Project: Virtual Waiter

**Mục lục**

[Chapter 1. Project description 4](#_Toc179902421)

[1.2 Problem description 4](#_Toc179902422)

[1.3 Business requirement 4](#_Toc179902423)

[1.3.1 Business rule 4](#_Toc179902424)

[1.3.2 Business analysis 4](#_Toc179902425)

[1.4 System’s actor 5](#_Toc179902426)

[1.5 Functional requirement analysis 6](#_Toc179902427)

[1.5.1 System Usecase Diagram 6](#_Toc179902428)

[1.5.2 Use case specification. 8](#_Toc179902429)

[1.6 Non-functional requirement 12](#_Toc179902430)

[Chapter 2. System Designing 13](#_Toc179902431)

[2.2 System Structure Modeling 13](#_Toc179902432)

[2.2.1 Receptionist’s Usecases: 13](#_Toc179902433)

[2.2.2 Customer’s Usecases: 13](#_Toc179902434)

[2.2.3 Kitchen’s Usecases: 14](#_Toc179902435)

[2.2.4 Usecase Waiter Update order 15](#_Toc179902436)

[2.3 Interaction Modeling 15](#_Toc179902437)

[2.3.1 Usecase Receptionist check-in 15](#_Toc179902438)

[2.3.2 Use case Xem thông tin hộ khẩu 15](#_Toc179902439)

[2.3.3 Usecase Chỉnh sửa thông tin hộ khẩu 15](#_Toc179902440)

[2.3.4 Usecase Xóa nhân khẩu 15](#_Toc179902441)

[2.3.5 Usecase Xem lịch sử thay đổi nhân khẩu 15](#_Toc179902442)

[2.3.6 Usecase Thêm nhân khẩu 15](#_Toc179902443)

[2.3.7 Usecase them hộ khẩu 15](#_Toc179902444)

[2.3.8 Usecase xem thông tin nhân khẩu 15](#_Toc179902445)

[2.3.9 Usecase thống kê 15](#_Toc179902446)

[2.3.10 Usecase Quản lý tạm trú 15](#_Toc179902447)

[2.3.11 Usecase quản lý tạm vắng 15](#_Toc179902448)

[2.3.12 Usecase Tách hộ khẩu 16](#_Toc179902449)

[2.4 Kiến trúc tổng thể của hệ thống 17](#_Toc179902450)

[2.4.1 Tầng trình bày 17](#_Toc179902451)

[2.4.2 Tầng ứng dụng 18](#_Toc179902452)

[2.4.3 Tầng nghiệp vụ 18](#_Toc179902453)

[2.4.4 Tầng dữ liệu 19](#_Toc179902454)

[Thiết kế chi tiết lớp 19](#_Toc179902455)

[2.4.5 Sơ đồ gói 19](#_Toc179902456)

[2.4.6 Gói Model 19](#_Toc179902457)

[2.5 Thiết kế dữ liệu 22](#_Toc179902458)

[2.5.1 Sơ đồ bảng dữ liệu 22](#_Toc179902459)

[2.6 Thiết kế giao diện 23](#_Toc179902460)

[2.6.1 Giao diện màn hình chính 23](#_Toc179902461)

[2.6.2 Chức năng thêm hộ khẩu 24](#_Toc179902462)

# Project description

## Problem description

A large restaurant needed to serve hundreds of customers at the same time. This requires processing multiple orders, servings, checkouts at the same time. It’s recommended to use an Information System to help customers to self-order, kitchen to monitor and prepare, employees to confirm orders and easily proceed checkout.

## Business requirement

### Business rule

Customer able to see and order dishes at the table through mobile devices correlated with the table.

* Dishes will have: Name, description, picture, price, size, discount.
* Discount will have 2 types: percentage discount on singular dish and combo discount. Percentage discount will automatically apply when customer select the food while combo discount will recommend customer whenever they select a dish that is in any discount combo.

Kitchen will see customer’s order and prepare the dish. Confirm that the dish is ready and waiters can serve the dish to the customer.

* Orders will have the dish ID, display name, size and number, table
* Kitchen can cancel the meal if a customer wish and they haven’t started on preparing

Waiters take the dish to serve, then confirm that the dish is served to customer.

Receptionist when check-in for customer, can check whether the tables are available and setup the tables

When check out, the system will calculate the total price and receptionist can proceed on payment.

### Business analysis

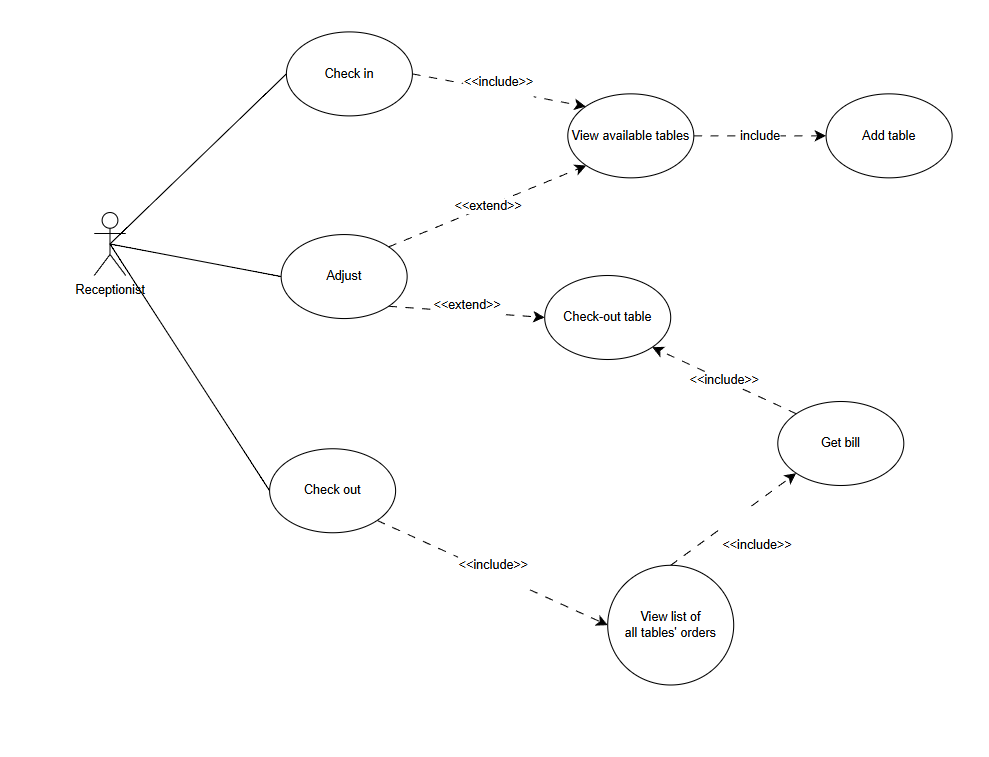
1. Customer can use the mobile device to order at the table. What they can do:
   * Browse through the list of dishes
   * Select the dish, amount, size and order
   * Check the dishes that they have ordered and their status (preparing, served)
   * Cancel the dish that they ordered, as long as the kitchen confirm that they can.
2. Kitchen can check the pending orders and set the status of the dish to preparing, ready or canceled:
   * Update the status of the order to preparing, ready, canceled.
3. Waiters take completed dishes from the kitchen and serve to customers, then update the status in the order to served.
4. Receptionist can organize tables, check-ins, check-outs:
   * Check whether any table available
   * Check-in for customer and setup the active table on the system
   * Change table for customer if request
   * Check-out for customer and calculate the bill.

