VIETNAM GENERAL CONFEDERATION OF LABOR

TON DUC THANG UNIVERSITY

FALCUTY OF INFORMATION TECHNOLOGY

Midterm Project

Subject: Service-Oriented Architecture

Topic: Customer Order Management Module

Students perform:
Tran Khai Hoang (520H0635)
Nguyen Nhat Thong (52000808)
Nguyen Gia Khiem (520H0464)

Advising teacher:
Mr. Duong Huu Phuc

April 11, 2023

DECLARATION

We guarantee that this research is our own, conducted under the supervision and guidance of Mr. Duong Huu Phuc. The result of our research is legitimate and has not been published in any forms prior to this. All materials used within this researched are collected by ourselves, by various sources and are appropriately listed in the references section. In addition, within this research, we also used the results of several other authors and organizations. They have all been aptly referenced. In any case of plagiarism, we stand by our actions and are to be responsible for it. Ton Duc Thang University therefore are not responsible for any copyright infringements conducted within our research.

ACKNOWLEDGEMENT

The first sincere thanks I want to give to Mr. Duong Huu Phuc, who enthusiastically taught and worked tirelessly to give me enough tools and skills to complete this report. He played an important role in improving my mathematical logic and knowledge. The second thanks we would like to give to the teachers of the Department of Information Technology of Ton Duc Thang University for giving me the opportunity to do this report, because it is not only a report but also a very important experience for me in the next year.
Our report can have some errors, we are very open to receiving feedback from teachers so that I can improve my report writing skills.
Finally, we wish you good health and success in your noble career.

Contents

1 System Overview

1.1 Introduction

This report will present an overview of the general architecture for some functions of the **Customer Ordering Module**, including the schema, specifications of the functions and the database of the module by using UseCase and ERD diagrams. In addition, this report will also introduce the technology that the team chooses to realize the module and the module after our team deploys it.

1.2 System Specification

For convenience and increased interaction between restaurant staff and diners, a restaurant wants to develop a **diners order management module**. The new module allows **diners, waiters, and kitchen staff** to interact with each other in real time. Diners can proactively order and control the dishes ordered, as well as the total amount of the bill, helping to reduce the work and omissions of the waiter. In addition, the module allows kitchen staff to control the dishes ordered from diners as well as their notes.

1.3 Topic Scope

This project is a **Customer Ordering Module** that only ensures some basic functions just enough to increase the interaction between restaurant staff and diners. The project was implemented and developed under the guidance of university faculty.

1.3.1 Objects and Functions Boundaries

- Objects Boundary: Manager, Waiter, Cashier, Chef and diners
- Functions Boundary:

- Diners: View Menu, Place Order, Pay Bill by Cash, View State of Dishes, Choose Dish, Add Note, Check Orders in Bill
- Waiter: Create Bill, Open Table
- Cashier: Print Bill, check and Confirm Bill, View Bill History
- Chef: Modify State of Dishes Ordered, Modify State of Dishes in Menu
- Manager: Manage Staff, Create Bill, Open Table, Print Bill, check and Confirm Bill, View Bill History

1.3.2 Technologies Boundaries

- Main technology used:

- Front-end: HTML, CSS, JavaScript, Bootstrap
- Back-end: ExpressJS
- Database: MongoDB

1.4 Practical Implications

- Improve customer general experiences
- Optimize food ordering process.
- Enhance communication between staffs and diners.

1.5 Thesis Layout

- Thesis consists of 4 parts:

- Part 1 System Overview: Introduction to the goal, scope and practical significance of the topic.
- Part 2 System Analysis and Design: Presents functional requirements, non-functional requirements, ERD diagram, physical database model, Use Case diagram and Use Case specification.
- Part 3 System Implementation: List the functions (APIs) of the module, in which, clarify Input/Output of each function.
- Part 4 References

2 System Analysis and Design

2.1 Functional Requirement

- Diners: View Menu, Place Order, Pay Bill by Cash, View State of Dishes, Choose Dish, Add Note, Check Orders in Bill
- Waiter: Create Bill, Open Table
- Cashier: Print Bill, check and Confirm Bill, View Bill History
- Chef: Modify State of Dishes Ordered, Modify State of Dishes in Menu
- Manager: Manage Staff, Create Bill, Open Table, Print Bill, check and Confirm Bill, View Bill History

2.2 Non-functional Requirement

- Operational: The system can work on any browser and on most devices with Internet
- Performance: System has a fast response time
- Security:
  - No employee can log in to another account without a password
  - the functions of the position are separate in the system and no position can use the functions of another position except Manager.
- Cultural and Political: None

