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

PROJECT REPORT

PRACTICE ON SOFTWARE ENGINEERING

Restaurant POS 2.0

Lecturer: Assoc. Prof. Quan Thanh Tho

Nguyễn Luật Gia Khôi 1952079

Phạm Bùi Minh Huân 1952056

Phạm Khánh Trình 1953044

Nguyễn Đình Khương Duy 1952207

Changelog

No.	Date	Changes	Actors
10	23/10/2021	“Class diagram” is modified.	Phạm Khánh Trình Nguyễn Luật Gia Khôi
9	21/10/2021	“Component Diagram” for the whole system is added.	Nguyễn Luật Gia Khôi Nguyễn Đình Khương Duy
8	19/10/2021	“Model-View-Controller” architecture for the whole system is added.	Phạm Khánh Trình Phạm Bùi Minh Huân
7	06/10/2021	“Class diagram” for the whole system is added.	Nguyễn Luật Gia Khôi Nguyễn Đình Khương Duy
6	04/10/2021	“Sequence diagram” for the “Pay bill” use-case is added.	Phạm Bùi Minh Huân
5	02/10/2021	“Activity diagram” - capture major functional requirements of the system.	Phạm Khánh Trình
4	22/09/2021	“Use-case diagram for one specific feature” and “table format the use-case diagram” are added.	Nguyễn Đình Khương Duy Nguyễn Luật Gia Khôi
3	21/09/2021	“Use-case diagram for the whole system” is added.	Phạm Khánh Trình Phạm Bùi Minh Huân
2	20/09/2021	“Functional requirements” and “Non-functional requirements” are added.	Nguyễn Luật Gia Khôi Phạm Bùi Minh Huân
1	19/09/2021	Identify: - Context of the project. - Relevant stakeholders. - Requirements. - The scope of the project.	Phạm Bùi Minh Huân Nguyễn Đình Khương Duy Nguyễn Luật Gia Khôi Phạm Khánh Trình

Table of contents

I. Introduction

Problem description: Develop a responsive web-based POS system that implements the current business flow and provides contact-less interactions between customer and staff.

Users:

- Restaurants' staff (including owner)
- Customer

Workflow: In order to serve the restaurant business, the web-application includes table reservations, ordering food, alerts, billing, credit card processing, and customer management. Customers can turn on the web-application, browse the menu, order and decide on takeaway options. After that, the web-application will automatically calculate, display bill and payment options. On the other hand, restaurants' owners can login to their restaurant account and access digital customer management options, insights.

II. About the project

1. Context

Overall: Restaurant POS 2.0 provides a contactless end-to-end system for restaurants in this new age. Restaurants' owners will have their business intelligence increased and wasted effort reduced by digitalization of the business flow. The service is also extendable to oversee multiple restaurants in the future.

Intended audience: Traditional restaurants (such as Hòn Đất restaurant, Gogi House restaurant chains…)

Payment: Payments are made after meals (or after finished ordering when takeaway). Payment options include: cash, card.

Order: Customers can browse the menu and have the option for takeaway.

Table reservation: The option to reserve a table beforehand is available.

Restaurants' owners managing options: After logging in, restaurants' owners can manage customers, access insights and update menus.

Menu: Dishes available can be updated in real-time.

2. Stakeholders

Primary stakeholders:

- Restaurants' owners.
- Staff.
- Clients.

Secondary stakeholders:

- Banks.
- Ministry of Industry and Trade.
- Ministry of Information and Communications.

3. **What is expected to be done?**

Restaurant POS 2.0 is an online web-based food-ordering technology that:

- Allow customers to browse menus, order food.
- Calculate the amount owed by the customer.
- Prepare an invoice for the customer.
- Indicate the customer's options to make payment.

4. **Scope**

Restaurant POS 2.0 is a responsive web-based POS system with:

- Transaction capability of about 300 orders per day.
- Be extendable to use for multiple restaurants.

III. Requirements

1. Functional requirements

All the functional requirements are as followed:

- **Table reservation**: Customers can reserve the table in advance by choosing the time and number of people to be expected.
- **Menus browser**: Customers and staff both can view the menus and see all dishes and their prices.
- **Food ordering**: Customers order their food and have it served at the restaurant or prepared for takeaway. Staff receive orders and pass them to the kitchen.
- **Payment**: Bill is calculated when orders are taken and sent to customers at the end. Customers choose the desired type of payment and proceed to make the transactions. Receipts are calculated when payments are made.
- **Login**: Restaurants' managers can use an official account to sign into the system to gain access to managers' insights and actions(customer management, menu update).
- **Customer management**: Restaurants' managers can oversee the customer ordering process, the number of customers ordered so far, and access insights.
- **Menu update**: Restaurants' managers can modify, delete, add dishes and their prices to the menu.
- **Implementation:** using Web technology and QR code, so customers will not have to install apps.

2. Non-functional requirements

All the non-functional requirements are as followed:

- Allow non-direct contact between Clerks and Customers. (no direct contact required throughout the process)
- Fast (elapsed time for an action/order < 3 seconds)

- Usable from a mobile device, a tablet device, or a normal computer/laptop. (available on Chrome, Firefox, Edge...)
- Capable of handling about 300 orders per day or more.
- Extendable to use in multiple restaurants in the future.

