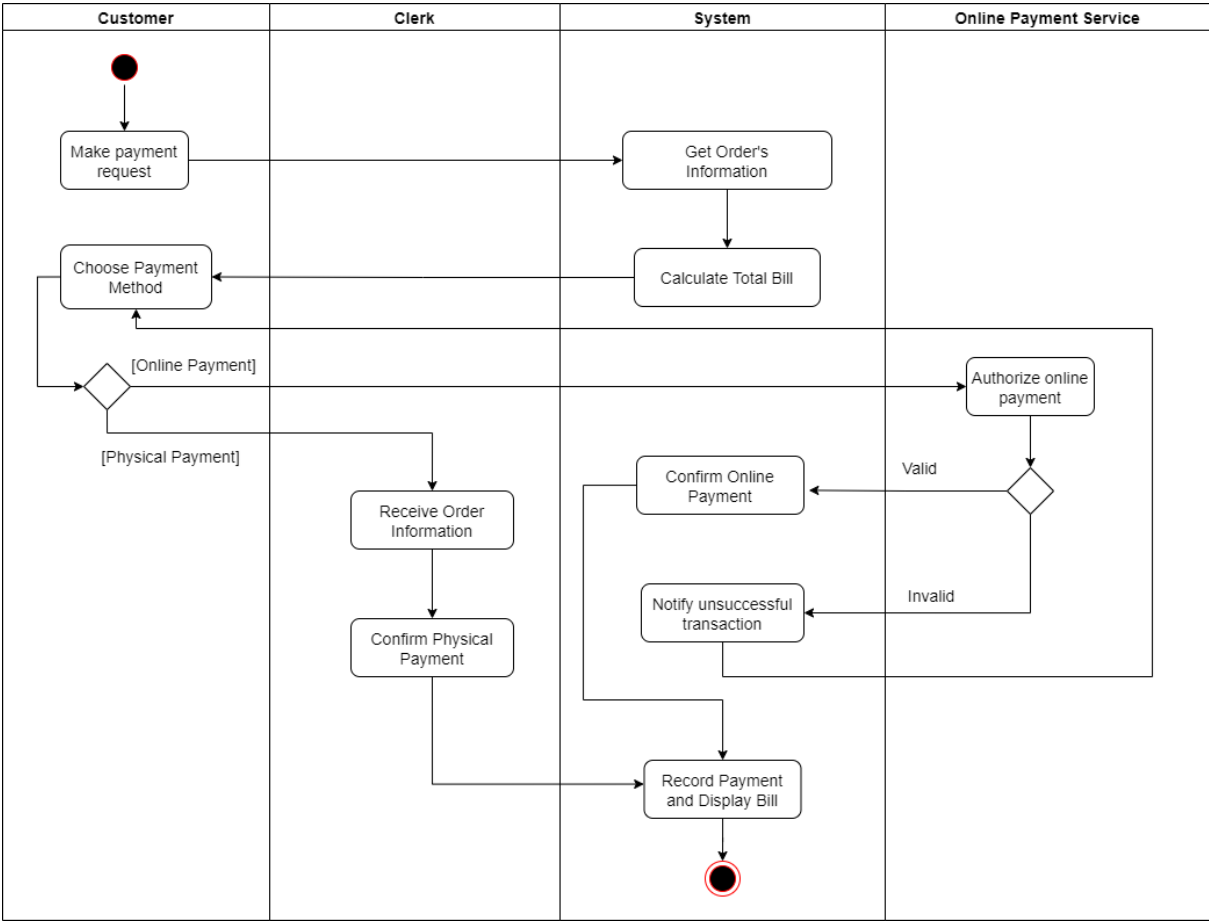


Order System Activity Diagram ([Link image](#))

Above is the activity diagram of Order System Process

1. Order system activity diagram is divided into 3 swimlanes: Customer, Clerk and the System
2. First, after the customer accessed the table through QR code, a 'Make Order' request sent from the customer's device to the System, system will display the menu and create a temporary cart to cache the update (add, remove dish) that customer made to the order.
3. Customers submit the order, System will save the order into the Order Table in the database as "Pending" state Order. Customer will be directed to the Payment Page
4. On the clerk's side, when refreshing the 'Order Management' page, the clerk will view all the pending orders. Clerk can access the details of each order, then click the confirm button to set the state of the order to "confirmed" Order state

Feature 3: Making Payment

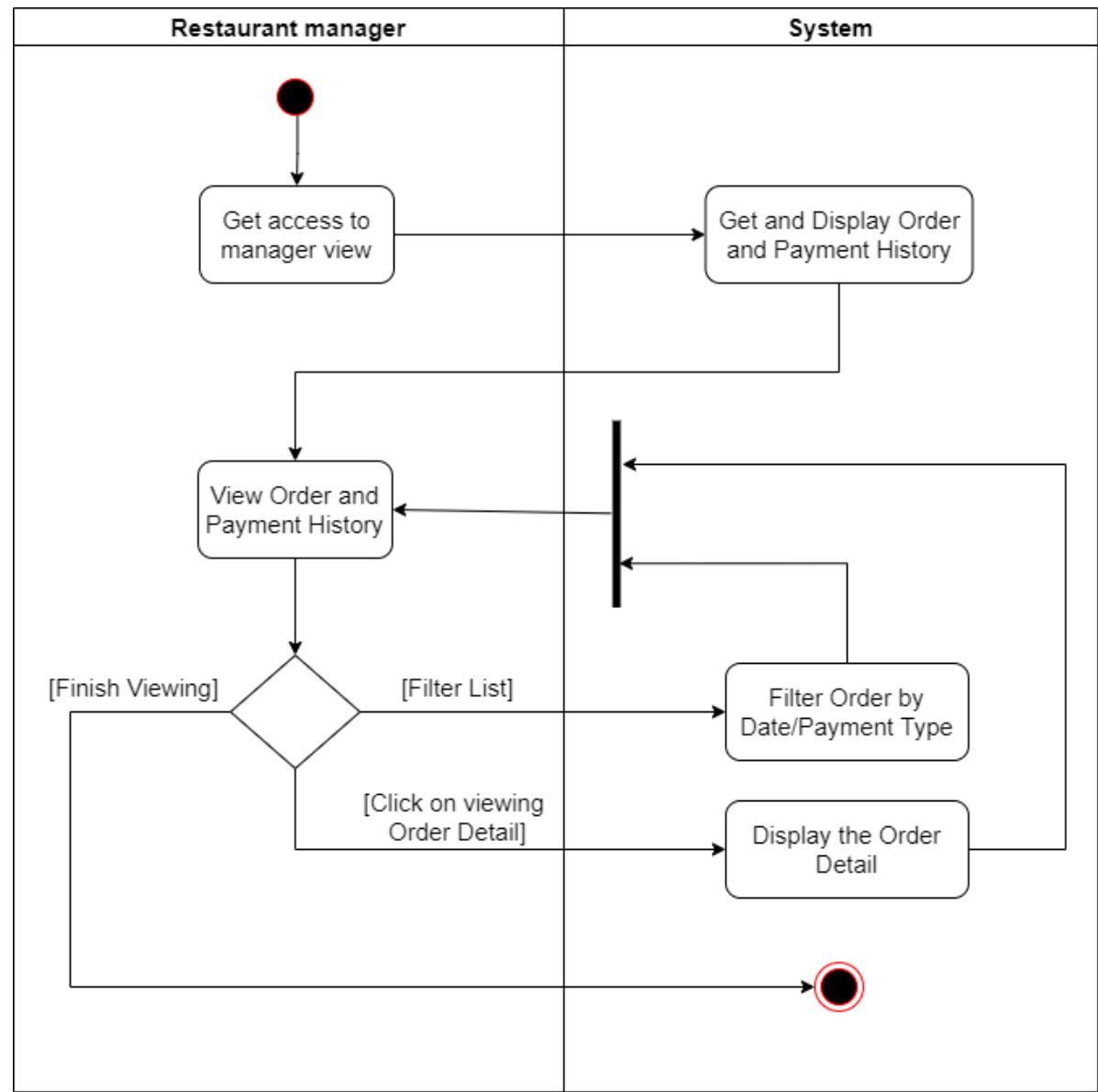


Making Payment Activity Diagram ([Link image](#))

Above is the activity diagram of Making Payment Process

1. The diagram is divided into 4 swimlanes: Customer, Clerk, System and Online Payment Service.
2. When the customers want to make their payment, they send a payment request to the system, the system asks for the payment method they want to use.
  - If customers choose to pay by physical payment method (cash, credit card), the payment request is recorded in the database as a “Pending” state payment. The customers then are directed to the bill page
  - If customers choose to pay by online payment method, their payment is directed to online banking service. Online banking service will verify their transaction and send back validation information to the system.
    - If payment is failed, customers are navigated back to choose the payment method page to choose a different bank or a different method
    - If payment is made successful, the payment is recorded in the database and the customers are directed to the bill page
3. Restaurant Clerk only confirms Physical Payment, while Online Payments are confirmed by online banking service. On the clerk's side, after the customers have finished their orders, the clerk will be called to receive the payment and confirm the payment is successful.

Feature 4: View Order History



View Order History Diagram ([Link image](#))



Above is the activity diagram of View Order History Process

1. The diagram is divided into 2 swimlanes: Restaurant Manager and System
2. When the manager gets access to Manager's view, System gets all Order and its Payment from Database and displays to the Manager's View
3. The manager views the list of Orders and Payments
4. When the manager finishes viewing, the manager finishes the window, the activity finishes.
5. If the manager chooses to see a specific Order Detail, System gets the Order Detail from Database and displays to the view
6. If the manager chooses to filter Orders, the manager chooses the Date or the Payment Type or both, System filters the list according to the selection