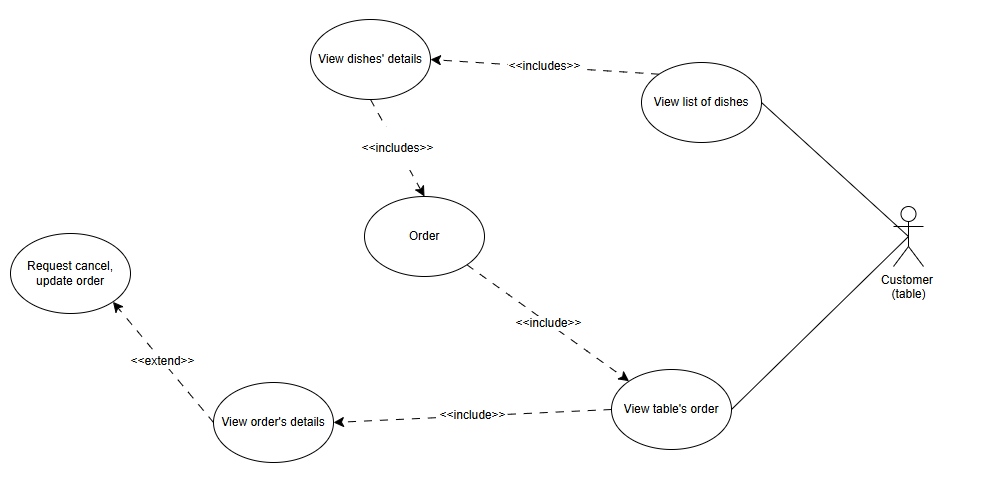
## System’s actor

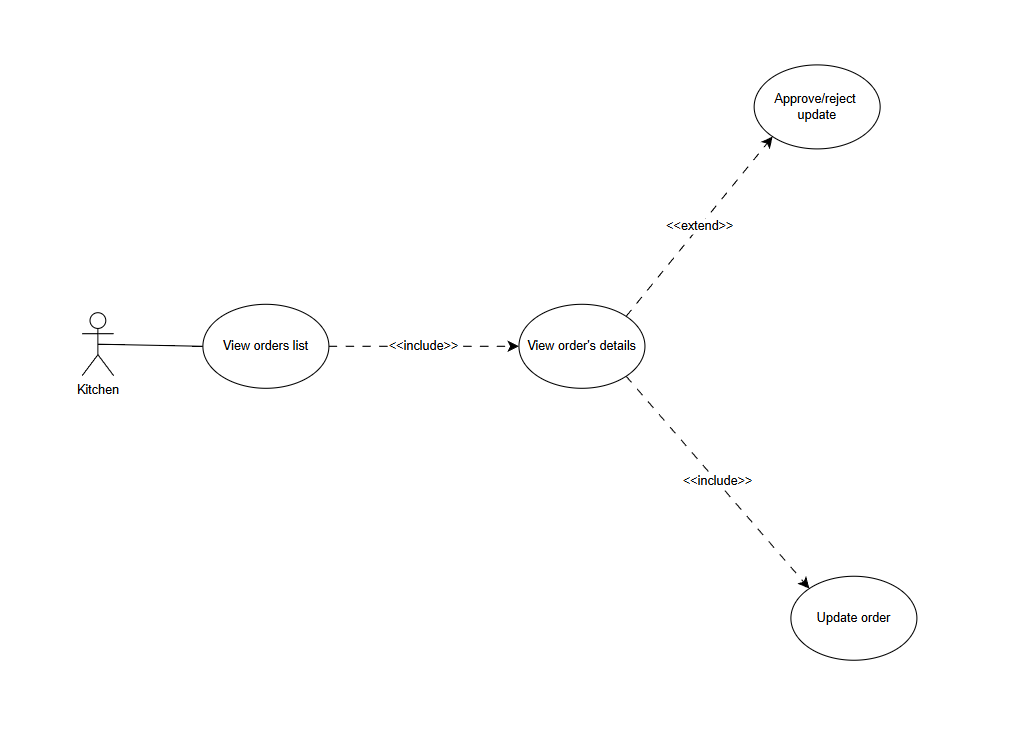
The system has 4 actors: Receptionist, Customer (mobile device on table), Kitchen, Waiter.

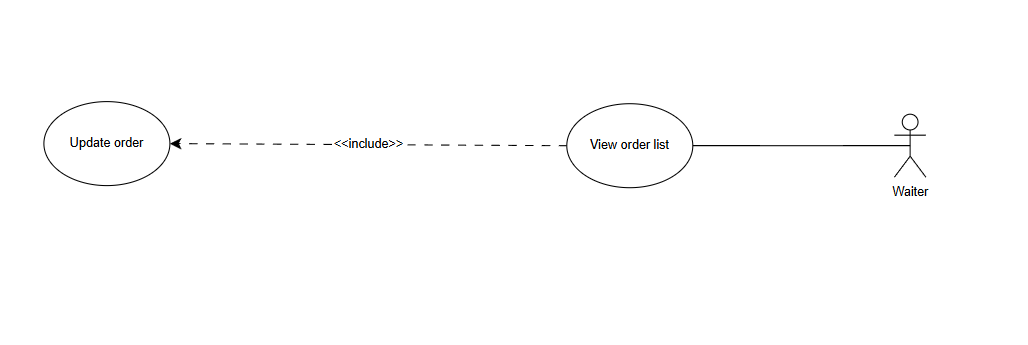
## Functional requirement analysis

### System Usecase Diagram









### Use case specification.

#### Usecase Receptionist Check-in

|  |  |  |  |
| --- | --- | --- | --- |
| **Usecase code** | UC01 | **Usecase Name** | Receptionist Check-in |
| **Actor** | Receptionist | | |
| **Goals** | Check-in customer, set up tables in the name of the customer | | |
| **Triggers** | Receptionist select Check-in | | |
| **Preconditions** | None | | |
| **Basic flow**  **(Success)** | |  |  |  | | --- | --- | --- | | **No.** | **Proceed by** | **Action** | | 1 | Receptionist | Select available tables, enter customer name | | 2 | System | Create a record for the customer | | 3 | System | Set tables as Being used | | 4 | System | Set correlation between tables and the customer record | | 5 | System | Announce succeed | | | |
| **Alternative flow** | b. Receptionist cancel. System goes back to homepage | | |
| **Post-conditions** | None | | |

#### Usecase Receptionist Adjust (remove)

|  |  |  |  |
| --- | --- | --- | --- |
| **Usecase code** | UC02 | **Usecase Name** | Receptionist Adjust (remove) |
| **Actor** | Receptionist | | |
| **Goals** | Change customer’s table (remove) | | |
| **Triggers** | Receptionist select Adjust | | |
| **Preconditions** | None | | |
| **Basic flow**  **(Success)** | |  |  |  | | --- | --- | --- | | **No.** | **Proceed by** | **Action** | | 1 | Receptionist | Select customer record | | 2 | System | Show customer’s active table | | 3 | Receptionist | Select table and remove | | 4 | System | Set table as available and refresh the cart for the tables | | 5 | System | Announce succeed, go to active table page | | | |
| **Alternative flow** | b. Receptionist cancel. System go back to homepage | | |
| **Post-conditions** | None | | |