2.3 Use Case Diagram

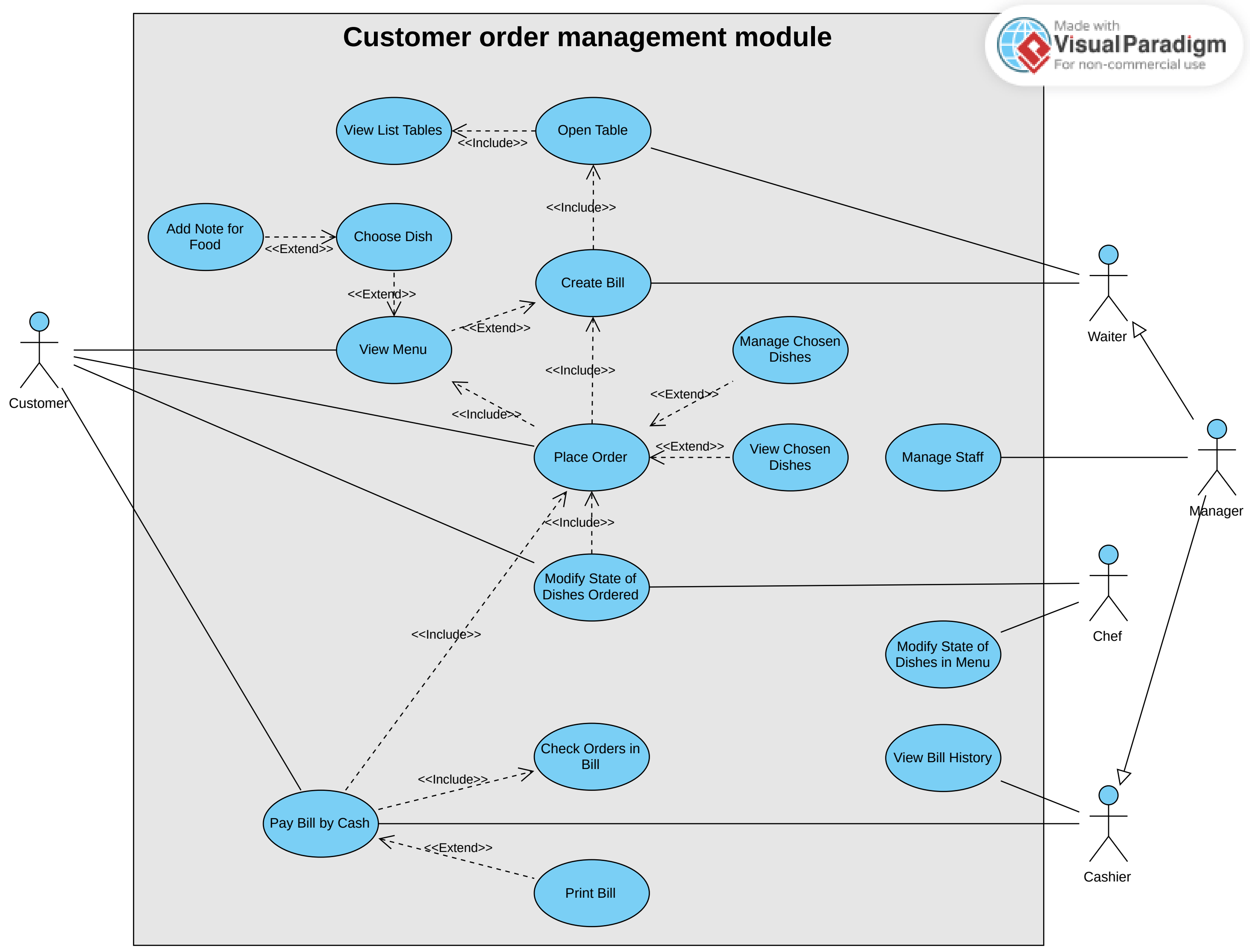


Figure 1: Use Case Diagram of Customer Order Management System

2.4 Use Case Specification

2.4.1 Use Case Login

Use case ID	UC001
Use case Name	Login
Primary Actor	Waiter, Cashier, Chef, Manager
Stakeholder and Interests	None
Summary	The function helps staffs log into the system before using other functions
Pre-Condition	Employee must have an account in the database
Trigger	Staffs click the Login button on the system interface
Post-condition	Staff successfully logged into the system
Relationships	Include: - All use cases in the diagram → Login
Main scenario	1. Staffs want to access the system to use the functions for their work through the link. 2. The browser sends a request to the system. The system displays the login interface for staffs. 3. Staff fill in all the information that the system requires and click the Login button. 4. The system checks that the information is sufficiently entered. 5. The system checks that the information entered by staffs is correct. 6. The system updates this Staff's login time to the database and changes the login interface to the interface suitable for the login Staff position.
Alternatives	4.a. If the system checks the information entered is missing. The system will send a notification of additional missing information 5.a. If the system checks the information entered is incorrect. The system will send a notification to the user to check the entered information.

Table 1: Login use case specification

2.4.2 Use Case Logout

Use case ID	UC002
Use case Name	Logout
Primary Actor	Waiter, Cashier, Chef, Manager
Stakeholder and Interests	None
Summary	The function helps staffs log out of the system
Pre-Condition	Employee successfully logged in system
Trigger	Staffs click the Logout button on the system interface
Post-condition	Staff successfully logged out of the system
Relationships	None
Main scenario	1. After the session ends, the employee wants to log out of the system. Employee presses the Logout button. 2. The system records and logs out the employee's account. The system updates the employee's logout time on the system
Alternatives	None

Table 2: Logout use case specification

2.4.3 Use Case Open Table

Use case ID	UC003
Use case Name	Open Table
Primary Actor	Waiter
Stakeholder and Interests	Manager
Summary	Waiters mark the table position that diners choose on the system
Pre-Condition	- The waiter must log in successfully. - On the system, there must be a table ID selected by the guest. - The table selected by the guest must be in an Empty state.
Trigger	The waiter selects the table ID from the list provided by the system and clicks the Open Table button.
Post-condition	- The table selected by the guest changes status to Serving on the system
Relationships	Include: Open Table → View List Tables Open Table → Login
Main scenario	1. After the waiter successfully logged into the system. The system switches to an interface containing a list of tables with an empty status (List provided by the system). 2. The waiter clicks to select the ID of the table on the system that matches the table position chosen by the diner or can nominate the table position for guests from the list provided by the system. 3. The waiter clicks the Open Table button. 4. The system records and sends notifications about the information of empty invoices.
Alternatives	1.a.1. If all the tables in the restaurant are in service. The system will return an empty list and the message "All tables are in use". 1.a.2. The waiter asked that customers could wait their turn.

Table 3: Open Table use case specification

2.4.4 Use Case View List Tables

Use case ID	UC004
Use case Name	View List Table
Primary Actor	Waiter
Stakeholder and Interests	Manager
Summary	Waiter checks the list of available tables to open for customers.
Pre-Condition	Waiter logged in successfully.
Trigger	After Waiter successfully logged in system.
Post-condition	Waiter can see a list of available tables and their information.
Relationships	Include: Open Table → View List Tables View List Tables → Login
Main scenario	1. After the service successfully logs on to the system. The system recognizes and switches to the interface containing the list of empty cork. 2. The Waiter clicks a table in the list. 3. The system displays a pop-up window containing the information of that table .
Alternatives	1.a. If there are currently no empty tables, the system will display a blank interface and send 1 notification of all desks in use.

Table 4: View List Table use case specification

2.4.5 Use Case Create Bill

Use case ID	UC005
Use case Name	Create Bill
Primary Actor	Waiter
Stakeholder and Interests	Manager
Summary	Waiter creates empty invoice for customer (Empty invoice is a 0 VND invoice because the customer has not ordered)
Pre-Condition	- The waiter must successfully logged into the system. - The waiter has opened the table (table chosen by the guest).
Trigger	Waiter presses Create Order button
Post-condition	- The blank invoice is generated by the system containing the basic information of the invoice and the ID of the selected guest's desk location.
Relationships	Include: Create Bill → Open Table Create Bill → Login
Main scenario	1. After the waiter has logged in and successfully opened the table on the system. The system will display the information of the empty invoice of the diner containing the ID of the table selected by the guest and other basic information. 2. The waiter gives the blank bill to the guest to check that the information is correct. 3. Waiter presses Create Order button. 4. The system records and adds a new invoice to the system with the information shown to the guest and the ID of the Servant who created this invoice. 5. The system updates the status of the table location selected by the guest. 6. The system switches to a system interface containing the customer's menu and real-time bill. 7. The system activates the function of adding items to orders on the customer's system interface.
Alternatives	2.a.1. Diners check the information if they find errors or want to change the information. Specifically, convert the table. 2.a.2. The waiter pressed the Cancel button. 2.a.3. The system returns to the table selection system interface. 2.a.4. Diners choose a new table position again. The service staff records the information and re-selects the table on the system. 2.a.5. The waiter clicks the Open Table button. 2.a.6. The system will display the system interface containing the information of the desk location and the basic information of the invoice changed by the customer. 2.a.7. The waiter gives the blank bill to the guest to check the information.

Table 5: Create Bill use case specification

2.4.6 Use Case View Menu

Use case ID	UC006
Use case Name	View Menu
Primary Actor	Customer
Stakeholder and Interests	None
Summary	Customers view the menu of the restaurant's dishes served that day on the system interface.
Pre-Condition	- Waiter successfully logs in. - The table has been successfully opened. - The diner's empty invoice has been successfully generated.
Trigger	Right at the original interface of diners
Post-condition	- List of dishes being served that day (Dishes with Available status).
Relationships	Include: View Menu → Login Extend: - View Menu → Create Bill - Choose Dish → View Menu - Add Note for Food → Choose Dish
Main scenario	1. After the waiter logs in, open the table and initialize the empty bill successfully. The system allows customers to add items to their orders. 2. Customers click to select 1 dish from the Menu. 3. The system records and sends a windows popup containing information of the dish. 4. Customers enter the number of dishes they want to order, add notes to the dishes (if desired) and press the Add button on the windows popup. 5. The system records and puts the data of ordered dishes into the waiting list with the status of Pending.
Alternatives	1.a. If the waiter has not opened the table and initialized an empty invoice for diners or initialization failed. Diners can only view menu and dish information and cannot add items to orders and proceed to order. 3.b. If the customer does not want to order the food being viewed, the customer can press the close button or press outside the windows popup to exit the windows popup and continue to select the dish on the menu.

Table 6: View Menu use case specification

2.4.7 Use Case Place Order

Use case ID	UC007
Use case Name	Place Order
Primary Actor	Customer
Stakeholder and Interests	None
Summary	Customers order selected dishes on the system
Pre-Condition	- Waiter successfully logged in. - The table has been successfully opened. - The diner's empty invoice has been successfully generated. - The customer added the selected dishes to the invoice.
Trigger	Customers click the Place Order button
Post-condition	- The dish selected by the customer is added to the empty invoice that the customer created earlier in the pending status. - Food information is transferred by the system down to the kitchen for inspection and reparation.
Relationships	Include: Place Order → Login - Place Order → Create Bill - Place Order → View Menu Extend: - Place Order ← View Chosen Dishes - Place Order ← Manage Chosen Dishes
Main scenario	1. After diners have viewed the Menu and added the food they want to order to the list of dishes to order. 2. Diners click the Place Order button. 3. The system records and sends a confirmation message that "you want to order the selected dishes". 4. Diners click the Accept button to agree to order. 5. The system records and saves the information of the dish to the diner's invoice as well as sends the dish information placed to the chef for processing.
Alternatives	1.a. If the diner does not want to order the selected item, the diner can remove the item from the wish list by pressing the Remove button located in the same row as the dish. 1.b. If you want to change the order quantity of the selected dish, you can increase or decrease the quantity by clicking the + and - buttons in the quantity column in the same row as the dish you want to increase or decrease the quantity. 1.c. If diners want to change the note for the dish, the diner presses the re-note button. A windows popup will appear for diners to enter a new note for the dish. 4.a. If the diner presses the Cancel Order button. The system will stop the ordering process and turn off the order confirmation notification.