3. Use-case diagram for the whole system

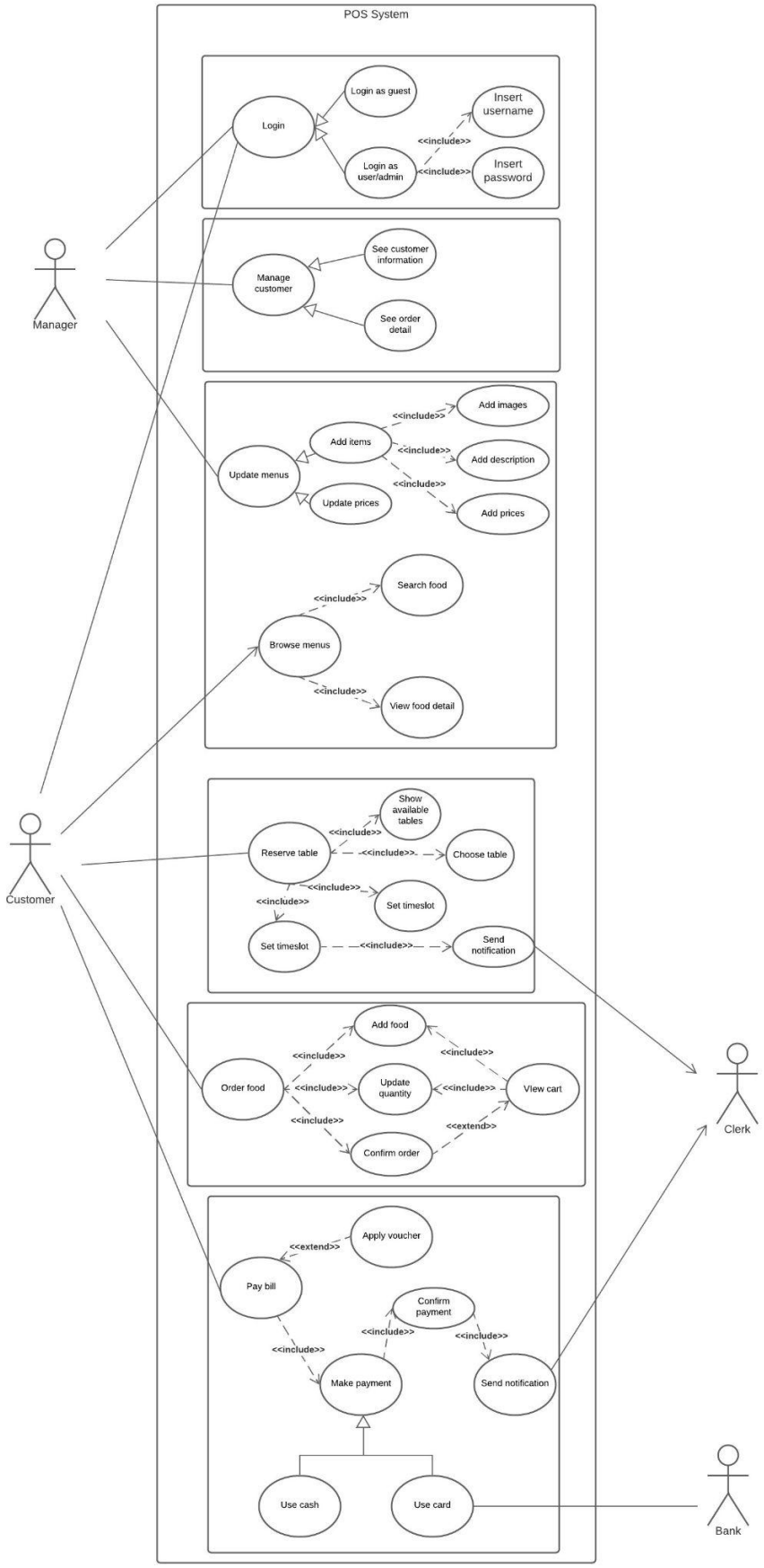

Figure 1: The use-case diagram for the whole system Restaurant POS 2.0.

IV. One specific feature

1. Feature:

We choose to demonstrate the use-case diagram of the "Pay bill" feature.

2. Use-case diagram:

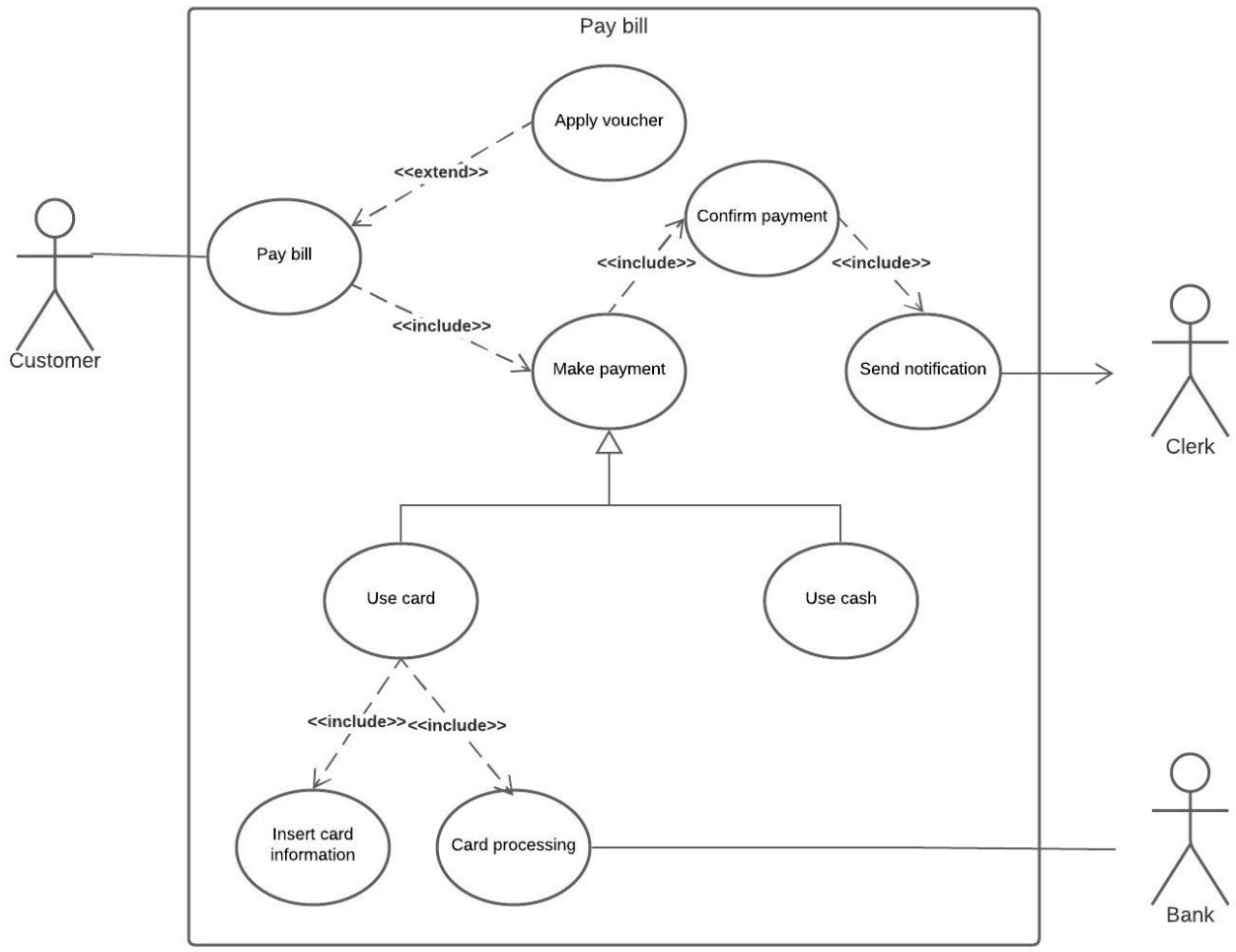

Figure 2: The use-case diagram for the "Pay bill" feature.