#### Usecase Receptionist Adjust (add)

|  |  |  |  |
| --- | --- | --- | --- |
| **Usecase code** | UC03 | **Usecase Name** | Receptionist Adjust (add) |
| **Actor** | Receptionist | | |
| **Goals** | Change customer’s table (add tables) | | |
| **Triggers** | Receptionist select Adjust | | |
| **Preconditions** | None | | |
| **Basic flow**  **(Success)** | |  |  |  | | --- | --- | --- | | **No.** | **Proceed by** | **Action** | | 1 | Receptionist | Select customer record | | 2 | System | Show customer’s active table | | 3 | Receptionist | Select add | | 4 | System | Show available table | | 5 | Receptionist | Select available table(s) | | 6 | System | Set tables as Being used | | 7 | System | Set correlation between tables and the customer record | | 8 | System | Announce succeed | | | |
| **Alternative flow** | b. Receptionist cancel. System go back to homepage | | |
| **Post-conditions** | None | | |

#### Usecase Receptionist Check-out

|  |  |  |  |
| --- | --- | --- | --- |
| **Usecase code** | UC04 | **Usecase Name** | Receptionist Check-out |
| **Actor** | Receptionist | | |
| **Goals** | Check-out for customer | | |
| **Triggers** | Receptionist select Check out | | |
| **Preconditions** | None | | |
| **Basic flow**  **(Success)** | |  |  |  | | --- | --- | --- | | **No.** | **Proceed by** | **Action** | | 1 | Receptionist | Select customer record | | 2 | System | Show customer’s active table | | 3 | Receptionist | Confirms tables with customer select check out | | 4 | System | Show list of orders and calculate the bill | | 5 | Receptionist | Proceed payment and confirm | | 4 | System | Set table as available and refresh the cart for the tables | | 5 | System | Announce succeed, go to active table page | | | |
| **Alternative flow** | b. Receptionist cancel. System go back to homepage | | |
| **Post-conditions** | None | | |

#### Usecase Customer Order

|  |  |  |  |
| --- | --- | --- | --- |
| **Usecase code** | UC05 | **Usecase Name** | Customer Order |
| **Actor** | Customer | | |
| **Goals** | Order dishes | | |
| **Triggers** | Customer uses the application to order | | |
| **Preconditions** | Table is active | | |
| **Basic flow**  **(Success)** | |  |  |  | | --- | --- | --- | | **No.** | **Proceed by** | **Action** | | 1 | Customer | Select order | | 2 | System | Show list of dishes | | 3 | Customer | Select a dish | | 4 | System | Show dish details | | 5 | Customer | Select order | | 6 | System | Show order page | | 7 | Customer | Enter amount, size | | 8 | System | Calculate price (with percentage discount) | | 9 | Customer | Place order | | 10 | System | Save order, announce succeed | | 11 | Customer | Select view order list | | 12 | System | Show order list | | | |
| **Alternative flow** | 5b. Customer select combo discount  6b. System show order page of combo  7b. Customer enter amount  11c. Customer select order more  12c. System shows list of dishes | | |
| **Post-conditions** | None | | |

#### Usecase Customer Request update order

|  |  |  |  |
| --- | --- | --- | --- |
| **Usecase code** | UC06 | **Usecase Name** | Customer Request update order |
| **Actor** | Customer | | |
| **Goals** | Customer request to update order | | |
| **Triggers** | Customer select update for an order | | |
| **Preconditions** | Order is not served or ready. | | |
| **Basic flow**  **(Success)** | |  |  |  | | --- | --- | --- | | **No.** | **Proceed by** | **Action** | | 1 | Customer | Select request update | | 2 | System | Show order details edit page | | 3 | Customer | Update details and submit | | 4 | System | Receive request | | | |
| **Alternative flow** | b. Customer cancel. System go back to order’s details | | |
| **Post-conditions** | None | | |

#### Usecase Kitchen accept/reject customer update

|  |  |  |  |
| --- | --- | --- | --- |
| **Usecase code** | UC07 | **Usecase Name** | Kitchen process update |
| **Actor** | Kitchen | | |
| **Goals** | Accept customer’s update or reject it | | |
| **Triggers** | Customer sent request, kitchen view will refresh to see the request | | |
| **Preconditions** | There are update from customer for the order | | |
| **Basic flow**  **(Success)** | |  |  |  | | --- | --- | --- | | **No.** | **Proceed by** | **Action** | | 1 | System | Shows order with updates | | 2 | Kitchen | Open order to check request | | 3 | System | Show request, comparing the changes | | 4 | Kitchen | Accept update | | 5 | System | Change order | | | |
| **Alternative flow** | 4b. Kitchen reject. System update request customer to rejected | | |
| **Post-conditions** | None | | |

#### Usecase Kitchen Update order

|  |  |  |  |
| --- | --- | --- | --- |
| **Usecase code** | UC08 | **Usecase Name** | Kitchen update order |
| **Actor** | Kitchen | | |
| **Goals** | Set the order to ready so waiters can take it to serve | | |
| **Triggers** | Kitchen finish preparing the dish | | |
| **Preconditions** | There are customer order | | |
| **Basic flow**  **(Success)** | |  |  |  | | --- | --- | --- | | **No.** | **Proceed by** | **Action** | | 1 | Kitchen | Open order detail | | 2 | System | Shows order detail | | 3 | Kitchen | Update order to Ready | | 4 | System | Announcing succeed and go back to order list | | | |
| **Alternative flow** | 3b. Kitchen cancel. System go back to order list | | |
| **Post-conditions** | None | | |

#### Usecase Waiter Update order

|  |  |  |  |
| --- | --- | --- | --- |
| **Usecase code** | UC09 | **Usecase Name** | Waiter Update order |
| **Actor** | Waiter | | |
| **Goals** | For waiter to update the order to served after serving the dish to customer | | |
| **Triggers** | Waiter successfully serve the customer | | |
| **Preconditions** | There are Ready order | | |
| **Basic flow**  **(Success)** | |  |  |  | | --- | --- | --- | | **No.** | **Proceed by** | **Action** | | 1 | Waiter | Open order details | | 2 | System | Show order details | | 3 | Waiter | Update the order to Served | | 4 | System | Update the information. And announce success | | | |
| **Alternative flow** | None | | |
| **Post-conditions** | None | | |