Table 7: Place Order use case specification

2.4.8 Use Case Modify State of Dishes Ordered

Use case ID	UC008
Use case Name	Modify State of Dish
Primary Actor	Customer
Stakeholder and Interests	Chef
Summary	Customers view the status of dishes that have been ordered. And the system will automatically update when there is a change from the chef
Pre-Condition	- Waiter successfully logged in. - The table has been successfully opened. - The diner's empty invoice has been successfully generated. - The customer has successfully placed the order. - The chef has successfully logged in.
Trigger	After the customer clicks Accept in the pop-up window, confirm the order.
Post-condition	- Diners can see the list of dishes that have been ordered and their status every time the chef makes a change.
Relationships	Include: Modify State of Food Ordered → Login - Modify State of Food Ordered → Place Order
Main scenario	1. After the customer clicks Accept button in the pop-up window to confirm the order. The system sends the information of all items with Pending status to the list of pending dishes of the chef. 2. The chef clicks to select 1 dish from the list on the chef's interface. 3. The system displays a pop-up window displaying information of the dish. 4. The chef reads the information and checks the ingredients to make the selected dish. 5. If there are enough ingredients, the chef presses the Accept button on the system. 6. The system records and disables the Deny button, Accept button and Cancel button on the pop-up window on the chef's interface. 7. The system updates the status of the dish on the system and the system interface of diners to Accepted. The system disables the Cancel Order, modify dish quantity and Remove button of the dish that switches the Accepted status on the customer's system interface. 8. After completing the dish, the chef presses the Complete button on the chef's system interface and contacts the waiter to bring the dish to the guests. The system updates the status of the dish on the system and the customer's system interface to Completed. 10.The system turns off the pop-up window of the existing dish and switches back to the system interface containing the list of dishes to be processed.

Alternatives	1.a. After confirming the order, the status of the dish has not changed to Accepted, Denied or Completed. Users can cancel their order by pressing the Cancel Order button in the same row as the item they want to cancel. The system will change the status of those items to Canceled 6.a.1. If there are not enough ingredients, the chef presses the Deny button on the system 6.a.2. The system records and closes the pop-up window of the current dish and transfers to the system interface the chef's list of dishes to be handled. 6.a.3. The system updates the status of the dish on the system and the system interface of diners to Denied. The system disables the Cancel Order button of the dish that has changed to Denied status.

Table 8: Modify State of Dish use case specification

2.4.9 Use Case Pay Bill by Cash

Use case ID	UC009
Use case Name	Pay Bill by Cash
Primary Actor	Customer
Stakeholder and Interests	Cashier, Manager
Summary	Customers proceed to pay and are provided by the system with the total amount of the invoice, the list of ordered dishes with Completed status and other relevant information.
Pre-Condition	- Waiter successfully logged in. - The table has been successfully opened. - The diner's empty invoice has been successfully generated. - The customer has successfully placed the order. - The cashier has successfully logged in. - Cashier confirms diner successful payment.
Trigger	After the diner clicks the Payment button on the diner's system interface.
Post-condition	- Diners complete bill payment.
Relationships	Include: Pay Bill by Cash → Login - Pay Bill by Cash → Check Orders in Bill - Pay Bill by Cash → Place Order Extend: - Print Bill → Pay Bill by Cash
Main scenario	1. After eating, diners want to pay the bill and click on the Payment button. 2. The system will switch to the system interface containing invoice information, a list of ordered items with Completed status, and a total amount. 3. Customer check the information is correct. 4. The system sends the invoice's information to the pending list on the cashier's system interface. 5. The cashier clicks to select an invoice from the list. 6. The system displays a pop-up window containing the invoice's information. 7. The cashier based on the desk ID included in the receipt will go to pick up the customer's cash, then check and confirm the successful payment on the system by clicking the Complete button in the pop-up window of the invoice. 8. The system will record and automatically adjust the status of the invoice to Completed and end the invoice. At the same time, change the status of the table position selected by diners to Waiting. 9. The system sends notifications to diners whether they want to print Bill or not on the customer's system interface. 10. Diners press the yes button. 11.The system records and automatically prints bills for diners. Then the cashier will take the printed bill and deliver it to the customer.

Alternatives	1.a.1. If diners press the Payment button while there are still dishes in Pending state. The system will send a message "Some dishes are still not served, do you still want to pay ". 1.a.2.a. If the diner presses the yes button on the notification. The system will automatically cancel orders of all dishes with Pending status and change the status of those dishes to Canceled on the system. 1.a.2.b. If the guest presses the no button. The payment process will be stopped. 1.b.1. If the diner presses the Payment button when the dishes are still in the Accepted state (the chef accepts the order and is processing). The system will send a message "Some dishes are being prepared, if you continue to pay you will have to pay for those items even though they have not been served. 1.b.2.a. If diners press the yes button on the notification. The system will summarize the invoice to include items with Accepted status instead of only items with Completed status. 1.b.2.b. If the guest presses the no button. The payment process will be stopped. 2.a. If diners change their mind and do not want to pay. Diners can switch back to the Menu interface through the sidebar. 3.a.1. If the information is incorrect, the diner clicks the Warning button. 3.a.2. The system records and sends notifications to the cashier to check the information. 3.a.3. The cashier who receives the notice will notify the waiter to come to the diner to find out why the invoice information is incorrect. If there is an unloaded dish although the system displays the Completed status, the waiter contacts the kitchen directly to prepare and bring it to diners. 7.a. Diners press the no button. The system will not print out the bill.

Table 9: Pay Bill by Cash use case specification

2.4.10 Use Case Check Orders in Bill

Use case ID	UC010
Use case Name	Check Orders in Bill
Primary Actor	Customer
Stakeholder and Interests	None
Summary	Customer check their ordered dishes before checkout.
Pre-Condition	- Waiter successfully logs in. - Waiters open tables for diners successfully. - Waiters create blank invoices for successful diners. - Successful ordering.
Trigger	After diners click the Payment button .
Post-condition	- Customers can see the list of ordered dishes
Relationships	Include: - Pay Bill by Cash → Check Orders in Bill - Check Orders in Bill → Login
Main scenario	1. After eating, diners want to pay the bill and click the Payment button. 2. The system will automatically switch to the interface containing all information of the invoice and the list of ordered dishes. 3. The client checks that the information is correct and clicks the Accept button. 4. The system records and sends information through to the cashier.
Alternatives	3.a.1 If the customer checks the information is incorrect and press the Warning button. 3.a.2. The system will send a notification to the cashier. 3.a.3. The cashier contacts the waiter to find out the reason for the misinformation and find a fix. 3.a.4. If the problem is resolved, then the client clicks Accept.

Table 10: Check Orders in Bill use case specification

2.4.11 Use Case Modify State of Dishes in Menu

Use case ID	UC011
Use case Name	Manage State of Dish in Menu
Primary Actor	Chef
Stakeholder and Interests	None
Summary	The chef checks the ingredients of the dishes and adjusts the status of the dishes in the Menu. Dishes with ingredients to be processed will appear on the diners' Menu and dishes with insufficient or no ingredients will not appear on the Menu.
Pre-Condition	- The chef has successfully logged in.
Trigger	After the chef presses the Manage Food in Menu button .
Post-condition	- The dishes are changed state by the chef are updated in the system and updated on the menu interface of the diners.
Relationships	Include: Manage State of Dish in Menu → Login
Main scenario	1. After the chef successfully logged into the system. The system record and switches to the interface containing the list of dishes waiting for the chef to prepare. 2. The chef clicks the Manage Food in Menu button. 3. The chef transfer system interface page contains all the dishes of the restaurant. 4. The chef checks the cooking ingredients. 5. If any dish has enough ingredients, the chef clicks the toggle button in the same row as the dish to switch to Available. If the dish is already available, the chef stays the same. 6. The system records and updates back on the system. 7. The system updates the Menu on the diner's interface .
Alternatives	5.a. If a dish does not have enough ingredients, the chef clicks the toggle button in the same row as the dish to change it to Unavailable. If the dish is already in an Unavailable state, the chef leaves it unavailable.