3. Description using table format:

Name	Pay bill
Actor	Customer, Restaurant, and Bank

Description	Customers have an option to apply vouchers. Customers have to choose a payment method. The payment information will be sent to the bank for authentication. The payment is confirmed and the restaurant will receive payment at the end of this use case.
Preconditions	The customer has confirmed the order and proceeded to payment.
Normal flow	1. (Extension point: The customer applies vouchers) 2. The customer chooses one of three payment options: Cash, Debit card, or Credit card 3. The customer makes payment. 4. The system displays the payment status to the user. 5. The use case ends.
Exceptions	*Exception 1: at step 3* 3a. If the card is invalid or expired, the system prompts a message to ask the customer to reenter the card information.
Alternative flow	*Alternative 1: at step 3* 3a. The customer uses a card: 1. The customer enters debit/credit card information. 2. The system sends card information to the bank for authentication and execution of the transaction. 3b. The customer uses cash: 1. The system proceeds to step 4

Use case scenario for the "Apply voucher" use case:

Name	Apply voucher
Actor	Customer.
Description	Customers have an option to apply vouchers. Customers must enter the code of their vouchers.
Preconditions	The customer has confirmed the order and proceeded to payment.
Normal flow	1. Customers finish choosing food and press the payment button. 2. The system proceeds to the payment page. 3. Customers choose whether to use the vouchers. 4. Customers enter their code. 5. Customers proceed to choose the payment option.
Exceptions	*Exception 1: at step 3* 3a. If the voucher's code is invalid or expired, the system prompts a message to ask the customer to reenter the code of the voucher.
Alternative flow	*Alternative 1: at step 3* 3a. The customers choose not to use the vouchers and proceed to choose the payment option.

Use case scenario for the "Make payment" use case

Name	Make payment
Actor	Customer, Restaurant and Bank
Description	Customers have to choose one of the two payment options and fill in the necessary information if a card is chosen for payment. The validation of the card will be carried out by the bank. Then, the payment is confirmed by the customer, the transaction is made and the restaurant is notified at the end of the use case.
Preconditions	The customer clicks to choose a payment method.
Normal flow	1. The system highlights the payment option that the customer clicked on. 2. The system displays the button to confirm the payment option. 3. The customer clicks on the button to confirm the payment. 4. The system displays the payment status to the user. 5. The use case ends.
Exceptions	*Exception 1: at step 2* 2a. If the card is invalid or expired or does not have enough money, the system prompts a message to ask the customer to reenter the card information.
Alternative flow	*Alternative 1: at step 2* 2a. The customer uses a card: 1. The system shows the form for the customer to fill in the card information. 2. The customer enters debit/credit card information.

	3. The customer clicks on the confirm button. 4. The system sends card information to the bank for authentication. 3. The system proceeds to step 3. 2b. The customer uses cash: 2. The system proceeds to step 3. *Alternative 2: at step 3* 3a. The customer uses a card: 1. The card transaction will be executed by the bank. 2. The system proceeds to step 4. 3b. The customer uses cash: 1. The system proceeds to step 4.

V. Activity diagram

Following is the activity diagram that captures the major functional requirements of Restaurant POS 2.0:

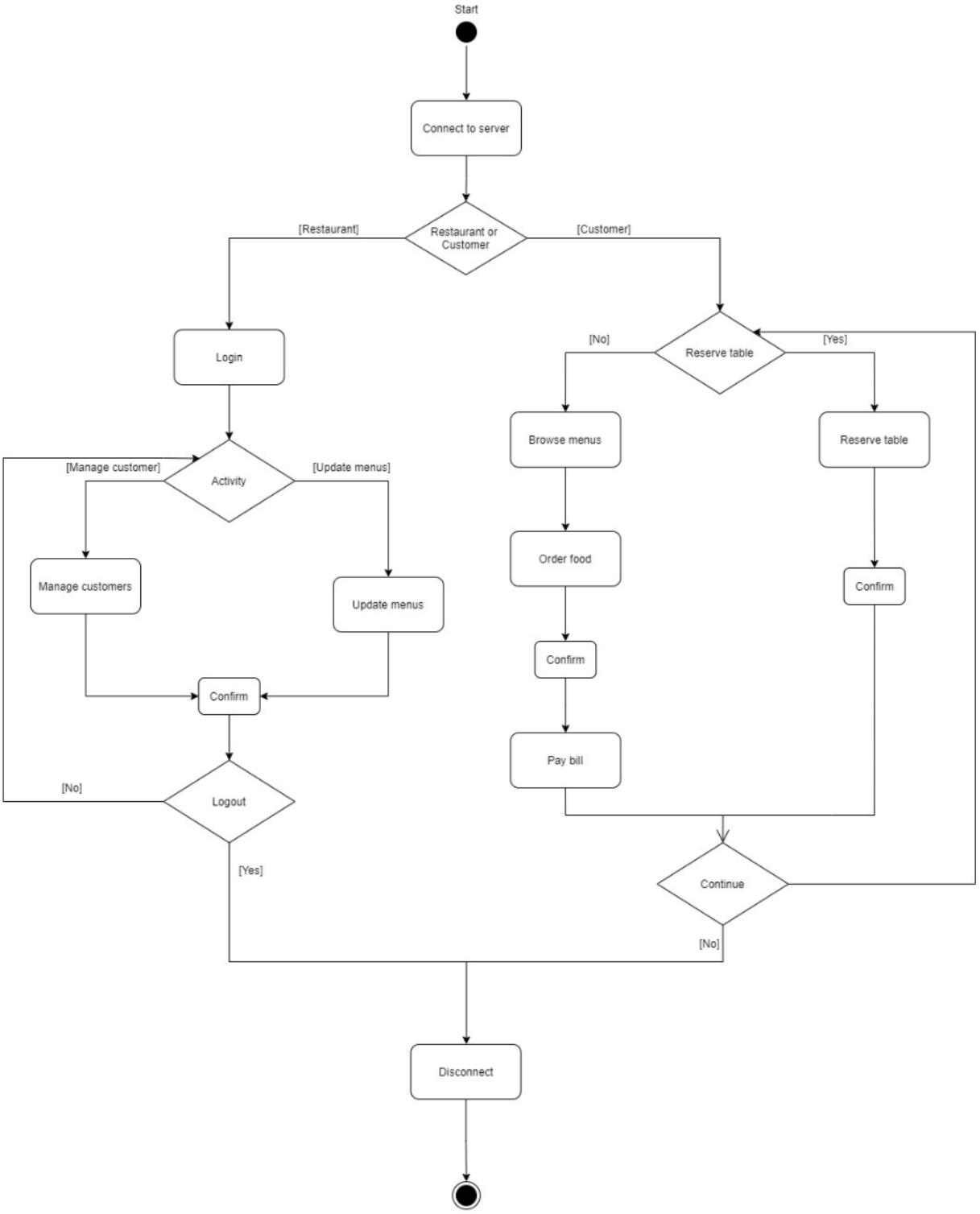

Figure 3: The activity diagram capturing major functional requirements of Restaurant POS 2.0.

VI. Sequence diagram

Following is the sequence diagram that represents the “Pay bill” use case of Restaurant POS 2.0.

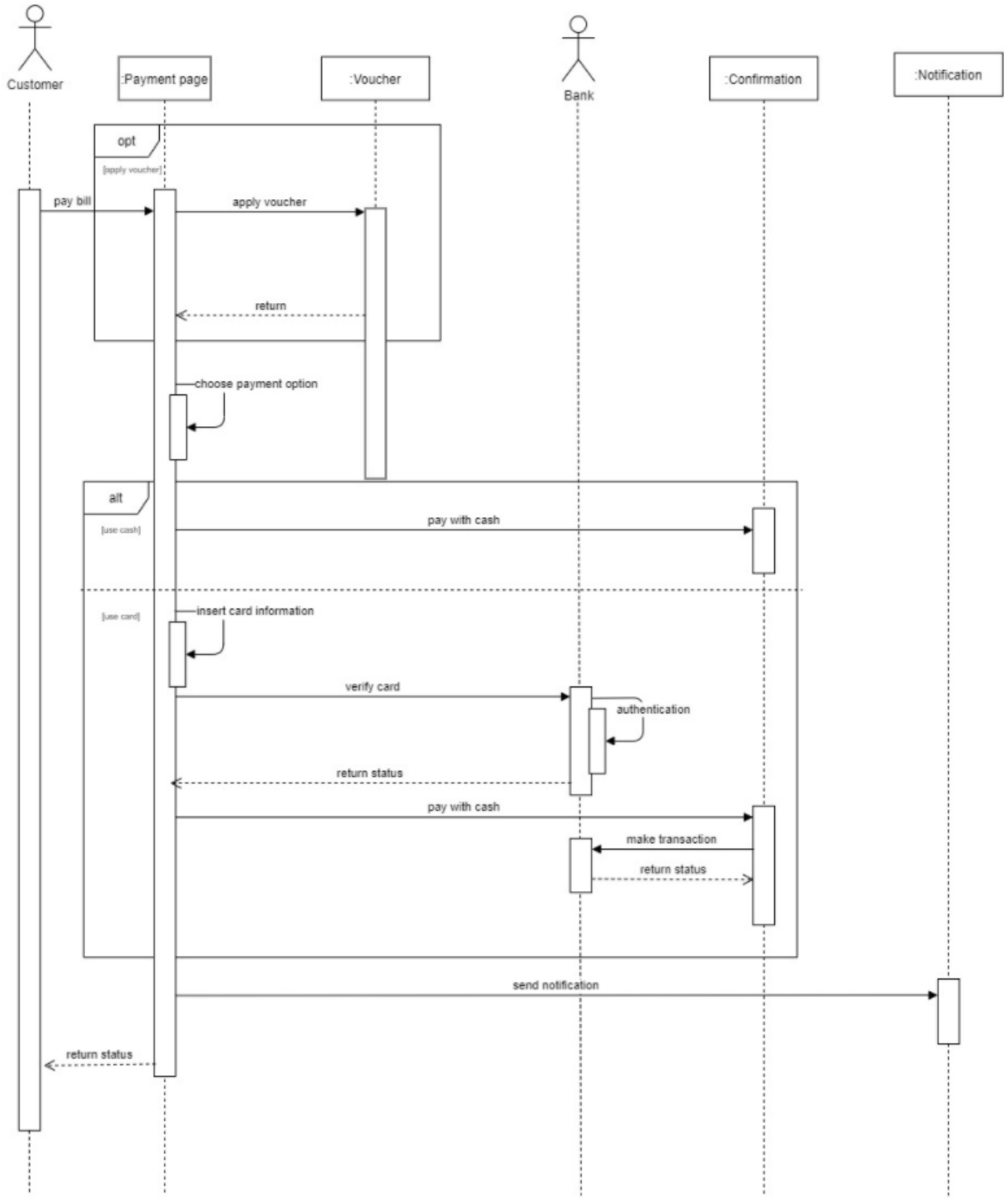

Figure 4: The sequence diagram for the "Pay bill" feature.

VII. Class diagram

Following is the class diagram of the whole Restaurant POS 2.0 system.