## Non-functional requirement

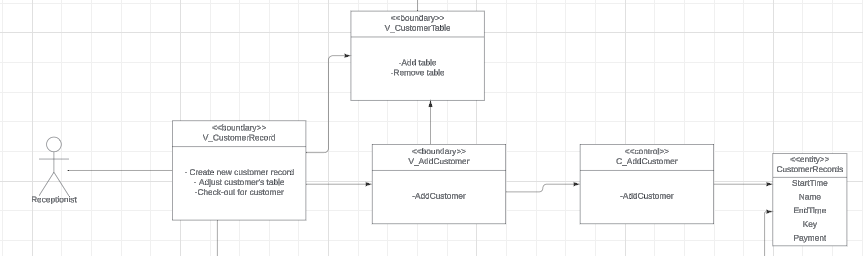
There is no non-functional requirement

# System Designing

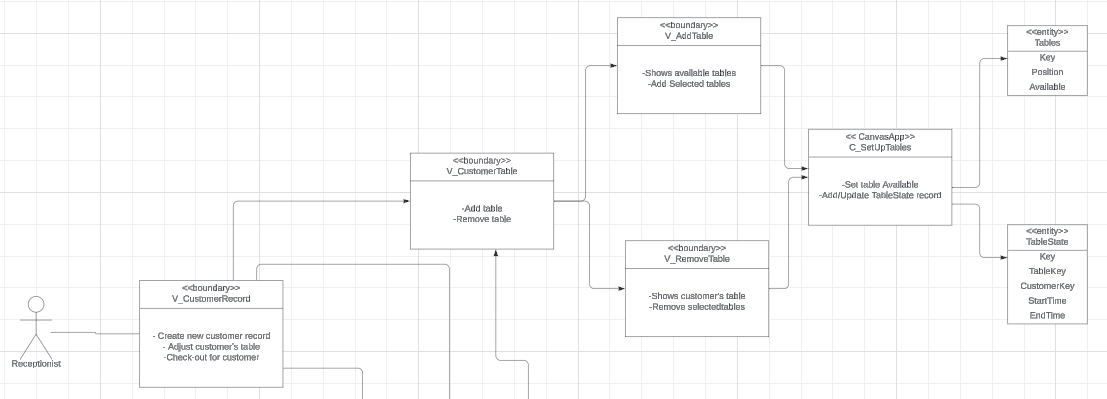
## System Structure Modeling

### Receptionist’s Usecases:

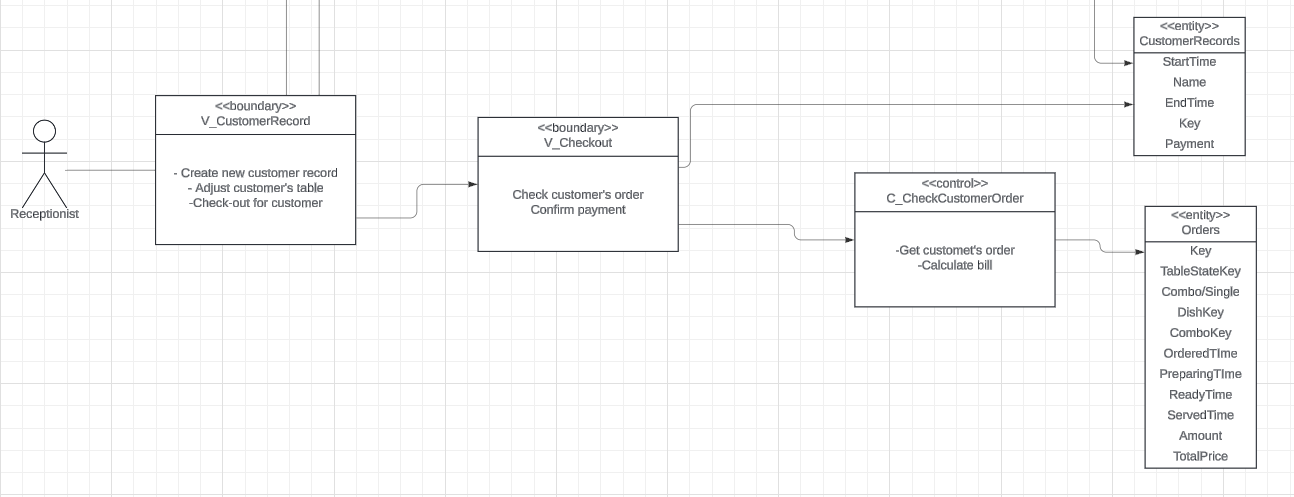
Usecase Receptionist Check-in:



Usecase Receptionist Adjust:

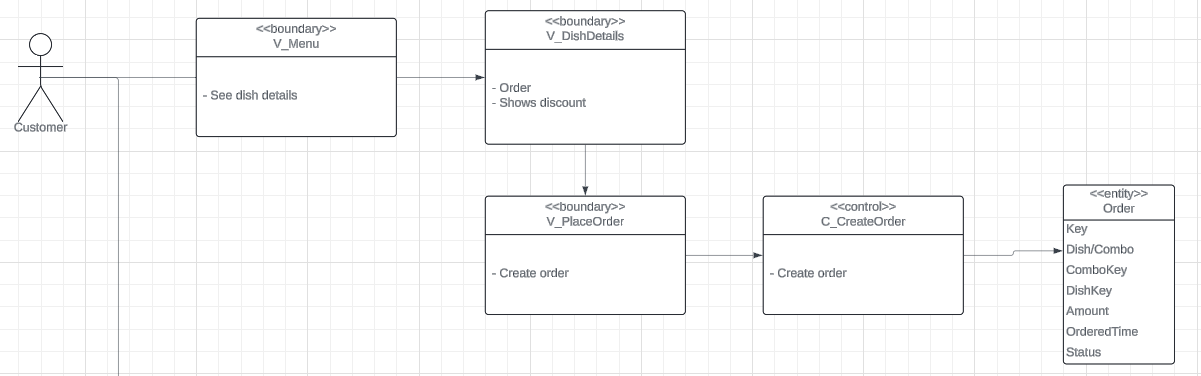


Usecase Receptionist Check-out

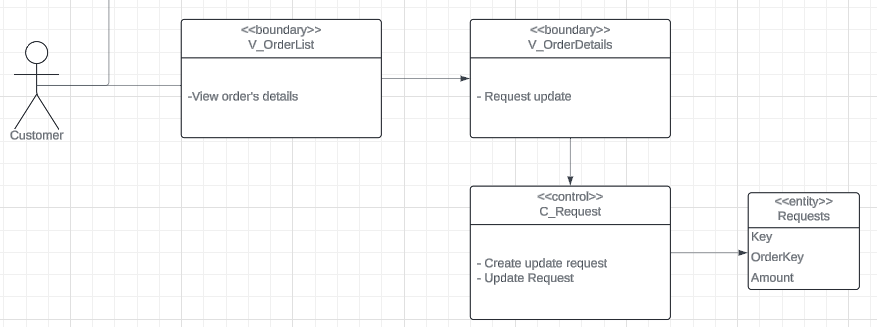


### Customer’s Usecases:

Customer Order Usecase:

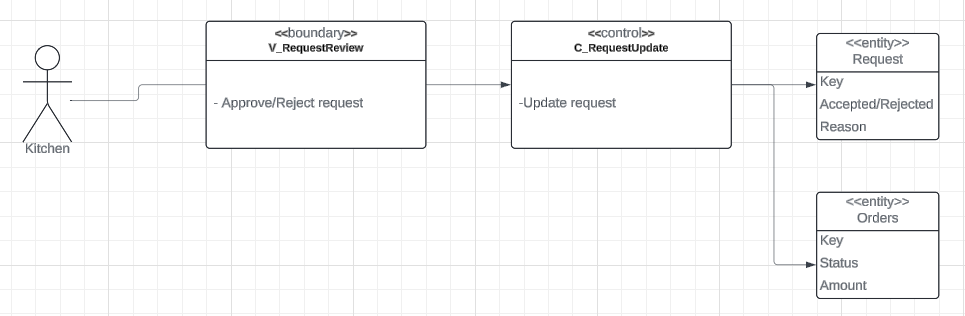


Usecase Customer Request

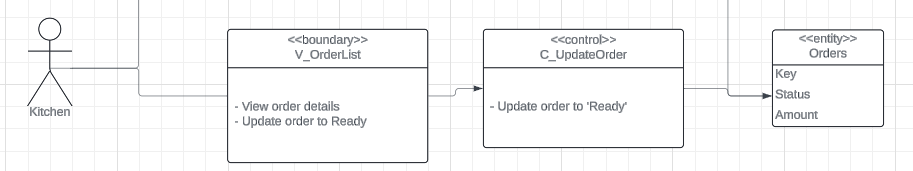


### Kitchen’s Usecases:

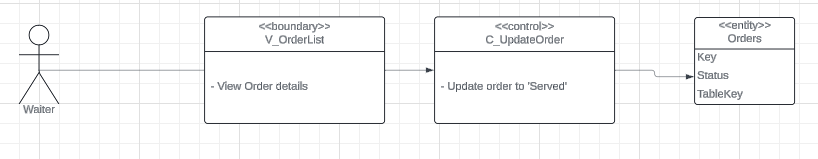
Usecase Kitchen Accept/Reject customer update:



Usecase Kitchen Update order:

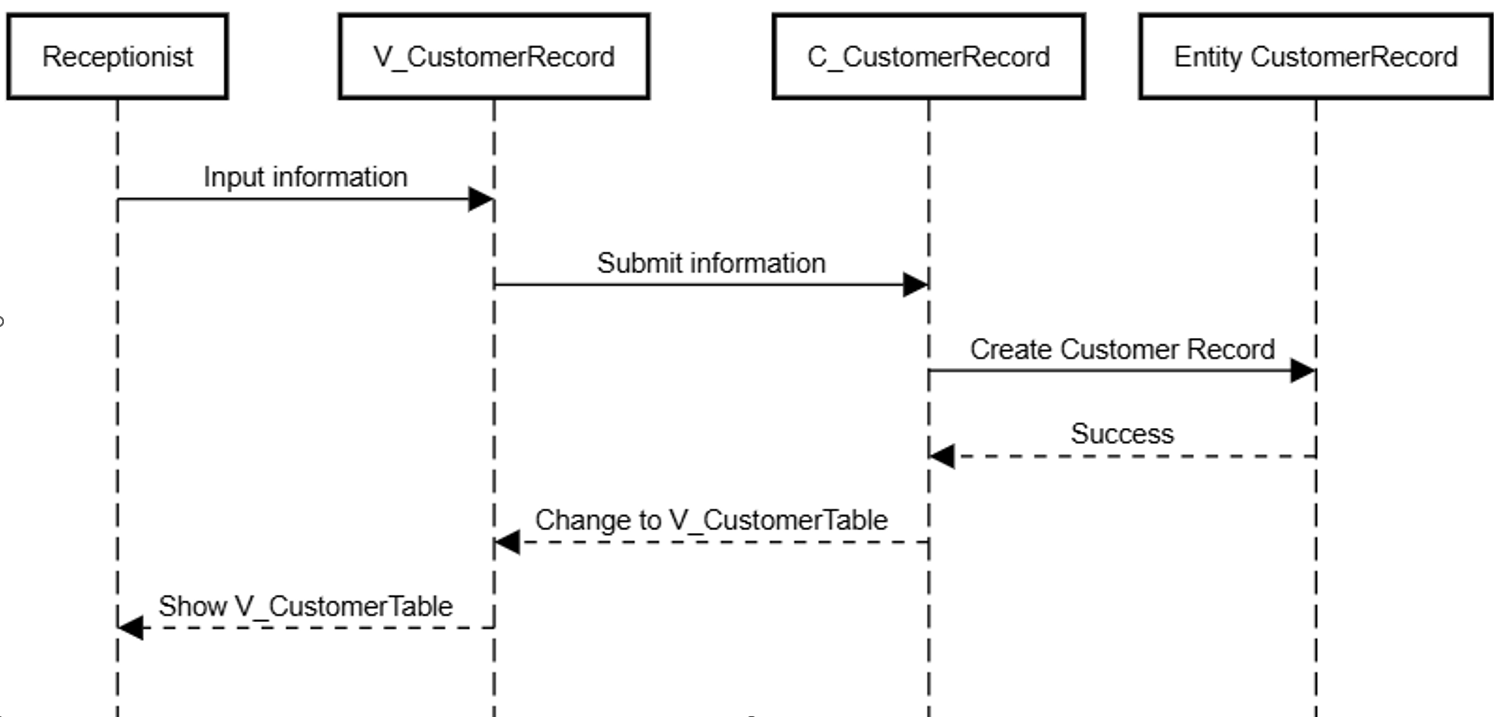


### Usecase Waiter Update order

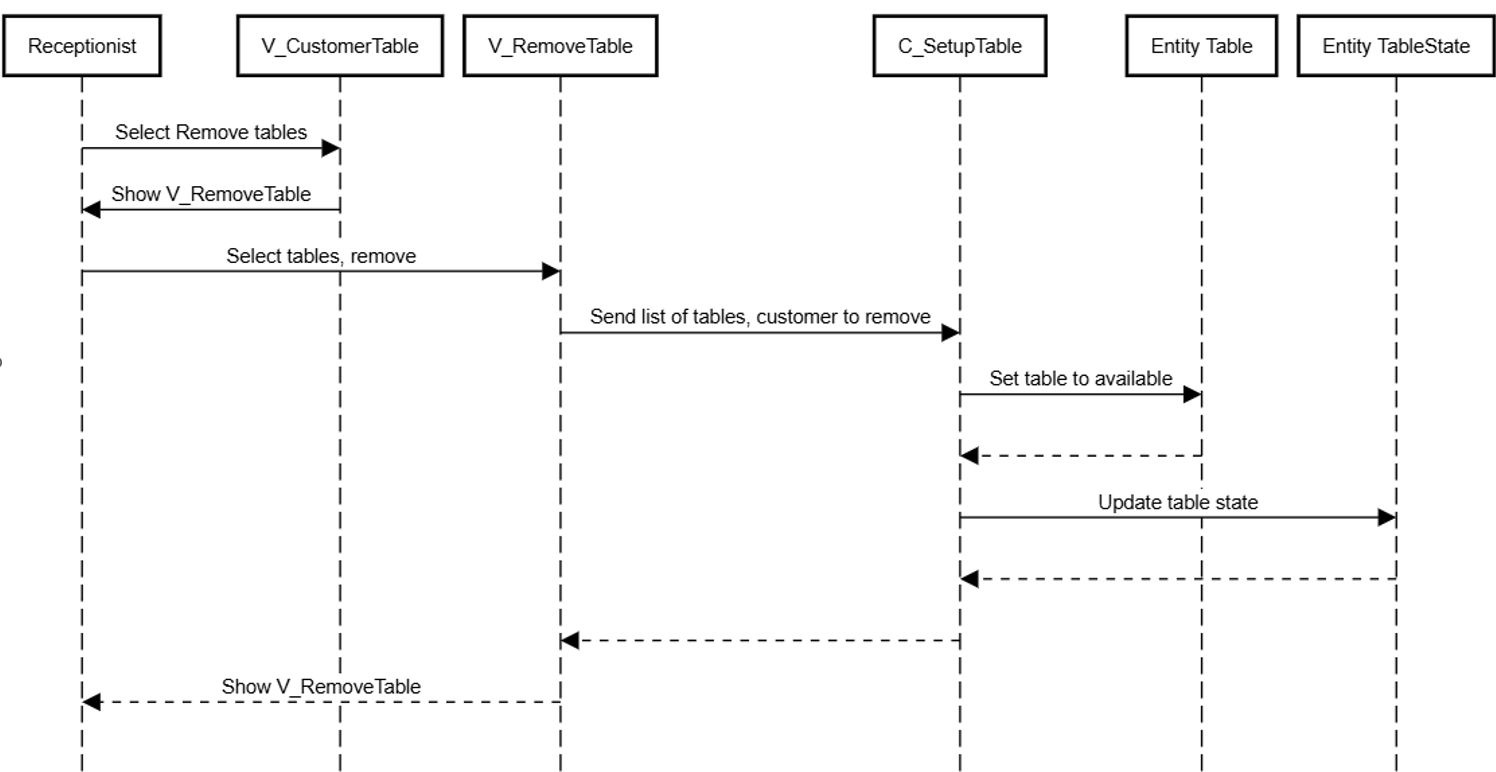


## Interaction Modeling

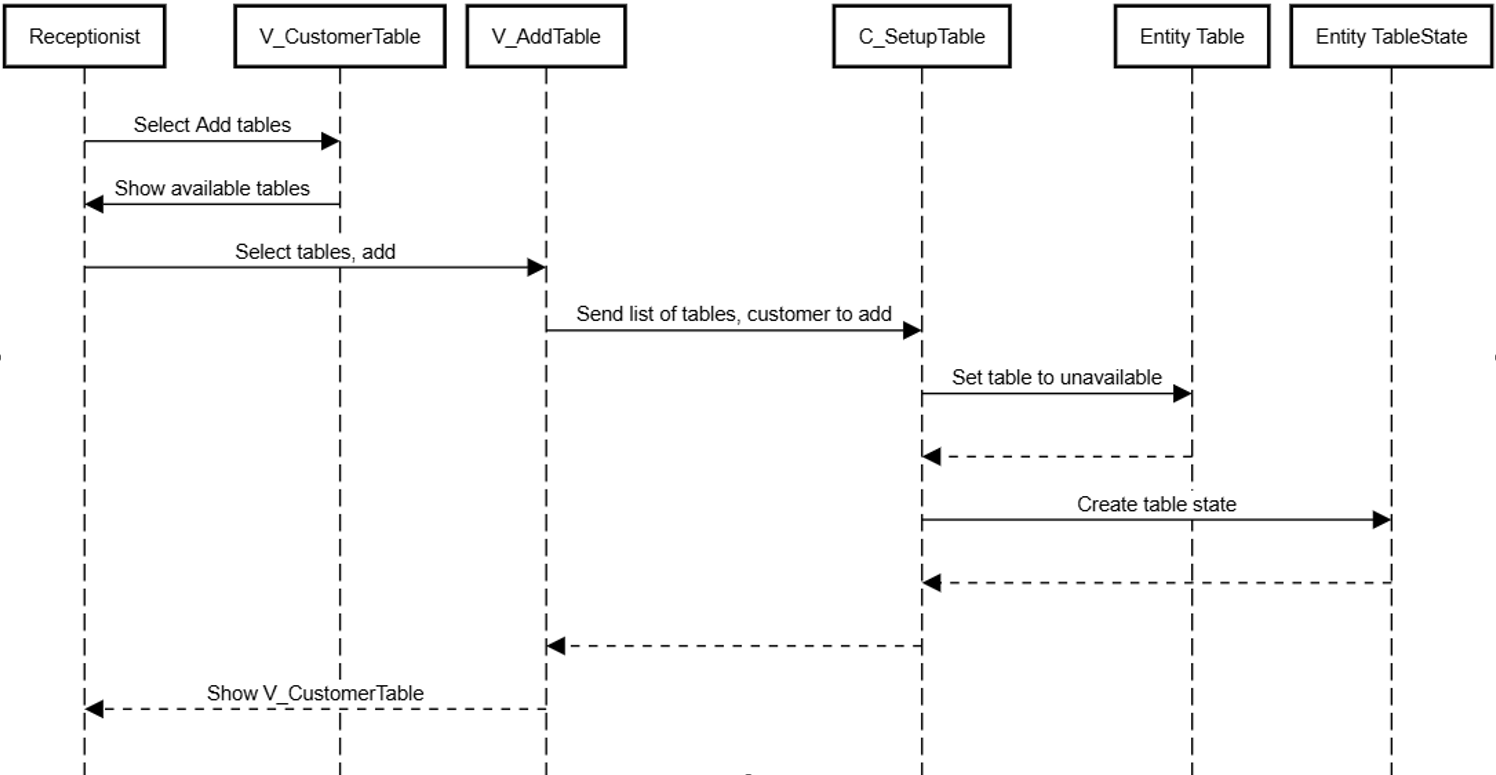
### Usecase Receptionist check-in



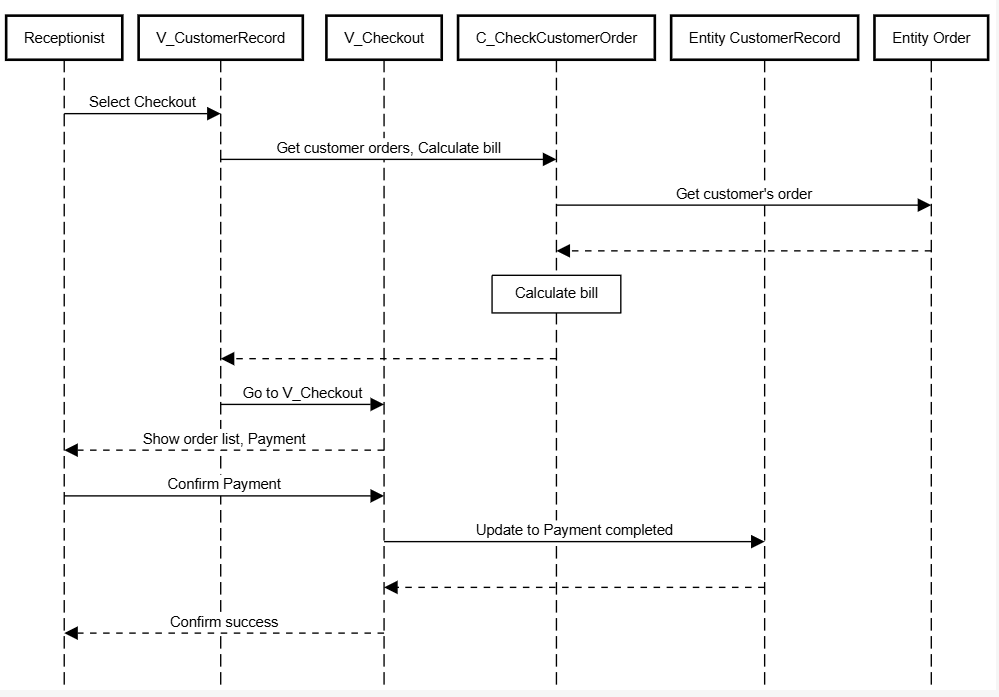
### Usecase Receptionist Adjust (remove)



