BK TP.HCM

CSE

SOFTWARE ENGINEERING
PROJECT REPORT

Point-of-Sale
System Design

This project was proudly done by...

No.	Student ID	Student's name	Email address
1	1952669	Vũ Hoàng Hải	hai.vu.tharios19@hcmut.edu.vn
2	1952088	Lê Nguyễn Tân Lộc	loc.lenguyentan@hcmut.edu.vn
3	1952536	Nguyễn Lê Thảo Vy	vy.nguyen8956@hcmut.edu.vn
4	1952937	Nguyễn Văn Quang	quang.nguyen.0310@hcmut.edu.vn
5	1952785	Nguyễn Lý Đăng Khoa	khoa.nguyen.bk_2019@hcmut.edu.vn

FOREWORDS

The following project is both a documentation and application on a basic point-of-sale terminal that is usually operated on restaurant businesses. It describes the focal point of a point-of-sale system that enables non-direct contact between clerks and customers, using Web technology and QR code, that is responsive enough to implement the current business flow below.

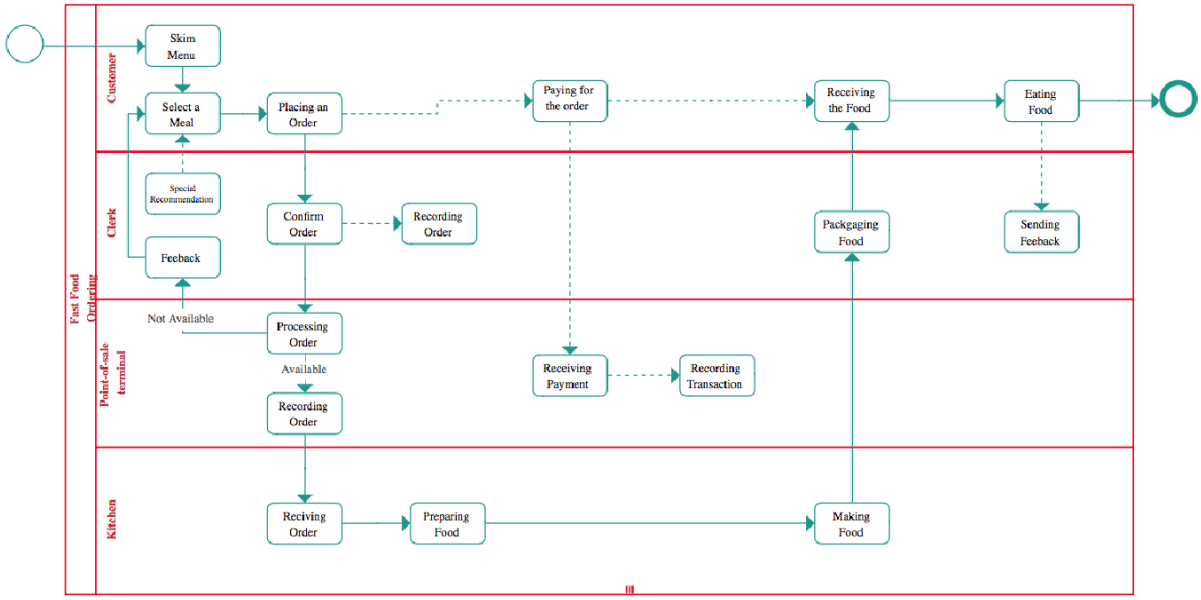


Figure 1: A universal restaurant point-of-sale system.

The proposed system in this project must be accessible from a mobile device, a tablet device or a normal computer, capable of error-free high workload and adaptive to future extensions.

This document is divided into multiple parts that will be progressively stated through the elicitation. Mainly, it will discuss requirement elicitation, system modelling, architecture design and two sprints implementation of the system.

Contents

A Requirement elicitation **3**

1 Project briefing **3**

1.1 Context 3

1.2 Stakeholders 4

1.3 Features 4

1.4 Scopes 4

1.4.1 In scope 4

1.4.2 Out of scope 5

2 Requirements and use-casing **5**

2.1 Functional requirements 5

2.2 Non-functional requirements 8

2.3 System use-case diagram 10

3 Food ordering use-case example **11**

B System modeling **14**

1 Activity diagram **14**

2 Sequence diagram **14**

3 Class diagram **14**

C System architecture **19**

1 Cloud-native architecture pattern **19**

1.1 Architecture foundation 19

1.2 Integration and security advantages 19

2 Functional requirement implementation **21**

D System implementation **23**

1 Github repository **23**

A Requirement elicitation

This section illustrates the contextual foundation idea of the entire project and give insights on how each functional block of the proposed system works. As such, the objectives of the system, the enriching scope, the sub-requirements for each interactive component are discussed in-depth.

1. Project briefing

1.1. Context

This project is targeted to be a web-based point-of-service (POS) system that empowers restaurants administrative with class management tools in order to operate their business efficiently. Apart from the regular dine-in service mode, this system also incorporates Internet-based and QR-code-enabled applications to address the difficulties occurred during the COVID-19 pandemic. It facilitates simplified, straight-forward and contact-less interactions between restaurant customers and staffs, e.g. food ordering, servicing, and payment in any convenience. The application also supports automation of various internal restaurant management tasks, such as order recording, order status managing, transaction recording, etc.

Figure 2: A universal restaurant point-of-sale system.

1.2. Stakeholders

The relevant stakeholders of this proposed system may include human resources such *restaurant owner, managers*, and *staff* (including clerks, chefs and janitors), and external consumerism, e.g. *restaurant customers/clients*.

1.3. Features

The system consists of the following features that enable the complete functionality:

- A *menu screen* for displaying the restaurant's menu; customers can make their selections in this screen.
- A *detail screen* for each food, which will provide pictures and/or related information of a specific food.
- A *cart* that visually display customer's selected food; this screen allows customers to re-check, edit and place their orders.
- A *payment screen* that offers different payments options.
- An *order processing screen* that will be displayed while the food being made; this screen contains status of customer's pending order.
- A *feedback screen* to receive customers' feedback on qualities of the restaurant's food and services.