# Sequence Diagram

## Feature 1: Table Management

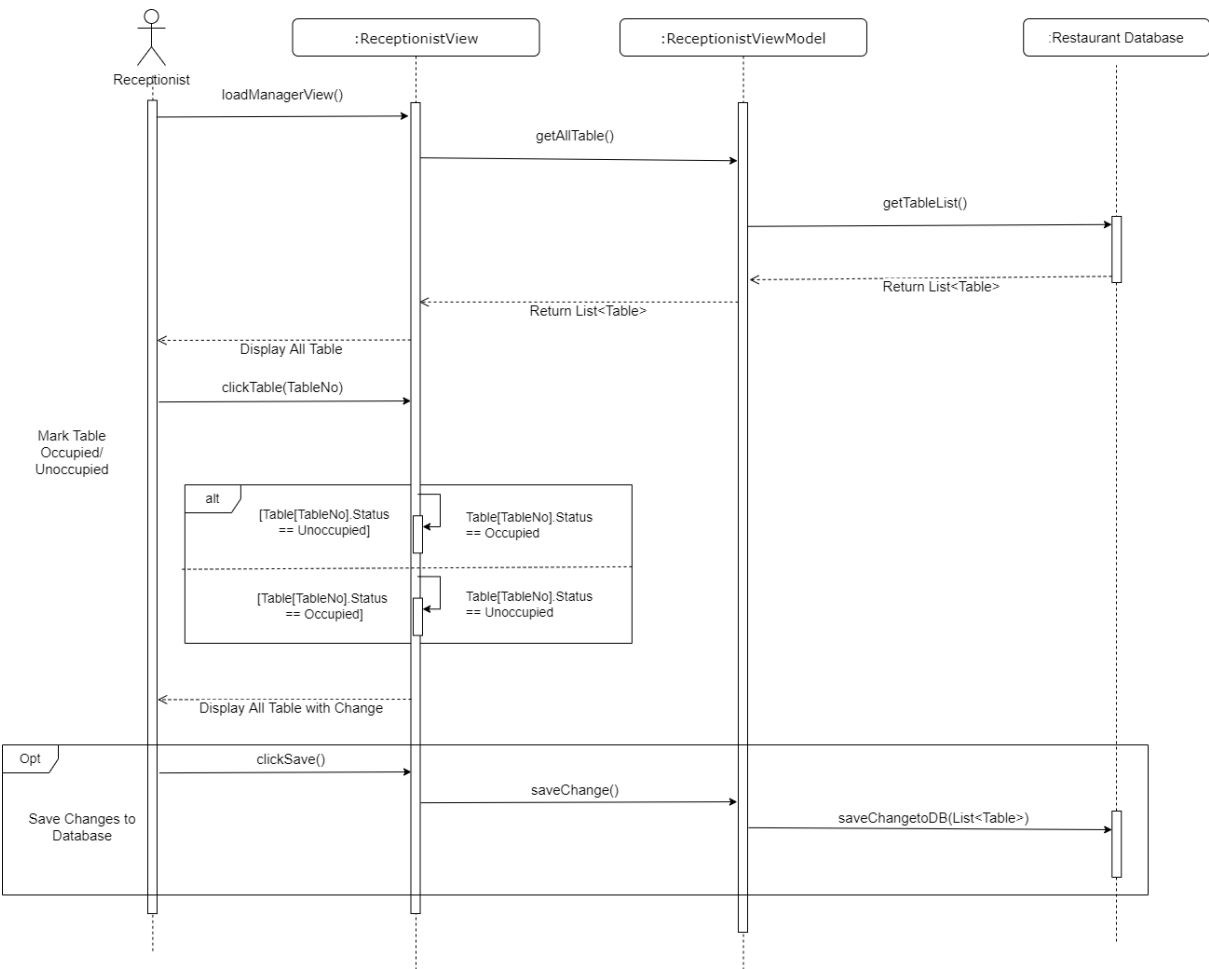
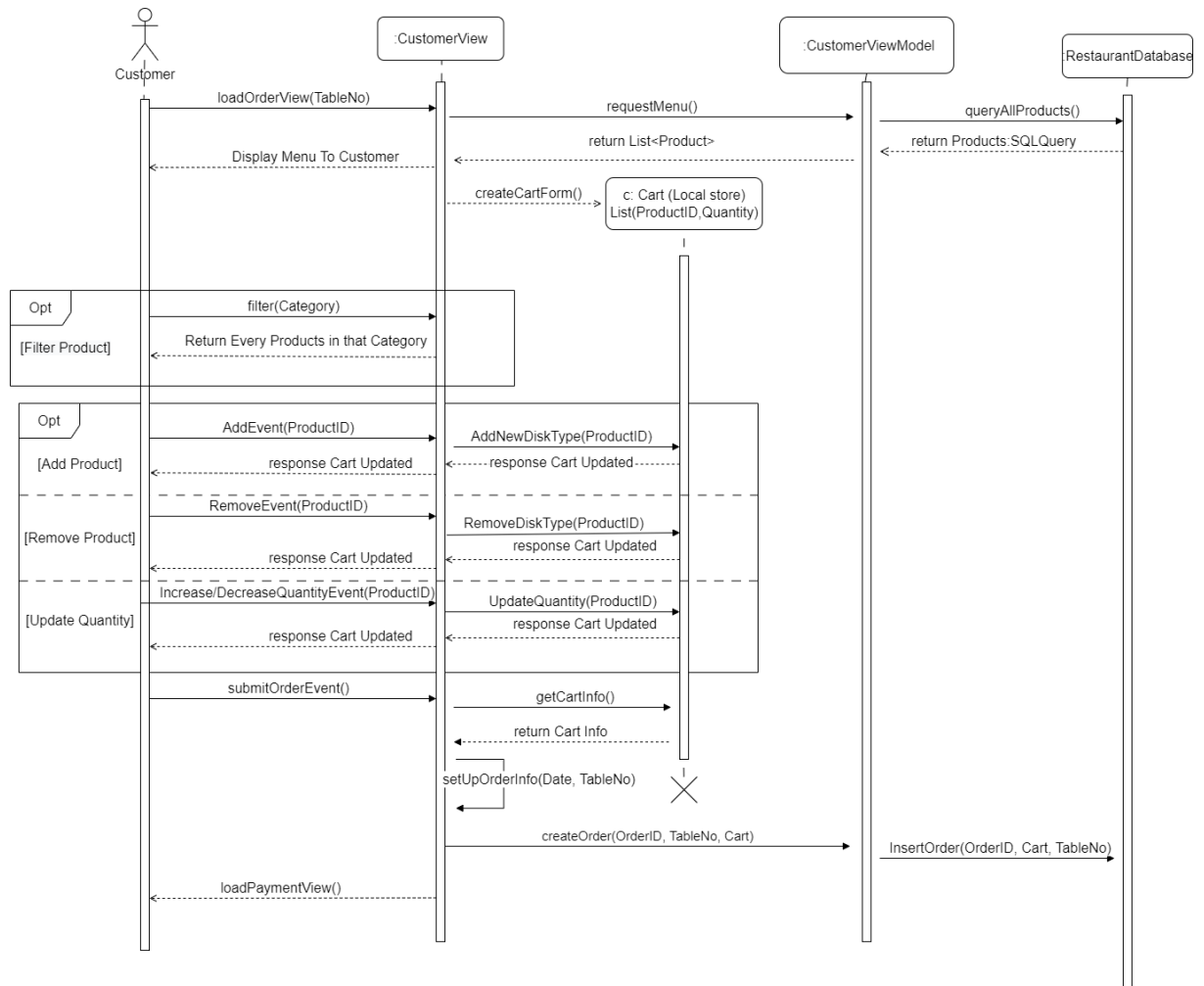


Table Management Sequence Diagram ([Link image](#))

The sequence diagram above presents Table Management Sequence:

1. The receptionist opens the Receptionist Tab to see all the table's status. System sends a request to get the status of all table lists from Database by GetAllTable() and the system returns a List<Table> (Table contains Table Number and Table Status).
2. A table status list will display on the screen to show all the tables with it's status.
3. Now, the receptionist can do as following:
  - The receptionist presses the update button below a table, its status will change from Occupied to Unoccupied or vice versa , this change will be saved locally on the View Model class. After updating the table status successfully, the system will show the table list again with changes.
  - The receptionist clicks the Save button (clickSave()) to save changes made locally to the Database system. The receptionist can continue to update another table or can choose to exit.

## Feature 2: Order System (Make Order - Customer View)

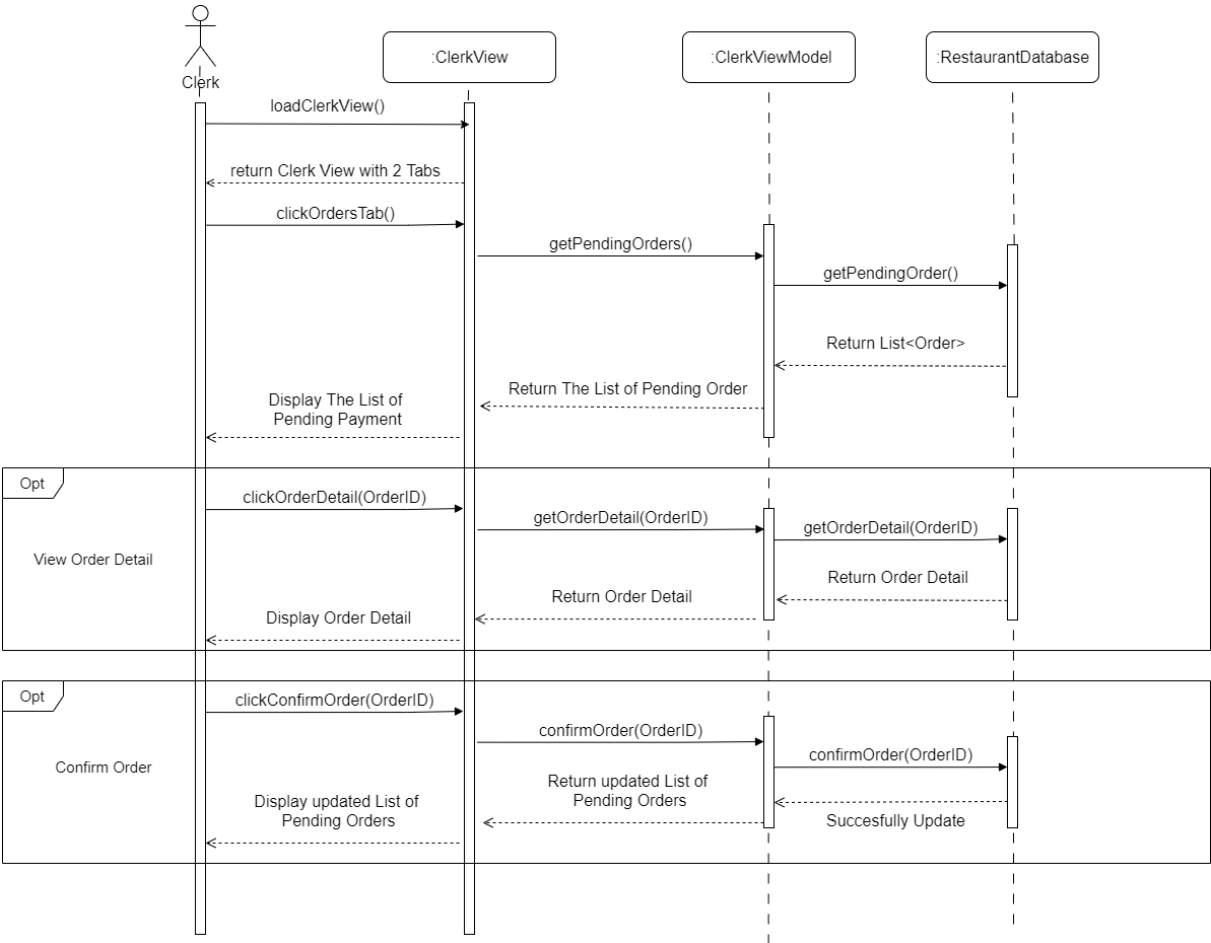


Order System Sequence Diagram - Customer ([Link image](#) - Main Make Order)

The sequence diagram above presents the Make Order process:

1. After the customer accesses the table through QR code, Customer View UI will render at the customer screen. Before displaying, the Customer View-Model will request to the Restaurant Database for Menu Info
2. A Cart object will be created and stored at customer local device to save the info about current order
3. Customers can view the menu and filter dishes according to category. Customers can update the quantities and click the add button to add the disk into the cart or remove added disk from the cart.
4. After finishing choosing the order, the customer clicks the submit button and the Order object will be created with information of Date time, Cart, Table number of the table that made this order, state as 'pending' state and inserted into the database.