Table 11: Manage State of Dish in Menu use case specification

2.4.12 Use Case View Bill History

Use case ID	UC012
Use case Name	View Bill History
Primary Actor	Cashier
Stakeholder and Interests	Manager
Summary	The cashier views the statistics of invoices made by day, month or year.
Pre-Condition	- The cashier has successfully logged in.
Trigger	After the cashier presses the Statistic button.
Post-condition	- The cashier can view the list of payment invoices according to the timeline selected.
Relationships	Include: View Bill History $\rightarrow$ Login
Main scenario	1. After the cashier successfully logs in. The system switches the interface to the list of invoices to be processed. 2. The cashier clicks the Statistics button. 3. The system displays all invoices paid for the current day. 4. The cashier selects 1 invoice from the list. 5. The system displays the details of that invoice in the pop-up window. 6. The cashier presses the outside of window or button Close in window 7. The system closes the window 8. The cashier selects the time period you want to watch. 9. The system displays all invoices paid during that period.
Alternatives	3.a. If there are currently no invoices paid by the system, the interface will be blank. 8.a. If during the selected period the system has no invoices paid, the interface will be blank.

Table 12: View Bill History use case specification

2.4.13 Use Case Manage Staff

Use case ID	UC013
Use case Name	Manage Staff
Primary Actor	Manager
Stakeholder and Interests	None
Summary	Management utilizes the system to manage staffs with tasks such as viewing the list and information of staffs, viewing login history, editing and updating staff information and permissions, disabling or enabling staff accounts.
Pre-Condition	Manager must successfully logged in to the system.
Trigger	Immediately after the manager successfully logs into the system.
Post-condition	Manager is able to perform tasks such as viewing the list and information of staffs, viewing login history, editing and updating staff information, adding and deleting staff from the list.
Relationships	Manage Staff → Login
Main scenario	1. After the manager successfully logged into the system. 2. The system switches to an interface containing a list of all the restaurant's employees and their status (whether working that day or not). 3. Manager selects a staff in the list. 4. The system switches to an interface that displays an staff's details including login history and interactions with that staff's information. 5. If the manager wants to adjust the staff's information, the manager clicks the Update button. 6. The system records and displays a pop-up window containing the employee's information fields for the Manager to modify the information. 7. Manage click Update button on the pop-up window. 8. The system checks that the information fields have entered the correct data. Information fields that managed to be blank, the system defaults that those information fields do not change. 9. The system sends an update confirmation message to the manager. 10. Manager click Confirm on the notification. 11. The system updates the information fields of that staff that are managed to enter new information on top of the system. The system updates the staff's details on the interface. 12. If the manager wants to disable an staff's account, click the Disable Account button in the interface containing the staff's details. 13. When finished checking or updating information, the manager can press the Back button to return to the list of all employees.

Main scenario	14. If the manager wants to add a new staff to the list, the manager clicks the Add button on the staff list interface. 15. The system record and displays a pop-up windows containing the information fields needed to add a new staff. 16. Manager enter information into the field and click the Add button. 17. The system checks the information entered correctly and sufficiently. The system sends a notification confirming the addition of new staff. 18. Manager click Confirm. 19. The system records and adds a new staff to the system. 20. If the manager wants to remove an staff from the list, the manager clicks the Delete button in the same line as the staff who wants to delete from the list. 21. The system records and sends confirmation messages. 22. Manager click Confirm. 23. The system recognizes and removes that staff from the system.
Alternatives	8.a. If management enters the staff's new information to update are wrong data type, the system will send a wrong location notification for management to correct. 10.a. If the manager presses the No button, the system stops the update information process. 12.a. If the manager wants to activate the staff's account, the manager clicks the Enable button in the interface containing the details of the staff who wants to activate the account. 17.a. If the manager enters information of the wrong data type or enters insufficient information, the system will send a notification of the wrong or missing location for the management to correct. 22.a. If the manager clicks No, the system will stop the staff deletion process.