1.4. Scopes

The central objective of this project is to propose a web-based solution for restaurant owners, to help them maintain the normal operation of their business, while limiting physical / direct interactions between restaurant customers and staff as much as possible. To illiterate in clearer details, the in and out scopes are as follows:

1.4.1. In scope

- A responsive website with attractive UI, categorization, filter and search functionality.
- A back-end system to provide support for multiple restaurants, with the ability of handling approximately 300 transactions per day.
- An online database that supports inventory management.

1.4.2. Out of scope

- Interactive platform to facilitate contactless interactions between restaurant staff.
- Restaurant staff management system.
- Delivery management system.
- Payment processing system.

2. Requirements and use-casing

The following section describes the all functional and non-functional requirements in context of the desired system. A use case diagram is also presented to cohesively illustrate the interactions of the said system.

2.1. Functional requirements

a. **Search and filter menu**:

- The system shall display all available options that the restaurant provides by default.
- The menu screen is visible to all inusers.
- The system shall display detailed information of a selected item.
- The system shall display detailed food categorization to user when needed.
- The system shall enable user to enter search text to the search box.
- The system shall display all matching results based on the search text.
- The system shall enable user to apply one or more filter(s) to the search result.
- The system shall allow user to navigate between the search results.
- The system shall notify user when no matching result can be found.

b. **Account registration (log in/log out)**:

- The system shall require staff members logged in before performing any task. Customers with affiliated membership programs can log in to enjoy exclusive benefits.
- The system shall keep record of all activities of each account.

- The system shall grant different privileges for each specific type of user: customer-type user can place order and have no access to internal information; clerk-type user / kitchen-type user / manager-type user will have permission to edit their position-related information on the system.
- The system shall require customer-type user to enter his / her telephone number and a password in order to create an account.
- The system shall allow manager-type user to create new staff-type account(s).
- The system shall allow manager-type user to edit restaurant menu, modify several types of inventory and view all internal data.
- The system shall allow clerk-type user and kitchen-type user to assign some specific statuses to pending order(s).
- The system shall enable kitchen-type user to modify cooking-related inventory, including ingredients, cooking supplies, kitchen utensils and appliances.

c. **Editorial menu**:

- The system shall only allow manager-type user to edit the menu.
- The system shall allow manager-type user to add / remove item(s) in the menu.
- The system shall allow manager-type user to add / remove / edit related information of any item.
- The system shall support different types of tags to be assigned to items, such as *"new"*, *"hot"*, *"sale"*, etc.
- Each item can have one or more tag(s) at a time.

d. **Update order status / update cooking progress**:

- Every order can have one status at any given time.
- The status of an order is visible to the customer who ordered it, the clerk who is in charge of it and the kitchen staff.
- The system shall allow clerk-type user to assign *"received"* status for newly-submitted order(s); *"not available"* in case of one or more item(s) in an order is unavailable; *"confirmed"* for fully-available order(s) which can be forward to the kitchen and *"completed"* for those that have been successfully finished.

- The system shall allow no change on order(s) marked as *"completed"*.
- The system shall allow kitchen-type user to make some updates on their cooking process, by assigning *"processing"* to orders that are being prepared and *"ready"* to those that are ready to serve / packed.

e. **Manage inventory**:

- The system shall only allow staff-type user (manager-type and kitchen-type) to access inventory information.
- The system shall enable kitchen-type user to add / remove / change the quantity of cooking related items (ingredients; cooking supplies; kitchen utensils and appliances).
- The system shall allow manager-type user to add / remove / change quantity of items that are not taken care by kitchen-type user (service supplies; packing materials; maintenance, repairs and operating goods).
- The system shall notify staff-type user (manager-type and kitchen-type) if the amount of any item in their charge goes below 10.
- The system shall record any change made on inventory information, including who made the change, its timestamp and the change description.

f. **Make an order**:

- The system shall allow customer-type user to select between eat-in, take-away or delivery.
- In case delivery is selected, customer shall be directed to a third party delivery platform.
- The system shall allow customer-type user to add minimum 1 and maximum 30 separate items to his / her cart.
- The system shall allow customer-type user to edit his / her cart.
- The system shall calculate and display the total price of the order.
- The system shall allow customer-type user to place his / her order.
- The system shall notify online clerk-type user when a new order is placed.
- The system shall notify kitchen-type user when a new order is forwarded to be cooked.
- The system shall notify customer-type user once his / her order is marked as *"completed"*, display the invoice and ask for his / her feedback.

g. **Make payment**:

- The system shall allow customer-type user to choose one from a list of online payment methods to proceed.
- The system shall forward customer-type user to his /her selected payment platform.
- The system shall check whether the transaction is made successfully and notify clerk-type user.

h. **Give / display feedback**:

- The system shall allow customer-type user to give feedback on food and service quality.
- The system shall display the feedback of every item.
- The system shall enable user to filter the feedback of each item by date and rating.
- The system shall allow customer-type user to edit his / her feedback.

i. **View and manage business data**:

- The system shall keep record of every completed invoice.
- The system shall allow manager-type user to access business data and statistics.
- The system shall allow manager-type user to filter the restaurant business data and sort them as needed.
- The system shall display an overall summary for statistics for manager-type user if needed.

2.2. Non-functional requirements

- The server is kept to be always online to ensure synchronization. Active hours includes from 6am to 10pm, customers can use all functions during the period. After those hours, customer can make reservation in advance.
- The website should have a consistent scale, design, look, and responsiveness on different devices and offers easy to use and have instructional videos for customers.
- Maintenance to the system is performed every 4 months. Each periodic and progressive system upgrades should not last more than 30 minutes and happen in the hour that has the least customer traffic to avert unintended congestion.