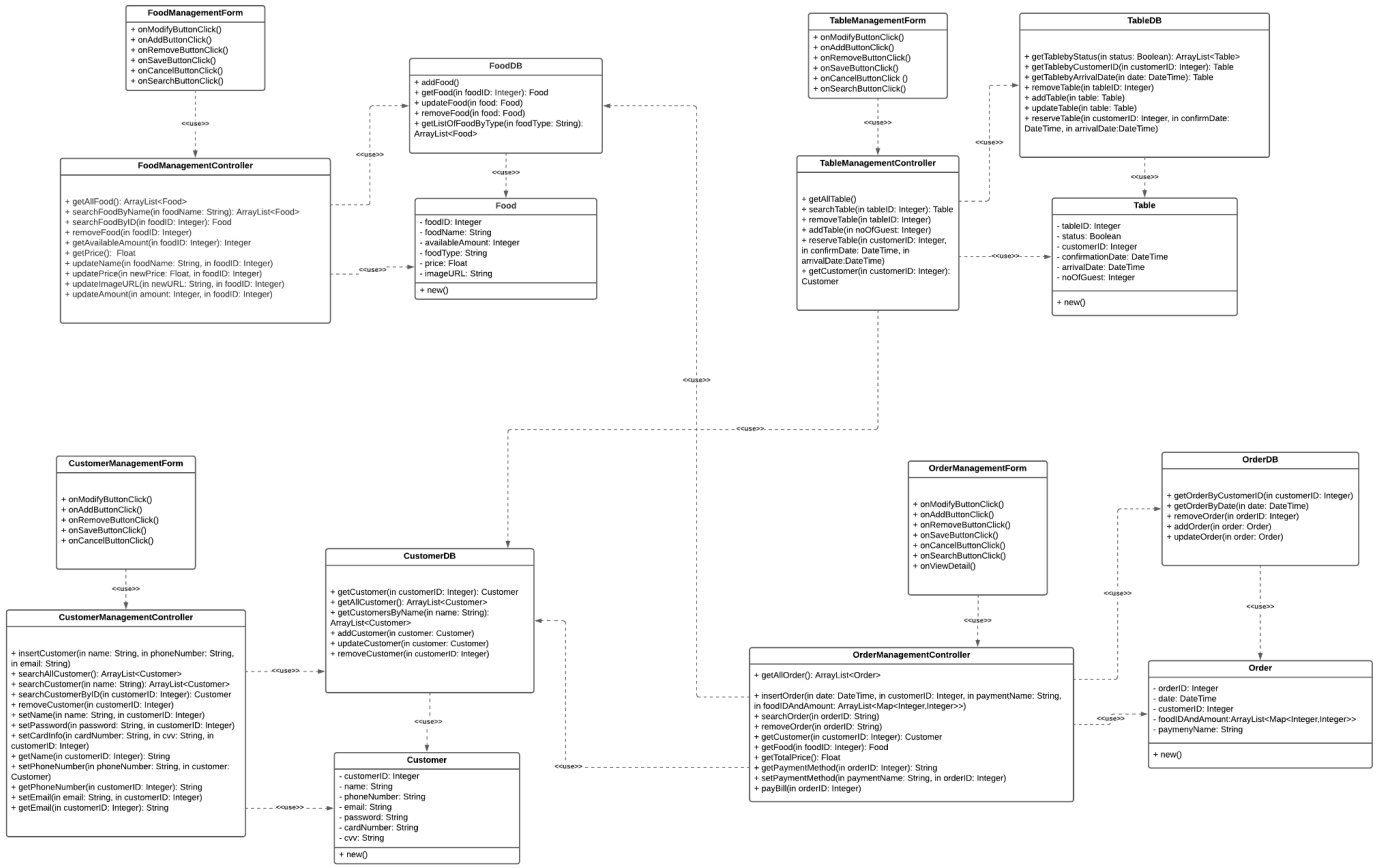

Figure 5: The class diagram of Restaurant POS 2.0.

VIII. Model-View-Controller (MVC)

For the architectural approach of the system, we decided to choose the Model-View-Controller framework. The following parts will supply in-depth definitions, reasons for choosing this framework and some samples.

1. What is a Model-View-Controller Architecture?

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components is built to handle specific development aspects of an application. Their functions can be briefly described as follows:

- The Model component corresponds to all the data-related logic that the user works with.
- The View component is used for all the UI logic of the application.
- The Controller acts as an interface between Model and View components to process all the business logic, incoming requests and to manipulate data using the Model component and interact with the Views to render the final output.

MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects.

2. Reasons for choosing MVC

Firstly, the code of Model-View-Controller (MVC) Architecture is easy to extend and grow, and MVC's components can be tested separately from the user. Secondly, development of the various components can be performed parallelly which means it helps you to avoid complexity by dividing an application into the three

units: Model, View, and Controller. Because it only uses a front controller pattern which processes web application use cases (such as Browse Menu, Update Menu, Pay Bill,...), it offers the best support for test-driven development and it works well for any web applications which are supported by large teams of web designers and developers. Third, all classes and objects (such as Customer, Food, Table,...) are independent of each other; therefore, they can be evaluated separately and independently. Last but not least, MVC Design Pattern allows logical grouping of related actions on a controller together which considerably reduces the size of a web application.

3. **Apply into the POS System.**

a. **Use-case: Browse Menu**

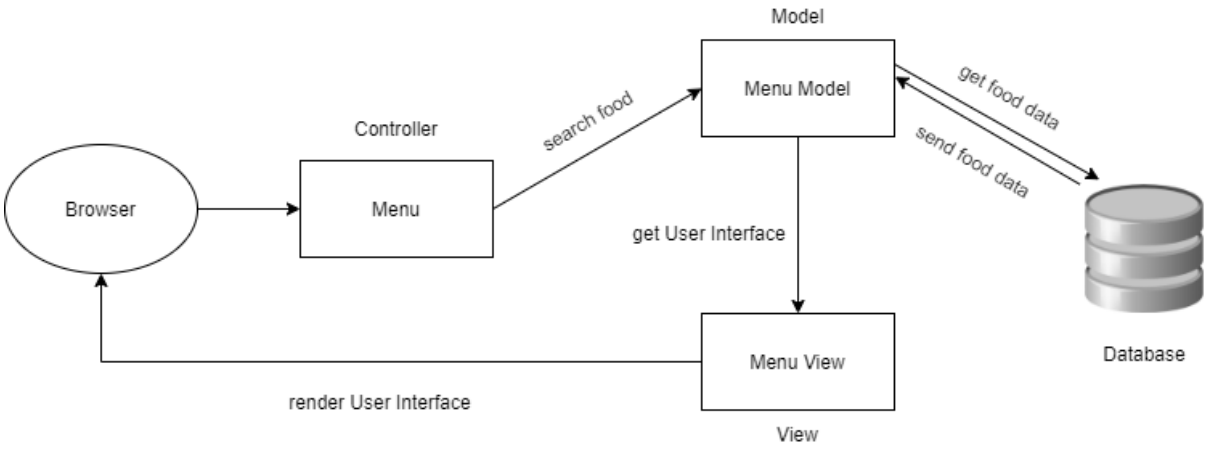

Figure 6: Apply Model-View-Controller Architecture to Browse Menu use case.