Table 13: Manage Staff use case specification

2.5 Entity-Relationship Diagram

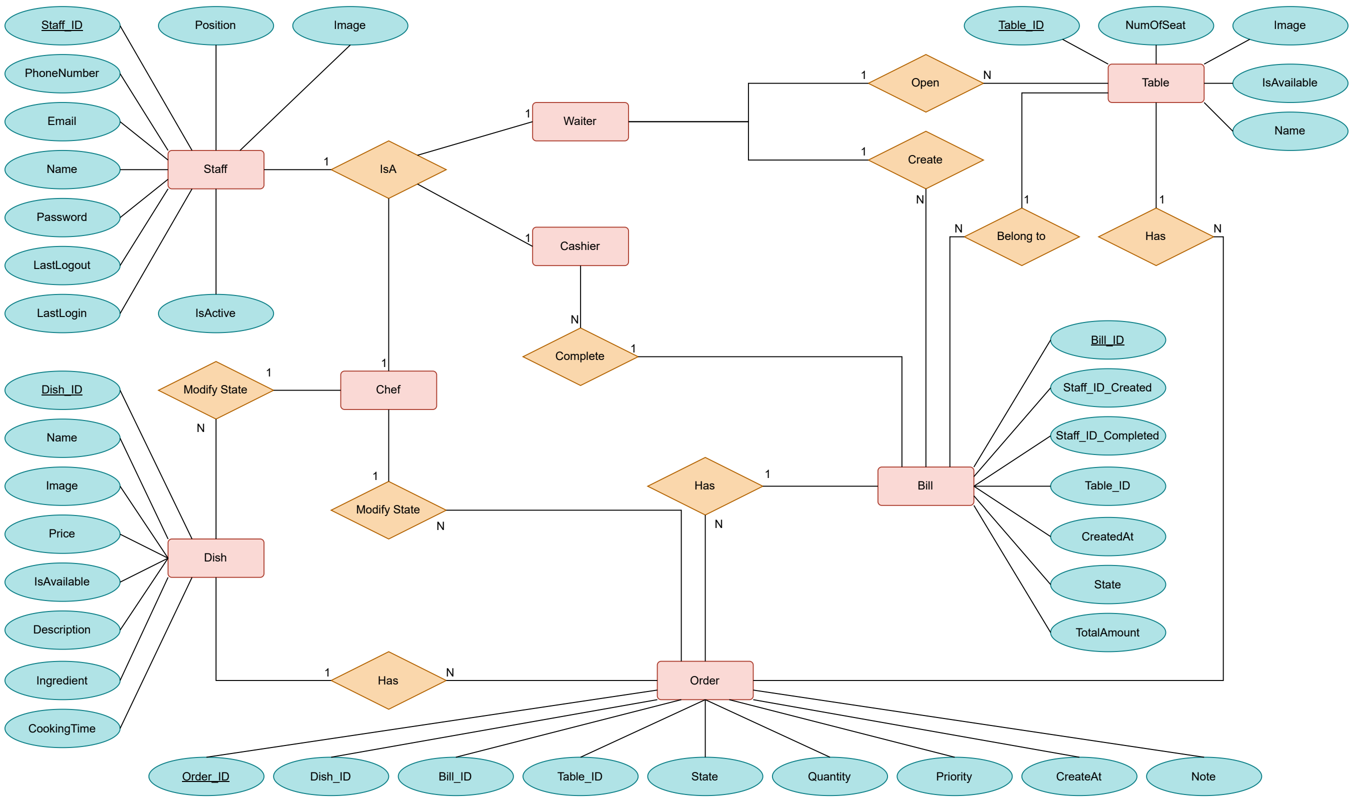


Figure 2: Entity-Relationship Diagram of Customer Order Management Module

2.6 Class Diagram

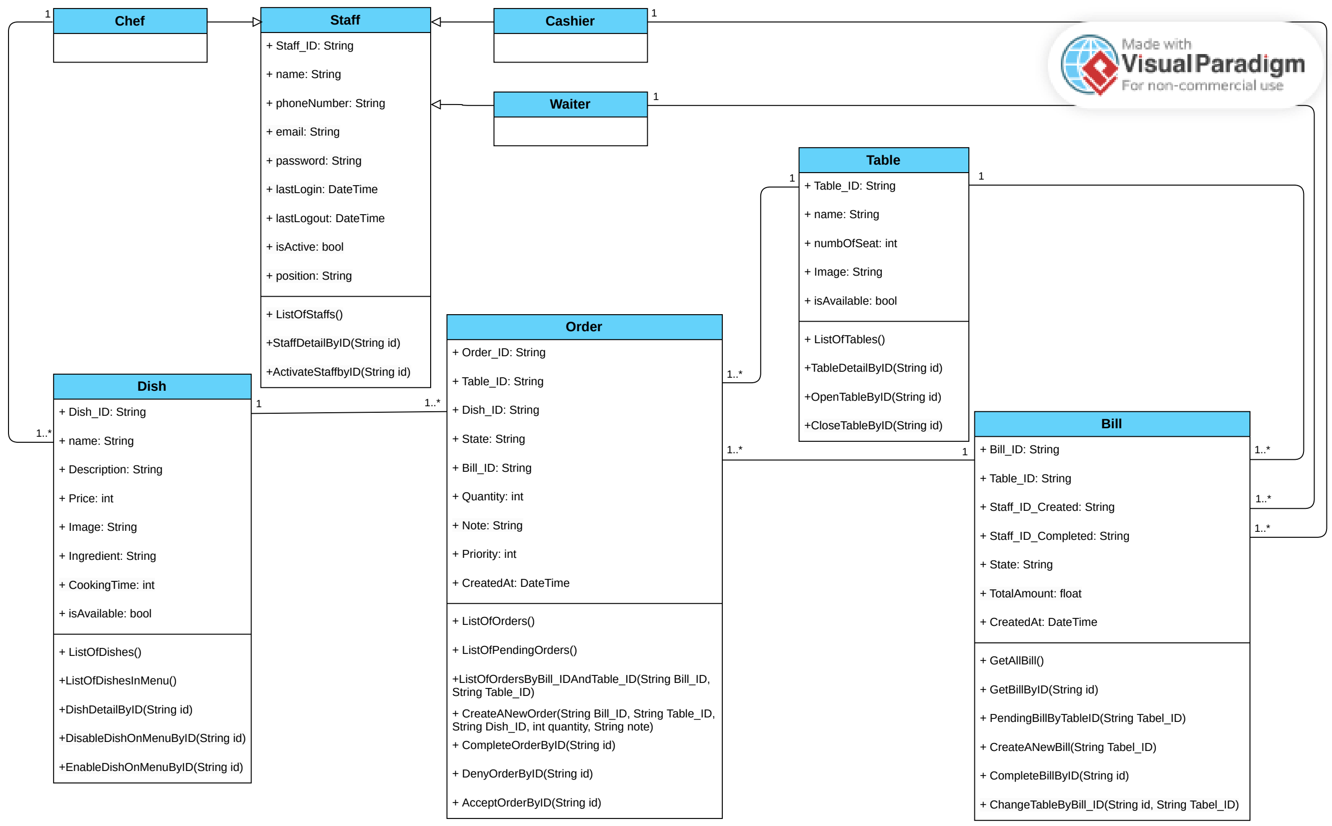


Figure 3: Class Diagram of Customer Order Management Module

2.7 Physical Database Model

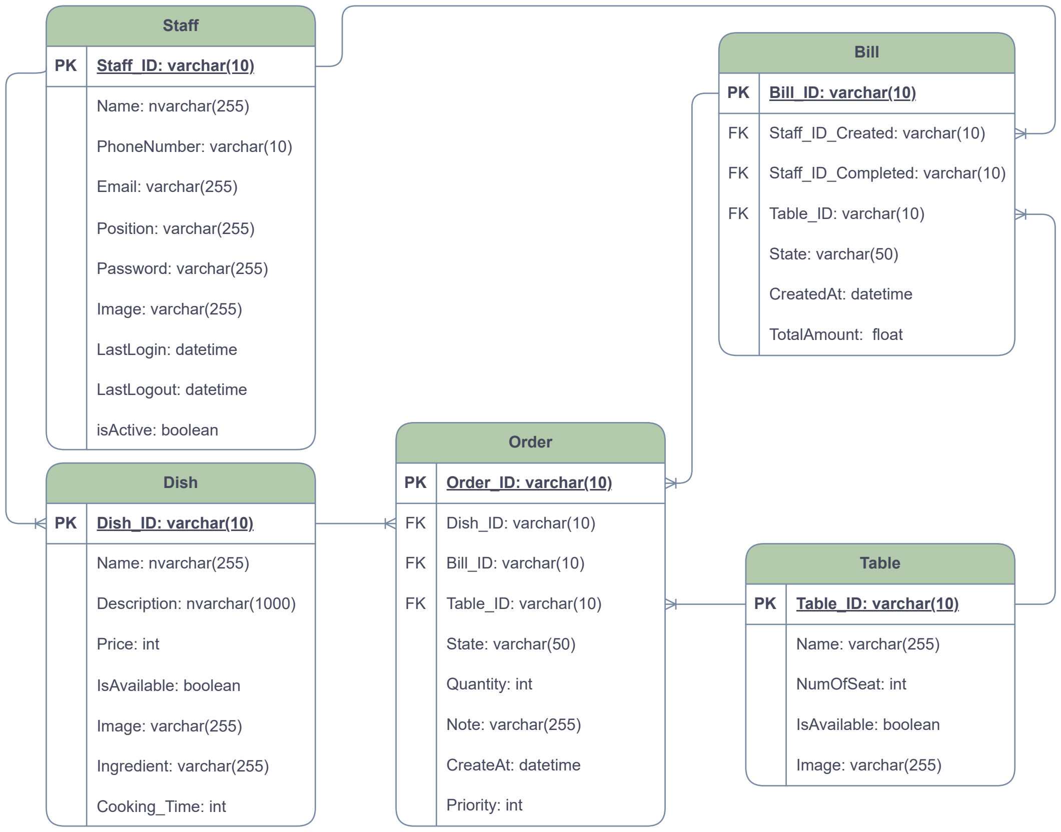


Figure 4: Physical-level Database Model of Customer Order Management Module

2.8 Finite State Machine Diagram

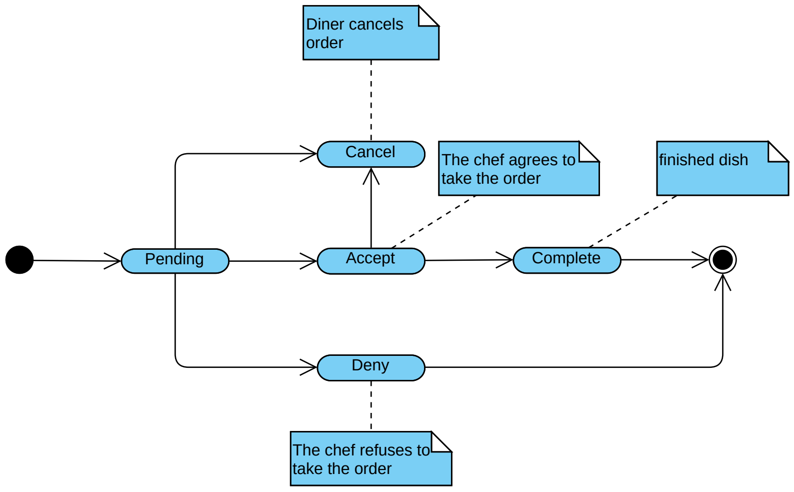

Diner cancels order
Cancel
The chef agrees to take the order
finished dish
Pending
Accept
Complete
Deny
The chef refuses to take the order