## Feature 2: Order System (Confirm Order - Clerk View)

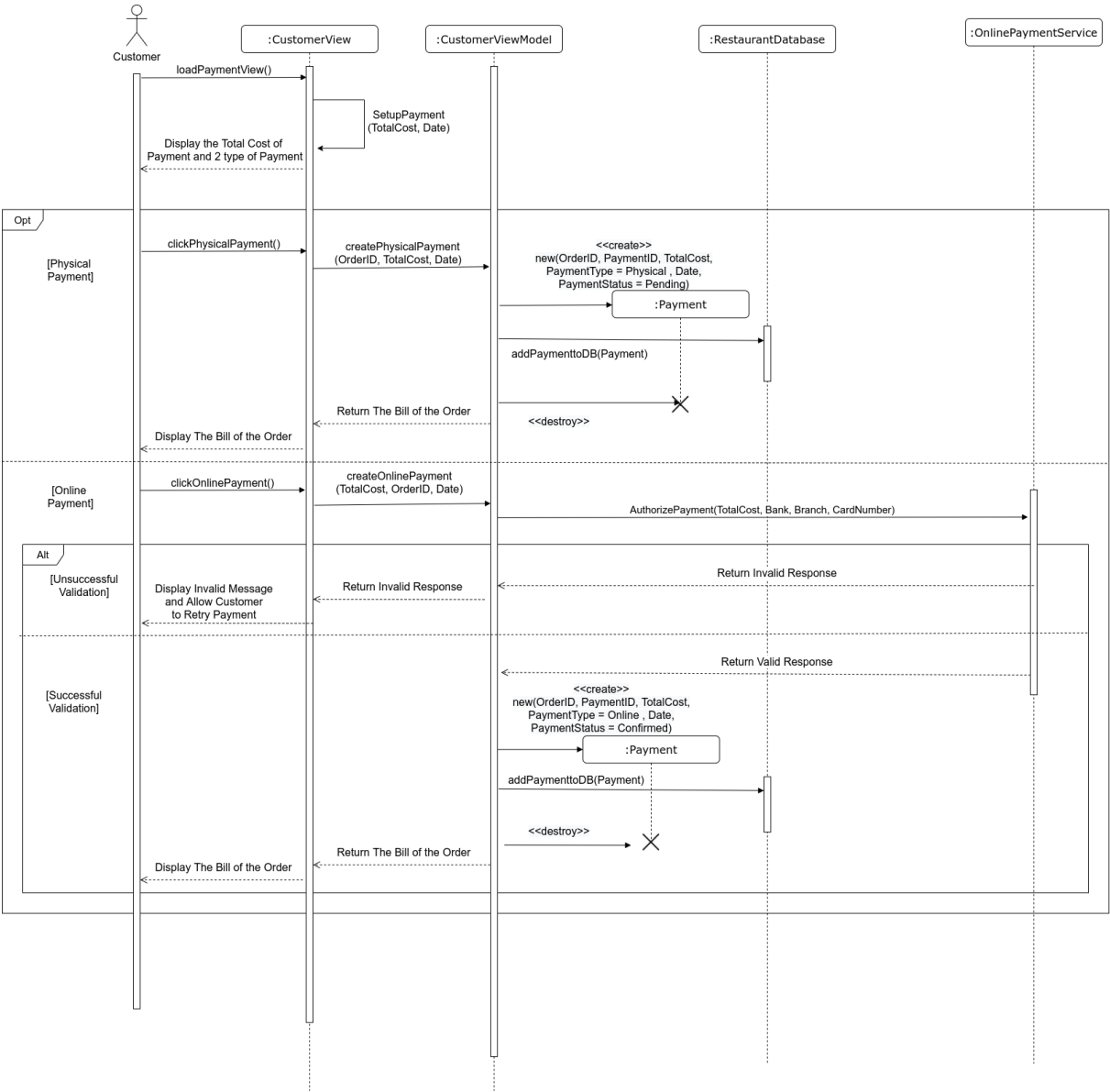


Order System Sequence Diagram - Clerk ([Link image](#) - Main Manage Order)

The sequence diagram above presents the Confirm Order process:

1. Clerk opens the Clerk View and open View Pending Order
2. Clerk clicks the refresh button to load new pending orders from the database
3. Clerk views the detail the order and asks the kitchen to make the order
4. When the kitchen has finished the order, the clerk click confirms on the order, set the order's state to 'confirmed' in the database

### Feature 3: Making Payment (Make Payment - Customer View)



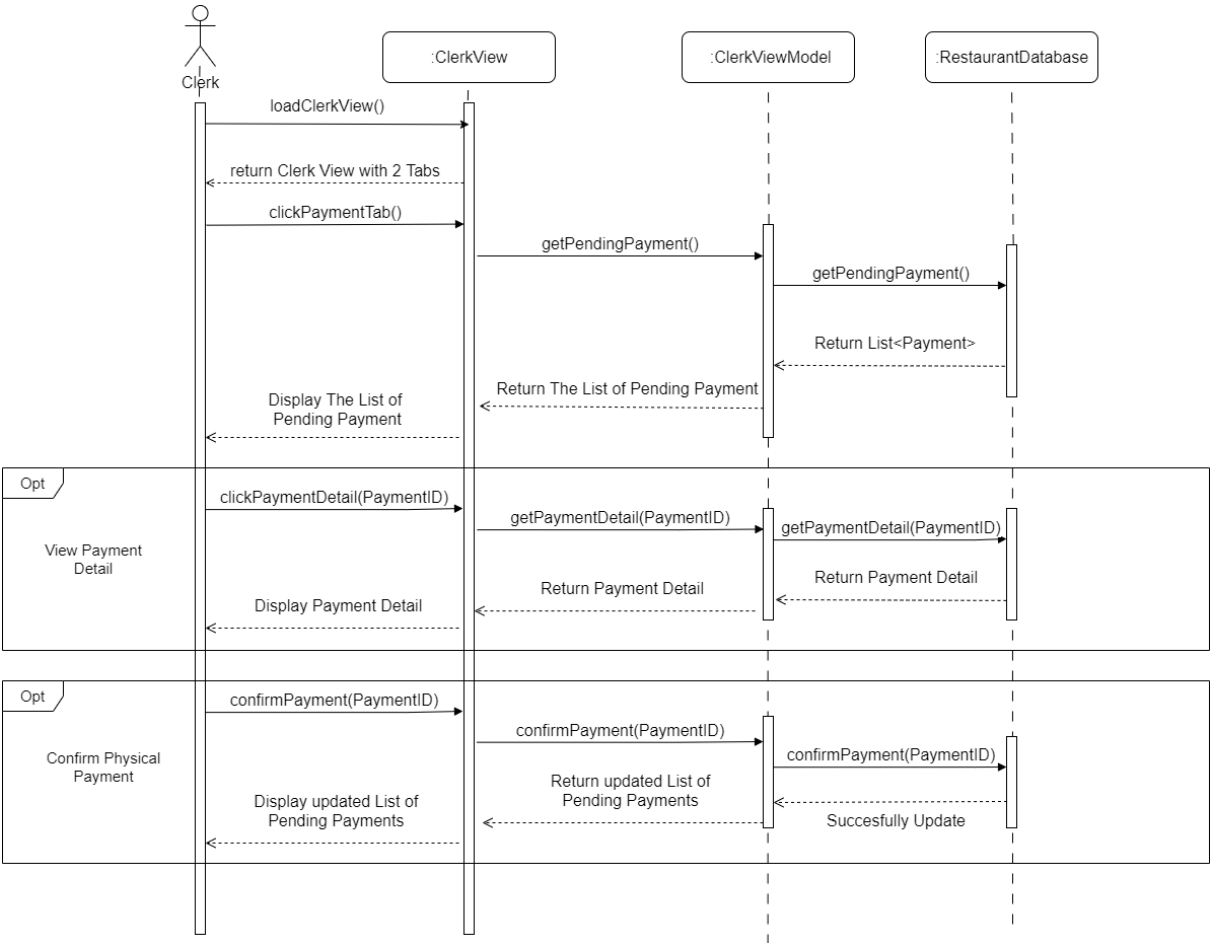
Making Payment Sequence Diagram - Customer ([Link image](#) - Customer)



The sequence diagram above presents Make Payment process:

1. Customers choose to make their payment, (clickPaymentTab()) is called and directs them to the payment page) after receiving the request, Customer View Model retrieves the order information, calculates their bills and displays 2 payment option
2. Customers can choose between two payment methods:
  - If customers choose to make physical payment, “clickPhysicalPayment()” is called, their payment request is sent to the Customer View Model to handle (createPhysicalPayment()). Customer View Model creates a payment object storing payment information, stores it to the database and customers are directed to the bill page
  - If customers choose to make online payment, “clickOnlinePayment()” is called, their payment request is sent to Customer View Model to handle (OnlinePayment()), then send payment request to an online payment service to authorize the transaction (AuthorizePayment(TotalCost)), the service handles the transaction and returns back validation information to the system.
  - If online payment is invalid, customer view will navigate back to the payment page, customer can choose to make online payment again or change to physical payment.
  - If online payment is valid, payment object will be created and store their payment information to the database and customers are directed to the bill page