Step 1: As customers search for food on the Menu, the Controller component acts as an interface using the Model component to process and the View component to render the final output.
Step 2: The Model component handles the request and sends a query request to the database.
Step 3: The Model component waits for the reply for food data from the database and sends the request to the View component to get the User Interface components (HTML, CSS,...)
Step 4: The View component receives the request and renders the user interface.

b. **Use-case: Update Menu**

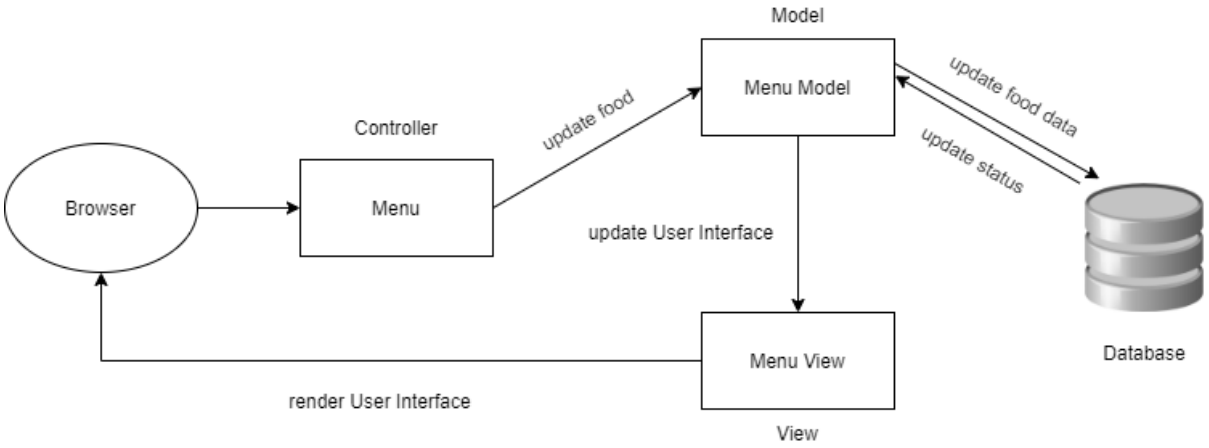

Figure 7: Apply Model-View-Controller Architecture to Update Menu use case.

Step 1: When the staff of the restaurant want to update the food on the Menu, the Controller component acts as an interface using the Model component to query the database and the View component to render the final User Interface.
Step 2: The Model component handles the request and updates (insertion, deletion, or change) the food data from the database.
Step 3: The Model component waits for the "update status" response from the database and sends the request for an update to the User Interface to the View component.
Step 4: The View component receives the request and renders the user interface.

IX. Component Diagram

1. What is a Component Diagram?

Component diagram shows components, provided and required interfaces, ports, and relationships between them. This type of diagram is used in Component-Based Development (CBD) to describe systems with Service-Oriented Architecture (SOA).

Components in UML could represent

- Logical components (e.g., business components, process components), and
- Physical components (e.g., CORBA components, EJB components, COM+ and .NET components, WSDL components, etc.),

along with the artifacts that implement them and the nodes on which they are deployed and executed. It is anticipated that profiles based around components will be developed for specific component technologies and associated hardware and software environments.

2. Apply into POS System

a. Following is the component diagram of Customer Management from Restaurant POS 2.0 system.

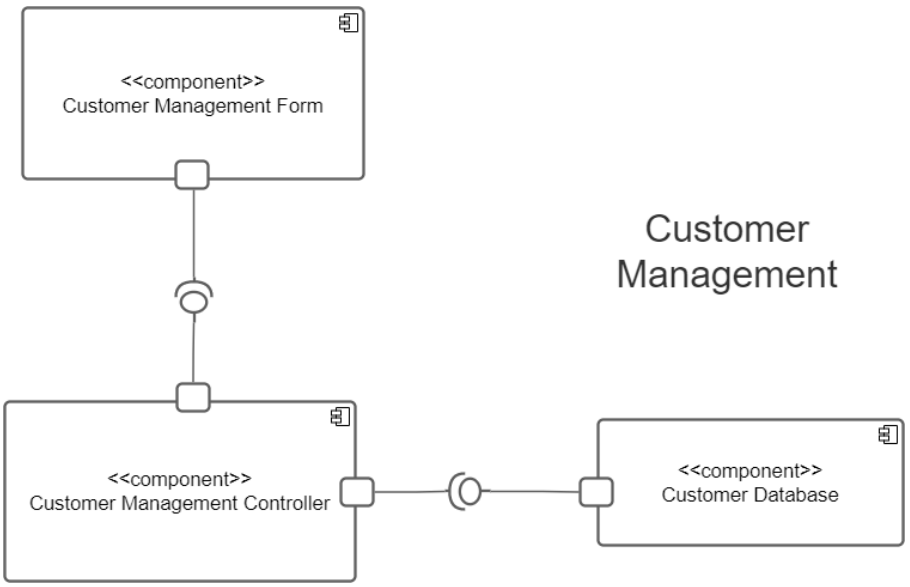

Figure 8: The component diagram of the Customer Management architecture.

b. Following is the component diagram of Menu Browsing from Restaurant POS 2.0 system.