Figure 5: Finite State Machine Diagram of Order in Customer Order Management Module

3 System Implementation

3.1 Project APIs

3.1.1 Table APIs

```
// [GET] List of Table -> /api/table/list
- Input:
    Params: {null}
    Body: {null}
- Ouput: [
  {
    "_id": "6427cd62f33bd700584352db",
    "Name": "Table 004",
    "Image": "/images/tables/table04.jpg",
    "isAvailable": true,
    "NumOfSeat": 10,
    "__v": 0
  },
  {
    "_id": "6427cd48f33bd700584352d7",
    "Name": "Table 002",
    "Image": "/images/tables/table02.jpg",
    "isAvailable": true,
    "NumOfSeat": 8,
    "__v": 0
  },
  {
    "_id": "63e8f248ec09e543cd357f74",
    "Name": "Table 001",
    "isAvailable": true,
    "NumOfSeat": 5,
    "__v": 0,
    "Image": ""
  },
  {
    "_id": "6427cd57f33bd700584352d9",
    "Name": "Table 003",
    "Image": "/images/tables/table03.jpg",
    "isAvailable": true,
    "NumOfSeat": 5,
    "__v": 0
  }
]
```

Listing 1: List of Tables

```
// [GET] Table Detail by ID -> api/table/:id
- Input:
    Params: {:id=6427cd57f33bd700584352d9}
    Body: {null}
- Ouput: {
  "_id": "6427cd57f33bd700584352d9",
  "Name": "Table 003",
  "Image": "/images/tables/table03.jpg",
  "isAvailable": true,
  "NumOfSeat": 5,
  "__v": 0
}
```

Listing 2: Table Detail by ID

```
// [PUT] Open Table by ID -> /api/table/open/:id
- Input:
    Params: {:id=6427cd57f33bd700584352d9}
    Body: {null}
- Ouput: {
  "message": "Open Successfully"
}
```

Listing 3: Open Table by ID

```
// [PUT] Close Table by ID -> /api/table/close/:id
- Input:
    Params: {:id=6427cd57f33bd700584352d9}
    Body: {null}
- Ouput: {
  "message": "Close Successfully"
}
```

Listing 4: Close Table by ID

3.1.2 Staff APIs

```
// [GET] List of Staffs -> /api/staff/list
- Input:
    Params: {null}
    Body: {null}
- Ouput: [
  {
    "_id": "63e898f7181ad06276241aee",
    "Name": "Tran Khai Hoang",
    "Position": "Manager",
    "Email": "tkh@sud.com",
    "Password": "$2a$10$riJaBmh5on2bdib/wuNAweDdOPTlyqGZdauhDJbDd4B8bGZaRVymu",
    "__v": 0,
    "Last_Logout": "2023-04-01T06:26:48.701Z",
    "Last_login": "2023-04-01T07:03:52.240Z"
  },
  {
    "_id": "6425319438d6e50a8fbe67c7",
    "Name": "Nguyen Nhat Thong",
    "Image": "/img/staff/staff01.jpg",
    "PhoneNumber": "0378807854",
    "Position": "Waiter",
    "Email": "thong@gmail.com",
    "Password": "$2a$10$riJaBmh5on2bdib/wuNAweDdOPTlyqGZdauhDJbDd4B8bGZaRVymu",
    "isActive": false,
    "__v": 0,
    "Last_login": "2023-03-31T01:56:20.069Z",
    "Last_Logout": "2023-03-30T15:35:54.348Z"
  },
  {
    "_id": "642661a0d4f8a068c454014f",
    "Name": "Nguyen Gia Khiem",
    "Image": "/img/staff/staff01.jpg",
    "PhoneNumber": "0378807854",
    "Position": "Chef",
    "Email": "khiem@gmail.com",
    "Password": "$2a$10$aXh6Jg9x51VaJPihipaFbuDoP3ljWe7Zgbt2gbtfoNQF42CJKx8IG",
    "isActive": false,
    "__v": 0
  }
]
```

Listing 5: List of Staffs

```
// [GET] Staff Detail by ID -> /api/staff/:id
- Input:
    Params: {:id=6425319438d6e50a8fbe67c7}
    Body: {null}
- Ouput: {
  "_id": "6425319438d6e50a8fbe67c7",
  "Name": "Nguyen Nhat Thong",
  "Image": "/img/staff/staff01.jpg",
  "PhoneNumber": "0378807854",
  "Position": "Waiter",
  "Email": "thong@gmail.com",
  "Password": "$2a$10$riJaBmh5on2bdib/wuNAweDdOPTlyqGZdauhDJbDd4B8bGZaRVymu",
  "isActive": false,
  "__v": 0,
  "Last_login": "2023-03-31T01:56:20.069Z",
  "Last_Logout": "2023-03-30T15:35:54.348Z"
}
```

Listing 6: Staff Detail by ID

```
// [PUT] Activate Staff by ID -> /api/staff/activate/:id
- Input:
    Params: {:id=6425319438d6e50a8fbe67c7}
    Body: {null}
- Ouput: {
  "message": "Activate Successfully"
}
```

Listing 7: Activate Staff by ID

3.1.3 Order APIs

```
// [GET] List of Orders -> /api/order/list
- Input:
    Params: {null}
    Body: {null}
- Ouput: [
  {
    "_id": "64267601ef15a475bef194f6",
    "DishID": "64264c08f8c8b3dbec9ad852",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Complete",
    "Quantity": 1,
    "Note": "Order 5",
    "CreatedAt": "2023-03-31T05:54:20.781Z",
    "Priority": 60,
    "__v": 0
  },
  {
    "_id": "642663c6fd92955f7b210c77",
    "DishID": "64264c35f8c8b3dbec9ad854",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Complete",
    "Quantity": 3,
    "Note": "Order 4",
    "CreatedAt": "2023-03-31T04:35:54.870Z",
    "Priority": 90,
    "__v": 0
  }
]
```

Listing 8: List of Orders

```
// [GET] List of Orders by BiiiID and TableID -> /api/order/list/:BillID/:
   TableID
- Input:
    Params: {:BillID=642663a9fd92955f7b210c74,
    :TableID=63e8f248ec09e543cd357f74}
    Body: {null}
- Ouput: [{
    "_id": "64267601ef15a475bef194f6",
    "DishID": "64264c08f8c8b3dbec9ad852",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Complete",
    "Quantity": 1,
    "Note": "Order 5",
    "CreatedAt": "2023-03-31T05:54:20.781Z",
    "Priority": 60,
    "__v": 0
  },{
    "_id": "6427da5137823db004306c7a",
    "DishID": "64264c08f8c8b3dbec9ad852",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Pending",
    "Quantity": 1,
    "Note": "Order 1",
    "CreatedAt": "2023-04-01T07:15:10.952Z",
    "Priority": 60,
    "__v": 0
  },{
    "_id": "642663c6fd92955f7b210c77",
    "DishID": "64264c35f8c8b3dbec9ad854",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Complete",
    "Quantity": 3,
    "Note": "Order 4",
    "CreatedAt": "2023-03-31T04:35:54.870Z",
    "Priority": 90,
    "__v": 0
  },{
    "_id": "6427da5e37823db004306c7d",
    "DishID": "64264c08f8c8b3dbec9ad852",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Pending",
    "Quantity": 2,
    "Note": "Order 5",
    "CreatedAt": "2023-04-01T07:15:10.952Z",
    "Priority": 120,
    "__v": 0
  }]
```

Listing 9: List of Orders by BiiiID and TableID

```
// [GET] List of Pending Orders -> /api/order/list/pending
- Input:
    Params: {null}
    Body: {null}
- Ouput: [
  {
    "_id": "6427da5137823db004306c7a",
    "DishID": "64264c08f8c8b3dbec9ad852",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Pending",
    "Quantity": 1,
    "Note": "Order 1",
    "CreatedAt": "2023-04-01T07:15:10.952Z",
    "Priority": 60,
    "__v": 0
  },
  {
    "_id": "6427da5e37823db004306c7d",
    "DishID": "64264c08f8c8b3dbec9ad852",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Pending",
    "Quantity": 2,
    "Note": "Order 5",
    "CreatedAt": "2023-04-01T07:15:10.952Z",
    "Priority": 120,
    "__v": 0
  }
]
```