### Feature 3: Making Payment (Confirm Payment - Clerk View)

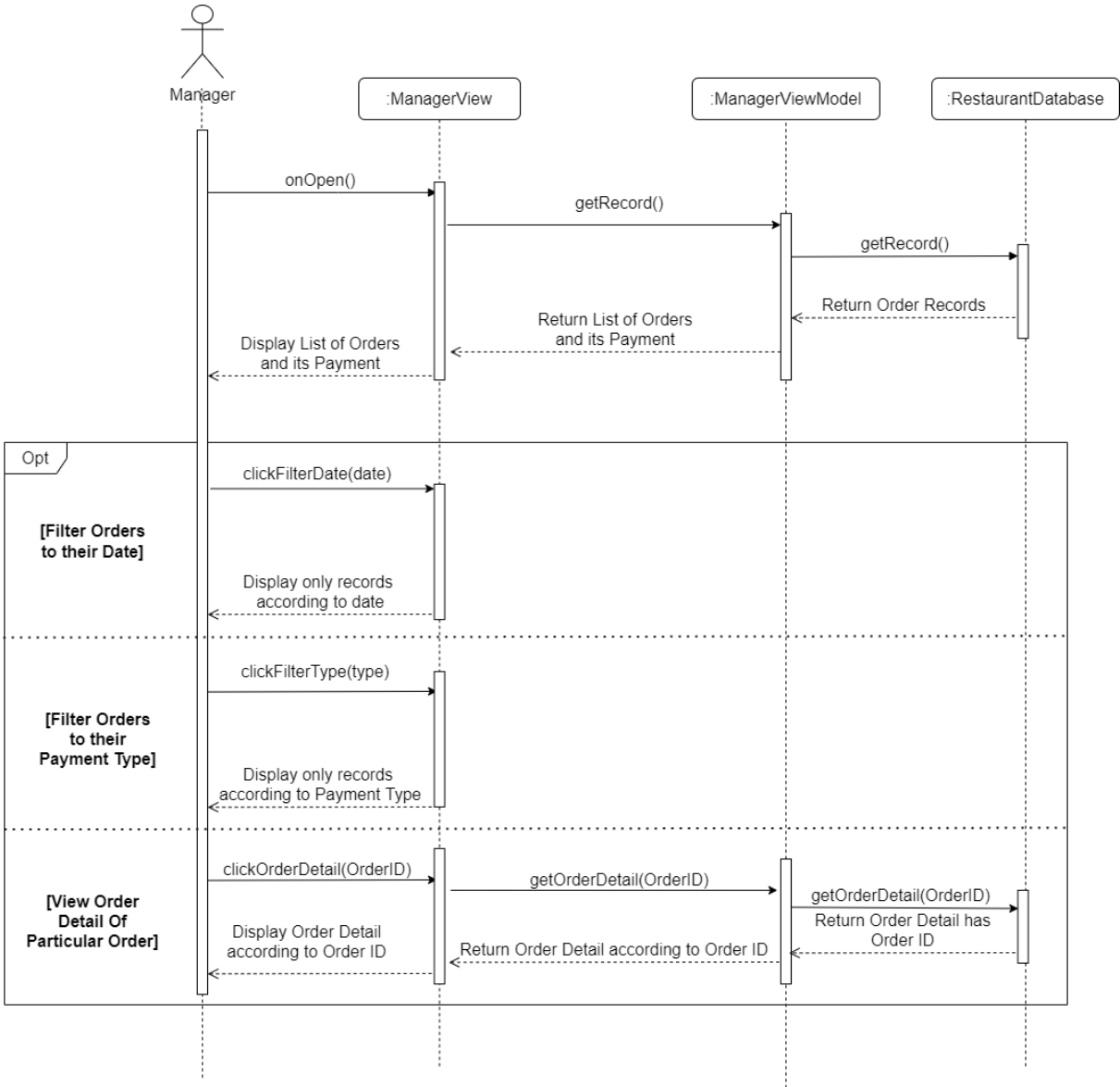


Making Payment Sequence Diagram - Clerk ([Link image](#) - Clerk)

The sequence diagram above presents the Confirm Physical Payment process:

1. Clerk opens the Clerk View and open View Pending Payment
2. Clerk clicks the refresh button to load new pending payments from the database through “getPendingPayment()”
3. Clerk views the detail the order and asks the kitchen to make the order
4. When the kitchen has finished the order, the clerk click confirms on the order, set the order’s state to ‘confirmed’ in the database

## Feature 4: View Order History



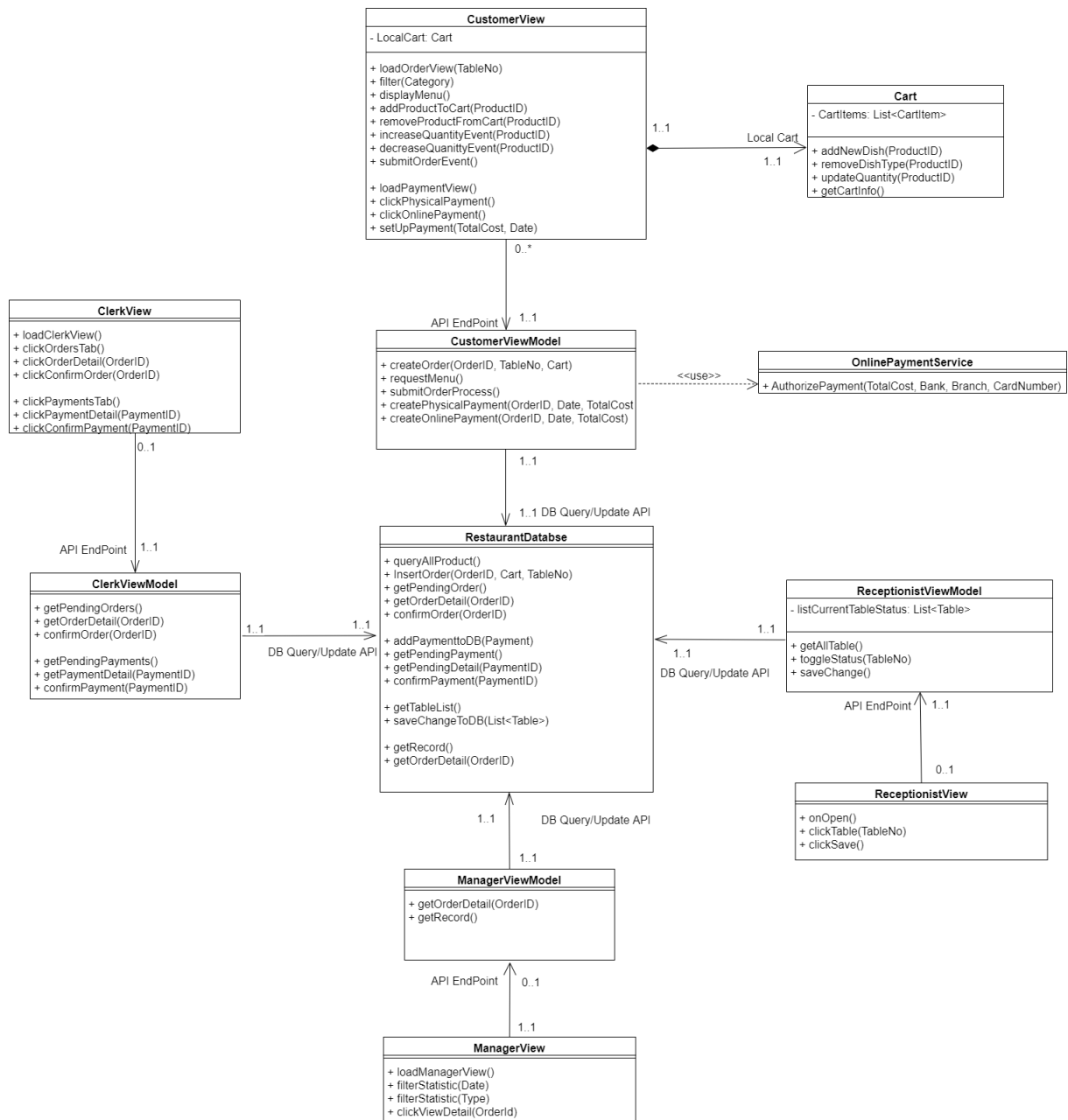
View Order History Sequence Diagram ([Link image](#))

The sequence diagram above presents the View Order History process:

1. The manager gets access to the manager view.
2. The system loads the data obtained by method ( `getRecord()` ) to the manager view.
3. The Manager View-Model handles that request by sending a request to the Database.
4. The Database returns Orders and Payment to the Manager View-Model and Manager View-Model sends to the Manager view.
5. If the manager chooses to filter Orders and its Payments by date or payment type (`filterDate(date)`, `filterType(type)`), the filter request is handled by the Manager View
6. If the manager chooses to view Order Detail, Manager View send the request to Restaurant Database through Manager View-Model and Restaurant Database returns the Order Detail back to the Manager View to display the order detail

## Class Diagram

We have drawn the activity diagram representing the flow of the work inside the system. Now we establish the class diagram in order to have a closer look at how the system is organized. Below is how the class diagram representing the system:



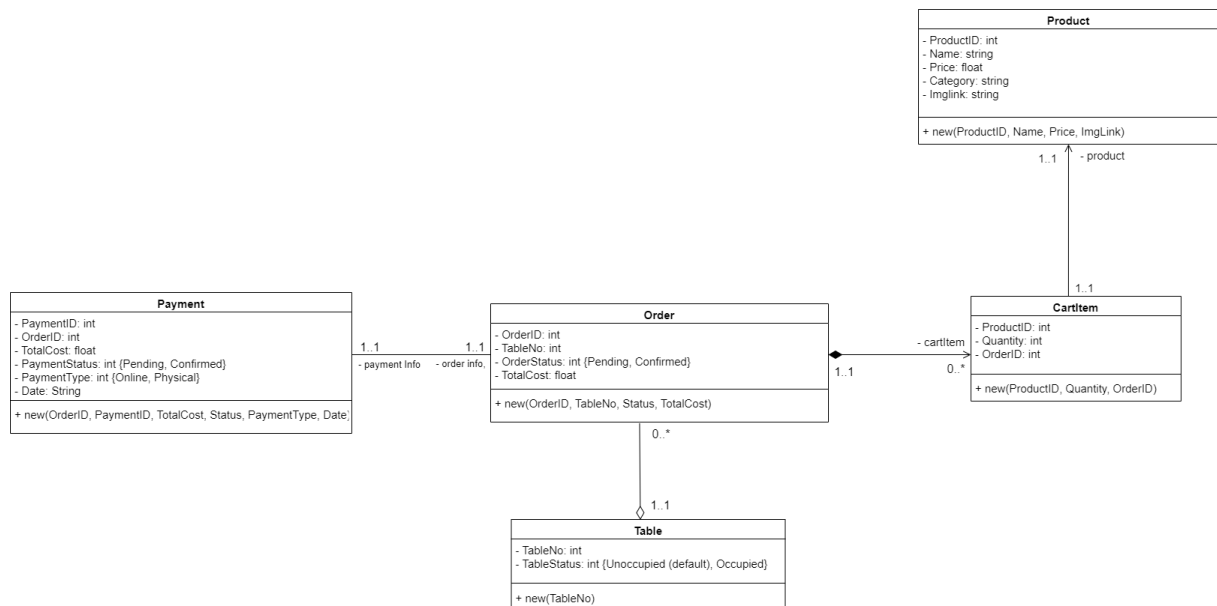
Class Diagram ([Link image](#) - Class Diagram)

The First Class Diagram presents all the active components that control the flow of the system, the association between components and the multiplicity of that relationship.

The diagram has simplified the structure of the system by combining multiple UI into 4 Class Views to interact with each type of Actors: Customer, Clerk, Manager, Receptionist.

Each View will call APIs provided by each corresponding ViewModel to access the database, and Each ViewModel will query or modify the database through APIs given by the Models.

There are addition auxiliary components such as the Cart Class owned by Customer View to handle the logic of the Orders' Cart in client side, or The Online Payment Service Class to authenticate and process Online Payment



Class Diagram - Data Class ([Link image](#) - Class Model)

The second Class Diagram presents all the Data Class for each relation stored in the Database, the multiplicity of each association between each relation, the properties for relation table.

# Architecture design

## 1. Introduction to MVVM

In designing our application, our group chooses the Model - View - ViewModel (MVVM) architecture.