- UI/UX should be modern and user-friendly, system respond time to users is allowed no more than 3 seconds.
- System must be decentralized, users have their unique permission to access different data. Customer databases must be encrypted end-to-end.
- The system e-invoice must follow the legal government laws of creating and using online invoices, as well as cyber security law. (*Circular No. 32/2011/TT-BTC, dating March 14th, 2011, and decree 119/2018/ND-CP dating September 12th, 2018*)

2.3. System use-case diagram

Figure 3 shows the use-case diagram representation for the entire system.

Figure 3: Overall system use-case of a proposed restaurant point-of-sale.

3. Food ordering use-case example

Figure 4 shows an in-depth use-case diagram of a *food ordering* system feature.

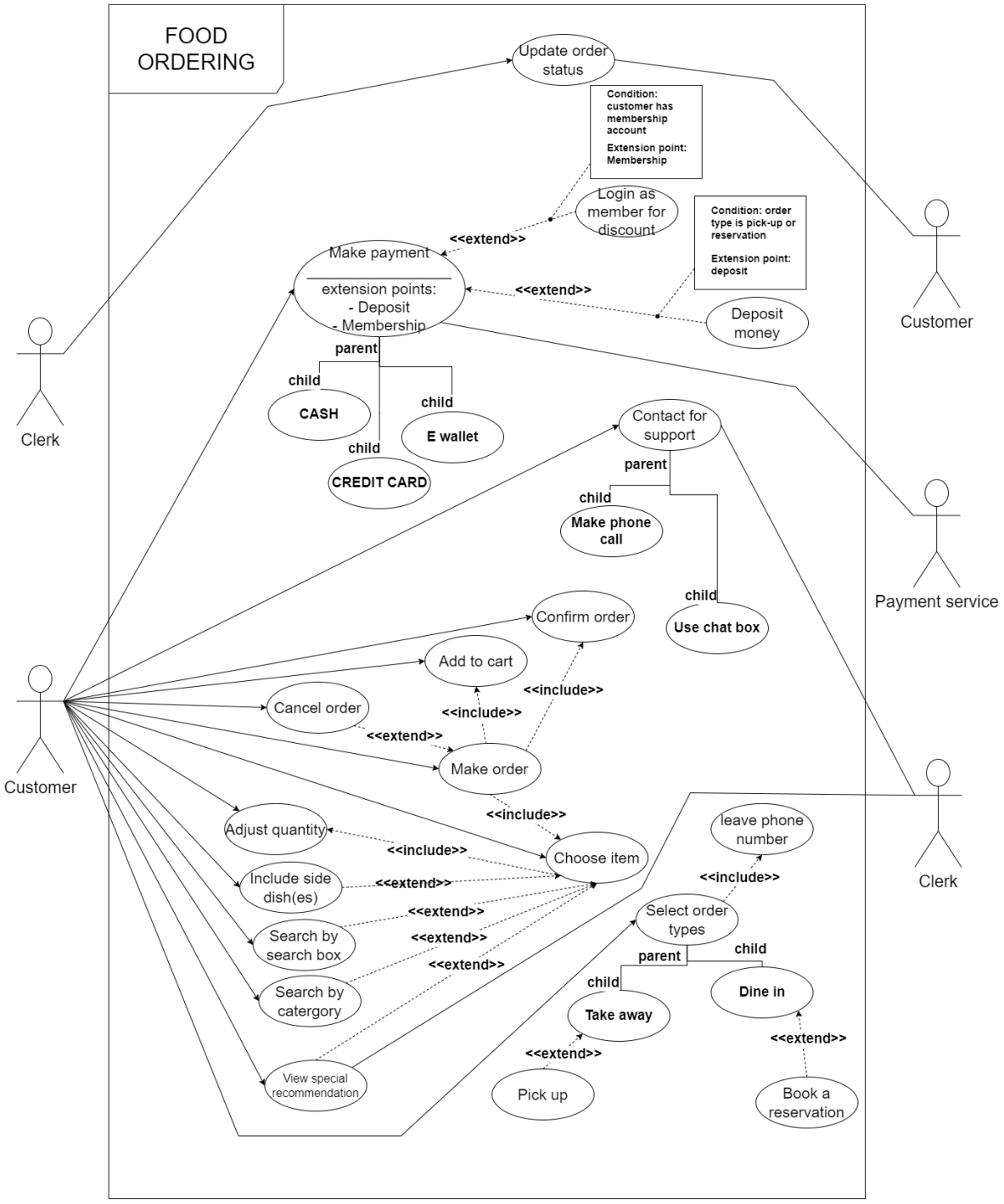


Figure 4: Food ordering use-case of a proposed restaurant POS feature.

In order to dissect this diagram in a friendly manner, the following table 1 will describe the use-case in details:

Use-case description	A feature of POS system that aims to help customer get access to all available services of the restaurant including make food order with different order types (dine-in, take-away), select meals via multiple searching methods (catergory searching, search by name) or view special recommendations of the restaurant. To ask for assistance, customer may contact the clerk via chat box or direct phone call. Further more, the feature supports multiple payment methods apart from cash such as credit card, E wallet.
Primary actors	Customer and Clerk.
Secondary actors	Customer, Clerk and Payment service.
Pre-condition	Customer must have access to the web page by scanning provided QR code or through Internet hyperlink.
Post-condition	The customer payment is verified and they have had their food served.
Main flow	1. Customers access the webpage by scanning the QR code or by Internet hyperlink. 2. Customers choose to dine-in without booking reservation in advance. 3. A menu pops up with different combinations of meals and drinks. 4. Customers select one of the displayed choice or search for a specific one via a search box or they can view some special recommendations from the restaurant in a window. 5. After choosing the meals, customers add them to cart. 6. Customers confirm their order. 7. Customers pay for their order via credit card, e-wallet, or direct cash. 8. Customer leaves phone number. 9. Once food is ready to serve, the system will inform the customer by their phone number SMS or notification.