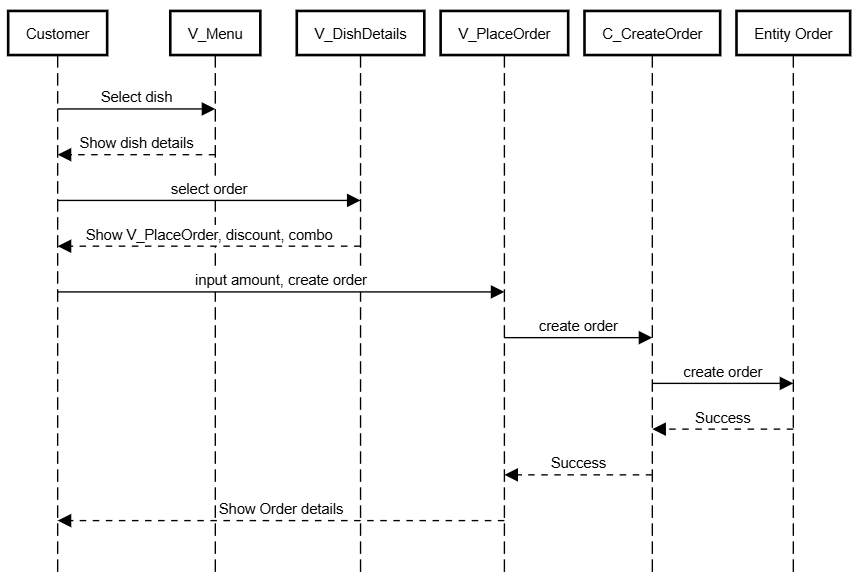
### Usecase Receptionist Adjust (add)



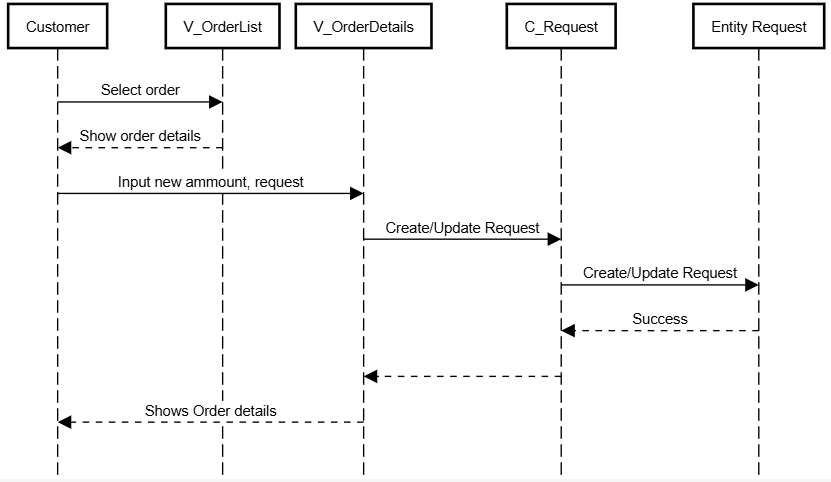
### Usecase Receptionist Check-out



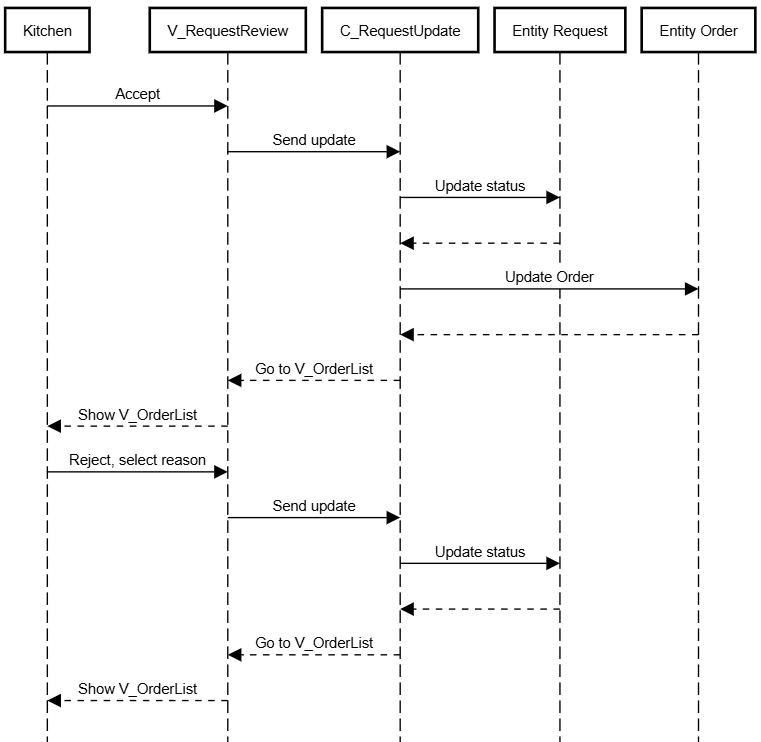
### Usecase Customer Order



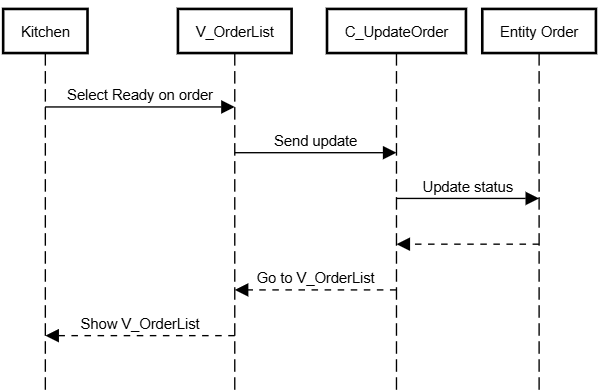
### Usecase Customer Request update order



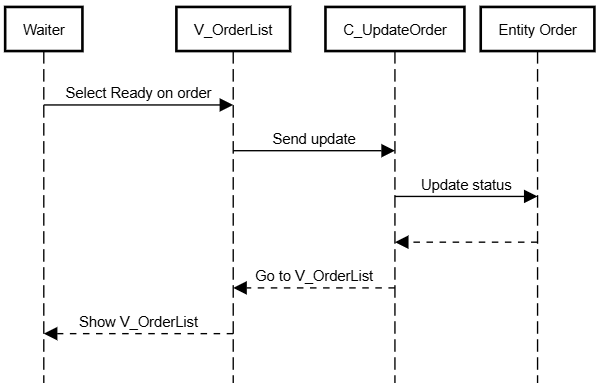
### Usecase Kitchen Accept/Reject customer’s update