Model - View - ViewModel is a web application architecture containing three main types of components Models, Views and the ViewModels.

Models represented for the storing of the application's data, defining the business logic and data logic for the application. The role of Models components in MVVM is similar to other architectures such as Model - View - Controller (MVC), or Model - View - Presenter (MVP).

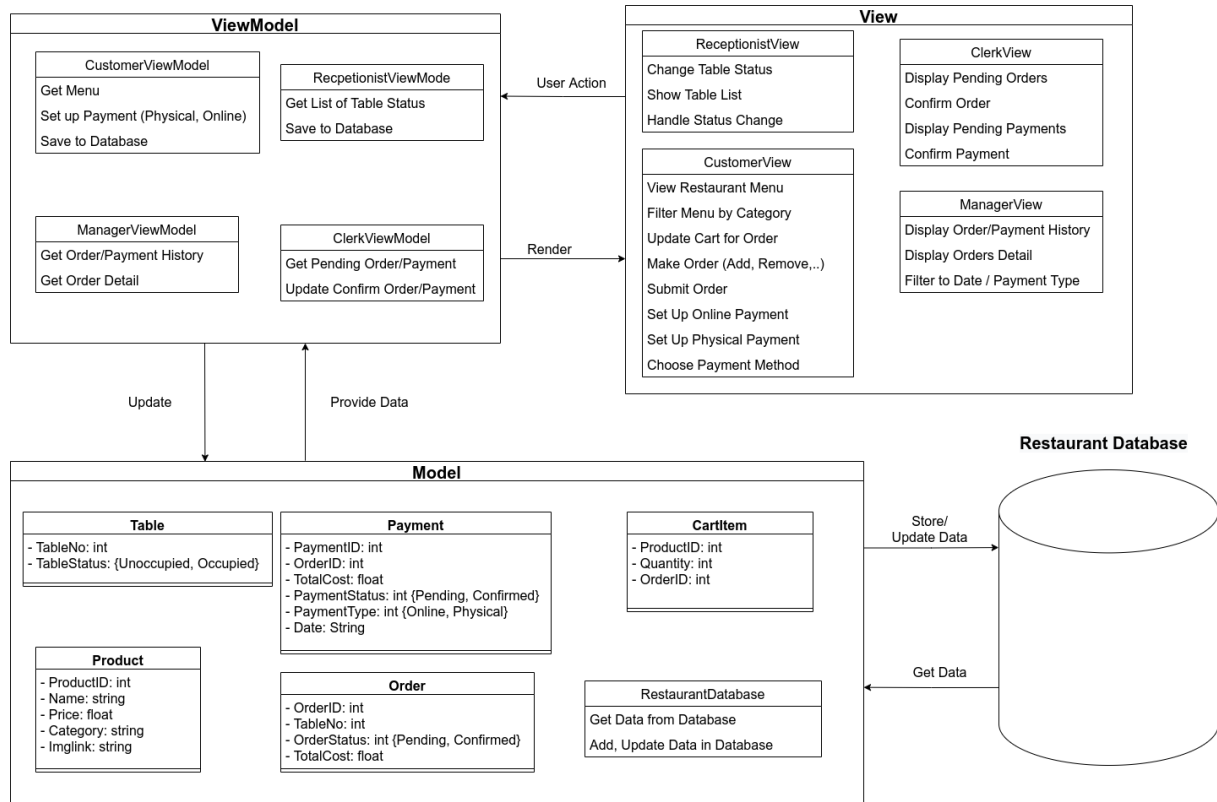
ViewModel is the middle man in MVVM architecture, receiving APIs for query and modified datas from Models. ViewModel components are responsible for updating the Models and notify the Views about the data modifications in the Models. ViewModel will also provide some APIs for the View to call whenever querying or modifying data events happens.

Views in MVVM handle multiple responsibilities from rendering and handle all the logic in the client-side. Unlike MVC or MVP, Views in MVVM has to specify the parameters before calling the ViewModel APIs to access the backend logic of the application, and has to handle the rerender process itself according to the response from the ViewModel APIs.

ViewModel being independent of the Views and the separation of Views in client side and ViewModel-Model in server side through sending HTTP requests helps our team be able to implement and debug both the front-end and back-end simultaneously. Since web applications emphasize on using resources on the client-side devices to relieve server resources and minimize the amount of data in the request between server and client, Model-View-ViewModel is the most appropriate architecture for our POS application.



## 2. Applying MVVM to our model



MVVM Diagram Design of the System ([Link image](#))

**View:** Application has 4 different views for different type of actors to the system:

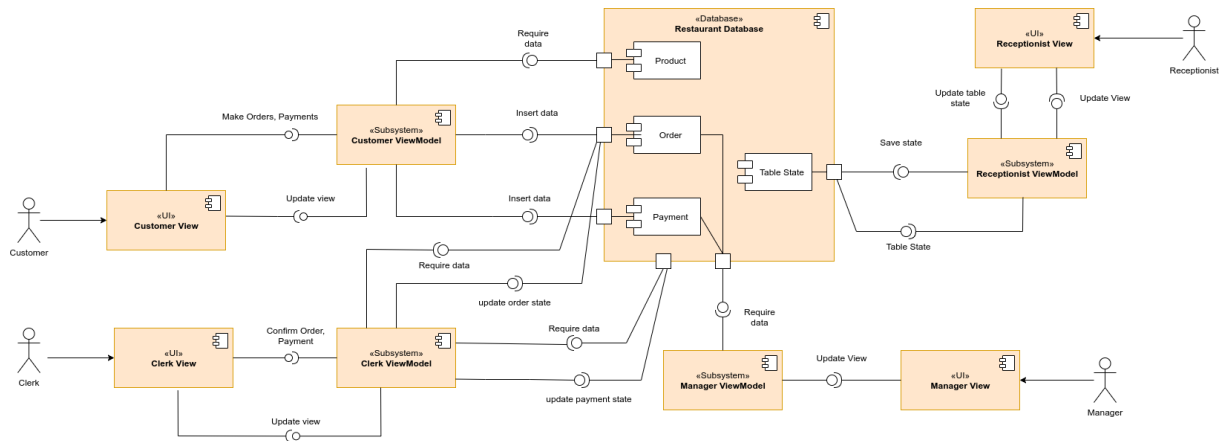
- Customer View: For customers in the restaurant want to place orders and make payments for the order
- Receptionist View: For the receptionist to use Table Management Feature
- Clerk View: For the clerk to receive and confirm pending physical payments and orders
- Manager View: For the restaurant manager to use View Order History Feature

**ViewModel:** Application is composed of 4 different ViewModel, each is responsible for each corresponding View for the same type of actors. Each ViewModel provides APIs to access the particular Models that are necessary for its corresponding Views to function. Views will utilize those APIs to manipulate the data as well as query the data.

**Model:** Application has 5 type of Model (Table, Payment, Cart, Product, Order) and a Restaurant Database class to query data from / to Restaurant Database

- Table: this will have a Table Number as key, and Table Status to indicate whether that table is available or occupied
- Payment: this will have a Payment ID as key, as well a Order ID for the Order, the TotalCost for the value of the Payment, Status for Pending and Confirm Status
- Order: this will have a Order ID as key, Table No for which table the order is made, Order Status for Pending and Confirm Status, and Date for the date the order was made
- Product: this will have ProductID as key, the Name of the Product, Price of the Product, Category for which this product belongs to and ImgLink for Image of the Product
- CartItem: this will have both ProductID and OrderID to store which Order have which Product with which quantity

### 3. Component Diagram



Component Diagram of the System ([Link image](#))

#### Description of the component diagram:

Each factor will have their own view and the views are displayed differently on screen: Customer View, Receptionist View, Clerk View, Manager View. Their views are displayed and updated and according to the data saved in the database.

Restaurant Database Component represents restaurant database data, which is data about Product, Order, Payment and Table Status.

Each View and data on it is processed through a corresponding ViewModel. Customer ViewModel processes everything involving customers, Receptionist ViewMode processes everything involving Receptionist, Clerk ViewModel processes everything involving the Clerk and Manager ViewModel processes everything involving the Manager.

# Implementation Technology

## 1. Front-end Implementation

**React** (or React.js) is a popular javascript front-end used to develop user interfaces for websites. React is an open source library, developed and maintained by Meta (facebook) React is used to build UI components, which makes web development easier. One of the features of React is rendering data not only in Server side but also in Client side.



### **Advantages:**

- Easy to use and maintain: React Components can be used with JSX syntax, extended JavaScript syntax, which allow developers to write components with HTML and JavaScript.
- Reusable components: UI is built by components, components can be used many times later, avoiding repeating similar codes.
- Large community with libraries: React is popular among developers, easy to find help from communities as well as a variety of custom libraries suitable for UI development.

## 2. Back End Implementation

**Expressjs** is a module from **NodeJs**. Express is used to build server side easier. **Expressjs** supports HTTP methods and middleware creating powerful and easy-to-use APIs



Some of main features:

- Handle HTTP requests.
- Define router for different actions on HTTP method and URL.
- Allow responding HTML pages based on arguments.

### 3. Database Management System

MySQL is an open-source relational database management system (RDBMS), developed, distributed, and supported by Oracle Corporation. MySQL is very fast, reliable, scalable, and easy to use. It is cross-platform which is ideal for both small and large applications.

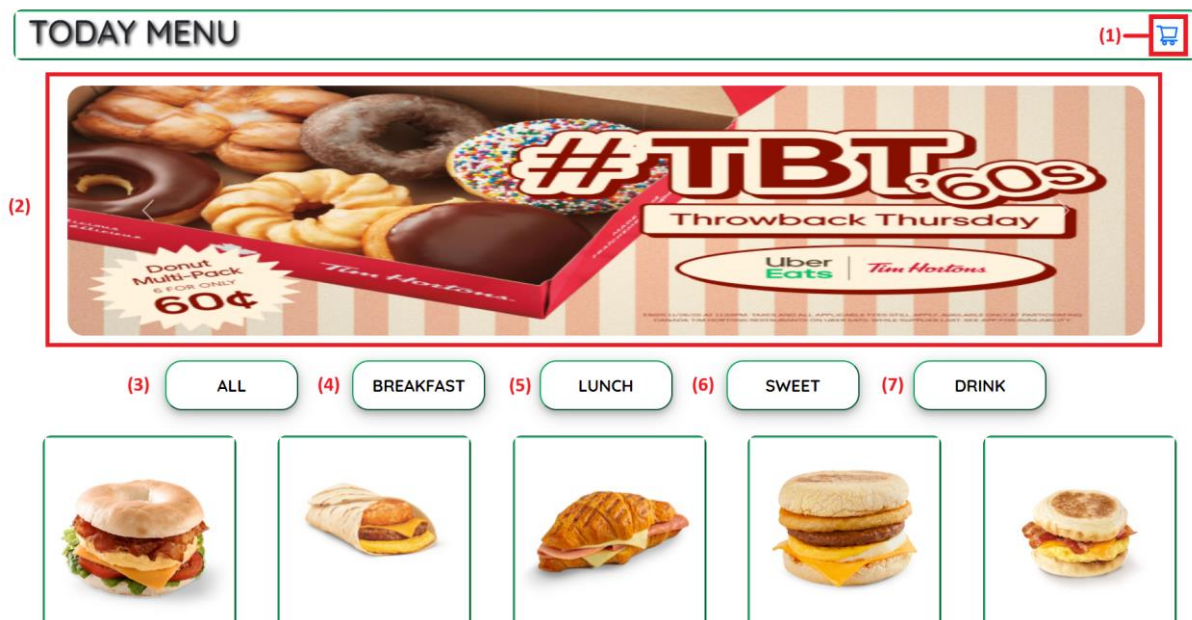
MySQL is very popular. A very large number of web developers around the world use MySQL and it is also used by huge websites like Facebook, Twitter, Airbnb, Booking.com, Uber, GitHub, YouTube, etc.