Alternative flow	1. Customers access the webpage by scanning the QR code or by Internet hyperlink. 2. Customer books a table reservation in the restaurant. 3. A menu pops up with different combos of meals and drinks. 4. Customer selects one of those displayed, search for a specific one via a search box or they can view some special recommendations from the restaurant. 5. After choosing the meals, customer adds them to cart. 6. Customer confirms the order. 7. Customer deposits money for the table reservation. 8. Customer leaves phone number. 9. The system will send SMS or notification to customer 15 minutes before the booked time.

Table 1: Food ordering glossary

B System modeling

This section will perform laid-out diagrams to capture the major functional requirements of the desired restaurant POS system. It is divided into three caconicals of an activity diagram, a sequence diagram and a class diagram.

1. Activity diagram

Figure 5 shows an activity diagram of a restaurant POS that condenses the major functional requirements of the entire system.

2. Sequence diagram

Figure 6 shows a sequence diagram of a restaurant POS that has object interactions arranged in predefined time sequence.

3. Class diagram

Figure 7 shows a class diagram of a restaurant POS. This depicts the structure of the system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects declared in **Section A**.

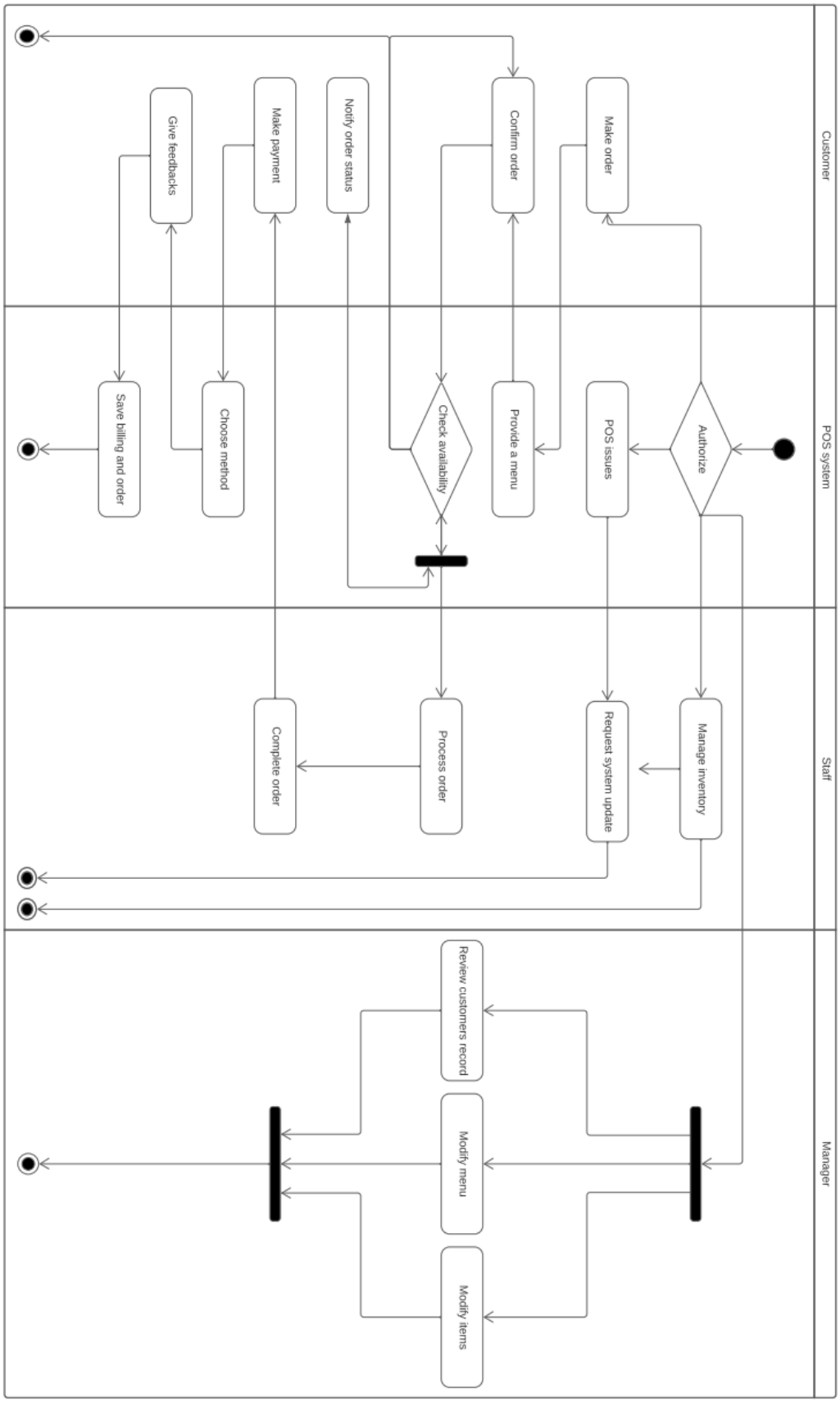
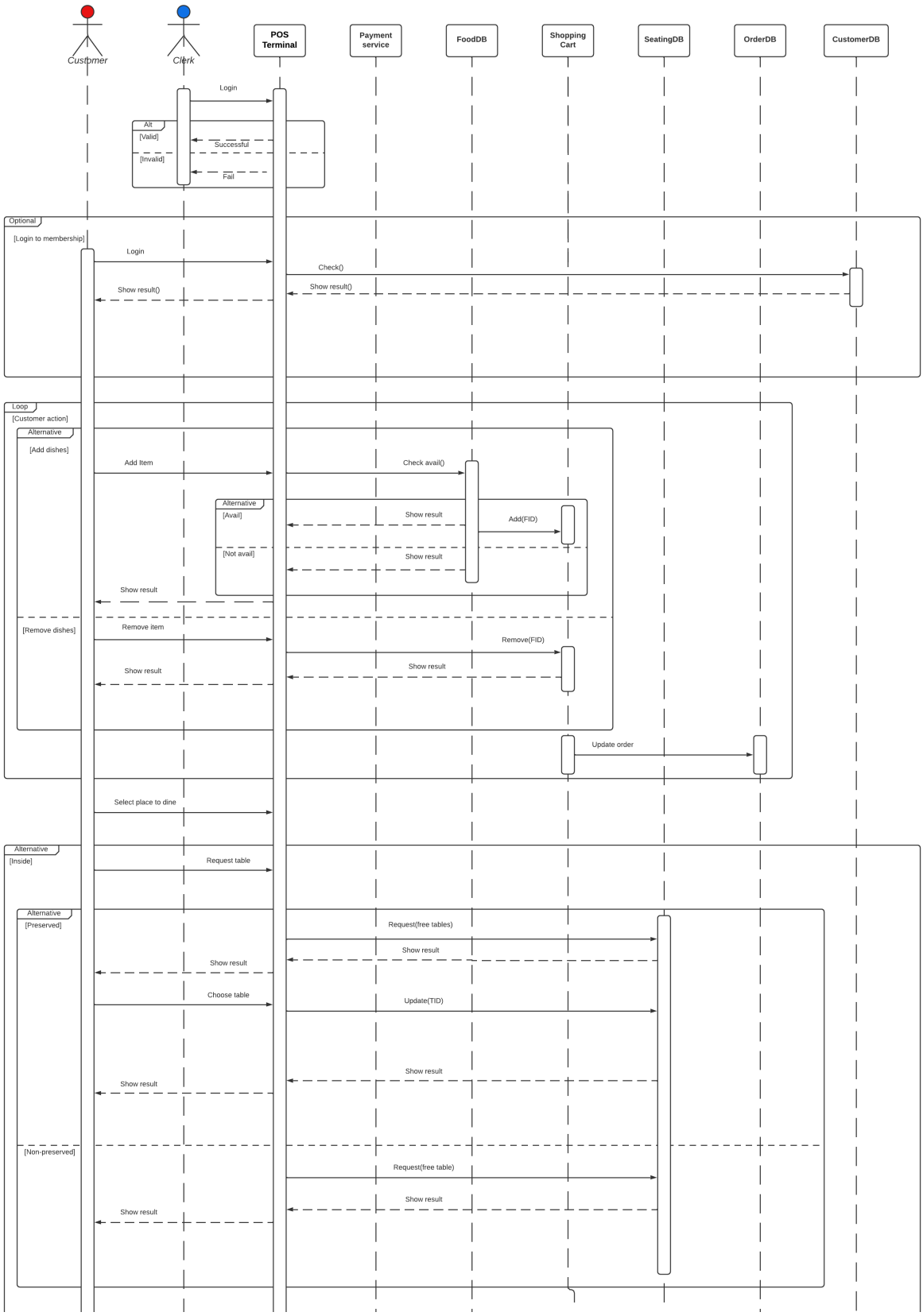