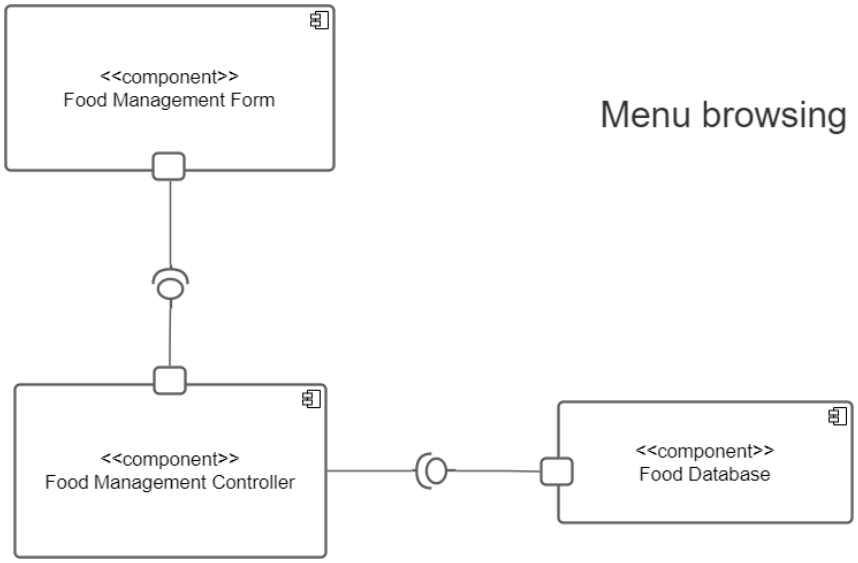

Figure 9: The component diagram of the Menu Browsing architecture.

c. Following is the component diagram of Table Reservation from Restaurant POS 2.0 system.

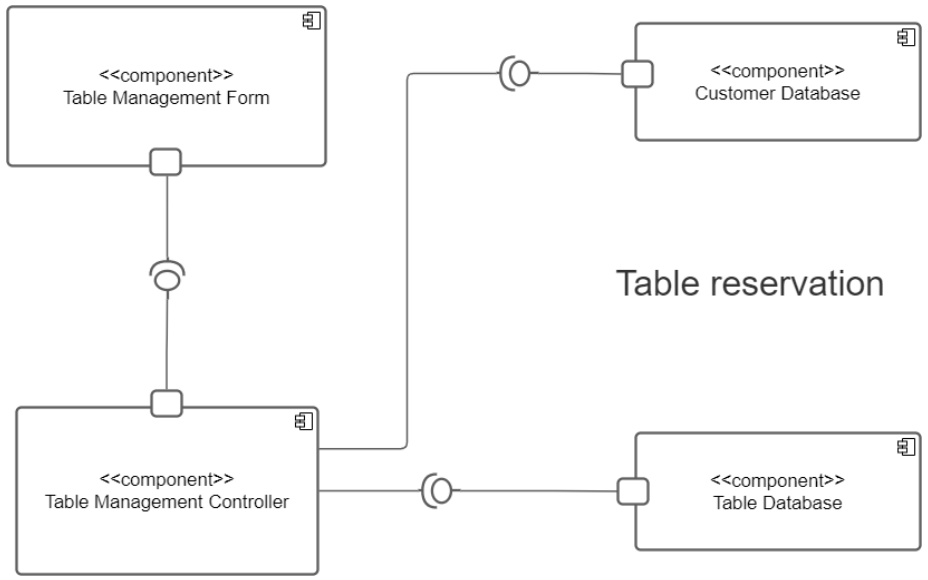

Figure 9: The component diagram of the Table Reservation architecture.

d. Following is the component diagram of Payment from Restaurant POS 2.0 system.

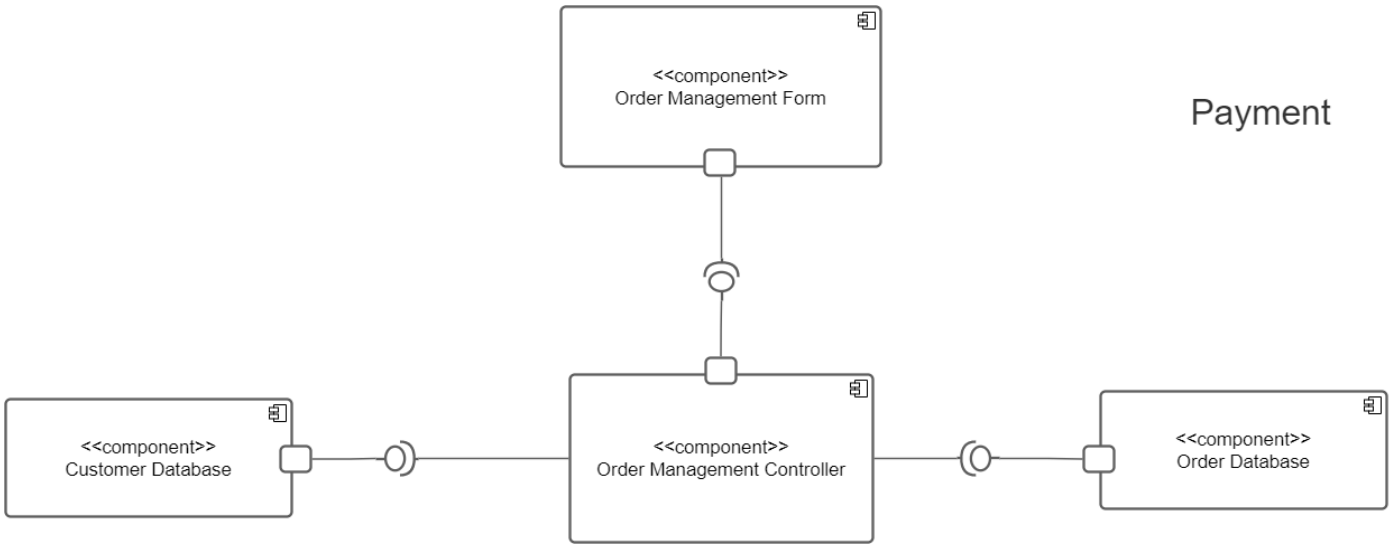

Figure 10: The component diagram of the Payment architecture.

e. Following is the component diagram of Menu Update from Restaurant POS 2.0 system.