# Implementation

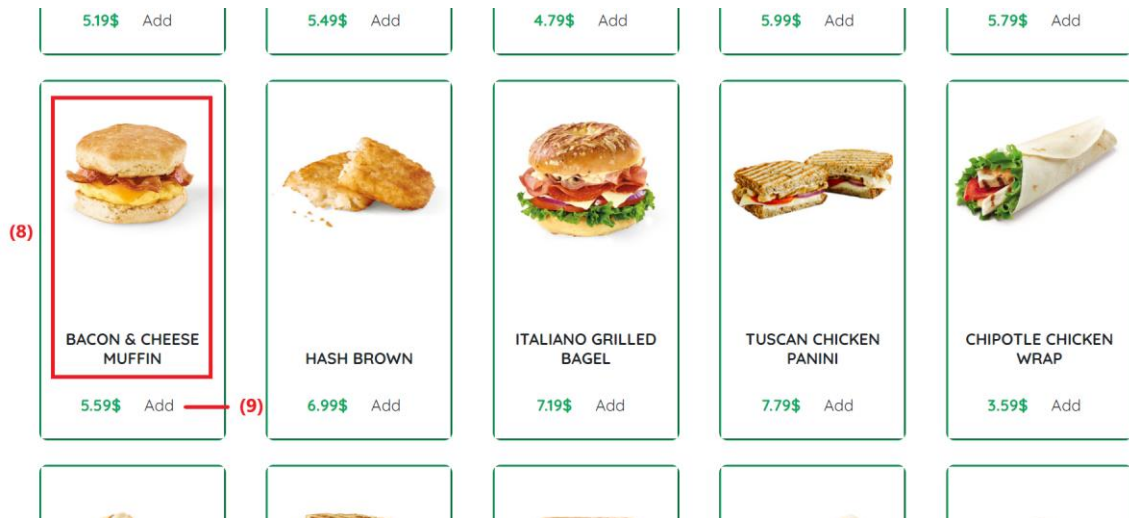
## 1. Customer View

After scanning the QR code placed on the table, the customer device is presented with the restaurant's Menu Page (UI 1.1) viewing all available food and a banner slide for special deal promotion (2). Below the banner slide is the categories filter bar with multiple for different food types: All (3), Breakfast (4), Lunch (5), Sweet (6), Drink(7), clicking on each category will filter and show only the list of products that belong to that category.



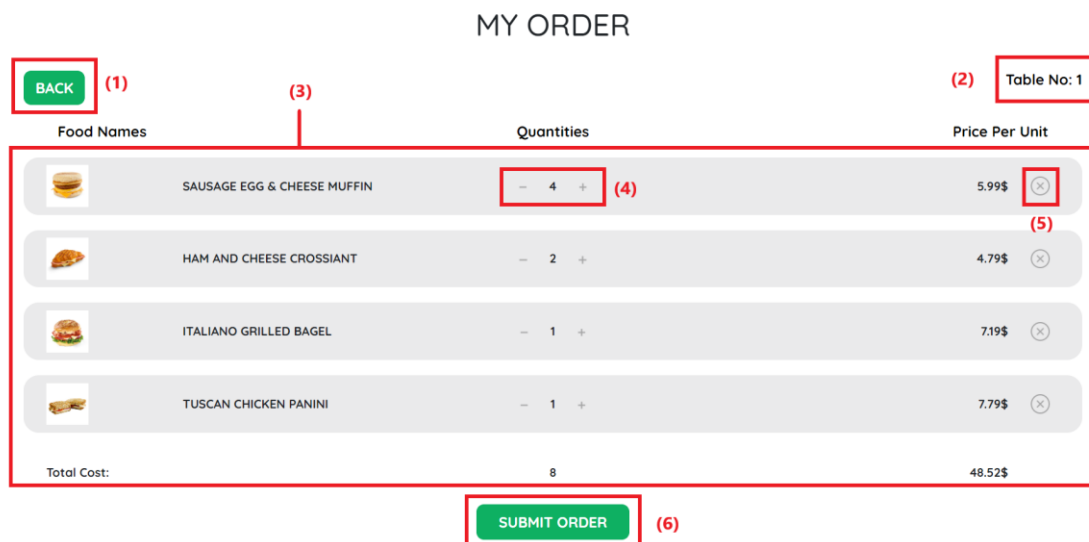
UI 1.1: Customer View start with Menu Page (1)

Below the navigation bar is the menu itself viewing multiple cards for each product (UI 1.2). Each product card contains an image illustrating the food (8), the price and a button to add the product to cart (9). Customers can browse and add the desired food to the local cart store on their device. After being satisfied with their order, customers can click on the cart button placed on the top-right side of the page (1) to route to the View Order page (UI 1.3)



UI 1.2: Customer View start with Menu Page (2)

The view order page shows the table number (2) and customer current order state (3). Customers can either modify the quantity of the same product that they want (4) or remove product from the cart (5). Customers can click on the back button (1) to continue browsing the menu. Finally, customers will click on the Submit Order button (6) to finalize the order and routing to payment page (UI 1.4) and start the payment process.



UI 1.3: Customer View of Their Order

In the Payment page (UI 1.4), there are some fields that show information about customer's detail (1): Date which is the date of the customer's order, Time which is the time



that a customer makes his/her order, Table No to show the table number that the customer is using and the total cost which shows the amount of money that the customer must pay.

### PAYMENT

**(1)** Date: 11/26/2021 Time: 11:01:46 PM Table No: 1 Total Cost: 48.52\$

Physical Payment **(2)** **Pay Later**

Online Payment **(3)** **Pay Now**

#### *UI 1.4: Customer View with Making Payment Feature*

**PAYMENT**

Date: 11/26/2021 Time: 11:01:46 PM Table No: 1 Total Cost: 48.52\$

Physical Payment **Pay Later**

Online Payment **Pay Now**

**(1)** Bank  
Bank name  
Branch  
Card Number  
Card Number  
**(2)** Valid **Invalid (3)**

#### *UI 1.5: Online Payment simulation with Valid and Invalid Option*

The customer can choose two options: Physical method or Online payment. If the customer chooses Physical payment, he/she can press the confirm button (2) in the physical payment box, and the customer will send their payment request to the Clerk. If the customer

chooses Online payment, they press the confirm button (3) in the online payment box, and the customer will receive a window that shows in the screen (UI 1.5).

After filling 3 fields Bank, Branch and Card number (1) to connect to his/her bank account, the Valid and Invalid Option is a simulation of the Online Payment Service, when the customer clicks on Valid, System registers the Payment as Valid and move to the Bill, while clicking on Invalid, System registers the Payment as Invalid and return customers back to the Payment Page, where Customer should choose a different payment method.

**BILL**

(2) 

Date: 11/26/2021  
Table No: 1

Time: 11:22:43 PM

Payment Method: Physical

(1)

| Food Names                  | Quantities | Price          |
|-----------------------------|------------|----------------|
| SAUSAGE EGG & CHEESE MUFFIN | 4          | 23.96\$        |
| HAM AND CHEESE CROSSIANT    | 2          | 9.58\$         |
| ITALIANO GRILLED BAGEL      | 1          | 7.19\$         |
| TUSCAN CHICKEN PANINI       | 1          | 7.79\$         |
| <b>Total</b>                |            | <b>48.52\$</b> |

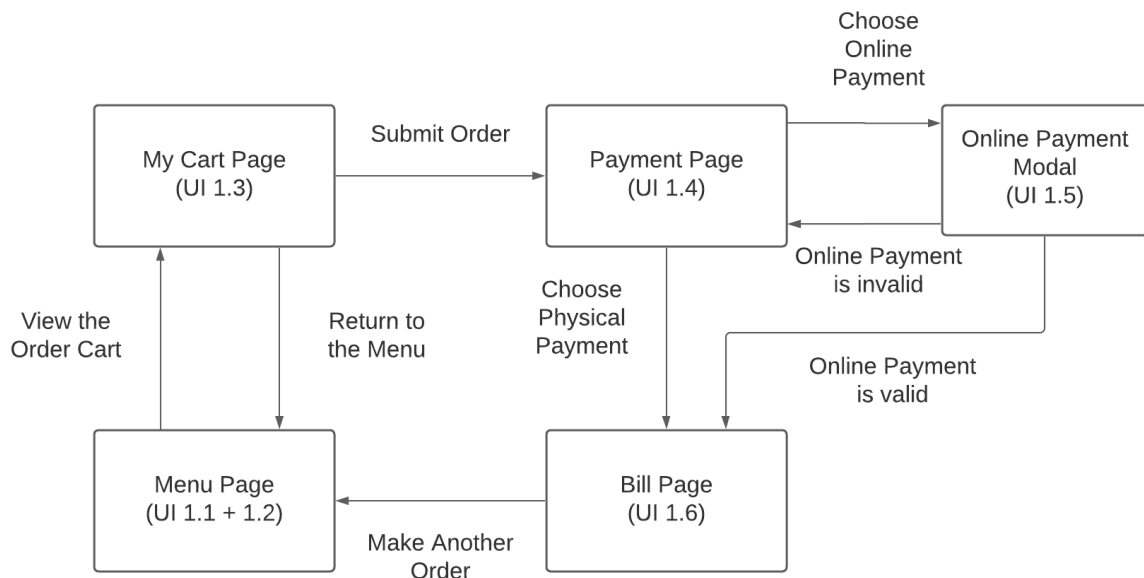
BACK

 (3)

*UI 1.6: System displays the bill for the Payment/ Order*

In Bill page (UI 1.6), there are four information (1) provided below the Bill header that are similar to Payment page, Payment Method is the method that customer chooses in Payment page. Below the information field is the summary table (2). The table provides several information about the order, these are Product Name, Quantities of each type Product, Price of each type Product and Total Payment. At the bottom of the page is Back button (3), which will return to the Menu page and reset all information in the cart when a customer presses it, used for making new orders.

## Screen flow of Customer View (From View Menu, Make Order to Make Payment)



Since this feature is an important feature, a screen flow diagram is needed.