Figure 5: Activity diagram of a restaurant POS.


Customer
Clerk
POS Terminal
Payment service
FoodDB
Shopping Cart
SeatingDB
OrderDB
CustomerDB
Login
Alt
[Valid]
Successful
[Invalid]
Fail
Optional
[Login to membership]
Login
Show result()
Check()
Show result()
Loop
[Customer action]
Alternative
[Add dishes]
Add Item
Check avail()
Alternative
[Avail]
Show result
Add(FID)
[Not avail]
Show result
Show result
[Remove dishes]
Remove item
Remove(FID)
Show result
Show result
Update order
Select place to dine
Alternative
[Inside]
Request table
Alternative
[Preserved]
Request(free tables)
Show result
Show result
Choose table
Update(TID)
Show result
Show result
[Non-preserved]
Request(free table)
Show result
Show result

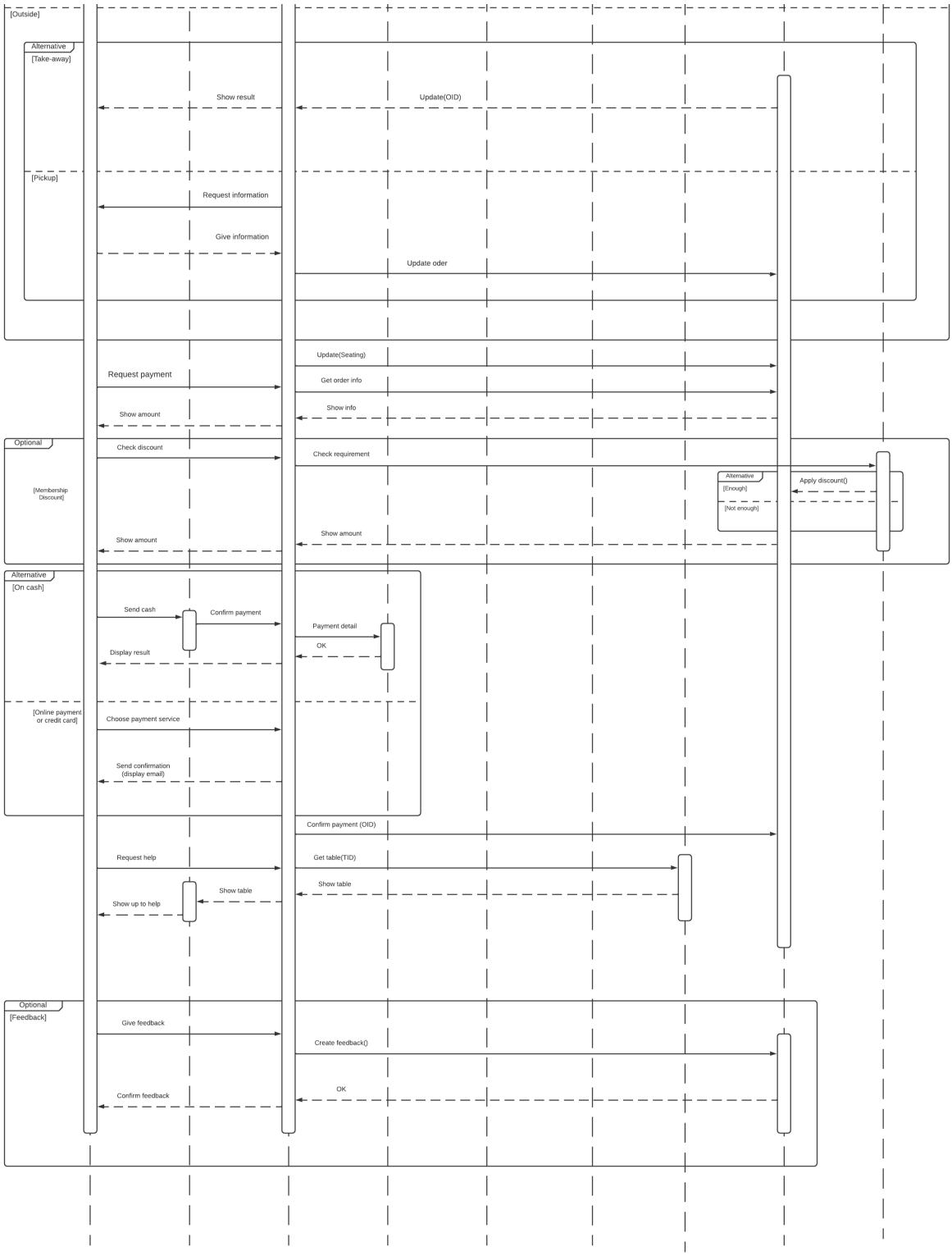


Figure 6: Sequence diagram of a restaurant POS.

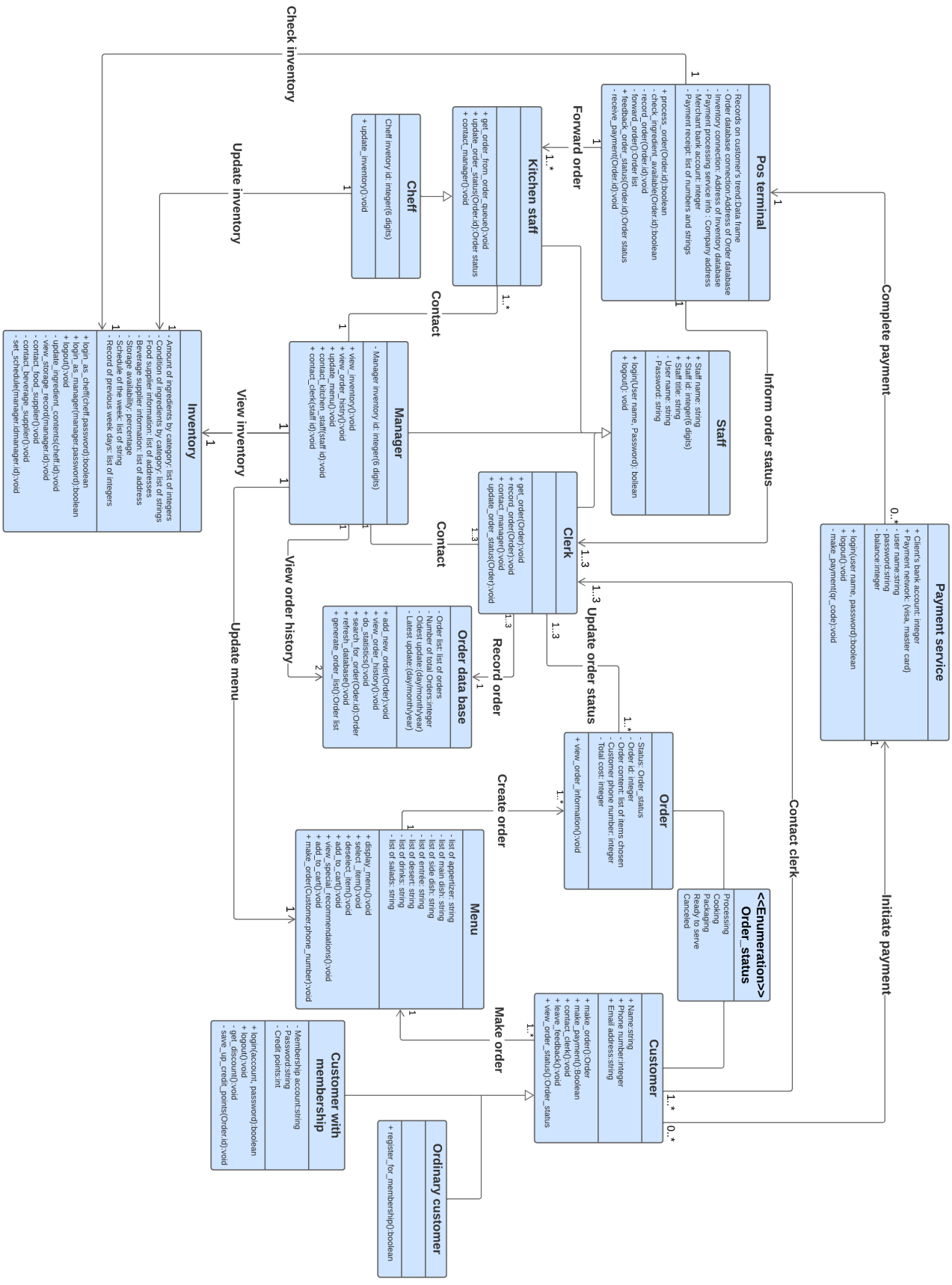


Figure 7: Class diagram of a restaurant POS.