### Usecase Kitchen Update order

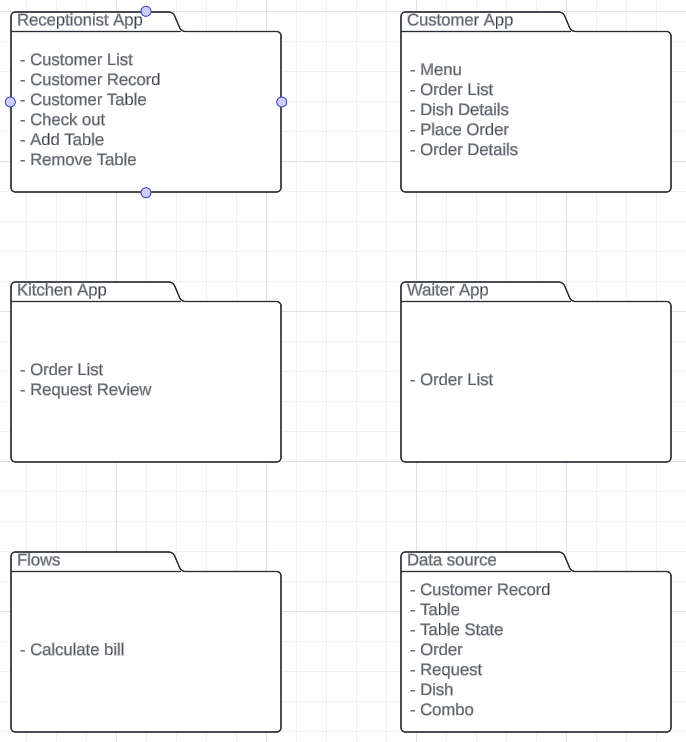


### Usecase Waiter Update order

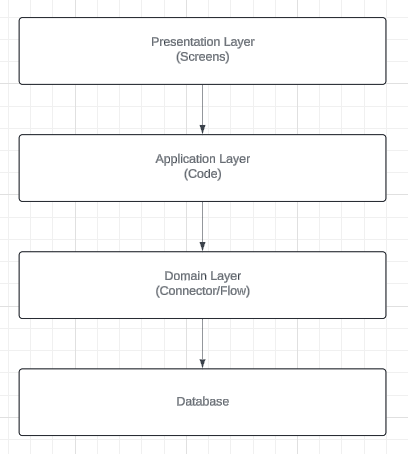


## System Architecture

Package architecture:



Layered architecture:



### Receptionist App

* + - * + Customer List:

Shows list of active customer

Name, Check-in time

> Open customer record

> Add new customer

> List of checked-out customer

* + - * + Customer Record

Customer’s name, check-in time

List of Customer’s tables

> Add Table

> Remove Table

> Check out

* + - * + Customer Tables

List customer’s tables

> Add tables

> Remove tables

* + - * + Check out

List orders from customer, total payment

> Update customer to Checked-out

* + - * + Add Table

List available tables

* + - * + Remove Table

List customer’s tables

### Customer App

* + - * + Menu

List dishes

> See dish details

> See Order list

* + - * + Order List

List orders

> See order details

> See Menu

* + - * + Dish Details

Show dish details: name, picture, descriptions, ingredients, price, discount, combo

> Order

* + - * + Place Order

> Input info: amount

Shows calculated price

* + - * + Order Details

Shows status, code, dish name, picture,

> Request change

### Kitchen app

* + - * + Order List

Show list of pending orders: Dish, table, notification for Update Request

> Update order to Ready

> View Request

* + - * + Request Review

Show old and new amount of order, order details

> Accept, reject + give reason

### Waiter app

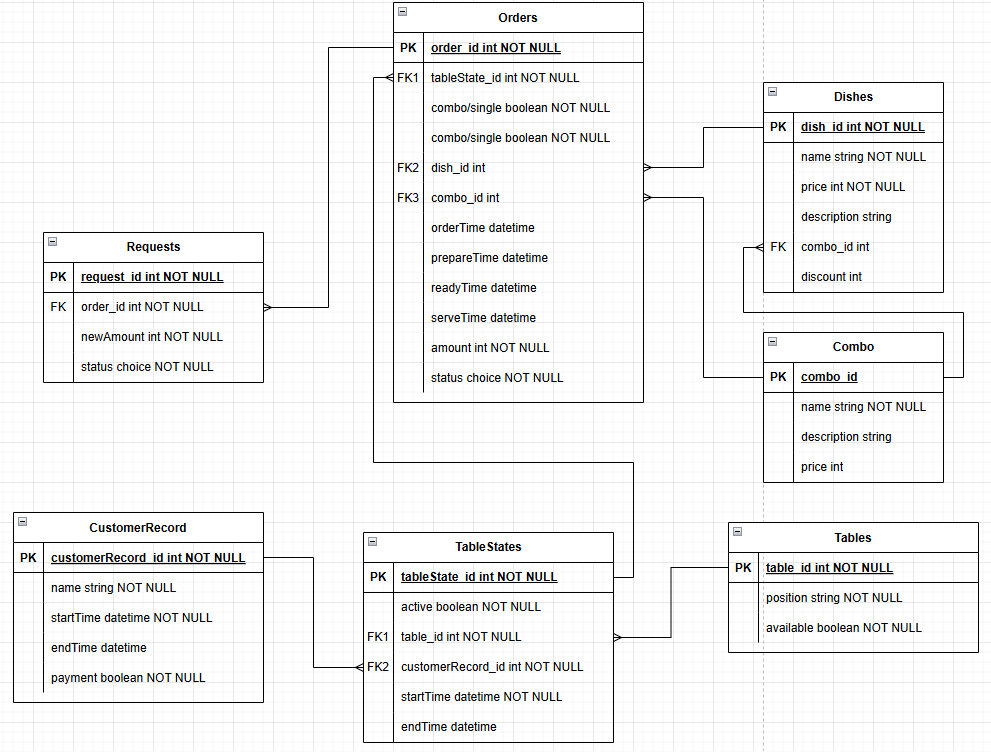
* + - * + Order List

Show list of ready orders: Dish, table

> Update order to Served

## Data source architecture

### Entity Relationship Diagram



### Entity description

#### CustomerRecord

* + - * + customerRecord\_id: ID of customer record
        + name: name of Customer
        + startTime: the time customer checked in
        + endTime: the time customer checked out
        + payment: whether customer paid or not

#### Tables

* + - * + table\_id: ID of the table in the restaurant
        + position: position of the table in the restaurant
        + available: whether the table is available or not

#### TableStates

* + - * + tableState\_id: ID of the record. When a customer started on a table
        + active: whether the customer is still using this table
        + table\_id: ID of the table that the customer is using
        + customerRecord\_id: ID of the customer who using the table
        + startTime: time when customer started to use the table
        + endTime: time when customer stop using the table (to checkout, to move to different table)

#### Dishes

* + - * + dish\_id: ID of the Dish
        + name: name of the dish
        + price: price for one dish
        + description: description about the dish, introduction, ingredients.
        + combo\_id: Combo that the dish is in
        + discount: amount of discount for the dish

#### Combos

* + - * + combo\_id: ID of the Combo
        + name: name of the combo
        + price: price for one combo
        + description: description about the combo, introduction, dishes included, ingredients.

#### Orders

* + - * + order\_id: ID of the order
        + tableState\_id: the table that ordered
        + combo/single: whether the order is for a dish or combo
        + dish\_id: ID of the dish in order
        + combo\_id: ID of the combo in order
        + orderTime: time where the order take place
        + prepareTime: time where the order prepared
        + readyTime: time where the order ready
        + serveTime: time where the order served
        + amount: amount of the dishes in the order
        + status: current status of the order

#### Requests:

* + - * + request\_id: ID of the request
        + order\_id: ID of the order that is requested to change
        + newAmount: new amount that changing in the order
        + status: current status of the request.