- Customers start as Menu Page (UI 1.1), can add Product to their Order and moves to My Cart Page by the Cart Button
- In My Cart Page, customer can go back to Menu Page through the Back Button, can change Quantity of current in-cart Product or clicks Submit Order for Clerk to Confirm, after their Order has been confirmed, customers go to Payment Page
- In Payment Page, customers between Pay Later with Physical Payment and Pay Now with Online Payment, if they choose Online Payment, a pop up will appear, require input information, click the Valid Option simulated the Online Payment Service validates the Payment and customer moves to Bill Page, otherwise, they remain in Payment Page to choose a different payment option
- In the Bill Page, customers can go back to the Menu Page to make a new Order

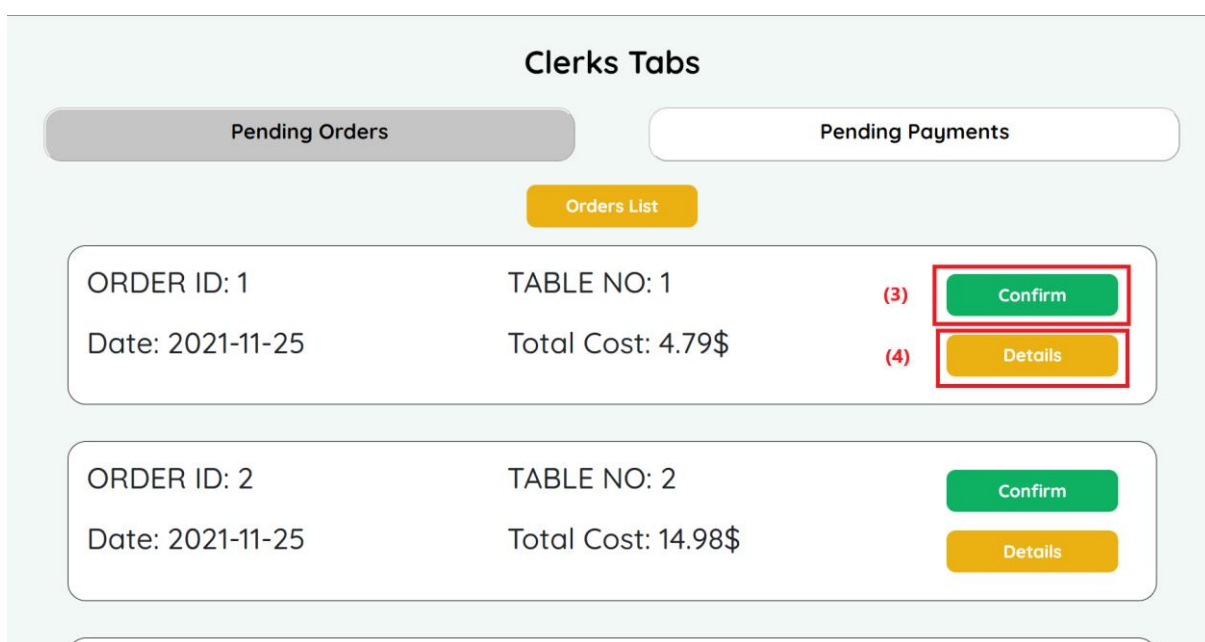
## 2. Clerk View

When starting up the view on the browser, the clerk is presented only with the navigation bar (UI 2.1) with two options, click on Pending Orders navigation link (1) views list of pending orders waiting to be confirmed (UI 2.2), the same with Pending Payments navigation link (2) views list of pending payments (UI 2.3).



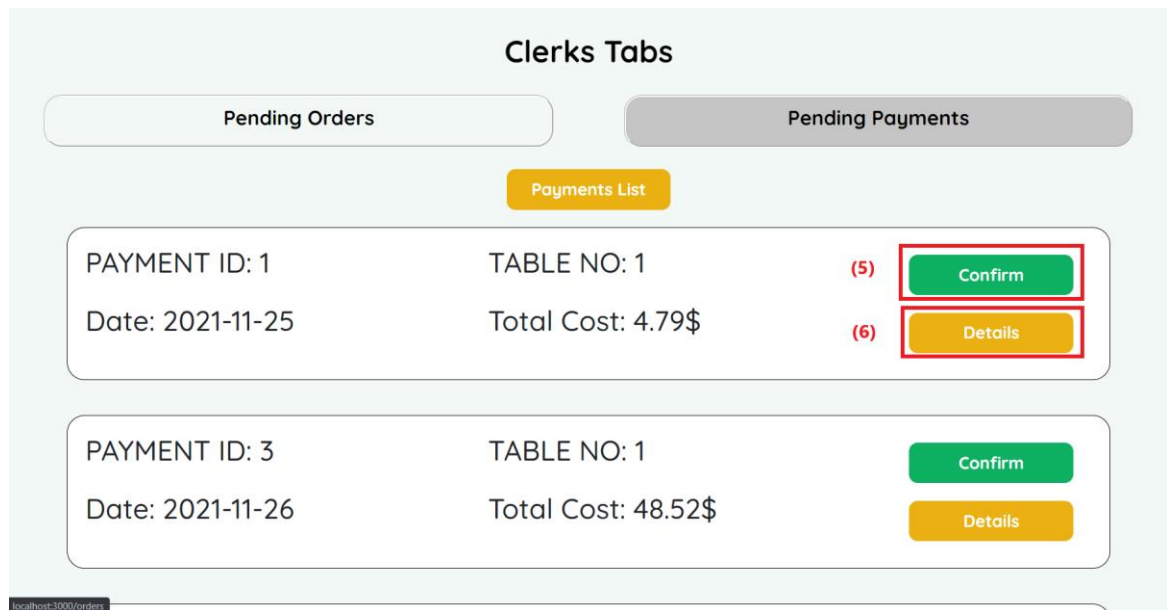
*UI 2.1: Clerk View Load Page*

On the Orders List page (UI 2.2), there are rows of items presented for each pending order, clicking on the confirm button (3) will send the accept update to backend and the item will disappear from the screen and clicking on detail button (4) will route to the Order Detail Page (UI 2.4).



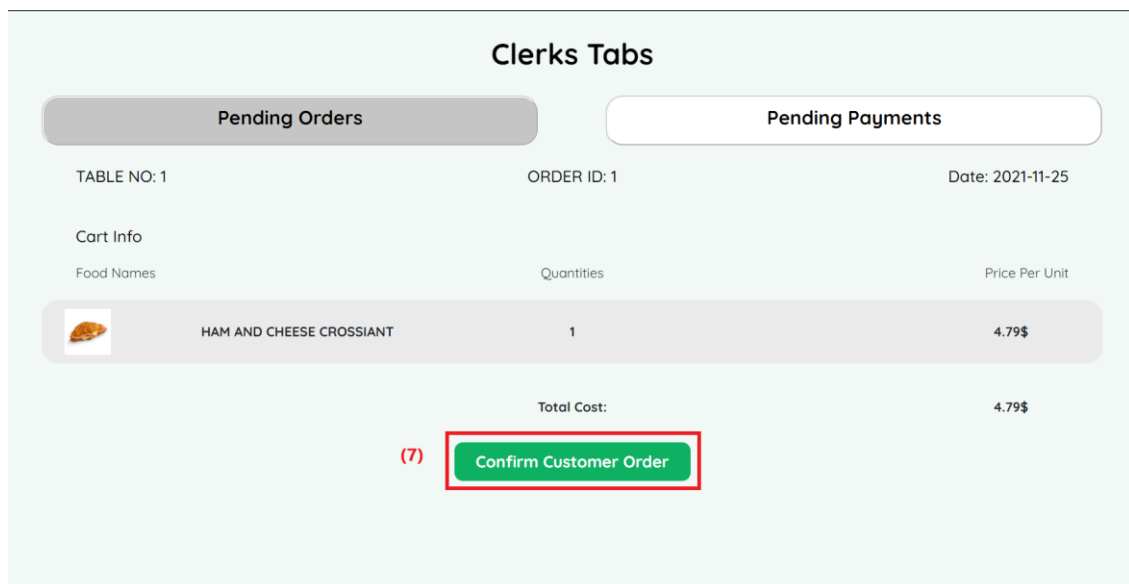
*UI 2.2: Clerk View Orders List*

The same with Payments List Page (UI 2.3), clicking confirm button (5) to accept the payment and clicking detail button (6) to view Payment Detail (UI 2.5).



UI 2.3: Clerk View Payments List

On the Order Detail page (UI 2.4) the details about the list of products ordered by the customer are presented with the prices and the total cost of the order, clicking the ‘Confirm Customer Order’ (7) will send an update request to backend. Clerk now will be return back to the Orders List page (UI 2.2) and the confirm order is removed from the list appeared UI



UI 2.4: Clerk View Order Detail

On the Payment Detail page (UI 2.5) the bill of the direct payment is presented. After the customer paid the bill, the clerk will click the ‘Confirm Customer Payment’ to confirm

payment and will be routed back to the Payments List page (UI 2.3) and the confirmed payment will disappear.

### Clerks Tabs

Pending Orders

Pending Payments

TABLE ID: 1PAYMENT ID: 1Date Time: 2021-11-25

Payment Info

| Food Names               | Quantities | Price  |
|--------------------------|------------|--------|
| HAM AND CHEESE CROSSIANT | 1          | 4.79\$ |
| Total                    |            | 4.79\$ |

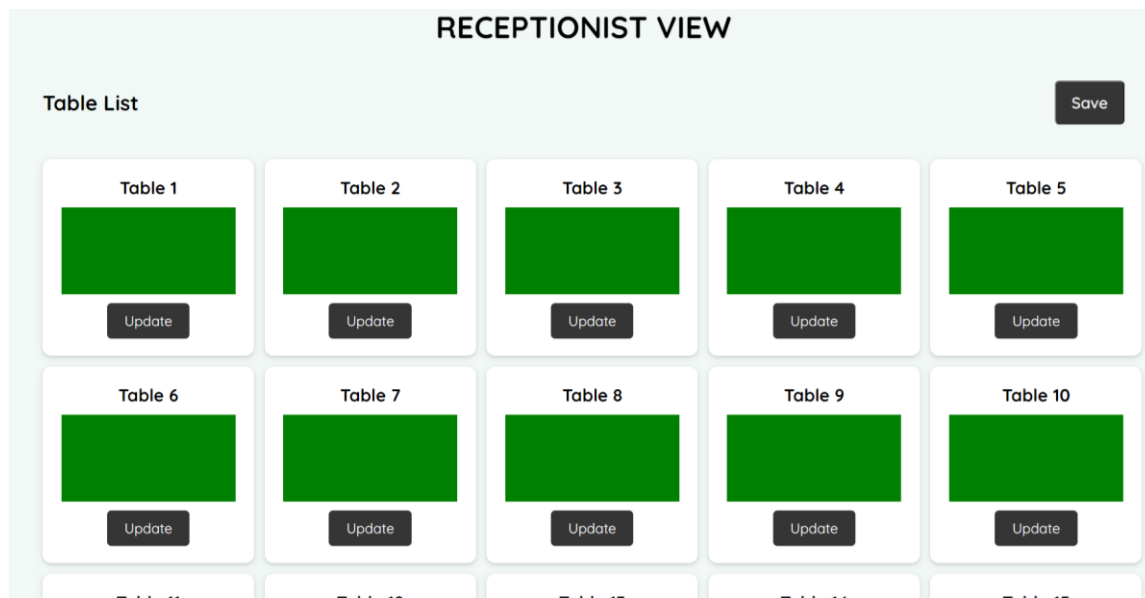
(8) 