C System architecture

A fundamental approach to visualize the structural integrity of the entire restaurant point-of-sale system presented in this project is via a cohesive blueprint of the overall architecture. In this section, all of the foundational components of the application are described in detail, with reasoning of choice and facilitation. They are then described further in implementation diagrams to better illustrate the said functional requirements in **Section A**.

1. Cloud-native architecture pattern

There are numerous architectural patterns that capture the design structures of various systems and elements of software so that they can be reused. For this project's system, to fully utilize the power of cloud-based ordering point-of-sale, using *application programming interfaces (APIs)* instead of a fixed model, a much more advanced version of *Model-View-Controller (MVC)* pattern is needed.

1.1. Architecture foundation

Cloud-native architecture is a suitable upgrade for MVC. It allows dynamic and agile application development techniques that take a modular approach (API with user-interfaces and services) to building, running, and updating software through a suite of cloud-based microservices and containerization with agility and dynamism versus a monolithic application infrastructure.

The entire architecture is depicted as a triangular representation in **Figure 8**. This is a close-knit relationship (represented by outer connecting edges) between sub-API gateways: **delegate** for the major operative branches of the restaurant system, **online** for activities done over the Internet, and **management** for internal affairs. Each of these has their own separate sets of user-interfaces, services and database buffers to handle tasks independently. Communication is established to the central core API to perform requests made from the sub-APIs and to keep synchronization between sub-API databases and the central database.

1.2. Integration and security advantages

It can be seen from the network the central database can only be accessed via the core API. This proposed architecture provides an efficient security layer that separates the master storage from

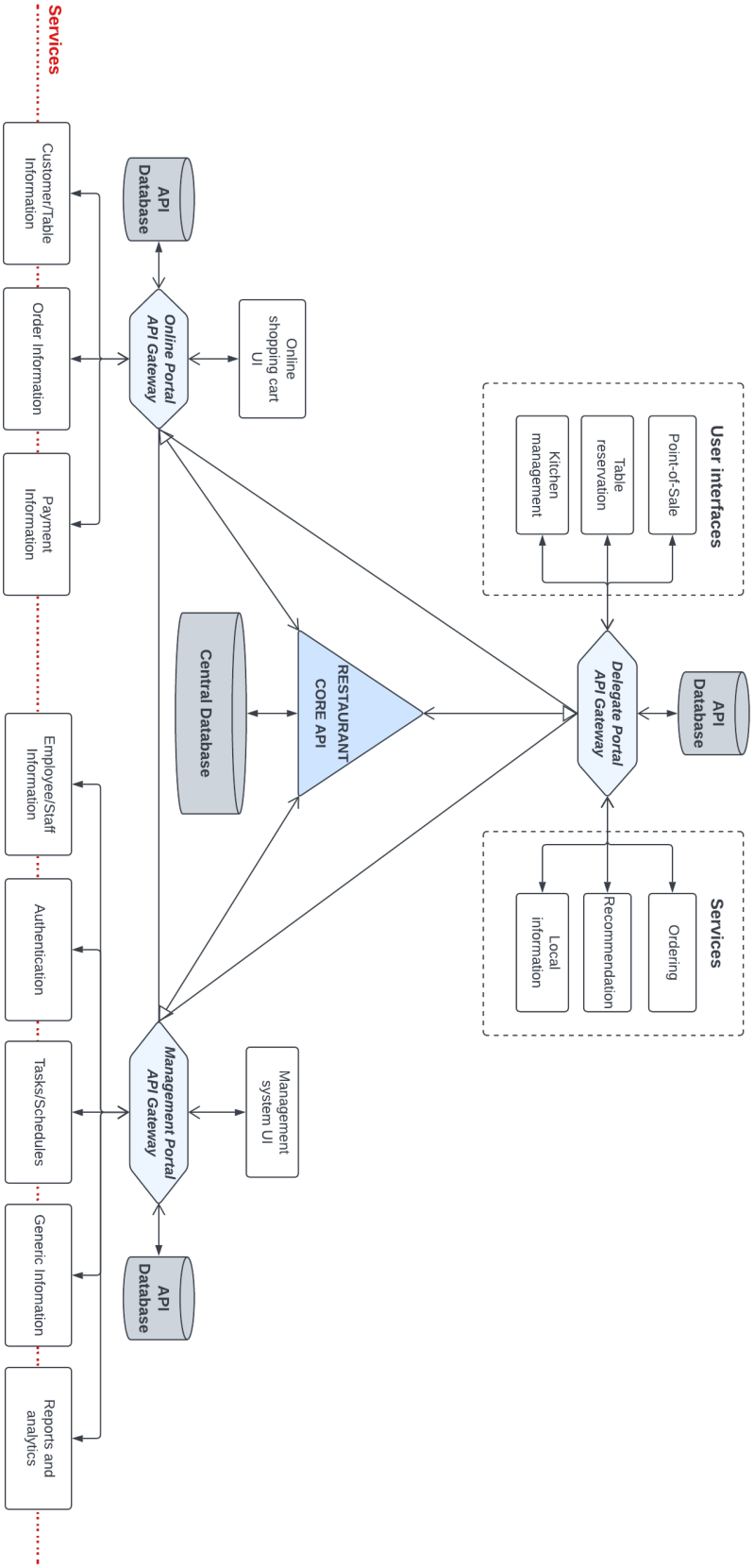


Figure 8: Cloud-native architecture of a restaurant POS.

the rest of the API network. Provided that one or more sub-API experience malfunction or undergoes maintenance/software updates, only the branch of that API is disabled while not affecting the rest of the network. If the central API is turned off temporarily, the local sub-API databases are utilized to store their own data and sends synchronization signal to other sub-databases to keep everything updated while access to the core is limited.