Listing 10: List of Pending Orders

```
// [POST] Create a New Order -> /api/order/create
- Input:
    Params: {null}
    Body: {
        "BillID"="642663a9fd92955f7b210c74",
        "TableID"="63e8f248ec09e543cd357f74",
        "DishID"="64264c08f8c8b3dbec9ad852",
        "Quantity"= 2,
        "Note"="Order 5",
    }
- Ouput: {
  "message": "New order has been created successfully"
}
```

Listing 11: Create a New Order

```
// [PUT] Complete Order by ID -> /api/order/complete/:id
- Input:
    Params: {:id=6427da5e37823db004306c7d}
    Body: {null}
- Ouput: {
  "message": "Complete Successfully",
  "order": {
    "_id": "6427da5e37823db004306c7d",
    "DishID": "64264c08f8c8b3dbec9ad852",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Complete",
    "Quantity": 2,
    "Note": "Order 5",
    "CreatedAt": "2023-04-01T07:15:10.952Z",
    "Priority": 120,
    "__v": 0
  }
}
```

Listing 12: Complete Order by ID

```
// [PUT] Deny Order by ID -> /api/order/deny/:id
- Input:
    Params: {:id=6427da5137823db004306c7a}
    Body: {null}
- Ouput: {
  "message": "Deny Successfully",
  "order": {
    "_id": "6427da5137823db004306c7a",
    "DishID": "64264c08f8c8b3dbec9ad852",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Deny",
    "Quantity": 1,
    "Note": "Order 1",
    "CreatedAt": "2023-04-01T07:15:10.952Z",
    "Priority": 60,
    "__v": 0
  }
}
```

Listing 13: Deny Order by ID

```
// [PUT] Accept Order by ID -> /api/order/accept/:id
- Input:
    Params: {:id=6427da5e37823db004306c7d}
    Body: {null}
- Ouput: {
  "message": "Accept Successfully",
  "order": {
    "_id": "6427da5e37823db004306c7d",
    "DishID": "64264c08f8c8b3dbec9ad852",
    "BillID": "642663a9fd92955f7b210c74",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Accept",
    "Quantity": 2,
    "Note": "Order 5",
    "CreatedAt": "2023-04-01T07:15:10.952Z",
    "Priority": 120,
    "__v": 0
  }
}
```

Listing 14: Accept Order by ID

3.1.4 Bill APIs

```
// [GET] Get All Bill -> /api/bill/all
- Input:
    Params: {null}
    Body: {null}
- Ouput: [
  {
    "_id": "6427dc5d5d7b855e4bb0a606",
    "StaffID_Created": "bReW-G_Eib0OGKWMWq5TGIsrOfnzR297",
    "TableID": "6427cd62f33bd700584352db",
    "State": "Pending",
    "CreateAt": "2023-04-01T07:21:13.144Z",
    "TotalAmount": 0,
    "__v": 0
  },
  {
    "_id": "642663a9fd92955f7b210c74",
    "StaffID_Created": "__AyJ5Yd0kUjMY8I9D14UK0OVKeskqqJ",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Pending",
    "CreateAt": "2023-03-31T04:35:54.866Z",
    "TotalAmount": 7620000,
    "__v": 0
  }
]
```

Listing 15: Get All Bill

```
// [GET] Get Bill by ID -> /api/bill/:id
- Input:
    Params: {:id=6427dc5d5d7b855e4bb0a606}
    Body: {null}
- Ouput: {
  "_id": "6427dc5d5d7b855e4bb0a606",
  "StaffID_Created": "bReW-G_Eib0OGKWMWq5TGIsr0fnzR297",
  "TableID": "6427cd62f33bd700584352db",
  "State": "Pending",
  "CreateAt": "2023-04-01T07:21:13.144Z",
  "TotalAmount": 0,
  "__v": 0
}
```

Listing 16: Get Bill by ID

```
// [GET] Pending Bill by TableID -> /api/bill/pending/:tableID
- Input:
    Params: {:tableID=6427cd62f33bd700584352db}
    Body: {null}
- Ouput:{
  "_id": "6427dc5d5d7b855e4bb0a606",
  "StaffID_Created": "bReW-G_Eib0OGKWMWq5TGIsr0fnzR297",
  "TableID": "6427cd62f33bd700584352db",
  "State": "Pending",
  "CreateAt": "2023-04-01T07:21:13.144Z",
  "TotalAmount": 0,
  "__v": 0
}
```

Listing 17: Pending Bill by TableID

```
// [POST] Create a New Bill -> /api/bill/create
- Input:
    Params: {null}
    Body: {"TableID"="6427cd62f33bd700584352db"}
- Ouput:{
  "message": "New bill has been created successfully"
}
```

Listing 18: Create a New Bill

```
// [PUT] Complete Bill by ID -> /api/bill/complete/:id
- Input:
    Params: {:id=642663a9fd92955f7b210c74}
    Body: {null}
- Ouput:{
  "message": "Successfully",
  "bill": {
    "_id": "642663a9fd92955f7b210c74",
    "StaffID_Created": "__AyJ5Yd0kUjMY8I9D14UK00VKeskqqJ",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Complete",
    "CreateAt": "2023-03-31T04:35:54.866Z",
    "TotalAmount": 7620000,
    "__v": 0,
    "StaffID_Completed": "GE7RtuxIrstAAIEjXv-CBnrh40Mo_j0i"
  }
}
```

Listing 19: Complete Bill by ID

```
// [PUT] Change Table by Bill ID -> /api/bill/change/:id/:TableID
- Input:
    Params: {:id=6427dc5d5d7b855e4bb0a606,
    :TableID=63e8f248ec09e543cd357f74}
    Body: {null}
- Ouput:{
  "message": "Change table Successfully",
  "bill": {
    "_id": "6427dc5d5d7b855e4bb0a606",
    "StaffID_Created": "bReW-G_Eib00GKWMWq5TGIsr0fnzR297",
    "TableID": "63e8f248ec09e543cd357f74",
    "State": "Pending",
    "CreateAt": "2023-04-01T07:21:13.144Z",
    "TotalAmount": 0,
    "__v": 0
  }
}
```

Listing 20: Change Table by Bill ID

3.1.5 Dish APIs

```
// [GET] List of Dishes -> /api/dish/list
- Input:
    Params: {null}
    Body: {null}
- Ouput:[{
    "_id": "64264c08f8c8b3dbec9ad852",
    "Name": "Crab and Beef Hotpot",
    "Image": "/images/dish/Crab-and-Beef-Hotpot.jpg",
    "Price": 990000,
    "isAvailable": true,
    "CookingTime": 60,
    "Ingredient": "Wild crab, fresh beef, cartilage, tofu, tomatoes, balut eggs,
     dried shallots, ginger",
    "Description": "The broth has a slightly sour taste from the use of rice
   vinegar, accompanied by the fragrant aroma of crab and tender sweetness of
   beef, including the chewiness of the cartilage and tendons. The fresh green
   vegetables make it easy to eat and perfect for the autumn weather.",
    "__v": 0
  }, {
    "_id": "64264c35f8c8b3dbec9ad854",
    "Name": "Stir-fried saft shell crab with salted egg yolk",
    "Image": "/images/dish/Stir-fried-saft-shell-crab-with-salted-egg-yolk.jpg",
    "Price": 1550000,
    "isAvailable": true,
    "CookingTime": 30,
    "Ingredient": "Soft-shell crab, Bread, Chicken egg, Salted egg",
    "Description": "The sweetness from peeled crabs combined with a little salt
   ...",
    "__v": 0
  }]
```