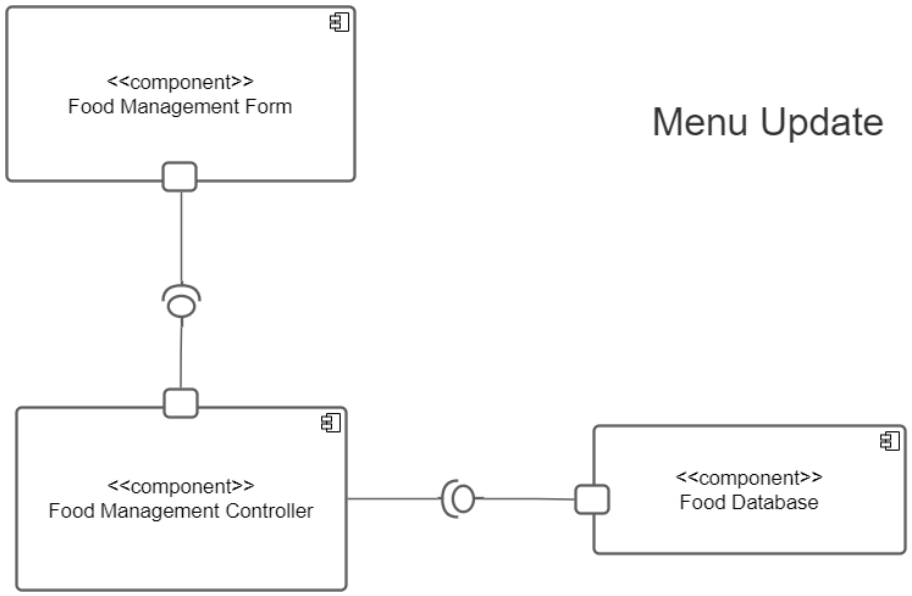

Figure 11: The component diagram of the Menu Update architecture.

f. Following is the component diagram of Food Ordering from Restaurant POS 2.0 system.

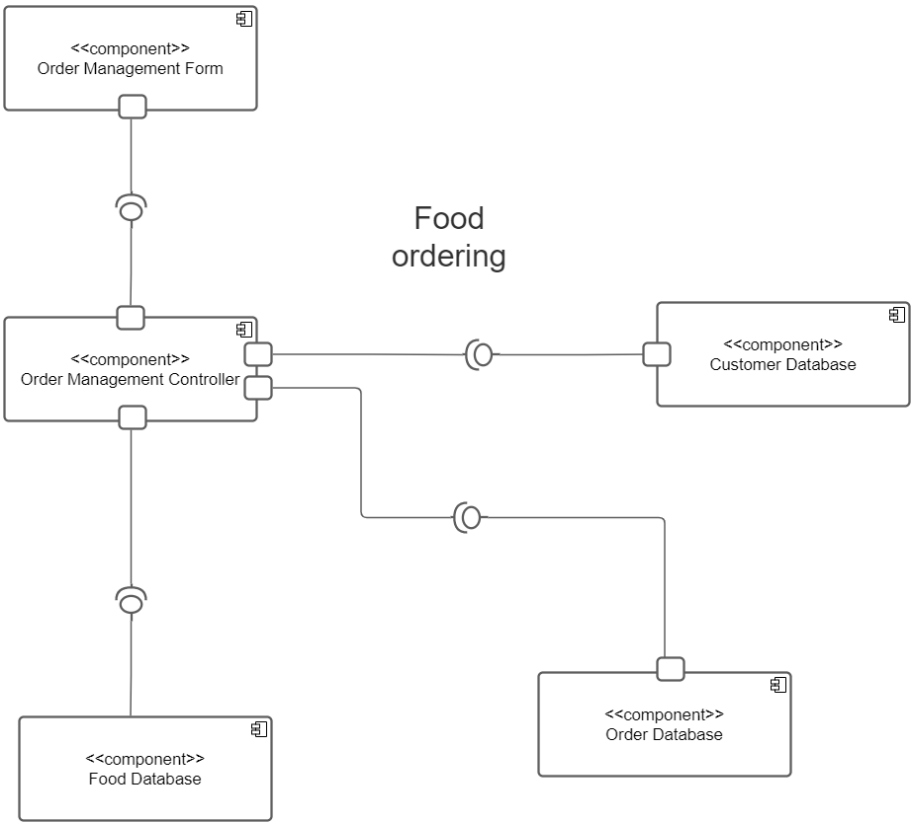

Figure 12: The component diagram of the Food Ordering architecture.

g. Following is the component diagram of Login from Restaurant POS 2.0 system.

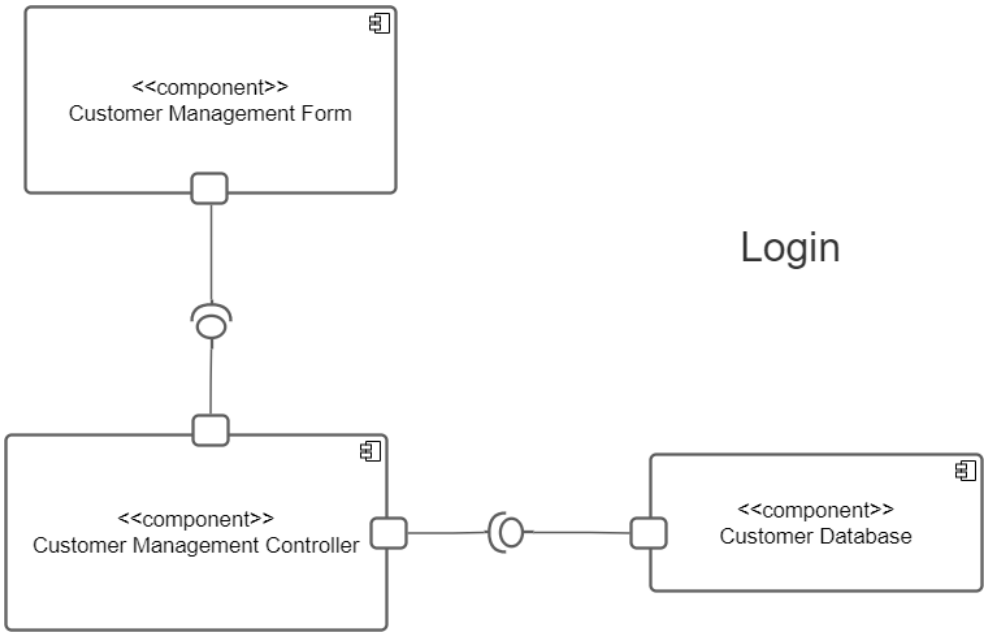

Figure 13: The component diagram of the Login architecture.