It is virtually impossible for external attempts to directly access the database without an appropriate sub-API intervention. Regardless of compromises from sub-API and its own database, it will not affect other nodes.

In addition to security benefits, the architecture can be expanded to accompany more sub-APIs in the future while keeping a cohesive footprint of the nodes in the network. It is even more efficient for interaction between multiple core APIs, e.g. the restaurant system to other networks of restaurants (layered cloud-native), providing wider business opportunities.

2. Functional requirement implementation

The following deployment diagram models the physical deployment of artifacts on nodes, in this case, the topology of the POS system outlines six logical components: website package, customer management, order, menu, payment and seating requirements.

When a user executes an order on the website, their operations are performed in a packaged online environment. One can visit the website, browse through a list of menus, make order and seating reservation, while optionally choose to sign up or sign in to their account. After everything is completed, the payment system shall be activated and can access the customer database if prompted.

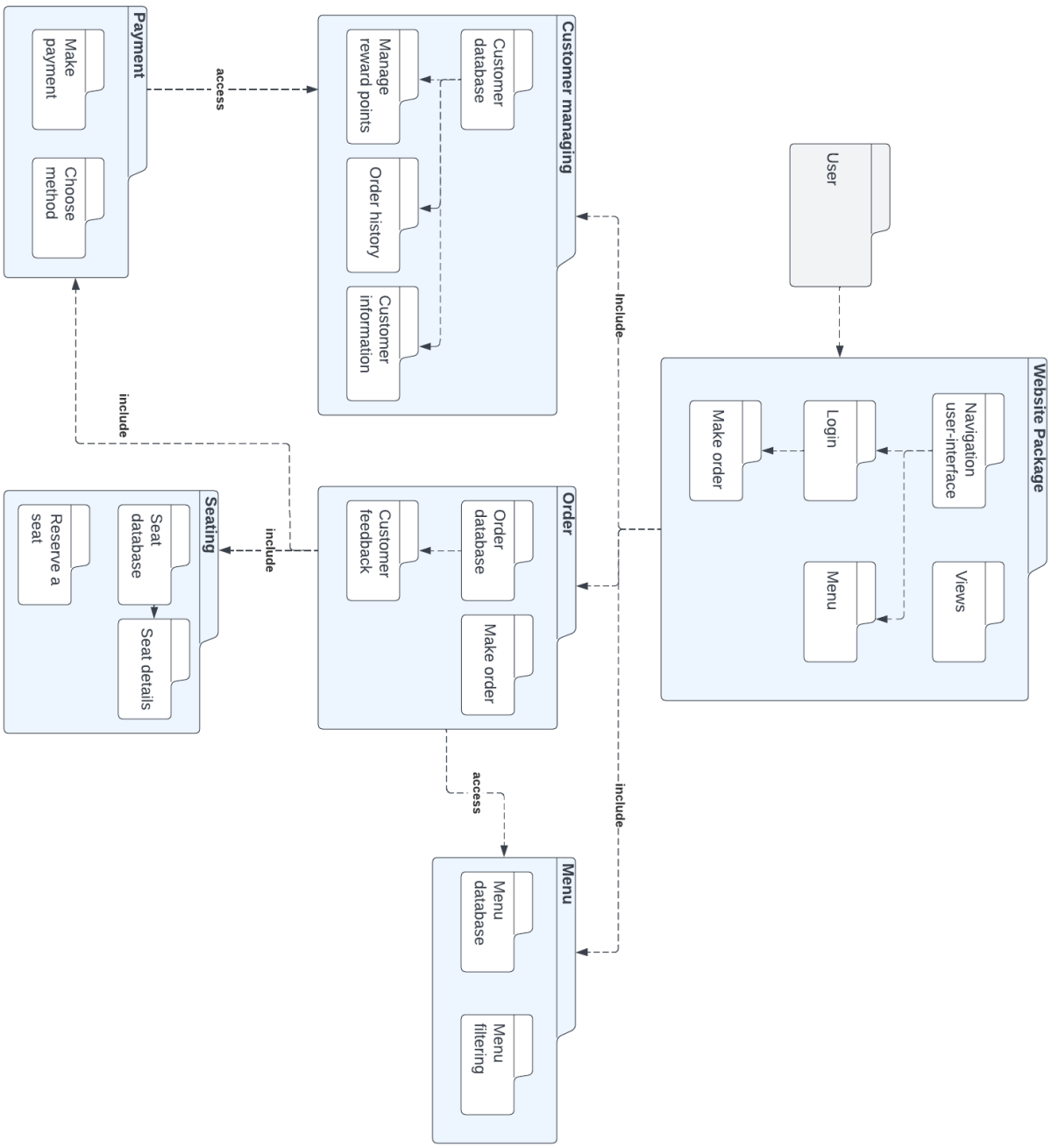


Figure 9: Deployment graph of a restaurant POS and its functional requirements.

D System implementation

This following section demonstrates how our point-of-sale system will be built from the elements instated in the previous chapters. The entire implementation will involve three stages: a front-end, back-end and the APIs that we have declared in the **System Architecture** chapter. These will connect to the cloud core structures and each version of implementation is controlled via a Github repository.

1. Github repository

In order to keep track of the project's underneath updates from multiple sources, Github is an efficient choice for version controlling the entire system. Collaboration is also a seamless work when members can publish and review versions of each other.

The project Github repository is at the following link, please use your designated web browser to view. If there is any trouble accessing the link, please contact us with our members email stated at the beginning of this project's report.

https://github.com/vy-nguyenlethao0510/HCMUT_CSE_POS_2.0

The structure of this repo is described in the figure below:

End of report part 3

This page is intentionally left blank