Listing 21: List of Dishes

```
// [GET] List of Dishes in Menu -> /api/dish/menu
- Input:
    Params: {null}
    Body: {null}
- Ouput:[
  {
    "_id": "64264c08f8c8b3dbec9ad852",
    "Name": "Crab and Beef Hotpot",
    "Image": "/images/dish/Crab-and-Beef-Hotpot.jpg",
    "Price": 990000,
    "isAvailable": true,
    "CookingTime": 60,
    "Ingredient": "Wild crab, fresh beef, cartilage, tofu, tomatoes, balut eggs,
     dried shallots, ginger",
    "Description": "The broth has a slightly sour taste from the use of rice
   vinegar, accompanied by the fragrant aroma of crab and tender sweetness of
   beef, including the chewiness of the cartilage and tendons. The fresh green
   vegetables make it easy to eat and perfect for the autumn weather.",
    "__v": 0
  },
  {
    "_id": "64264c35f8c8b3dbec9ad854",
    "Name": "Stir-fried saft shell crab with salted egg yolk",
    "Image": "/images/dish/Stir-fried-saft-shell-crab-with-salted-egg-yolk.jpg",
    "Price": 1550000,
    "isAvailable": true,
    "CookingTime": 30,
    "Ingredient": "Soft-shell crab, Bread, Chicken egg, Salted egg",
    "Description": "The sweetness from peeled crabs combined with a little salt
   ...",
    "__v": 0
  }
]
```

Listing 22: List of Dishes in Menu

```
// [GET] Dish Detail by ID -> /api/dish/:id
- Input:
    Params: {:id=64264c35f8c8b3dbec9ad854}
    Body: {null}
- Ouput:{
  "_id": "64264c35f8c8b3dbec9ad854",
  "Name": "Stir-fried saft shell crab with salted egg yolk",
  "Image": "/images/dish/Stir-fried-saft-shell-crab-with-salted-egg-yolk.jpg",
  "Price": 1550000,
  "isAvailable": true,
  "CookingTime": 30,
  "Ingredient": "Soft-shell crab, Bread, Chicken egg, Salted egg",
  "Description": "The sweetness from the soft-shell crab combined with the
   slightly salt...",
  "__v": 0
}
```

Listing 23: Dish Detail by ID

```
// [PUT] Disable Dish on Menu by ID -> /api/dish/disable/:id
- Input:
    Params: {:id=64264c35f8c8b3dbec9ad854}
    Body: {null}
- Ouput:{
  "message": "Disable Successfully",
  "dish": {
    "_id": "64264c35f8c8b3dbec9ad854",
    "Name": "Stir-fried saft shell crab with salted egg yolk",
    "Image": "/images/dish/Stir-fried-saft-shell-crab-with-salted-egg-yolk.jpg",
    "Price": 1550000,
    "isAvailable": false,
    "CookingTime": 30,
    "Ingredient": "Soft-shell crab, Bread, Chicken egg, Salted egg",
    "Description": "The sweetness from the soft-shell crab combined with the
   slightly salt...",
    "__v": 0
  }
}
```

Listing 24: Disable Dish on Menu by ID

```
// [PUT] Enable Dish on Menu by ID -> /api/dish/enable/:id
- Input:
    Params: {:id=64264c35f8c8b3dbec9ad854}
    Body: {null}
- Ouput:{
  "message": "Enable Successfully",
  "dish": {
    "_id": "64264c35f8c8b3dbec9ad854",
    "Name": "Stir-fried saft shell crab with salted egg yolk",
    "Image": "/images/dish/Stir-fried-saft-shell-crab-with-salted-egg-yolk.jpg",
    "Price": 1550000,
    "isAvailable": true,
    "CookingTime": 30,
    "Ingredient": "Soft-shell crab, Bread, Chicken egg, Salted egg",
    "Description": "The sweetness from the soft-shell crab combined with the
   slightly salt...",
    "__v": 0
  }
}
```

Listing 25: Enable Dish on Menu by ID

3.1.6 User APIs

```
// [POST] Login -> /api/user/login
- Input:
    Params: {null}
    Body: {"Email"="tkh@sud.com", "Password"="hoang9999"}
- Ouput:{
  "message": "Login successful.",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
   eyJfaWQiOiI2M2U4OThmNzE4MWFkMDYyNzYyNDFhZWUiLCJpYXQiOjE2ODAzMzAzNTl9.
   WKQGv7dLKChldN9SvL_1sPaAyhXgz0i5KPJffUKkeJs"
}
```

Listing 26: Login

```
// [POST] Logout -> /api/user/logout
- Input:
    Params: {null}
    Body: {null}
- Ouput:{
  "message": "Logout successful."
}
```

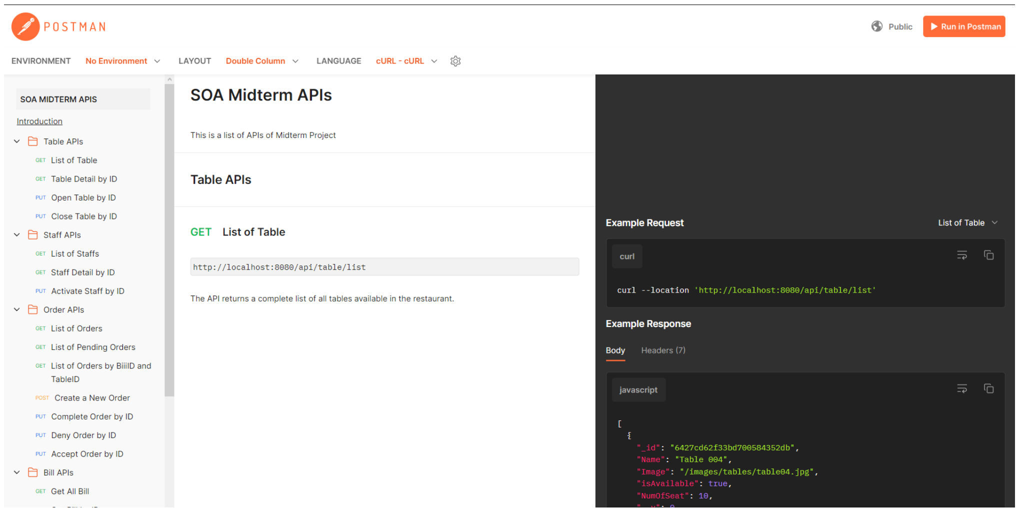
Listing 27: Logout

```
// [POST] Change password -> /api/user/changePassword
- Input:
    Params: {null}
    Body: {"Password"="hoang9999", "NewPassword"="hoang"}
- Ouput:{
  "message": "Password changed successfully"
}
```

Listing 28: Change password

3.2 APIs documentation using Postman

Please follows this link for the complete Postman documentation: Postman Documentation



ENVIRONMENT No Environment LAYOUT Double Column LANGUAGE cURL - cURL

SOA MIDTERM APIS

Introduction

- Table APIs
  - GET List of Table
  - GET Table Detail by ID
  - PUT Open Table by ID
  - PUT Close Table by ID
- Staff APIs
  - GET List of Staffs
  - GET Staff Detail by ID
  - PUT Activate Staff by ID
- Order APIs
  - GET List of Orders
  - GET List of Pending Orders
  - GET List of Orders by BiiilD and TableID
  - POST Create a New Order
  - PUT Complete Order by ID
  - PUT Deny Order by ID
  - PUT Accept Order by ID
- Bill APIs
  - GET Get All Bill

SOA Midterm APIs

This is a list of APIs of Midterm Project

Table APIs

GET List of Table

```
http://localhost:8080/api/table/list
```

The API returns a complete list of all tables available in the restaurant.

Example Request — List of Table

curl

```
curl --location 'http://localhost:8080/api/table/list'
```

Example Response

Body Headers (7)

javascript

```
[
  {
    "_id": "6427cd62f33bd700584352db",
    "Name": "Table 004",
    "Image": "/images/tables/table04.jpg",
    "isAvailable": true,
    "NumOfSeat": 10,
```

Figure 6: APIs documentation preview via Postman

References

[1] Phuc H.Duong, M.Sc. **Requirement Determination & Use-Case Analysis**. Ton Duc Thang University. 2022
[2] Phuc H.Duong, M.Sc. **Process Modeling & Data Modeling**. Ton Duc Thang University. 2022