Confirm Customer Payment

*UI 2.5: Clerk View Payment Detail*

### 3. Receptionist View

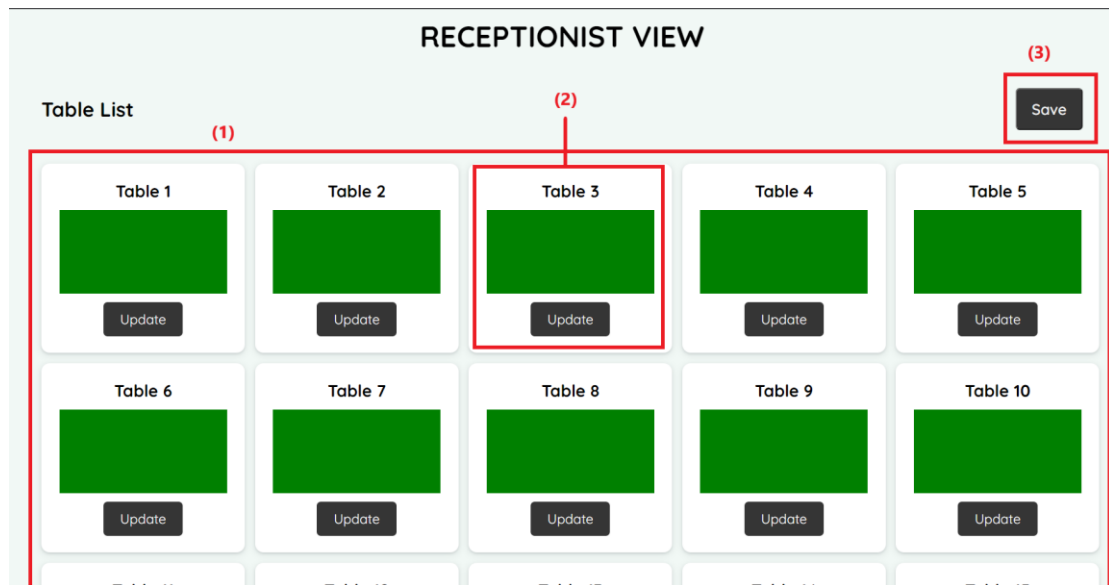
When start up the page, the System will execute the procedure to get 25 table's status from the Restaurant Database and display them to the Receptionist



*UI 3.1: Receptionist view with Feature Table Management*

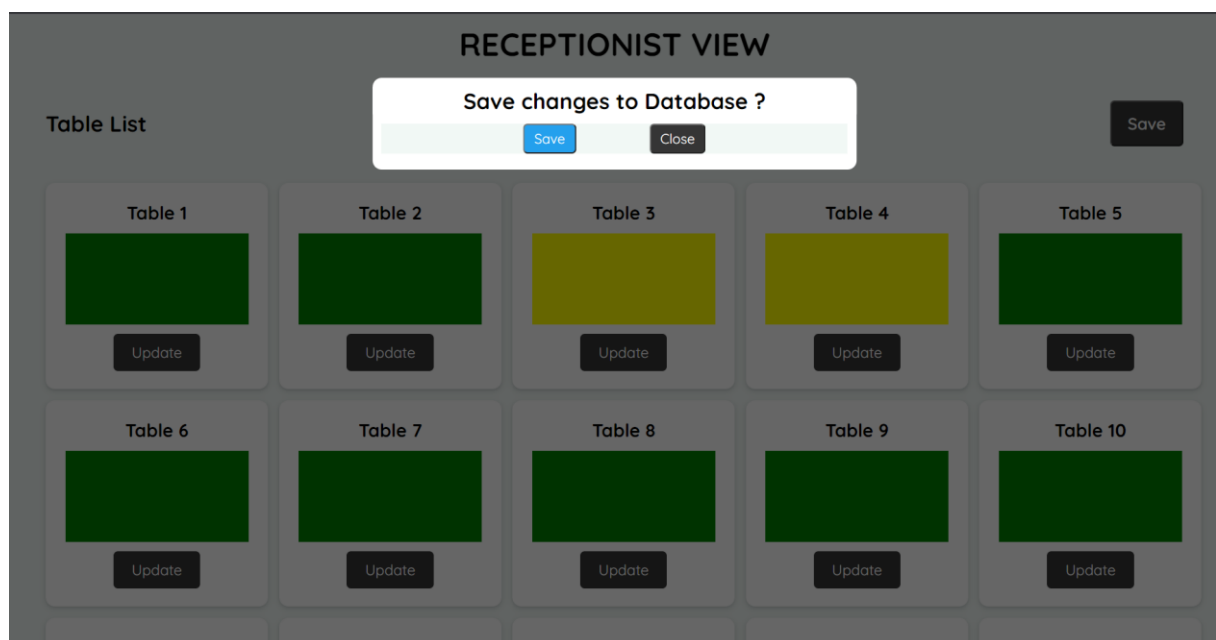
Each table is displayed as each card (Object 1) in a grid, inside each card contains table number, table status represented by a colored box (green as available and yellow as occupied) and a "Update" button to change its status.

When receptionist presses "Update" button in a selected table, the colored box toggles its (from yellow to green and vice versa) and the system saves that change to its local storage



*UI 3.2: Table Management annotation*

When receptionist has finished their update, the receptionist clicks the Save Button (2), which pops up a modal asking to confirm the choice and if receptionist chooses to save, the changes will be updated to the Restaurant Database



*UI 3.3: Save Changes to Database popup*



## 4. Manager View

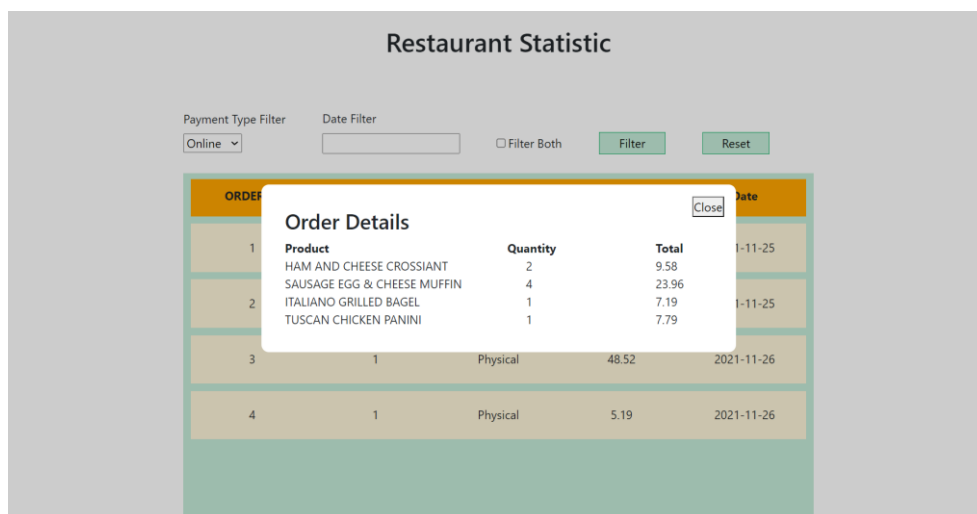
When starting up the page, the System will execute the procedure to get all orders and payments recorded from the Restaurant Database and display them to the Manager in pages. Each Record is displayed as each row in the Data Table (6), inside row contains Order ID, Table of the Order, Payment Type, Total Bill and Date



| Payment Type Filter | Date Filter          | <input type="checkbox"/> Filter Both | Filter      | Reset      |
|---------------------|----------------------|--------------------------------------|-------------|------------|
| Online ▾            | <input type="text"/> |                                      |             |            |
| ORDER ID            | Table                | Payment                              | Total (USD) | Date       |
| 1                   | 1                    | Physical                             | 4.79        | 2021-11-25 |
| 2                   | 2                    | Online                               | 14.98       | 2021-11-25 |
| 3                   | 1                    | Physical                             | 48.52       | 2021-11-26 |
| 4                   | 1                    | Physical                             | 5.19        | 2021-11-26 |

UI 4.1: Manager view screen

When Manager clicks a row in the data table, system displays a modal showing the order detail about that order/ payment



| Payment Type Filter | Date Filter          | <input type="checkbox"/> Filter Both | Filter      | Reset      |
|---------------------|----------------------|--------------------------------------|-------------|------------|
| Online ▾            | <input type="text"/> |                                      |             |            |
| ORDER ID            | Table                | Payment                              | Total (USD) | Date       |
| 1                   | 1                    | Physical                             | 48.52       | 2021-11-26 |
| 4                   | 1                    | Physical                             | 5.19        | 2021-11-26 |

**Order Details**

| Product                     | Quantity | Total |
|-----------------------------|----------|-------|
| HAM AND CHEESE CROSSIANT    | 2        | 9.58  |
| SAUSAGE EGG & CHEESE MUFFIN | 4        | 23.96 |
| ITALIANO GRILLED BAGEL      | 1        | 7.19  |
| TUSCAN CHICKEN PANINI       | 1        | 7.79  |

Close

UI 4.2: Pop-up screen

Manager can filter records by date when choosing a day in the Date Picker (2) and presses the filter button (4), payment method by using Payment Type Filter (1) and presses the filter button (4) or both by choosing input for (1) and (2), check the filter both box (3) and then presses the filter button (4) .

After filtering data, the manager can press the Reset button (5) to view all records.

**Restaurant Statistic**

(1)  
Payment Type Filter  
Online ▾

(2)  
Date Filter

(3)  
☐ Filter Both

(4)  
Filter

(5)  
Reset

| ORDER ID | Table | Payment  | Total (USD) | Date       |
|----------|-------|----------|-------------|------------|
| 1        | 1     | Physical | 4.79        | 2021-11-25 |
| 2        | 2     | Online   | 14.98       | 2021-11-25 |
| 3        | 1     | Physical | 48.52       | 2021-11-26 |
| 4        | 1     | Physical | 5.19        | 2021-11-26 |

(6)

UI 4.3: Manager view annotation

## Demo Video

Video Link: [https://drive.google.com/file/d/1dIkPiWjmnVLo\\_3Jg8\\_lneALi\\_WUlxF7-/view?usp=sharing](https://drive.google.com/file/d/1dIkPiWjmnVLo_3Jg8_lneALi_WUlxF7-/view?usp=sharing)

## GitHub repository

For further detail, please visit the work at our GitHub repository

<https://github.com/remsoakawaii1/CNPM>

The final product is in branch /main.

The development branch is branch /customer

The document branch is branch /document

The document branch is branch /document